Scalable Natural User Interfaces for Data-Intensive Exploratory Visualization: Designing in the Context of Big Data

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Dane M. Coffey

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Adviser: Daniel F. Keefe

July, 2014

Acknowledgements

This dissertation, and the work contained in it, would not have been possible without the support and contribution of many people. I would like to thank all those involved.

First and foremost, I would like to thank my peers at the University of Minnesota, including all the members of the Interactive Visualization Lab and the larger computer graphics group. Specifically, I am grateful for the contributions of Bret Jackson, David Schroeder, Lauren Thorson, Fedor Korsakov, Vamsi Konchada, Nicholas Malbraaten, and Seth Berrier. In addition to the collaborations and research we performed together, these individuals also provided support simply from going through the same challenges together. Becoming friends and working together in the lab with all of you has made my 5 years here truly a joyful struggle.

I want to thank all of my non-academic friends who have supported me through the years. In particular, I'd like to thank Jacob Kincaid, Bret Samelson, Adam Smith, Eli Weinmann, Dylan Murray, Ehren Whigham, Stephen Rashid, and Chris Weiser. Although we all live spreadout across the country now, you've all been a source of continual support throughout the years. In my pursuit of this accomplishment, It's been both difficult and rewarding seeing many of you get married and succeed in your careers. I also want to give special thanks to Robin Palmer for being such a great and supportive friend my last year and a half.

Perhaps the most thanks goes to my adviser and friend Daniel Keefe. I could not have asked for a better adviser throughout my research career. Your advice and guidance has helped me get past tough research problems, all the while you've afforded me the freedom to explore ideas and progress on my own. Working in this lab, playing with cool toys, building awesome input and output devices, and seeing the lab grow has been a blast.

My appreciation also goes to the the faculty members that have advised and guided me through the years. This includes my disseration commitee members Arthur Erdman, Victora Interrante, and Gary Meyer. Specifically, Arthur Erdman has been a critical advisor throughout my 5 years here, giving me the advice and resources to succeed. The interdisciplinary aspect of my research significantly strengthens it, and without you this would not have been possible. I would also like to thank John Carlis for his advice and support as well, particuarly in regards to my writing.

I would also like to thank my direct collaborators and fundings sources that supported and contributed to this research. This includes Chi-Lun Lin, John Huss, Cory Schaffhausen, Trung Le, and Fotis Sotiropoulos. The funding for this work has come from many sources, including Boston Scienfitic, the National Science Foundation, the National Institutes of Health, and the University of Minnesota.

Abstract

This dissertation investigates new exploratory visualization tools in data-intensive domains that are built upon natural user interface technology, including multi-touch surfaces and virtual reality. Scalable interactions are developed and evaluated in several software systems that are targeted toward supporting design workflows that make use of big data.

A new immersive and highly-interactive multi-touch workbench is presented, along with a theoretical framework and evaluation of how visualizations may be developed on it. Building upon this foundation, two different exploratory visualization software systems are presented that address distinct challenges faced by designers working in data-intensive domains. The first of these systems is called Slice World-In-Miniature (WIM), which is designed to overcome the difficulty associated with exploring large-volume data, where the complexity of the data often leads to the designers becoming disoriented. Using Overview+Detail techniques to provide context, the designer navigates inside of complex volumes using multi-touch gestures. This Slice WIM system is applied to a number of medical device design applications and evaluated by domain experts in this field. The second system is called Design by Dragging, which addresses the information overload associated with comparing and navigating many sets of interelated simulations. Design by Dragging gives the designer the power to explore high-dimensional simulation design spaces by using natural direct manipulation interactions. This system is applied to several problems in medical device design and in visual effects simulation, and a domain expert evaluation is presented.

The big data paradigm is integrally tied to the future of computing. The major contribution of this dissertation is its investigation into the effectiveness of natural user interfaces as a means of working in this paradigm. Although natural user interfaces have become ubiquitous in our daily lives, they are typically used only for simple interactions. This dissertation demonstrates that these technologies can also be effective in aiding design work in the context of big data, a result that could shape the future of computing and change the way designers work with computers.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

We begin by providing the background of and motivation for the core contributions of this dissertation research by placing it within the context of current challenges in the computer science research.

1.1.1 The Fourth Paradigm: Data-Intensive Computing and Visualization

Modern computing power has changed the way we perform science across all domains, so much so that many argue we have now entered a so-called "fourth paradigm" that will bring with it new scientific techniques that go beyond the current computational science we now employ [1, 2]. The study of the development of science makes clear that scientists operated in the first two paradigms for many hundreds of years as they moved from an experimental paradigm onto a theoretical paradigm. With the relatively recent development of computing, we have seen science move beyond those first two paradigms into a third, a computational paradigm that relies upon the simulation of increasingly complex phenomena. The new fourth paradigm, referred to as data-intensive science, is characterized by the abundance of data that scientists now have available at their disposal (i.e., Big Data [3, 4, 5]). These data arrive from many different sources (e.g., captured by instruments or generated through simulations) in such quantity that

scientists often find themselves overwhelmed by their volume.

Given that vision is the highest bandwidth input to the human cognitive system [6], visualization will play a critical role in the development of this data-intensive science. This developing subarea is just now receiving the recognition it deserves in publications and journal special issues specifically addressing the numerous and specific challenges associated with big data visualization [7]. One of the most fundamental of these challenges is developing new algorithms and data structures capable of handling the high computational cost and complexity of data processing at extremely large scales. To handle data that continue to increase in simulation resolution and temporal sampling, new streaming data structures and out-of-core processing algorithms are being researched to handle large datasets [1]. Researchers in many different scientific domains have investigated this problem. In the microscopy domain, researchers have developed a system for interactive volume visualizations of petascale data [8]. In climate simulation, researchers are developing new tracking algorithms for meteorological features that are able to handle very large simulations [9]. Although many of these systems are domain-specific, many of the algorithms being developed are generalizable and will contribute to the ability of other visualization systems to handle big data.

The breadth of challenges associated with the visualization of big data goes far beyond the algorithmic and data structure complexity needed to manage large data. For example, consider the following list of identified data-intensive visualization challenges:

1. The high computational cost and complexity of processing data at extremely large scales [1].

2. The difficulty of maintaining a sense of provenance during exploratory creative data understanding sessions in increasingly large and diverse datasets [2].

3. The difficulty of transitioning between local and global views of data at ever-increasing scales [1].

4. The visual clutter and information overload associated with combining many different types of data (i.e., multi-modal data) [10].

Although these challenges are too numerous to iterate in whole, these examples represent the scope of the difficulties and opportunities facing the research community.

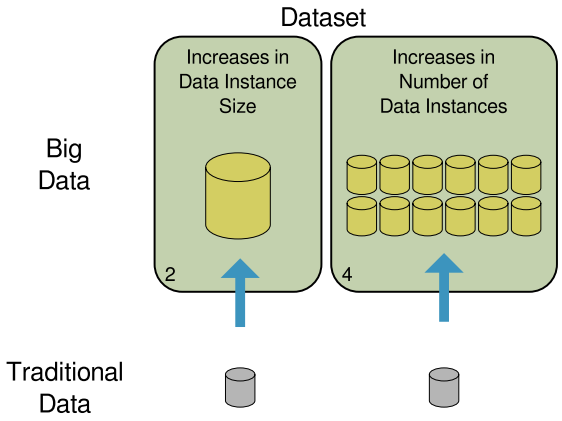

Figure 1.1: Two measures that cause a dataset to transition from traditional data to big data as it increases.

In this dissertation, we use a specific abstraction of the general challenges associated with data-intensive visualization. This abstraction considers two dataset measures that, as they increase, limit the application of existing visualization techniques by requiring higher bandwidth access to the data. If we consider the entire dataset in any given visualization problem domain, it will be composed of a set of individual instances (e.g., a single simulation "run" that represents indivisible elements of the domain. The first limiting measure of a dataset is the simulation instance size (i.e., the resolution of the simulation run). The second limiting measure of a dataset is the number of instances it contains. As shown in Figure 1.1, as these measures increase, traditional data is transformed into big data, as indicated by the blue vertical arrows.

Consider a user visualizing a dataset composed of many instances of scalar fields. In terms of the first measure, as the resolution of the individual field instances increases, the spatial complexity associated with the increased number of feature points in the instance will eventually overcome both the capability of a desktop monitor to convey the content of a field and the user's cognitive ability to perceive that many features. In terms of the second measure, as the number of inter-related scalar field instances increases, the combinatorial complexity associated with grouping these fields for comparison will eventually become too difficult for a user relying on existing systems to perceive. New data-intensive visualization systems are needed to overcome these limitations.

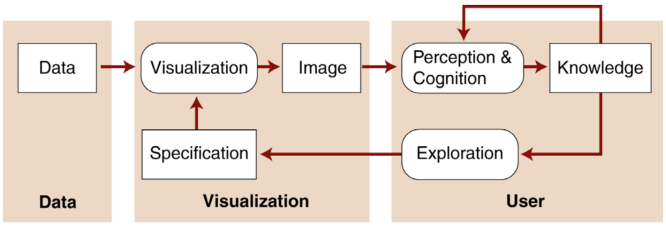

Figure 1.2: The ideal visualization discovery process goes beyond just the perception of static visuals and includes a feedback loop with the user at its center. In this loop, the user creates new specifications that affect the visualization, which in turns changes the user's perception of the data and leads to new insights.

1.1.2 Exploratory Visualization

In a data-intensive paradigm, the visualization techniques used for understanding data will not be static or rely on visuals alone. Instead, they will include a visualization discovery process, as outlined by Jarke van Wijk [11, 12] and shown in Figure 1.2. This process features a highly interactive feedback loop, with the user at the center of the loop. The integration of the visualization discovery process with data-intensive science is so critical that it has been continually identified as one of the most significant challenges facing the field of visualization[1, 11].

Although the tasks included in the visualization discovery process are numerous, a significant class of visualization tasks relies upon performing exploratory data analysis. Exploratory visualization (i.e., exploratory data analysis via interactive graphics) is employed when examining data without an explicit goal [13], such as when a user is becoming familiar with a new dataset or defining new hypotheses. The factor that distingiushes visual exploratory data analysis from other types of data analysis is that it places the human at the center of the process in a visual-interactive feedback loop that is used to guide exploration inside of a dataset, much like the visualization discovery process itself.

Although exploratory visualization is an active subarea of research within the visualization research community [11], previous work in this area has focused primarily on more traditional data, such as the multiple coordinate views strategies used for geospatial data [14] and biomechanics data [15]. When exploring traditional data, the

visual-interactive feedback loop is most commonly implemented by a basic desktop visualization and mouse interaction. A big data visual-interactive feedback loop, however, must support higher bandwidth access than that used with traditional data in order to overcome the two limiting measures of big data and to support data-intensive exploratory visualization.

1.1.3 Natural User Interfaces

New data-intensive exploratory visualization systems must focus on both the visual and interactive elements of the feedback loop. In the field of human-computer interaction (HCI), one particular class of interfaces, natural user interfaces (NUI), stands out as being particularly well-suited to overcome the limitations associated with big data. Natural user interfaces are interfaces that evoke a feeling of effortless interaction for the user. They accomplish this by borrowing elements from the natural ways in which humans interact with their surrounding environment. For example, virtual reality (VR) technology uses stereoscopic and head-tracked graphics to mimic the way humans view the world by moving their head around. Multi-touch technology allows users to reach out and touch objects displayed on screen similarly to the way that humans manipulate real-world objects.

The affordances granted by NUI technology have the potential to greatly increase the bandwidth of the visual-interactive feedback loop. A common characteristic of this technology and interaction is that the interface feels almost invisible to the user. This often results in users' having a sense of a direct connection to their data, allowing them to see their data in new ways or to make new connections that would otherwise be missed if they had been distracted by the interface. Ideally, using natural user interfaces should allow users to gain more insight into their data.

In the commercial realm, the ubiquity of input and output devices that support NUI has increased dramatically [16, 17]. This includes the prevalence of touch screens and the commoditization of virtual reality (VR) technology, technologies that are no longer limited to the research lab due to their high cost. iPads, touch-sensitive smartphones, and Kinects have become devices that we interact with daily. The low point of entry and familiarity of these technologies make this an ideal point in time to leverage these technologies to use with data-intensive exploratory visualization.

Despite their ubiquity, most current devices supporting NUI are typically used only for simple interactions and for playful if often still functional applications. It remains to be seen if these technologies are useful in applications that require more complex workflows, such as the advanced workflows of engineers, artists, and scientists. Thus, a major goal of this dissertation is to investigate the effectiveness of NUI technology in this capacity and its potential to advance beyond simple desktop and mice devices to shape and change the way users will work in the future.

1.2 Our Approach

This section describes our approach to investigating new exploratory visualization techniques that can handle big data.

1.2.1 A New Approach for Data-Intensive Exploratory Visualization: Scalable Natural User Interfaces

This dissertation introduces new natural user interfaces for exploratory visualization (i.e., exploratory visualization tools in which natural user interfaces are used to understand data). Furthermore, these natural user interfaces are designed to work in data-intensive domains by scaling the two measures of a dataset that increase as they transition toward big data. Thus, the contributions of this dissertation are new scalable natural user interfaces for data-intensive exploratory visualization.

A summary of the computational framework for the scalable natural user interfaces examined in this dissertation is presented in Figure 1.3. Both sides of the figure show the visual-interactive feedback loop, focused on the human at its center. The left loop represents the feedback loop for traditional data, which typically uses mouse and keyboard input and a desktop monitor. The right represents the transition to big data, in which a higher bandwidth feedback loop is needed, as indicated by the thicker green arrows. This new visual-interactive feedback loop is built upon the foundations of NUI technology and theoretical evaluations to guide the design of the exploratory visualizations.

Figure 1.4 shows one specific interface implementation of this new high-bandwidth visual-interactive feedback loop. This implementation builds upon two different types of

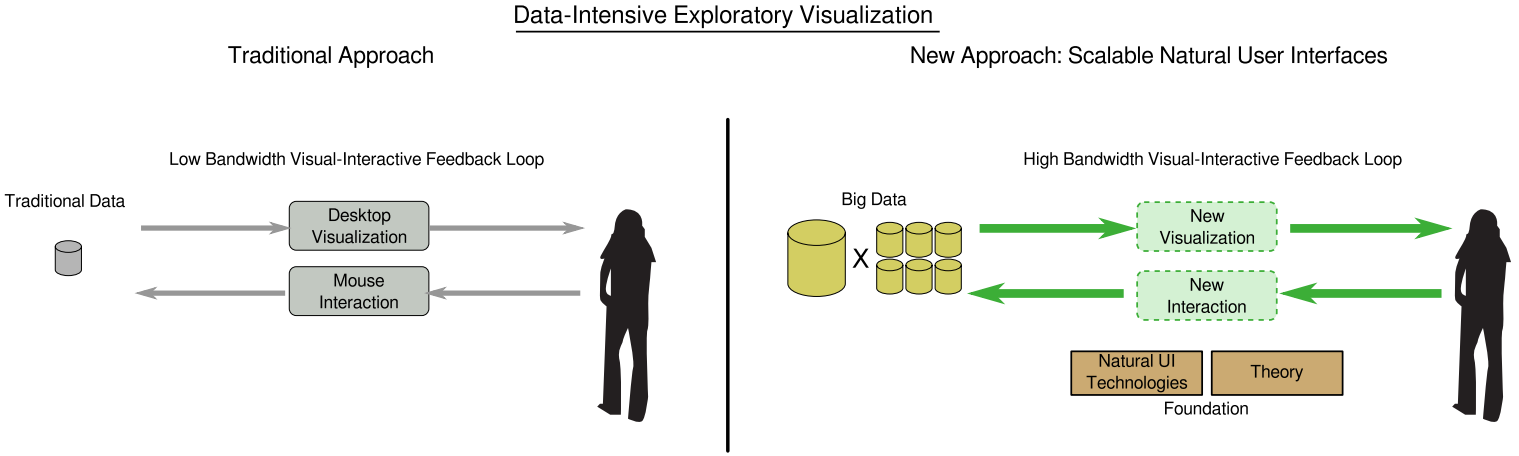

Figure 1.3: Computational framework, showing the limits of existing exploratory visualization systems (left). New visuals and interactions (right) are needed to enable a high-bandwidth visual-interactive feedback loop that supports data-intensive exploratory visualization. This system must handle data that scales both in the size of individual data samples and the number of data samples to explore. Building these systems on a foundation of interaction theory and natural user interface technologies will maximize the bandwidth between the designer and the dataset.

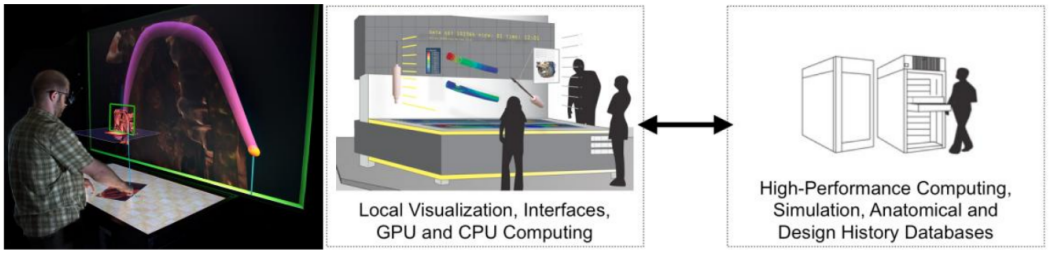

Figure 1.4: The image on the left side of the figure illustrates the multi-touch virtual reality interface I developed to explore anatomical volumetric datasets; the illustration on the right shows how this virtual workbench can be used in the medical device design process by connecting it to high-performance computing resources.

NUI technology: multi-touch input and virtual reality (VR). The interface presented in this figure represents the vision of this dissertation, in which these technologies are used to provide new means of exploring and analyzing complex data. The particular interface shown is being used to explore a complex 3D anatomical dataset resulting from patient-specific medical imaging. Using the rich input space provided by the touch-sensitive surface, the user is able to explore and interrogate the environment. We view this particular hardware combination as a potential workbench of the future, one on which a user is able to perform exploratory visualizations of big data. The right side of the figure presents a concept sketch of how many different sources of data could be integrated into this workbench and how the power of this workbench may be dramatically increased by connecting it to high-performance computing resources. Such a connection would give the user direct access to the raw horsepower needed to generate big data on the fly, greatly increasing the bandwidth of the visual-interactive feedback loop.

1.2.2 A Challenge: Designing in the Context of Big Data

The new data-intensive paradigm of science is impacting many fields. For example, in engineering fields, rapid advances in our ability to run simulations that generate massive amounts of raw data are shifting the workflow of engineers across many fields toward data-intensive computing. Engineers are beginning to use simulation to help with engineering design work in addition to its traditional role as a validation tool. For example, in the medical device field, engineers are now applying finite element

analysis (FEA) and computational fluid dynamics (CFD) software in a design capacity by evaluating many design alternatives through an iterative process. This software has only recently been able to be used in this way due to the power of widely available high-performance computing (HPC) resources that can generate many simulations in rapid succession. Designers struggling to effectively work with big data [18] need new exploratory visualizations to aid their efforts.

Although the work presented in this dissertation is generalizable to many different problem domains, the dissertation focuses specifically on medical domains, in particular the medical device design process and how it may be improved. Numerous medical device examples are presented as applications of our novel exploratory visualization techniques. The medical device design process has tremendous potential to be impacted by and benefit from data-intensive exploratory visualization techniques, and as such provides an ideal use case to demonstrate and evaluate the effectiveness of the research presented.

The current practices of engineers in the medical device design process are very costly and time-consuming, with many inefficiencies. The typical cycle is to identify a need and design a prototype, followed by many lengthy animal and clinical trials that are the main method of validating the device for the approval process. The core issue with this cycle is that it becomes increasingly difficult to iterate on the design after the initial phases of the cycle. The cost of changing the design after these trials are started is so high that a design freeze is typically placed on the device once trials are begun, discouraging further iterations or optimization of the design beyond this point. Thus, there is a tremendous benefit to encouraging iteration in early phases of this cycle and to receiving evaluative feedback on these iterations before moving to costly trials.

Simulation and modeling are ideal for iterating and evaluating alternatives early in the design cycle. Although these tools are already used to some extent in the medical device design process, they are usually employed for detailed validation of a specific aspect of the device late in the design cycle or for only preliminary inquiry into a design. Expanding the capabilities of the tools available to engineers will allow iterations to be performed early in the process by simulating thousands of design alternatives and identifying flawed designs long before entering animal and clinical trials. In this way, simulation and modeling are poised to impact medical device design much as they have

impacted such other fields as automotive and aerospace design.

In addition to providing key advances in the field of scientific visualization and human-computer interaction, the research presented in this dissertation will have a direct impact beyond computer science to benefit the medical device design research and industry community.

1.3 Overview

This section provides the thesis statement and an overview of the dissertation to come.

1.3.1 Thesis Statement

The goal of this dissertation is to evaluate the following thesis statement:

Scalable natural user interfaces can increase the effectiveness of data-intensive design tasks, overcoming limitations of current exploratory data visualization tools and enabling useful new strategies for working with big data.

Our research tests this thesis statement through four separate research thrusts, following the framework laid out in Figure 1.3. These thrusts are as follows:

- Research Thrust 1: Develop novel hardware and user interface technologies that support scalable natural user interactions with big data.
- Research Thrust 2: Create the foundations of a theoretical framework for evaluating and predicting how users perform exploratory big data analysis tasks.
- Research Thrust 3: Develop new data-intensive exploratory visualization techniques and interaction metaphors for handling larger individual data instances.
- Research Thrust 4: Develop new data-intensive exploratory visualization techniques and interaction metaphors for handling increases in the number of data instances contained in a dataset.

In pursuing these research thrusts, we develop new interaction hardware, theoretical results, and techniques that provide valuable contributions to current knowledge in the general area of data-intensive exploratory visualization. We expect that these

contributions will apply to a wide array of application domains that face the challenges associated with big data. To demonstrate the effectiveness of these contributions, we focus primarily on medical domains and the design tasks that medical device engineers face. Throughout this dissertation, we provide specific examples applying our research in these domains that have also been assessed by domain-expert scientists. Through these applications, this dissertation supports the claim that scalable natural user interfaces are critical to the exploratory visualization of big data.

1.3.2 Overview of the Dissertation

The following discussion of this research begins in chapter 2 with a review of the relevant research in the area of data-intensive exploratory visualization. Chapter 3 introduces a novel hardware configuration we have developed for supporting scalable natural user interfaces that combines multi-touch and virtual reality technology. This chapter primarily covers our efforts toward Research Thrust 1. Chapter 4 presents our efforts toward Research Thrust 2, introducing our theoretical foundations through an interaction taxonomy and design study for developing exploratory visualizations. The results of this design study are in turn used to inform the design decisions in two data-intensive exploratory visualization tools developed in this dissertation. Chapters 5 and 6 describe the first of these tools, Slice WIM, which is used to perform exploratory visualization in large volumetric datasets. These two chapters comprise our efforts toward Research Thrust 3. Chapters 7 and 8 then present the second tool developed in this dissertation, Design by Dragging, which is a tool targeted toward enabling exploratory visualization with many sets of simulations inside of large design spaces. These chapters encapsulate our efforts toward Research Thrust 4. Chapter 9 discusses our conclusions and their implications for practice and for future research.

Chapter 2

Related Work

This section provides an overview of related work in a number of specific computer science research areas that have implications for data-intensive exploratory visualization.

2.1 Virtual Reality for Scientific Visualization

The history of immersive virtual reality technologies' aiding scientists through interactive visualizations dates back to early work in which head-mounted displays (HMDs) have been used to depict 3D simulation data [19]. One of the breakthrough examples of VR applied in this context is the well-known Responsive Workbench system [20, 21] that provided a new VR form factor for scientists. The Responsive Workbench consists of a horizontal tabletop display that is rear-projected and viewed through shutter glasses to provide 3D imagery to the user. The user's head is tracked to provide the correct perspective to the user, and the user employs gloves and a stylus that are also tracked and can be used to interact with virtual objects in the tabletop environment. Although this technology is now commonplace, this system was one of the first to combine these technologies in a workbench form factor that scientists could use to visualize their data and has had a lasting impact on the development of further work in this direction.

Following the development of the Responsive Workbench, the research community quickly recognized the potential of using immersive virtual reality to aid in the visualization of data [22]. As the technology continued to advance, many different form factors, including CAVEs [23], head-mounted displays [24], and fish tank systems [25],

were all being used to visualize data across a wide array of scientific domains. For example, VR has been used by scientists within medical domains to explore patient imaging datasets [26] and to immerse oneself inside of a bloodflow simulation at extreme scales [27, 28]. In geology, scientists have used VR to explore and understand complex 3D reconstructions at multiple scales and to view finite element method simulations of the interaction between tectonic plates [29]. In archaeology, scientists have used VR to recreate and explore ancient cities like Rome [30] and to perform virtual excavations that are driven by data collected across many years.

Despite the success of many of these applications, immersive virtual reality technologies have yet to be widely adopted as an essential asset in data visualizations. One reason for this may be the inaccessibility of the underlying hardware, which until recently had been largely limited to dedicated VR research labs due to the high cost. However, recent technological advances are rapidly driving down the cost of this hardware. Commodity projectors that are capable of displaying a 120Hz stereo image are now available, and bright, high-resolution LCD displays are cheap enough that they can be tiled to form extremely high resolution walls, leading to a new generation of immersive virtual reality environments, such as the 72 megapixel CAVE2 [31], that are capable of displaying large scale data in extreme detail.

The data-intensive exploratory visualization interfaces developed in this dissertation build upon immersive virtual reality technology to take advantage of the benefits this technology brings to the visualization process. These advantages include improved spatial relations and context, the ability to collaborate in new ways, and the power to immerse and scale oneself within the data [32]. Furthermore, part of this dissertation work includes the design and development of a new low-cost VR configuration ideally suited for data-intensive exploration that is intended to these benefits when exploring big data.

2.2 Natural User Interaction and Direct Manipulation Interaction in Visualization

The quality of the interactions a scientist performs when interfacing with an exploratory visualization greatly impacts the success of the visualization. Scientists need to be able

to perform their tasks with an interface that complements the way they work, as opposed to getting in their way. Creating interfaces that accomplish this remains a significant challenge for the scientific visualization community [33]. Natural user interfaces are an ideal class of interactions to build upon for this purpose. Their innate characteristics often make the interface appear invisible to scientists, giving them a sense of direct access to the entirety of their data and thus making NUIs an ideal candidate for scientific visualization interaction [34].

Direct manipulation interfaces are a specific type of natural user interface that are well-suited for exploring data. Direct manipulation interfaces are those that encompass three principles: [35]

- continuous representation of the object of interest,
- physical actions or labeled button presses instead of complex syntax, and
- rapid incremental reversible operations whose impact on the object of interest is immediately visible.

Interfaces that follow these principles will allow a scientist to fluidly and rapidly navigate and quickly reverse mistakes, thereby encouraging exploration of the data. Researchers are recognizing a need for these interfaces, given that traditional control panels are not capable of handling the scale of big data, and are exploring the potential of direct manipulation for the visual analytics of common navigation tasks such as model steering, feature selection, and feature extraction [36]. Incremental continuous actions are also ideally suited to working with big data, where data processing is too complex to compute in real time [37].

Direct-touch surfaces, particularly multi-touch-capable surfaces, have recently become synonymous with NUI and direct manipulation, and there are many example of visualizations that use touch surfaces as input devices [38]. For example, touch input is being used as a way to explore and create visualizations of 2D flow fields [39, 40]. Multi-touch interfaces are being used to develop new interaction metaphors and gestures for navigation and manipulation in complex 3D visualization domains [41, 42, 43]. Touch interaction is also being used for visualization tasks in specific scientific domains, including serving as an intelligent selection mechanism for 3D point cloud datasets [44] or as a means to peel and split layers of 3D data in reservoir visualizations [45].

In this dissertation, we apply the principles of NUI and direct manipulation interaction along with multi-touch surface technology to the challenges associated with data-intensive exploratory visualization. For example, we use multi-touch input to directly manipulate the shadows of floating virtual objects as a means of exploring large-scale 3D scientific environments. We also use these principles and technology to directly manipulate the outputs of simulation data as a means of exploring very large simulation ensembles. These tasks would be difficult to complete using traditional user interfaces and are made accessible only through the use of NUI principles.

2.3 Context-Preserving Techniques for Exploratory Visualization

The first limiting measure, as described in the introduction, of existing exploratory visualization systems when working with big data is the size of individual data instances. As the scale and detail of these data sources continue to increase, the limited capabilities of static, single-view visualizations quickly becomes evident. Cockburn et al. [46] summarize the essence of this problem: "users need to interact with more information and with more interface components than can be conveniently displayed at one time on a single screen." The data sources they are referring to may range from large 2D documents to complex 3D scenes. This problem is driven by many factors, including display technology, humans' ability to perceive visuals, and the practical concerns of laying out information on a display screen. It is also a problem that is exacerbated by moving to larger and larger datasets.

Given the importance of viewing and navigating large datasets, a number of interaction and visualization techniques have been developed to facilitate that process by allowing navigation while maintaining context. Researchers have identified three categories into which these techniques can be grouped [46, 47]: zooming, overview+detail, and focus+context.

In the most basic of these categories, zooming style interfaces have been introduced that provide users with the capability to move through the dataset, either through panning within 2D domains or translating and rotating within 3D domains, and to zoom on demand to examine data in more detail. Cockburn et al [46] define zooming

interfaces as those that provide a temporal separation between views, meaning that the user must move through time to arrive at a more detailed view. Zooming and panning techniques have become standard within current computing environments, and everything from web browsers to text editors typically support shortcuts for zooming interactions. These interfaces are derived from early research like the Pad system [48], one of first desktop environments that was fully zoomable, or the work of Furnas and Bederson [49] that introduced a conceptual framework called Space-Scale diagrams that captures basic navigation using zooming and panning. One major limitation of zooming interfaces is that the user's spatial understanding can easily become confused as data moves around different locations of the screen depending on the currently applied scale and translation.

The second category of interfaces are overview+detail interfaces that provide separate (at least two) distinct views of a dataset at different scales. Cockburn et al. [46] define these as interfaces that provide a spatial separation between views, meaning that the views depicted to the user are shown at the same time but in spatially distinct locations on the screen. The simplest examples of this technique include two viewports of the data, such as a detailed view combined with an overview map that indicates the exact location of the detailed viewport. In more complex interfaces, many different coordinated windows can be displayed at once to show relationships across different locations in a dataset. For example, Butkiewicz et al. [14] present a system for exploring geospatial data using probes that create new linked windows to provide detail on demand. Sayle and Milner-White [50] describe a molecular viewer that depicts complementary 3D views of molecules with different levels of abstraction. Using multiple views to enable overview+detail visualization is a powerful concept, but can also lead to information overload if done improperly. Wang et al. [51] present a set of eight guidelines for weighing these tradeoffs when designing multiple view systems.

The final category of interfaces, focus+context, includes interfaces that embed a higher fidelity view (focus) within a lower fidelity view (context). Cockburn et al. [46] define these as interfaces that eliminate both spatial and temporal separation by displaying multiple fidelities within a single continuous view. This is accomplished by distorting the visuals presented to the user, most commonly through a "fisheye" style lens that may be positioned across space. One of the earliest examples of this style of interface

is the Perspective Wall [52], which provided a rectangular focal region surrounded by two perspective regions on either side that receded backwards. The theoretical basis for more generalized fisheye views was introduced by Furnas [53] and has led to numerous examples of fisheye lens exploration techniques [54, 55, 56]. The advantage of the focus+context interfaces is that they explicitly show the global relationship between views in a seamless display, although this benefit comes at the cost of distorting the space.

Many researchers have studied these three categories of context-preserving interfaces and investigated the tradeoffs between them. For example, Baudisch et al. [47] provide a quantitative evaluation comparing a type of focus+context screen that embeds a higher-resolution area inside a larger lower-resolution area against both a zooming interface and an overview+detail interface. They found that the focus+context screen was faster and reduced error rates when compared to the other interfaces, at least in the types of datasets they investigated. Plumlee and Ware [57] present a mathematical theory grounded in visual memory research for how users perform visual comparisons when using zooming interfaces as compared to when using multiple-windowed approaches. They found that the performance of the two interfaces has a cross-over point at which the comparisons become complex enough that they cannot be held in visual working memory.

The challenges outlined in this dissertation require a new consideration of the three categories of context-preserving navigation interfaces. For example, when using immersive VR, the constraints for these interfaces change. Immersive zooming interfaces will have to handle a much larger range of extreme scales in order to fully leverage the advantages of VR technology. Additionally, in VR, it is more difficult to implement focus+context interfaces that distort space than in traditional 2D focus+context environments, as doing so can confuse and disorient the user. Based on these considerations, our work builds primarily upon overview+detail style interfaces and how they can be adapted to work in VR. Overview+detail interfaces have the added benefit that they can easily include more than two views at once, an important feature when exploring large datasets that must show relations between and integrate many different data sources.

2.4 High-Dimensional Parameter Space and Ensemble Visualization

The second limiting feature of existing exploratory visualization systems for working with big data, as noted in the introduction, is the complexity associated with scaling to handle many data instances. There is a large body of related research in the field of ensemble visualization, or the visualization of sets of data composed of many related datasets. Much of this research focuses specifically on the domain of visualizing and interacting with high-dimensional parameter spaces, similar to the design challenges faced by medical device designers described earlier.

The basic task in parameter space visualization is to select a set of optimal parameters that produce a desirable result. The simplest system for performing this task is to let a user adjust each parameter and view the result. Given the difficulty of this process in more complex spaces, a number of computer-aided methodologies have arisen: interactive evolution, inverse design, and design galleries.

In the interactive evolution methodology, genetic algorithms are combined with manual evaluation via a human oracle. A set of sample outputs is derived using the genetic algorithm, which the user evaluates to evolve the next generation of samples. Karl Sims [58] presents one of the earliest examples of this methodology. Using random mutations followed by user observation and selection, complex cellular automata systems evolve over time, creating interesting dynamical systems that would otherwise be difficult to design by hand. In the graphics domain, a number of systems have been developed for applying interactive evolution to the 3D modeling and animation domains [59, 60, 61]. A major advantage of these techniques in data-intensive design is that they feature the human at the center of the visual-interactive feedback loop. However, they are limited in that they require the evolution (e.g., the simulation) to be close to real time for the loop to be fluid, and they require a genetic algorithm that is able to capture the user's implicit criteria.

The second methodology commonly used for high-dimensional parameter space exploration is inverse design. Systems that rely on this methodology are usually close to being fully automatic, typically requiring only a user-defined objective function or explicitly defined specification up front. Using optimization techniques together with the

objective function, the parameters are varied to converge on a solution. Inverse design has been applied to many problem domains in computer graphics and rendering [62]. For instance, early examples focused on how this methodology can be used to find optimal lighting conditions for 3D scenes [63, 64]. In one example, users paint their desired lighting on the scene, and the system finds the lighting parameters that best match it. Research has also used similar inverse techniques for specifying spacetime constraints to help animators design the motions of linked figures [65, 66]. This methodology has the advantage of allowing users to focus solely on their goals at hand by precisely specifying the desired output. However, such techniques require an objective function to optimize and are typically automatic and thus do not allow much user control aside from the initial specification. In design problems, defining an objective function that captures all of the desired goals or inherent tradeoffs is often quite difficult, and a human is needed to input design criteria interactively.

The third and final methodology, design galleries, was created to overcome many of the weaknesses of the first methodologies and represents the methodology most similar to the work developed in this dissertation. The first example was introduced by Marks et al. [67]. In this methodology, the design space is sampled to produce the most diverse set of outputs possible. This set is then presented to users in a gallery from which they can select their optimal output. Many follow-on examples have adopted this methodology and applied it within different domains, including visual effects design [68], trajectory-based character animation [69], disaster management [70, 71], and engineering design [72, 73]. This approach is appealing to the data-intensive paradigm, as it is becoming computationally easier to generate denser sets of samples than ever before. The weakness of traditional design gallery techniques is that they often suffer from information overload associated with displaying many results at once to the user.

The data-intensive paradigm and the computing power to densely sample high-dimensional spaces allow us to reexamine the way we perform parameter exploration tasks. The design galleries methodology is perhaps best suited to work in this new paradigm, but suffers from information overload in larger design spaces. In this dissertation, we work to overcome this limitation and to develop new techniques by bringing elements from the additional methodologies of interactive evolution and inverse design. Namely, we focus on making the human as central as possible in the visual-interactive

feedback loop, much like interactive evolution interfaces. We also focus on including the inverse elements from inverse design techniques that allow users to abstract the design space and focus on their current goals.

Chapter 3

Input and Output Hardware for Scalable Natural User Interfaces

We begin by introducing a new hardware platform that will be built upon throughout the dissertation to develop new scalable natural user interfaces. This hardware platform makes use of virtual reality (VR) technologies. VR offers many advantages to scientists trying to understand their data, and simply viewing data in VR often results in new insights previously missed regarding a dataset. Given their stated goal of simulating physical presence, immersive VR technologies provide one of the most natural existing user interface technologies. By embedding oneself within a dataset using VR technologies, a designer is able to view data at extreme scales while maintaining a physical relation to the data [74], making these technologies ideal for building scalable natural user interfaces.

Despite the advantages of immersive VR technologies, their traditional platforms also have limitations that constrain their use. CAVE or head-mounted environments suffer from issues that make their extended use difficult, as users often become physically fatigued after short periods of using six-degree-of-freedom (DOF) input devices (i.e., wands). Additionally, many elements of the user's more traditional work practices and tools must be left behind when using VR technologies. For example, the sophisticated UIs available in a desktop PC environment are difficult to fully recreate in 3D, and simple things like lab notebooks are no longer usable in VR.

This chapter introduces a novel VR hardware setup utilizing multi-touch input and explores the potential of using these technologies together to support data-intensive exploratory visualization. In particular, we introduce a new multi-surface VR display configuration that combines a large, table-sized multi-touch surface with a VR stereoscopic display wall, a configuration we call the Immersive Touch Workbench (ITW). We believe that the resulting hybrid environment can provide rich and valuable new modes of interacting with many VR applications by utilizing the strengths of both technologies while overcoming the limitations of using each independently for scientific visualization.

Recently, the widespread availability of affordable 3D input and 3D display devices, such as the Nintendo Wii motion controller, NaturalPoint Optitrack optical tracking, and 3D DLP TVs, has opened up new applications for VR, making it increasingly practical for VR to be used in labs, offices, and classrooms without the significant expense previously typical of VR hardware. Paralleling these advances has been an exciting hardware revolution in the area of multi-touch display devices. From the large-scale Microsoft Surface to the hand-held Apple iPad, many multi-touch devices are now commercially available, and a growing community of researchers and hobbyists has established viable approaches (e.g., [75]) for building multi-touch displays using low-cost commodity components.

Compared to traditional VR input devices used with 3D applications (e.g., 6-DOF trackers, VR wands), multi-touch surfaces may at first appear limited in that they fundamentally capture just 2D input on the surface. However, there are several reasons why multi-touch surfaces may be particularly well suited to VR applications. First, despite the fact that the touch events generated are 2D, multi-touch surfaces actually support very high-dimensional input because they support simultaneous tracking of tens to hundreds of touch points. Second, recent research has demonstrated that some 3D tasks (e.g., precise relative positioning) are better performed using 2D views than 3D ones [76, 77]; it follows that 2D multi-touch surfaces may be better suited for certain 3D interactions than 6-DOF devices. Finally, many of the most successful current multi-touch applications are built upon fluid, gestural, and direct-manipulation styles of input, which are also common characteristics of many VR applications. Thus, we believe multi-touch input can be just as expressive (or even more expressive) as traditional VR inputs, better suited to certain 3D tasks, and seamlessly integrated with many common

VR interfaces and interaction metaphors.

In the next section, we will contrast the ITW hardware configuration with existing technological and research advances. Following this, we will explain our new configuration in more detail, including two versions, a low-cost do-it-yourself version and a commodity version built with commercial hardware. Then we will present the fundamental 3DUI metaphor and a set of touch gestures upon which ITW applications can be built. Finally, we end with a discussion of the design tradeoffs in these workbenches and our conclusions.

3.1 Related Work

Several previous systems have explored interaction techniques that utilize multiple display surfaces. (Lachenal and Dupuy-Chessa present an ontology of this design space [78].) Closely related to our work, Ajaj et al. [79] combined a horizontal table and vertical screen and used a map on the table to control the camera view of a 3D scene displayed on the vertical screen. Aliakseyeu et al. [80] used a similar smaller-scale setup in which a tracked frame prop is moved above a tablet to control the view shown on a stereo screen. Spindler et al. [81] also explored above-the-table interaction using a tracked paper prop as an additional display surface. Hachet and Guitton [82] used a tracked tablet mounted on a pillar to enable interaction in large-display VR environments. LaViola et al. [83] used a pair of slippers to interact with the floor surface in a CAVE environment, freeing the user's hands for other tasks performed above the floor in the 3D space of the CAVE.

Although these systems all utilize a multi-surface configuration and mix 2D inputs and displays with 3D displays, we know of no existing multi-surface configuration that combines multi-touch table input with a head-tracked stereoscopic wall display. Our work has focused on this configuration because we believe it can be particularly effective for VR. For example, the configuration we propose can be used to make virtual objects appear to float in the air above the multi-touch table, which leads naturally to interacting with a shadow or projection of the 3D scene on the table, a metaphor we explore in the interface described later in the paper.

Within the multi-touch research community, Han's seminal work on frustrated total

Figure 3.1: Using the multi-surface VR setup to explore a biomedical environment.

internal reflection (FTIR) [75] has spurred a variety of follow-on research and a growing community commited to designing and supporting low-cost multi-touch hardware. An important result emanating from this community has been the tangible user interface objects (TUIO) protocol for describing and transmitting touch events [84]. Servers and clients that implement this protocol are available for a variety of platforms [85], and we have found that this infrastructure makes it easy to integrate touch input within a typical VR application structure. For example, the processing of TUIO events distributed to an application via a network connection can be integrated within a VR application alongside typical VR event-handling routines (e.g., handling tracking data distributed via VRPN).

3.2 Multi-Touch, Multi-Surface VR

The ITW combines multi-touch input with virtual reality by utilizing two separate display surfaces. These two display surfaces are placed at right angles to each other in an L-shaped configuration, one horizontal and one vertical, as shown in Figure 3.1. The horizontal surface is a multi-touch sensitive surface that serves as the main input to the system. The vertical surface is a stereoscopic and head-tracked surface that renders 3D graphics. Using this configuration, 3D environments can be rendered stereoscopically behind the vertical surface, presenting a view to the user that is similar to looking through a window. Additionally, 3D objects can be rendered in front of the surface such that they appear to be floating above the touch surface.

The horizontal touch surface brings several advantages to this hardware configuration. The first is as an input device, as the support of rich multi-touch gestures enables many novel ways to interact with the immersive 3D environment shown on the vertical surface. This includes the ability to interact with objects that float above the table surface, as described later in section 3.3. Second, the touch surface serves as an additional 2D display surface. Traditional VR systems are often limited in that traditional GUI or other 2D information is difficult to display and to interact with in the virtual world. By using the touch surface to display additional graphical elements, these components can be easily integrated into and interacted with in 3D virtual environments. For example, this combination of surfaces is ideal for mixing 3D scientific visualizations with linked 2D information visualizations.

We have designed and built two different versions of the ITW. These two versions differ in the implementation of the horizontal touch surface. The first (and original) version was built using low-cost commodity hardware with a do-it-yourself design of the multi-touch input device. Since the original development of this ITW, commercial hardware has advanced to the point where multi-touch tracking at a scale this large is both robust and affordable enough that turnkey systems are in many ways preferable to the do-it-yourself approach. In the following sections, we begin by first presenting the details of the vertical surface used in both versions, then the two designs for the multi-touch surface.

3.2.1 Vertical Surface

The vertical surface displays a three-dimensional stereoscopic view of the VR application. A mid-range stereo 720p projector mounted behind the screen generates the image, as seen in Figure 3.2. Shutter glasses are worn by the user, and the IR synchronization emitter is mounted behind the vertical screen. A 9' x 4.5' Rosco Grey projection material is used for the display screen and is mounted in a simple wooden frame by suspending it with an elastic cord wrapped around the frame and through grommets placed around the edge of the screen. The user's head is tracked by 6 Optitrack NaturalPoint cameras mounted on both the wooden frame and the surrounding ceiling.

3.2.2 Low-Cost Touch Surface

Our low-cost multi-touch surface uses a bottom-projected image and FTIR technology to implement the touch-sensitive display, a schematic of which is shown in Figure 3.2. FTIR-based multi-touch systems rely on IR light shining onto the edges of a sheet of transparent material. The light is reflected internally, but the reflection can be disrupted if another object makes sufficiently close contact - an effect enhanced by using a thin, flexible material known as a compliant surface [75]. A camera with appropriate filters can then detect the touch as a blob of infrared light. A projector displays an image onto the projection surface, which often serves as the base of the compliant surface. More detail on the low-cost multi-touch surface components is available in the original work on this topic [86].

3.2.3 Commercial Touch Surface

Since the development of the original low-cost ITW, commercial multi-touch technology has rapidly progressed to the point where very large and robust optical touch tracking solutions are available. In addition, as these technologies have advanced, the costs have dramatically decreased. We have developed a second version of the ITW that makes uses of these technologies, specifically the G4 Touch overlay from PQLabs. This second iteration of the ITW is shown in Figure 3.3. The PQLabs touch overlay is designed to be placed atop a television. Our design uses a Samsung 55-inch LCD television with a

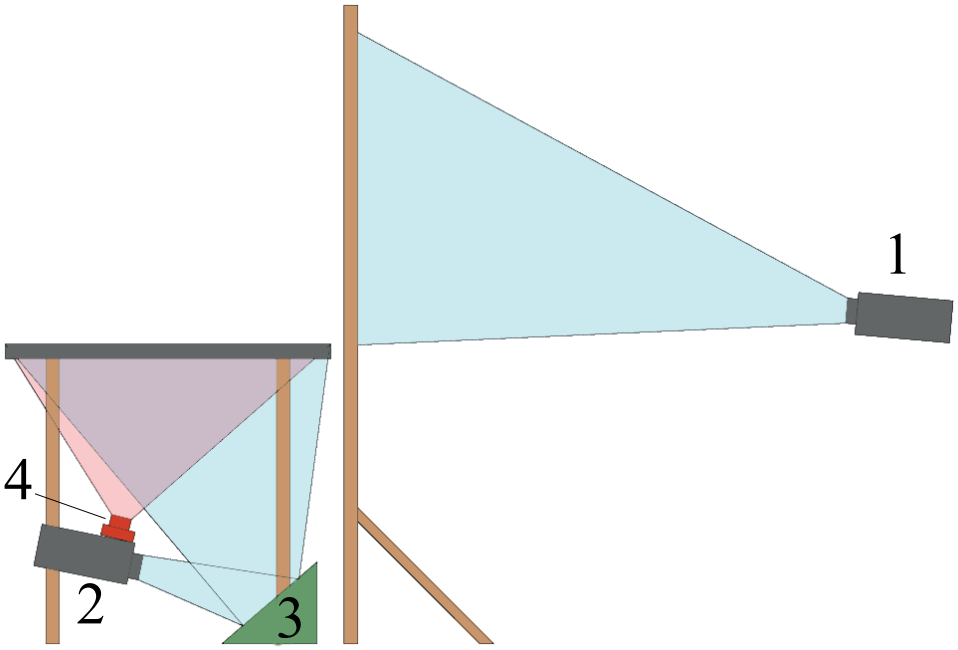

Figure 3.2: Right: the rear-projected image on the vertical screen originates from a 3D projector (1) mounted behind the screen. The horizontal table's image originates from a projector (2) mounted under the table and bounces off the front surface mirror (3). An IR camera (4) is mounted on top of the table's projector and is used to track points of contact on the table.

thin bezel. The additional benefit of using a television rather than the projector as in the original design is the full 1080p resolution and the increased brightness of the LCD TV.

3.3 Shadow Grab: 3DUI for Immersive Touch Workbenches

This section describes Shadow Grab, a 3DUI technique that is made possible by combining multi-touch with VR using the multi-surface ITW configuration. This basic interaction technique forms the basis for the new visualization techniques described in later chapters of this dissertation. In Shadow Grab, the user interacts with a 3D scene that appears to float above the table by manipulating its shadow on the table surface. This concept builds upon early work by Herndon et al. [87], who, using a mouse-based

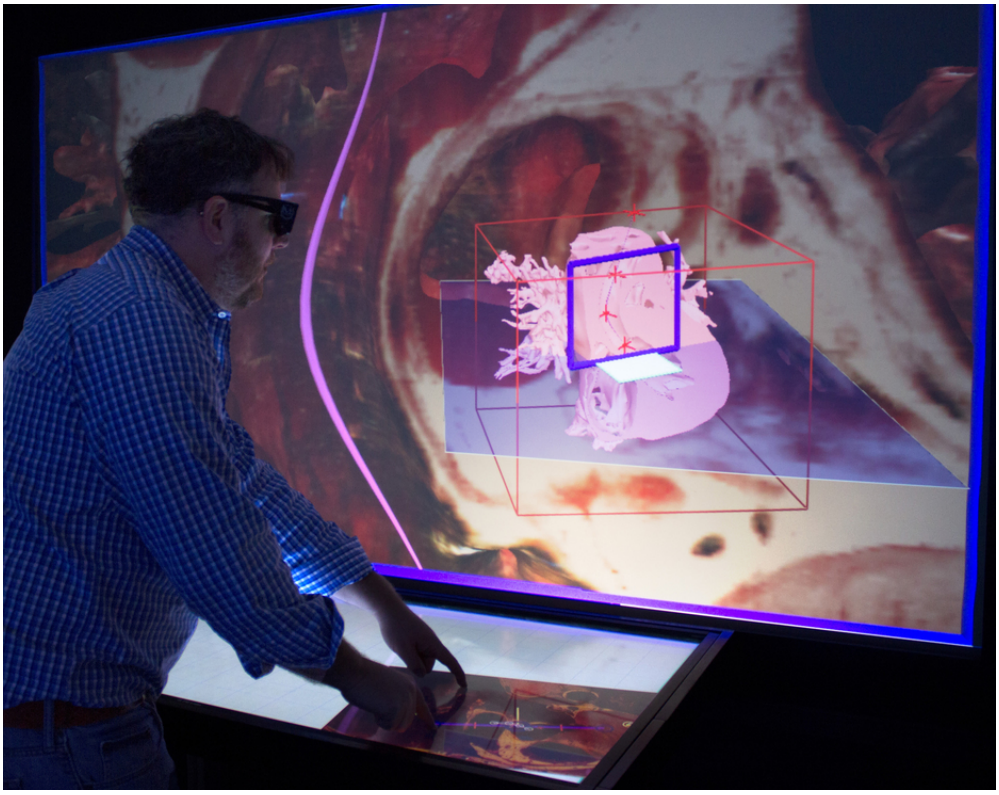

Figure 3.3: The second version of the ITW using a thin-bezeled LCD television combined with a PQLabs touch-sensitive overlay.

desktop system, showed that 3D scene interaction and manipulation can be facilitated by interacting with a set of 2D shadow widgets. This style of shadow interaction is ideal for the proposed multi-surface hardware setup because the rich 2D multi-touch input provides a direct way to manipulate the shadow.

Shadow Grab has two visual components. The first component is the floating 3D object displayed in stereo on the vertical screen, which in our example is an anatomical model of an aorta, as shown in Figure 3.4. The second component is the shadow cast by the 3D object downward onto the table surface, also shown in Figure 3.4. The shadow projection provides a 2D view of the outline of the object that the user can manipulate using multi-touch gestures. Using the gestures illustrated in Figure 3.5 (a), the object can be scaled, translated, and rotated. A single point of contact allows translation in the horizontal and depth dimensions; the height of the 3D object above

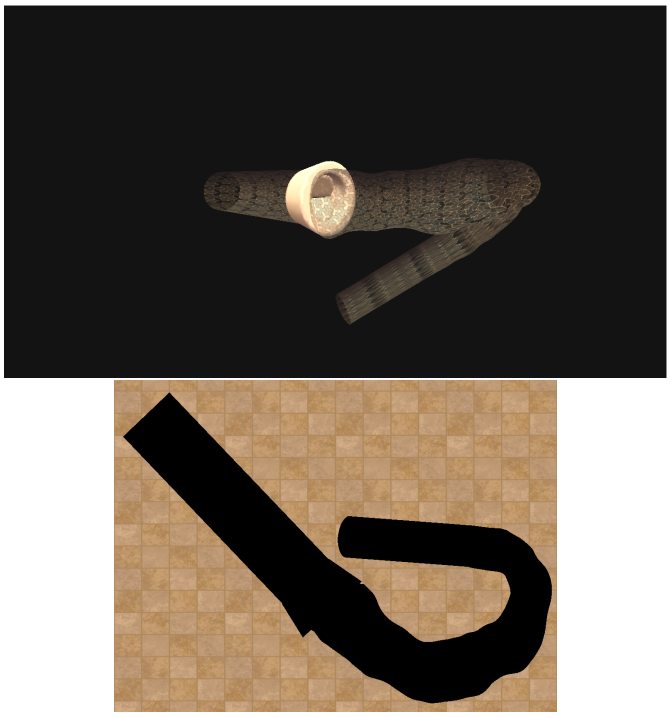

Figure 3.4: The Shadow Grab interface is used to manipulate 3D objects floating above the tabletop by interacting with their shadows projected on the table. Using multi-touch gestures, the objects can be translated, scaled, and rotated.

the table remains fixed. Two points of contact are used to scale the object and to rotate it around a vertical axis. Each of these interactions is similar in spirit to the 2D direct manipulation multi-touch interfaces that have become popular in photo browsers and similar applications. Figure 3.5 (b) illustrates an extension of these traditional gestures that we have implemented to support 3D interaction. Moving two points of contact together in the horizontal direction rolls the scene around an axis pointing to the vertical wall (the depth dimension), and moving two points of contact together in the depth dimension rotates the scene around a horizontal axis, tilting it toward or away from the viewer.

Initially, we implemented the shadow as a parallel projection of the scene downward along vertical projector rays. In practice, we found that when users interacted with the shadow positioned near the front of the table, this brought the 3D "floating" version of the scene very close to the user's head, which was uncomfortable for many users. Our

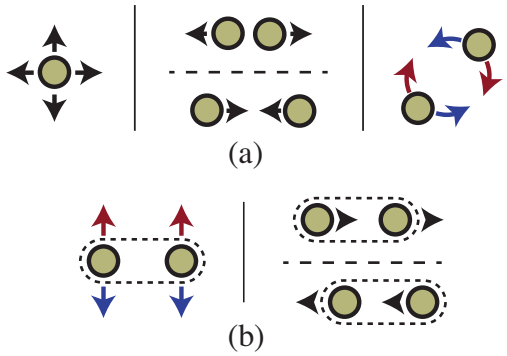

Figure 3.5: The set of multi-touch manipulation gestures used in the Shadow Grab interface. Gestures that translate, scale, and rotate the object in the plane of the table are shown in (a). Gestures that tilt and roll the environment are shown in (b).

solution to this issue is to apply a slight offset to the shadow, moving it six inches closer to the user than it should be. This dramatically improves the usability of the interface, and surprisingly, none of the users we have worked with have noticed this offset.

3.4 Discussion

3.4.1 Manipulating Objects in Front of Low-Cost Displays

As we built this VR system, we discovered that the current generation of low-cost VR displays is quite limited in its ability to produce the illusion of a scene floating in front of the vertical display. Our original design utilized a Samsung 60 DLP stereo television as the vertical stereo surface, but we found that the significant ghosting that occurred with this display when objects were positioned more than a few inches out of the screenplane made the stereoscopic imagery very difficult to fuse, and hence, the technique was almost unusable with the Samsung display. For this reason, our current implementation uses a mid-range stereo projector, which provides a higher quality image, albeit at an increase in cost. We anticipate that as 3D technology progresses, the quality of stereoscopic images produced by low-cost displays will improve considerably, and we submit that achieving the high-quality rendering of objects that appear significantly in front of a vertical VR display will be a particularly important benchmark for low-cost display manufacturers and researchers to achieve, as it can enable a new class of multi-surface

interfaces in the style of Shadow Grab.

3.4.2 Six-Degree-of-Freedom Tracking

In the current hardware setup, six OptiTrack NaturalPoint cameras are used for 6-DOF head tracking. Depending upon the application and the budget, 6-DOF tracking could play either an enhanced or a decreased role in the VR environment. With techniques such as Shadow Grab, 2D multi-touch interfaces could provide an affordable replacement for more traditional VR 6-DOF direct manipulation interfaces, and thus the optical tracking system used in our implementation could be smaller (so as to capture only limited head movement) or eliminated (if head tracking is not essential to the application). This would greatly reduce the overall cost of the VR environment.

On the other hand, it is also possible to make increased use of the 6-DOF tracking than in our current setup, which provides an exciting direction for future research. For example, the multi-touch gestures could be combined with 6-DOF hand-tracking above the surface.

3.4.3 The Choice of Touch-Tracking Method

FTIR is a fairly mature technology but by no means the only way to capture multi-touch input. Other methods, such as diffuse illumination (DI) and diffuse surface illumination (DSI), may also be considered. An important consideration in selecting the most appropriate technology is the size of the surface. Our design called for a rather large surface, and in this situation, FTIR is more cost-effective than DSI (which requires a more expensive type of acrylic material). For large surfaces, FTIR tends to provide more uniform performance than DI (which may suffer from uneven IR light distribution).

3.4.4 Infrared Interference Reduction

The original low-cost setup relies heavily on 850 nm IR lights. Touch-tracking, head-tracking, and the synchronization of the shutter glasses all use IR light close to this wavelength, and cross-interference can cause spurious inputs to the interactive surface or disrupt the synchronization of the shutter glasses. This can be counteracted in 3 ways. First, the brightness of the Optitrack IR LEDs can be decreased and the shutter

glasses synchronization strobe carefully positioned. Second, an OptiHub device recently developed by NaturalPoint can be used to synchronize the cameras' IR LEDs with the synchronization strobe for the shutter glasses. Third, non-IR syncing glasses can be used, such as the NVidia 3D Vision Pro RF glasses.

3.4.5 Seamless Projection

The metal frame of the multi-touch table used in our current implementation creates a small seam where the table and the vertical wall touch. An alternative design might be able to avoid this seam through engineering a seamless joint, perhaps inspired by the joints used in CAVEs or the Responsive Workbench.

3.4.6 Extending to Support Tangible Interfaces

Tangible interaction props marked with fiducials are more commonly associated with DI systems (e.g., Reactable [84]) as opposed to FTIR systems. Nevertheless, it is possible to augment the interaction with FTIR systems by using IR-emissive tangible objects as well as narrow-beam IR sources. The table would provide a natural resting place for these interaction props. Thus, an interesting area of future work is exploring the potential of prop-based user interfaces tailored to this multi-surface VR configuration.

3.5 Conclusion

We have presented a low-cost system that combines a 2D tabletop multi-touch interface with virtual reality that we call an Immersive Touch Workbench. In addition to the implementation details and design lessons learned in creating this system, we provide an sample 3D user interface that demonstrates the potential of the unique hardware configuration. Since multi-touch surfaces provide many simultaneous touch events, they are particularly rich (high-dimensional) input devices that may enable many new styles of interacting with VR applications. If appropriate mappings can be devised to interpret multi-touch input within the context of VR applications, the rich input afforded by multi-touch surfaces may prove beneficial in a variety of VR applications. Combining multi-touch and VR technologies is becoming increasingly practical given the growing

online community providing both hardware and software support. Due to the combination of the relatively low-cost rich interaction and existing open standards, the VR community is poised to take advantage of multi-touch surface computing. For these reasons, we believe the ITW may be the workbench of the future for designers working in the context of big data and is ideal for desiging new scalable natural user interfaces.

Chapter 4

Theoretical Foundations: Empirically Driven Design Guidelines for Scalable Natural User Interfaces

This chapter presents our work toward developing a theoretical understanding of the foundations of the data-intensive exploratory visualization process. The end goal of this chapter is a set of emprically driven design guidelines for developing scalable natural user interfaces. Specifically, to arrive at these guidelines, we build and then evaluate a theoretical framework using the ITW hardware presented in chapter 3. We explore the fundamental design choices available when creating exploratory visualizations using this platform through evaluative user studies. This foundation is explored using biomechanical motion data as the representative data in the visualizations.

Studies of motion are fundamental to science and engineering, yet designing effective motion data visualizations for today's study of biomechanics, fluids, and other space-time phenomena remains a challenging task. Our work is motivated specifically by data analysis challenges in the field of biomechanics, in which our collaborators collect rich high-resolution data but lack appropriate visualization tools. Data utilized in current

studies of biomechanics in both humans and animals can now capture the internal motions of bones at rates of 100 Hz or more. By leveraging advances in medical imaging and computer vision, these raw motion data can be correlated with 3D imaging so that 3D spatial reconstructions of the bones can be made at each step in time with sub-millimeter accuracy [88]. The resulting motion datasets have great potential to lead to new discoveries. In practice, however, the complexity of analyzing such detailed motions means that only the most easily extracted trends are currently used in biomechanics analyses, leaving many opportunities for developing visualization and other computational tools to help scientists better analyze these data.

In recent years, the desire to gain more understanding from these and other scientific motion datasets has motivated several advances within the visualization community (e.g., visualizations of wrist bone motions [89], animal chewing motions [90, 91], and bat wing motions [92]). This work has led to useful visual analysis tools, including new virtual reality (VR) visualization techniques, but it has not approached the problem of visualizing high-resolution scientific motion data in a systematic manner.

In this chapter, we attempt to lay a foundation for a systematic study of scientific visualizations of motion, with an emphasis on visualizing motion data in interactive VR environments. This foundation is intended to be used as a basis for desiging new visualization techniques capable of handing big data. Our approach is threefold. First, we present a scientific motion visualization design space taxonomy. This taxonomy is characterized as a matrix with two dimensions: (1) design choice for time and (2) design choice for space. We argue that three of the most fundamental design decisions that must be made along both of these dimensions is whether to depict the data via animation, interaction, or static presentation. This leads to a classification of motion visualizations that includes specific visualization designs, such as interactive time/static space, animated time/interactive space, and so on. Second, we present sample implementations for 8 of the 9 possible visualization designs identified in the taxonomy[1]. We also report the results of a quantitative user study that evaluates the success of these 8 visualization designs for a motion analysis task. Third, we report on a qualitative

[1]During development, we found that only 1 of the possible 9 designs, the static time/static space design, was too complex to be practical, leaving a total of 8 potentially valuable designs.

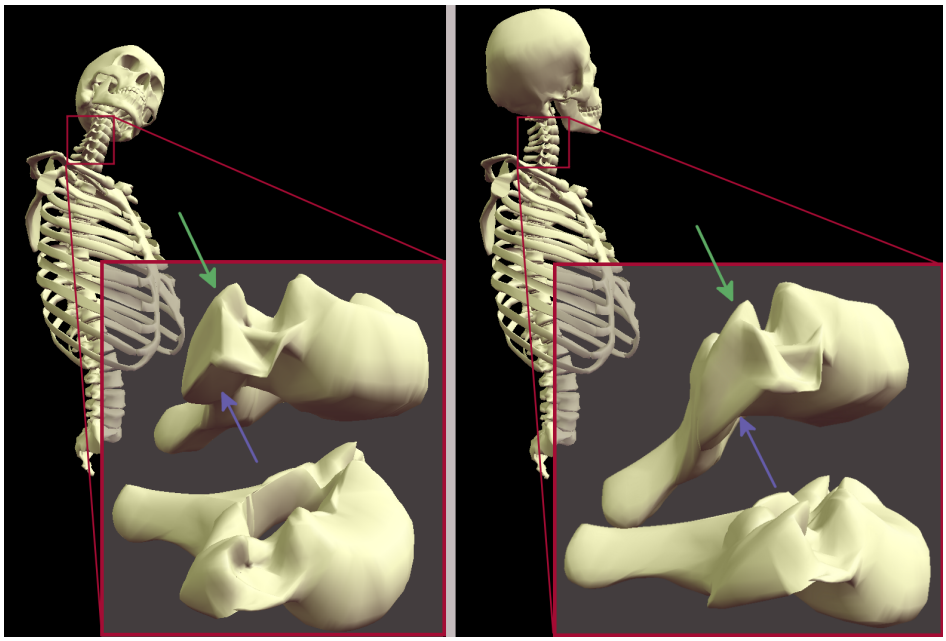

Figure 4.1:]
Studying the biomechanics of the human spine requires analyzing complex motions of the vertebrae.

assessment conducted with the collaborating biomechanics researchers. In this application case study, the visualization designs were evaluated with both a generic synthetic dataset and a neck circumduction dataset from an active musculoskeletal biomechanics research project (Figure 4.1).

Our work draws inspiration from low-level studies of vision and motion perception, studies of user interfaces and head-tracked stereoscopic environments, and visualization applications developed for real-world problems. We describe related work in these areas in the next section. We then present details of the motion visualization design space, the quantitative study, and the qualitative case study.

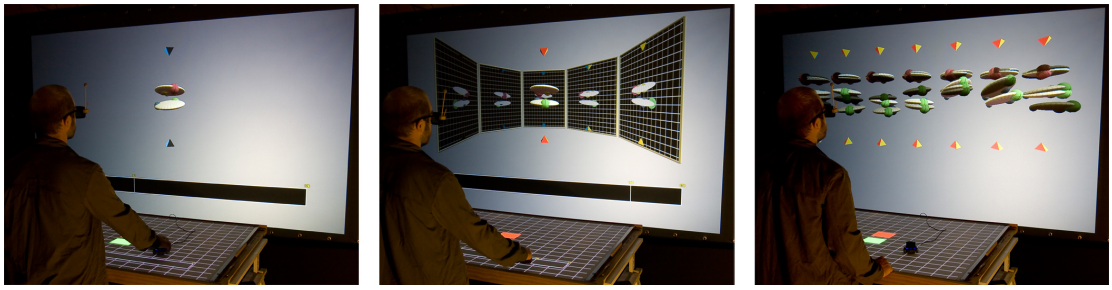

Figure 4.2: Three of the eight visualization designs studied. Left: Interactive Space/Animated Time - A 3D input device controls scene rotation; the time dimension is controlled automatically via animation. Center: Static Space/Interactive Time - The scene does not move but 3D projection planes are included to facilitate spatial judgments; time is controlled interactively by touching a widget on the table. Right: Animated Space/Static Time - Space is animated by automatically rotating the objects back and forth; time is presented statically through a timeline of key 3D poses of the motion.

4.1 Related Work

4.1.1 Perceptual Issues in Visualizing Motion

Our work closely connects with recent work in applied perception that has examined the role of animation in visualization and methods for facilitating accurate judgments of spatial relationships. It is now well known that the style of rendering used to depict 3D geometries has a direct impact on spatial perception (e.g.,[93]). Our work draws upon these findings to present the 3D forms that we visualize with appropriate lighting, texture, and other visual cues. It is also known that motion of a 3D scene, which is achieved by automatically animating the camera along a 3D path to produce motion parallax or by interactive camera control, improves spatial judgments[94]. Ware and colleagues also found that stereoscopic displays improved spatial perception. Interestingly, animated or interactive camera control is not always preferable to a stationary camera; interaction can slow down analysis, and cleverly designed static visualizations using the ExoVis technique, for example, can be just as accurate as animated and interactive visualizations[95]. Our work advances this line of research by determining the extent to which the results obtained for 3D stationary scenes also apply to 4D time-varying data that are relevant to scientific analyses and presented in VR.

From a background in cognitive and perceptual science, Tversky et al.'s seminal article, "Animation: Can it Facilitate?" makes a strong argument against the use of animation[96]. More recently within the visualization community, Robertson et al. responded to public excitement over Hans Rosling's compelling animated trend visualizations[97] by providing empirical evidence that, although animation is a compelling way to present already-identified trends, static displays are more effective for analyzing and identifying these trends[98]. On the other hand, animated visualizations may also have a number of benefits, as, for example, in detecting relative motion, determining the 3D structure of moving objects, and integrating coherently moving parts into a single object[99]. Despite significant attention paid by researchers to animation, controlled studies of motion visualization have predominantly focused on low-level perceptual effects and more abstract uses of motion (e.g.,[100]), not the design of VR visualizations to elucidate space-time relationships in experimentally collected or simulated motion data.

4.1.2 Displaying Space-Time Data in Virtual Reality

One of the most well-established benefits of VR environments is that they make use of natural human modes of interaction (e.g., using head-tracking, VR displays are updated naturally as the user physically moves through an environment). Thus, in designing VR visualizations of data that inherently describe motion, a logical choice is to display the data by re-animating the motion, often within a virtual reconstruction of the lab in which the data were collected. We argue that although this is a natural default design decision, it is likely, given the recent perceptual research highlighted above, that it may not be the most effective means of presenting motion data in virtual environments. By designing a user-based experiment to explore and compare design options in this space, our work continues an important line of research on perceptual issues in VR, including accurate spatial judgments[93, 101], change blindness[102], depth resolution[103], and more.

4.1.3 Visualizing Motion and Biomechanics

Finally, researchers in the scientific visualization community have adopted several techniques for conveying changes in spatial patterns over time that, like ours, involve the strategy of "freezing" time [104, 105, 106]. Our static visualization techniques bring similar concepts to virtual environments. We also draw upon a growing body of work in the visualization of biomechanics [90, 91, 89]. The fundamental visualization strategies that we explore in this chapter are intended to readily incorporate the visualization widgets (tracers and the like) introduced in these related systems to further enhance our visualizations.

4.2 Taxonomy of Motion Visualization Design Space

Several fundamental design choices must be made when developing motion (space-time) data visualizations, which we organize into the taxonomy diagramed in Figure 4.3.

These include three primary design choices regarding how to represent the time dimension of the data: (1) The user can have interactive control over time, such as via a knob or slider by which to advance the view forward or backward in time. (2) Time can be animated, with motion replayed in a loop. (3) Time can be shown statically, with multiple snapshots displayed in a single scene.

There are three analogous options for the space dimension: (1) The user can interactively control the scene, rotating and repositioning space to make spatial judgments from multiple viewing angles. (2) The scene can be rotated automatically, such as rocking back and forth to create motion parallax. (3) The scene can be placed in a stationary position and orientation, with clever additional visualization widgets used to enhance depth cues (e.g., the ExoVis technique[107]). Note that with head-tracked stereoscopic displays, some form of "camera" movement is always supported via tracking the user's head position; thus, we use "scene" here to refer to the positioning and orientation of the computer graphics scene relative to the physical space of the display and user.

Each box in Figure 4.3 represents a specific design for visualizing motion data. We conceive of these nine designs as fundamental in the sense that within each category, many variations are certainly possible, but leaving room for some variation, there is

		TIME		
		Interactive	Animated	Static
SPACE	Interactive	I_S,I_T •interactive camera •interactive time-line	I_S,A_T •interactive camera •animated timeline	I_S,S_T •interactive camera •multiple times in one scene
	Animated	A_S,I_T •auto camera motion •interactrive time-line	A_S,A_T •auto camera motion •animated timeline	A_S,S_T •auto camera motion •multiple times in one scene
	Static	S_S,I_T •multiple camera views in one scene •interactive time-line	S_S,A_T •multiple camera views in one scene •animated timeline	S_S,S_T •multiple camera views in one scene •multiple times in one scene

Figure 4.3: Taxonomy of motion visualization designs.

much to be learned from analyzing the relative strengths and weaknesses of these potential designs. In preliminary testing, we found that, given our motivating applications, each box in the taxonomy represents a potentially useful design for motion visualization except for Static Space/Static Time. We were unable to construct a visualization in this style that did not suffer from excessive clutter.

4.3 Examples of Motion Visualizations

The following sections describe our implementation of eight sample visualization designs from the taxonomy above. We introduce first the VR hardware used and then the synthetic motion data that we generated for the quantitative study described in Section 4.4, since these appear in several of the figures we use to illustrate the visualization designs.

4.3.1 Virtual Reality System Hardware

Our VR visualizations are implemented using a version of the Immersive Touch Workbench descirbed in chapter 3 and pictured in Figure 4.2. This environment fits within a popular class of VR form factors based on rear-projected imagery (e.g., see also the

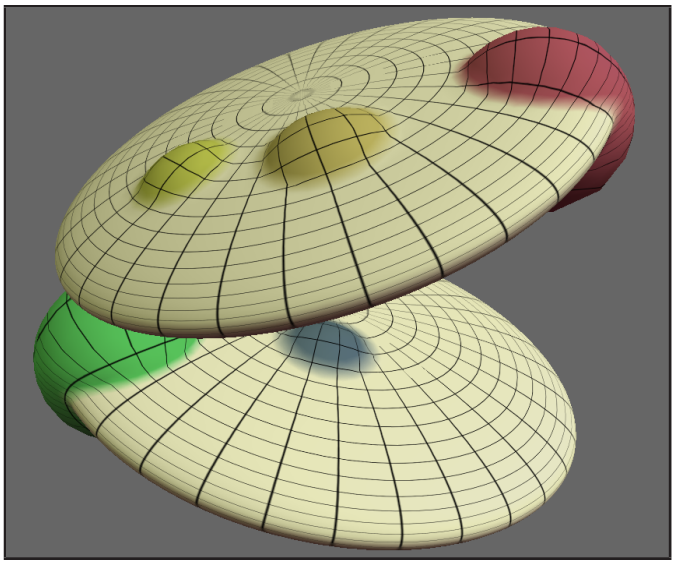

Figure 4.4: The 3D "bumpy disc" forms with highlighted surface features used in the motion visualization experiment.

CAVE and Responsive Workbench). The vertical 9' x 4.5' screen provides a stereoscopic head-tracked view via a DepthQ DQ-WXGA (1280x720 resolution) projector positioned behind the screen. The user wears NuVision shutter glasses, and five Optitrak Natural-Point cameras mounted on the vertical screen frame and the surrounding ceiling provide head tracking data to update the perspective projection of the scene in real time. The second screen is a 4' x 3' horizontal table that supports multi-touch input via FTIR sensing. The table display is driven by a ViewSonic PJD5351 (1024x768 resolution) projector. The two surfaces are perpendicular to each other, sharing an edge at the back of the table. Input is captured via two interaction devices. The first is the multi-touch table, the second a six-degree-of-freedom SpaceNavigator device (3DConnexion) placed comfortably on the table surface.

4.3.2 Synthetic Motion Dataset

Similar to the shape of the vertebrae found in the human spine, the 3D "bumpy disc" forms (Figure 4.4) in the synthetic motion dataset are designed to be abstractions of organic boney shapes. These types of forms are found in many current scientific motion visualizations. In addition to the applications to neck kinematics discussed in this

chapter, another active application area for our research has been understanding the unusual chewing motions of pigs, a project we have undertaken in collaboration with evolutionary biologists [90, 91].. We drew upon both these applications and others (e.g., analysis of gliding joints in the wrist [89]) in designing the abstract forms and the motion to drive them. All of these real-world applications require analysis of motion in joints: areas where two rigid bodies move in close proximity to each other. Often, these rigid bodies include important features, such as valleys, ridges, and extreme points, that are important to track (e.g., the superior and inferior articular facets highlighted in Figure 4.1, teeth that occlude with each other in chewing motions). Motions of these forms include both translation and rotation. Often, understanding changes over time in the distance between features on the rigid bodies is key to the analysis. For example, if uneven motion of the facets of adjacent vertebrae in the spine is observed at a certain time in a motion sequence, this could correlate with a point at which the patient experiences pain.

The abstract discs (Figure 4.4) are generated by flattening a spherical form into a disc and adding features (bumps) of varying radii to it. The dataset contains six different disc geometries created in this style, with the number of features on each disc varying from 2 to 4. On each disc, one feature is set to be larger than the others (i.e., the red and green highlighted bumps in Figures 4.4-4.5), which serves as the feature of interest for the analysis tasks conducted in the experiment.

Two classes of motions are used to control the bumpy discs. In the first, the two discs, oriented one on top of the other, spin and tilt but have no translational component (i.e., their center points are fixed at a constant distance away from each other). Eventually they tilt in such a way that one disc's feature point collides with the other disc (i.e., the two 3D geometries interpenetrate slightly). The second class of motions is defined by movement with a dominant vertical component, the result of two objects coming apart and together repeatedly, as in chewing, walking, or other cyclic motions. In these motions, the two rigid bodies move toward and away from each other with relatively large vertical translations as they tilt and rotate about their midpoints. Again, eventually a feature point on one disc collides with the other disc. The motions are generated algorithmically via a combination of piecewise functions sampled at 180 time steps (6 seconds of motion data when displayed at 30 frames per second).

4.3.3 Visualization Designs

In each visualization design, the motion data are displayed entirely on the vertical head-tracked stereoscopic display; the table is used only for simple 2D interface components. Users stand in front of the display, as shown in Figure 4.2, and, when desired, can take a step or lean to one side or the other, using head-tracking to view a slightly different perspective of the scene. The bumpy disc forms are displayed at the same scale in all of the designs. However, with the head-tracked view, users can easily view the geometries in any of the designs in more detail by moving closer to them. When this is done, the size of the projection of the geometry on the VR display wall increases to cover more pixels. Beyond this core functionality, the mapping from data to visuals and the interfaces used vary for each specific visualization design.

The distinguishing features of each of the designs are described below. Note that each of the designs identified in Figure 4.3 requires a choice (interactive, animated, or static) for both the time and space dimensions; thus, for each design, we select one choice from section 4.3.4 and combine it with one from section 4.3.5. Each of the the designs described below was generated over a period of months through an iterative design process conducted by our interdisciplinary team of computer scientists, biomechanics researchers, and a traditionally trained graphic designer. Throughout this process, our goal was to develop designs that capture the most defining characteristics of each category while recognizing that variation within each of the categories is certainly possible. For example, additional visualization widgets could be easily added to our Animated Time design; however, the features that are fundamental to Animated Time designs (e.g., automatic animation, a timeline widget that indicates the current state of the animation) would stay the same.

4.3.4 Design Choices for Time

Interactive Time. In Interactive Time designs, a timeline widget is drawn on the front edge of table. Users scrub forward and backward through time or jump directly to a new time simply by touching the timeline with a finger. A bar drawn on the line indicates the currently displayed time. As the user changes that time, both the timeline and the 3D visualization displayed on the vertical screen update accordingly.

Figure 4.5: The Static Time designs use sets of 3D key poses to convey change over time.

The widget is wide, and touches made anywhere in its vicinity are considered valid input, so by design, there is little overhead with regard to the time required to touch the widget. A non-interactive timeline is also displayed at the bottom of the vertical screen to facilitate operation while maintaining a visual focus on the vertical screen.

Animated Time. In Animated Time designs, the timeline described above is again displayed on both screens as a visual aid. However, the time is advanced automatically at 30 frames per second. When the end of the motion sequence is reached, the display returns to the beginning (time zero) and continues animating in a loop.

Static Time. In Static Time designs, movement over time is conveyed through a series of key frame 3D poses (Figure 4.5)). Rather than a simple set of key poses arranged either on top of each other or left to right across the display, we use a grouping strategy in this design to strike a happy medium between showing enough key poses to adequately describe complex motions and showing so many key poses in a single view that occlusion becomes a problem. The key poses are subdivided into groups that are arranged spatially across the visualization display in increasing order of time from left to right. Within each group, the key poses are rendered using a color mapping that increases in lightness from a dark gray to a warm white color in proportion to the advance in time. For example, in Figure 4.5), seven groups of keyframes are displayed, each containing 3 key poses. The darkest discs in the leftmost group make up the first key pose, the darkest discs in the second group make up the fourth key pose, and so on.

4.3.5 Design Choices for Space

Interactive Space. In Interactive Space designs, users rotate the scene using the SpaceNavigator device. To distinguish between rotation applied interactively by the user and any rotation of the discs themselves, which occurs regularly during their motion, colored prisms placed above and below the scene are used to establish a grounding context. To limit the complexity of the interface for evaluation purposes and since many biomechanical motions have a default "up" direction, we limited the rotational control provided by the SpaceNavigator when used in the quantitative user study to include only rotation about the vertical axis. The speed of rotation is set to increase linearly with the rotational force applied by the user, up to the threshold of 90 degrees per second.

Animated Space. In Animated Space designs, the scene is also rotated about a vertical axis, but the rotation is done automatically without any user input. The speed of rotation is set to 90 degrees per second, and the direction of rotation reverses after every two full rotations of the scene.

Static Space. In Static Space designs, the visualizations avoid the need for either interactive or animated rotation of the scene through a clever approach to capturing multiple perspectives of the scene in a single display (Figure 4.2 middle). This approach is inspired by the ExoVis technique [95], which is a tested successful approach for visualizing 3D spatial relationships in a static 3D picture (although not in VR). In a typical ExoVis implementation, a 3D object of interest is visualized together with a series of projection planes arranged at orthogonal orientations just outside the bounds of the object. A parallel projection of the object is then texture-mapped onto each projection plane, providing a view of the object from several vantage points within a single 3D scene.

Through iterative design, we developed the new VR implementation of ExoVis, shown in Figure 4.2 middle. The array of 5 projection planes behind the 3D object is a change from the original ExoVis design in which two projection planes are often placed at angles in front of the 3D object and the viewer looks through the "window" in the space between the side planes to see the 3D object. One positive aspect of this original design is that the side planes are very easy to see, and since they are close to the viewer in space, they are quite large, making them easy to interpret. The drawback of this

approach is that in head-tracked VR, these side views, which capture a perspective of the scene from roughly 45 degrees away on either side of the user's default head position, are very easy to generate simply by taking a step in either direction and looking at the scene from that new viewpoint, while the views that are "missing" in the Static Space situation are views from behind the objects, which is what our final Static Space design emphasizes.

4.4 Quantitative User Study Experiment

Using the VR system, visualization designs, and motion data described above, we conducted an exploratory user study. Participants in the study were tasked with analyzing spatial and temporal relationships in the synthetic dataset of moving bumpy disc forms. Accuracy, speed, and confidence were assessed for each of the 8 fundamental visualization designs described earlier.

Hypotheses. To guide the exploratory study, we formulated a series of 5 core testable hypotheses. The first 4 test the design choices that we believe will be best for speed and for accuracy across both dimensions of the design space. The final hypothesis tests users' ratings of confidence in the design.

- *H1:* Users will be *faster to complete* space-time analysis tasks for motion visualizations designed with the *Animated Space* strategy as compared to (a) *Static Space* and (b) *Interactive Space* strategies.
- *H2:* Users will make *fewer errors* in space-time analysis tasks for motion visualizations designed with the *Interactive Space* strategy as compared to (a) *Static Space* and (b) *Animated Space* strategies.

The rationale for these two hypotheses about the design choice for space is that we know from previous experiments performed in other contexts that animated displays (i.e., using automatic camera rocking) have outperformed other designs in terms of the speed of analysis [95]. Although our hypothesis is that this finding will also hold in our situation, we target several differences in the data visualization (e.g., using data that fundamentally represent motions, using VR technology) that are interesting to

test. With respect to accuracy, the closest related work did not find a significant difference between the different display conditions; however, we intuitively believe the users should be most accurate in analyzing this type of motion data when they have complete interactive control over the visualization.

- *H3:* Users will be *faster to complete* space-time analysis tasks for motion visualizations designed with the *Static Time* strategy as compared to (a) *Interactive Time* and (b) *Animated Time* strategies.

- *H4:* Users will make *fewer errors* in space-time analysis tasks for motion visualizations designed with the *Interactive Time* strategy as compared to (a) *Static Time* and (b) *Animated Time* strategies.

The rationale for these two hypotheses about the design choice for time is as follows. First, following a similar rationale as for space, we intuitively believe that users should be most accurate in analyzing these data when they have complete interactive control over the visualization. However, while we believe that Interactive Time is likely to produce the most accurate results, it will be interesting to see whether Static Time and Animated Time are nearly as accurate. One reason that we hypothesize that users will be fastest analyzing Static Time designs is that visualizations that depict time statically (i.e., simultaneously showing more than one instant in time) have outperformed animated visualizations in other contexts[98]. In those trend visualizations, changes over time were read quickly and accurately by users when a full trace of the trajectory over time was displayed, but trends were more difficult to identify when animation was used, despite the compelling aesthetic of the animation. We believe a similar effect will hold true in this situation motivated by complex 3D scientific motions.

- *H5:* Users will be *most confident* in space-time analysis tasks for motion visualizations designed with the *Interactive* strategies (both Interactive Time and Interactive Space) as compared to the non-interactive strategies.

This final hypothesis tests the common understanding that the feeling of "being in control" provided by explicit user interfaces will lead users to be most confident in motion analyses using these designs.

Design. The experiment followed a 3x3 within-subjects design. The independent variables were (1) design choice for time (Interactive Time, Animated Time, Static Time) and (2) design choice for space (Interactive Space, Animated Space, Static Space).

Task. Participants were asked to perform a task that requires both spatial understanding (judging distances and collisions between two organic 3D geometries) and temporal understanding (finding the first occurrence of multiple similar events). While viewing visualizations of the motion of the two bumpy disc forms, participants were required first to detect collisions of two highlighted features (one shown in red on the top disc, and one in green on the bottom disc) with the opposite discs and second to indicate which feature was the first to collide with the other disc by touching the appropriate button (colored red or green) on the touch table. A collision was defined as occuring whenever any part of the highlighted feature intersected any part of the opposite disc.

Training. The experiment began with an initial training period consisting of 18 tasks, 6 with each of the Interactive Space/Interactive Time, Animated Space/Static Time, and Static Space/Animated Time designs. These were chosen strategically to introduce each participant to the full range of design choices encountered in the experiment. During training, visual feedback was provided to indicate correct and incorrect answers.

Experiment Sessions. During the data collection portion of the experiment, tasks were presented in blocks of 26 motions, one block for each visualization design. Before each block, on-screen instructions informed the user of the design choices used for both space and time and the user interface employed in the upcoming trials. Users were also informed that the first two trials in each block were to be considered practice trials. After each block, participants were asked to rate their confidence in the most recent visualization design using a 7-point Likert scale (1 = not confident at all, 7 = very confident). The ranking was indicated by pressing a numbered button on the table surface. In addition to the confidence ranking, the other dependent variables recorded for each subject were time taken and accuracy for each trial. The order of presentation for the 8 visualization designs was randomized across subjects using a balanced Latin square. The order of presentation for the motion sequences (not including the 2 practice trials) was also randomized within each block.

Participants. 5 females and 11 males recruited from a university graduate student

Design	Errors		Time Taken		Confidence	
	Mean	SD	Mean	SD	Mean	SD
I_S, I_T	2.13	1.58	16.99	4.83	5.25^{D}	1.13
I_S, A_T	2.60	1.49	12.88	4.85	4.19	1.47
I_S, S_T	3.10	1.48	16.74	6.67	4.13^{C}	1.08
A_S, I_T	2.40	2.09	14.75	4.11	5.19^{ABC}	0.98
A_S, A_T	3.06	1.58	12.38	6.59	4.19^{A}	1.28
A_S, S_T	3.10	1.36	16.91	7.01	4.25	1.13
S_S, I_T	2.42	1.74	18.72	7.94	4.63	1.09
S_S, A_T	2.98	1.52	15.00	7.56	3.69^{BD}	1.35

Table 4.1: Experimental results for each visualization design. For the confidence measure, significantly different pairwise results are indicated by corresponding superscripts.

population (mainly from computer science) participated in the study and were compensated for their time. The mean age of the participants was 24.56, with a standard deviation of 2.56; one was left-handed. Of the participants, 8 reported playing video games regularly, 6 sometimes, and 2 never; 13 reported little to no prior experience using virtual reality, and 3 reported using virtual reality more than 20 times. Time to complete the study, including practice sessions, ranged from 1 to 2 hours.

4.5 Quantitative Study Results

Each participant produced raw performance data from which we calculated the average time taken for each visualization design and the total number of errors for each visualization design. The raw data, organized by visualization design, are reported in Table 4.1. We analyzed the data using standard error plots and mixed-model univariate analysis of variance (Anova). We modeled our experiment as a repeated-measure design that considered the observer as a random variable and the design choice for space, design choice for time, and time between collisions as fixed. The confidence measures reported by participants were analyzed separately.

The main effect of the *design choice for time* was significant for both the number of errors ($F(2, 345) = 12.560, p < 0.05$) and the time taken ($F(2, 345) = 30.643, p < 0.05$). The main effect of the *design choice for space* was not significant for the number of errors ($F(2, 345) = 1.660, p = 0.192$) but was significant for the time taken

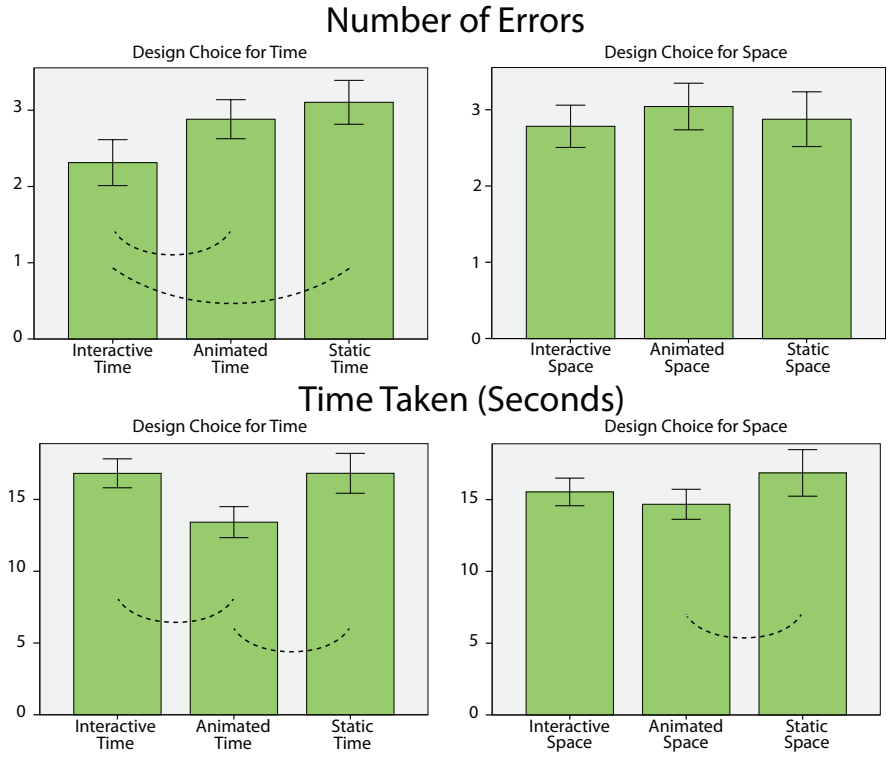

Figure 4.6: Experimental results for number of errors and time taken. Significantly different pairwise measures are indicated via dashed lines. Error bars are +/- 2 SE.

($F(2, 345) = 11.948, p < 0.05$). No significant cross-interactions were found. Follow-on pairwise comparisons were performed with the Bonferonni correction applied. The pairwise comparisons are summarized graphically in Figure 4.6.

Interactive Time designs produced the fewest number of errors. The difference in the number of errors was found to be significant when comparing Interactive Time with both Animated Time (Part a of H4, $95\%CI = [0.941, 0.198], p < 0.05$) and Static Time (Part b of H4, $95\%CI = [1.207, 0.376], p < 0.05$); thus, we accept H4. For the design choice for space, no significant differences in the number of errors were found; thus, we fail to accept H2.

Participants took the least amount of time to analyze the Animated Time designs as compared to both Static and Interactive Time. This does not agree with H3, in which we thought that users would be fastest analyzing Static Time designs, and thus

we fail to accept H3. Although not one of our original hypotheses, we note that the observed difference in time taken was found to be significantly less for Animated Time than for both Interactive Time ($95\%CI = [4.685, 2.116, p < 0.05$) and Static Time ($95\%CI = [4.842, 1.970], p < 0.05$). The difference between Interactive Time and Static Time was not significant. A significant difference in time taken was also found among the design choices for space. Animated Space designs were significantly faster than Static Space (Part a of H1, $95\%CI = [3.620, 0.748], p < 0.05$); the mean time taken for Interactive Space designs was just slightly more than for Animated Space, but the differences were not significant. Thus, we accept hypothesis H1a, but we fail to accept hypothesis H1b.

Participants' ratings of confidence are summarized in Table 4.1. Judging by mean responses, participants were most confident in the Interactive Space/Interactive Time design, closely followed by the Automatic Space/Interactive Time design. (Both of these designs received mean ratings above 5.0.) Table 4.1 indicates significant differences in these ratings based on a Friedman test comparing the 8 visualization display conditions ($\chi^2(7, N = 16) = 38.229, p < 0.05$) and follow-on pairwise Wilcoxon tests. Automatic Space/Interactive Time was significantly different relative to three other designs: Automatic Space/Automatic Time ($z = 2.906, p < 0.05$), Static Space/Automatic Time ($z = 3.688, p < 0.05$), and Interactive Space/Static Time ($z = 2.844, p < 0.05$). Additionally, findings for Interactive Space/Interactive Time were significantly different than those Static Space/Automatic Time ($z = 3.406, p < 0.05$). Revisiting H5, the data partially support this hypothesis. Users were more confident regarding Interactive Time designs as compared to Animated Time and Static Time, and with Interactive Space designs as compared to Static Space but not to Animated Space.

4.6 Application and Domain Expert User Feedback

To understand the potential of the design conditions in a realistic use case, we performed a separate qualitative evaluation with domain experts. We applied the 8 visualization designs to the problem of visualizing the spinal kinematics dataset shown in Figure 4.1. This dataset integrates experimentally collected motion capture of points on the head and back of patients at 100Hz and sub-millimeter spatial resolution [108]

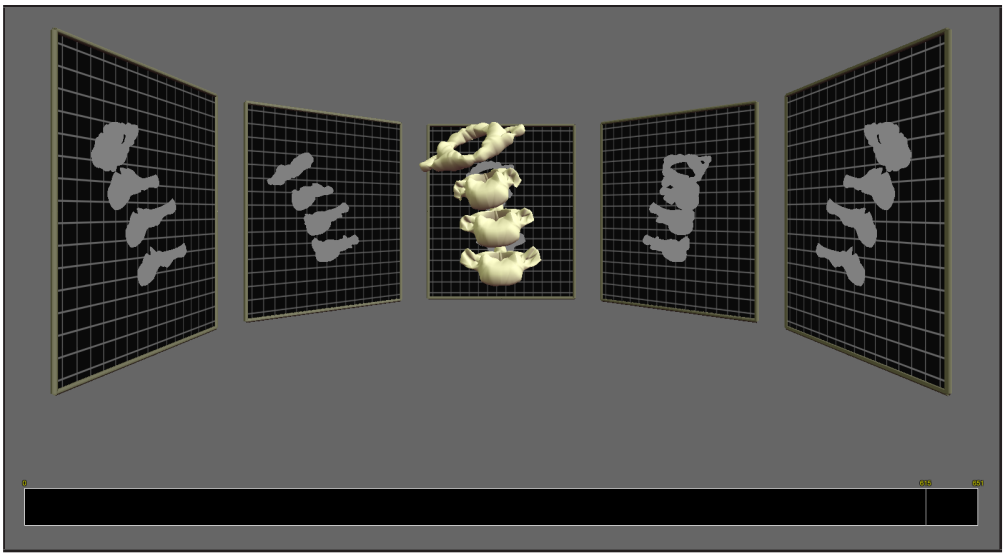

Figure 4.7: An application of the Static Space/Interactive Time design to data from a spinal kinematics study, one of the 8 designs we qualitatively evaluated with domain scientist collaborators.

with simulated internal motions of each vertebra of the cervical spine [109]. A university professor researching musculoskeletal biomechanics and his graduate student, who were our collaborators studying these data, participated in the evaluation session. Together, we critiqued each visualization design based on its potential to succeed in biomechanical engineering analyses and in clinical surgical use. We used a talk-out-loud protocol while viewing the visualizations to facilitate open-ended discussion. After viewing the 8 spinal kinematics visualizations, a final critique was conducted to evaluate the synthetic motion data and geometry used in the quantitative user study.

Figure 4.7 shows the Static Space/Interactive Time design applied to the spinal kinematics dataset. This design sparked considerable discussion during the evaluation. It was viewed as an appealing strategy for visualizing the data in large part because it provides 2D projections of the 3D data that match the views of the anatomy with which surgeons often work (e.g., sagittal, coronal, transverse). Closely mirroring the results of the quantitative evaluation, the domain scientists expressed that the interactive control of time in this design was essential and agreed that, overall, providing this interactive control is likely to be one of the most important decisions in optimizing visualizations

to better support motion analysis.

The Animated Space designs were for the most part met with criticism by the domain scientists. A specific rationale offered for this was that in this type of analysis, maintaining a solid anatomical frame of reference is critical; the scientists indicated that just as they were beginning to understand the spatial orientation of the spine, the visualization rotated away, or even worse, changed the direction of rotation, which complicated their awareness of whether the spine was being viewed from the right or left lateral positions.

4.6.1 Hybrid Designs for Biomechanics

Although Static Time designs performed poorly in the quantitative study, the domain scientists appreciated several aspects of these designs. They suggested that if the Static Time presentation could be used in a hybrid design together with Animated or Interactive Time–for example, switching to a static presentation after one loop of an animated sequence–then Static Time could prove a valuable technique for biomechanics analysis. For example, one specific analysis task that is required to identify aberrant motions of the spine is tracking the relative motions of the superior and inferior articular facets on adjacent vertebrae (highlighted in Figure 4.1). For a flexion-extension exercise, unhealthy motion could be characterized by too little, too much, or uneven motion of the facets. On the negative side, it was clear from the evaluation that the Static Time designs required significantly more explanation and instruction to use than the animated or interactive time designs.

Additional ideas for hybrid visualization designs emerged from the discussion. One was a design that would include a more subtle use of Animated Space. For example, after interactively indicating a feature of interest on the 3D geometry, it may be useful to orbit or rock the geometry around this point to increase motion parallax and spatial perception. In this situation, a point of interest would already be identified by the user, so the complications in maintaining a spatial awareness during animation reported by our domain scientists might be much reduced.

4.6.2 Relevance of Quantitative Study to Biomechanics

In addition to viewing the design conditions applied to the neck data, our collaborators also evaluated the abstract geometry and motion task used in the quantitative user study. During this evaluation, we confirmed the relevance of the abstract forms and task. For example, it was suggested by the domain scientists that analyzing the motion of the highlighted feature points within the bumpy disc abstraction introduced earlier could be seen as directly analogous to the task of tracking the facets of the vertebrae over time and understanding how they interact with (i.e., contact and slide against) each other.

4.7 Interpretation and Conclusions

Results of both the quantitative and qualitative studies support the hypothesis that Interactive Time designs would be more accurate than alternatives.

Results of the quantitative study did not support the hypothesis (H2) that Interactive Space designs would be more accurate than the alternatives. This result is consistent with those of another study [95] that tested spatial-only data in a non-VR environment. If these results are to be trusted and there is some flexibility in terms of the most accurate design choice for space, then the best choice with respect to the speed of analysis could be selected and/or Static or Animated Space could be selected so that other aspects of the visualization could be mapped to the available VR interaction devices. With that said, the qualitative study provides valuable additional guidance that may narrow the options for biomechanics applications. Our collaborators were certain in their evaluation that Animated Space was to be avoided, except perhaps in the hybrid design discussed above in which the researcher first indicates a feature of interest through a user interface and space is then animated around this point. Between the Interactive Space and Static Space designs, there was less of a clear preference; however, the strengths of the Static Space design that were discussed were mostly related to the fact that users who work regularly with medical imaging data have training that could be applied directly to interpreting visualizations in the style of ExoVis.

Time taken to complete the task is also important, although in scientific applications it is often valued less than accuracy. Results of the quantitative study partially support

hypothesis H1, as Animated Space designs were faster to analyze than only the Static alternative. Static Time designs were not, as hypothesized in H3, faster to analyze than the alternatives. Perhaps the combination of the complexity in both space and time in these data requires static visualizations that are simply too complex to analyze quickly, even though they present the user with all of the information needed to make a judgment immediately rather than requiring them to adjust time interactively or wait for an animation to play. Finally, although it was not one of our original hypotheses, we note that inspection of the quantitative study data plots together with the post-hoc pairwise comparisons (Figure 4.6) does suggest a significant result in terms of the speed of analysis, but in this case for Animated Time designs, which were analyzed more quickly than both Static and Interactive Time designs.

The most critical design decision for supporting scientific motion analysis in interactive VR environments appears to be to provide the user with interactive control over time. We come to this conclusion based on the following reasoning. First, since users value accuracy over speed, we look for guidelines first in terms of the accuracy of the different designs. Second, for the design choice for time, the quantitative results suggest that Interactive Time designs are more accurate than alternatives, which is also supported by the qualitative results.

Although there is some quantitative evidence that coupling Interactive Time with Animated Space could produce the best design for supporting a combination of both accuracy and speed, this is contradicted by the qualitative results, in which participants were strongly against using Animated Space for biomechanics analysis, except perhaps in a new hybrid design. As such, it is difficult to arrive at a concrete recommendation for the best design choice for space. One interpretation is that for general-purpose motion visualization in VR, Interactive Time/Animated Space may be the best choice for balancing accuracy and speed; however, Interactive Space or Static Space are likely preferable in biomechanics applications like ours.

In the future, we believe the design space taxonomy presented here, along with additional research building on this study, can play an important role in enabling a systematic exploration of scientific motion visualization. In the remainder of this dissertation, the interaction and visualization techniques we develop are built upon the

foundational guidelines emerging from this design study. Specifically, we develop hybrid styles of data-intensive exploratory visualization that take interactive control over both time and space to their extreme in order to provide expressive and direct ways to control these two dimensions.

Chapter 5

The Slice WIM Metaphor: Overview+Detail Visualization of Large Volumes

This chapter introduces the first of the two new scalable natural user interfaces for performing exploratory visualization and analysis of big data that are presented in this dissertation. As described in the introduction, big data stresses the limits of existing visualization tools. For example, consider the first of the two measures of a dataset that limit existing tools: the individual instance size. In volumetric domains, very large and complex datasets are now common to many fields within science and engineering, where they are routinely generated via imaging, experimental data capture, and/or simulation. As the size and complexity of these data increase, data analyses become increasingly difficult. For spatially complex data, visualization via virtual reality environments has proven useful due to the increased spatial understanding resulting from head-tracked stereoscopic viewing [28, 110, 94]. However, navigating through spatially complex volumetric datasets within virtual environments remains a challenge.

One of the most pressing challenges in virtual reality visualization is supporting effective navigation through the environment while viewing data at multiple scales. For example, in fluid flow visualizations, a typical immersive visualization strategy is to "shrink oneself" to the size of a small particle in the flow to zoom in on local flow

features. At this small, zoomed-in scale, the local flow features are visible and the head-tracked stereoscopic VR environment facilitates the process of identifying local flow patterns. The problem is that, at this small scale, the grounding context provided by bounding geometry or larger-scale flow features cannot be seen. This makes it easy for users to become lost within the virtual environment and very difficult for them to understand relationships between local and global features. Thus, current approaches to visualizing and exploring high-resolution volume datasets in VR do not appropriately support overview+detail data analysis.

In visualization research, maintaining context while focusing on a subset of data is not a new problem. However, since prior work on overview+detail visualization has been conducted primarily within desktop/non-immersive visualization settings, it is not clear how to interpret and apply the core findings of that work (e.g., the importance of multiple coordinated views [47], the benefits of complementing 3D displays with 2D displays [76, 77]) to interactive VR applications.

Within the VR research community, the existing interfaces that are closest in spirit to desktop overview+detail visualization techniques are based on the World-in-Miniature (WIM) metaphor [111, 112]. Although several valuable extensions to VR-based WIM techniques have been recently introduced [113, 114], these have been applied only to architectural models and cityscapes. Determining how best to utilize WIMs to navigate complex scientific volume datasets while supporting both overview and detail views of the data remains an open problem. To further complicate this problem, the goal of most real-world scientific or engineering analyses is not simply to view a dataset, but rather to also interrogate the data by measuring, selecting, and querying 3D data.

In this chapter, we present Slice WIM, a new World-in-Miniature VR interface built upon the Immersive Touch Workbench described in chapter 3 that addresses these challenges while also exploring the new potential of interfaces that combine multi-touch input with VR. The significance of the ITW hardware for interactive techniques such as Slice WIM is that, just as in a CAVE [23] or Responsive Workbench [20, 21], it supports stereoscopic viewing of a 3D model in such a way that the model appears to float in the air above the horizontal surface. However, in contrast to other VR configurations, the horizontal surface here is a multi-touch table, enabling a number of new interaction techniques to be explored. For example, Slice WIM utilizes the Shadow

Figure 5.1: A miniature version of the 3D data appears to float in the air above the table surface. (A digital rendering is superimposed on the photograph to demonstrate the effect.) A cutting plane through the volume data is projected (like a shadow) onto the table below, where multi-touch gestures are used to navigate and interrogate the data. After navigating to a useful view of these imaging data of a heart, the user is now defining a smooth 3D curve (e.g., the shape of a catheter delivery system) relative to the anatomical dataset.

Grab metaphor described in chapter 3 for manipulating 3D objects that float above the table by touching their shadows projected on the table. Figure 5.1 provides an overview of the resulting Slice WIM framework, highlighting the ability to navigate volumes within medical imaging data (here a high-resolution scan of the heart).

The contributions of this chapter include: (1) a new VR WIM interaction technique that specifically addresses the challenges of exploring scientific volume data (e.g., supporting both overview and detail views, linking 2D and 3D data displays within a VR environment); (2) a demonstration of a new approach for utilizing multi-touch input to explore 3D scientific data; and (3) a quantitative design study run to evaluate the capabilities of the Slice WIM metaphor in a navigation task.

The remainder of the chapter begins with a discussion of relevant related work. The

concept of the Slice WIM framework and its use for navigation is then described in general terms. Following this, we describe the details of quantitative design study we ran to evaluate the Slice WIM metaphor.

5.1 Related Work

Our work builds upon three main areas of related research, as described in the following sections.

5.1.1 Overview+Detail Visualization

A wealth of research in both the scientific and information visualization communities has addressed overview+detail visualization (e.g. [51, 47, 115, 46]). However, almost all of this prior work has targeted desktop and/or non-immersive visualizations, using strategies such as multiple coordinated views arranged in separate windows (e.g. [14, 116, 117]). We aim to incorporate design lessons from this prior work while reinterpreting them to apply them to virtual environments. For example, Slice WIM presents multiple closely coordinated views of the 3D data [47] and mixes 3D visualizations with complementary 2D visualizations of the same data [76, 77], although rather than implementing these data displays as side-by-side windows on a desktop, they are implemented as a head-tracked stereoscopic display linked with a 2D multi-touch table display, which supports both mixed 2D and 3D views, and multiple linked 2D views through the persistent slices feature.

5.1.2 World-In-Miniature Interfaces

World-In-Miniature (WIM) interfaces have been used extensively in VR. Introduced by Stoakley et al. [112] and Pausch et al. [111], the core WIM concept is to provide a small (e.g., handheld) model of the virtual environment that can act as both a map and an interaction space as the user explores the large-scale environmentessentially a world within a world. Operations that are difficult to perform in the large-scale world (e.g., navigating a great distance, selecting/manipulating an object that is far away) can be done easily in the WIM. To this core concept and related interaction techniques, researchers have more recently added features to support scaling and scrolling the WIM

for use in very large environments [114] and automatic selection of optimal WIM views, including handling issues of occlusion, for use in complex architectural models in which the 3D structure of the model can be analyzed algorithmically [113]. Although this recent research has focused on architectural and geospatial applications, scale and occlusion are also important considerations when working with scientific volume datasets. Our work uses a gestural touch interface both to set cutting planes within the WIM (addressing occlusion issues) and to scale, rotate, and translate the WIM (addressing scaling and scrolling).

It is often natural to think of WIMs as resting on a surface, either a handheld prop [112], the floor of a CAVE environment [118], or as several investigators suggest, a table [112, 119]. Interactive Slice WIM uses a table, but extends previous approaches by specifically considering volumetric data, which do not include a ground plane.

Outside of VR, a desktop-based implementation of the WIM metaphor provides inspiration for our intended applications to scientific visualization. Li and Hanson [120], for instance, designed a mouse-based WIM interface for exploring astronomical data that present similar overview+detail visualization challenges because of their extreme scale and sparsity.

5.1.3 Touch Interfaces for 3D Visualization

Touch and tabletop interfaces have an established record of effective use in 2D applications. In addition to their recent commercial popularity, several important advantages of 2D multi-touch interfaces have been documented by researchers, including effective selection of on-screen targets [121], improved user awareness in collaborative interaction tasks [122], and improved feedback for virtual and real-world interactions [123].

Only recently, however, have the fluid, gestural styles of interaction typically afforded by these interfaces begun to be leveraged for exploring 3D data [124, 125, 126, 42]. A common theme in this new area of research is overcoming the perceptual problem that occurs when using a stereoscopic multi-touch display: as users reach out to touch the surface, their hand occludes their view of the display, ruining the illusion of an object floating on top of the display. Rather than using an augmented reality display [124], a true 3D display (this technology is still emerging) [125], or adjusting the parallax to a nonphysically realistic value as the user reaches out to touch the display [126],

we solve this problem by employing a multi-surface hardware configuration that allows viewing an unobstructed object floating in space and touching its shadow projected onto a table [127].

This approach is motivated in part by the success of prior desktop interfaces that utilize the metaphor of manipulating 3D objects via interactive shadows [87]. Perceptual research reinforces the validity of this approach. It is now well known that 2D views of 3D scenes provide complementary information to users and that, depending upon the task, judgments based on these 2D views can be more accurate than those based on 3D views [76]. Further, combining 2D and 3D views can be particularly effective for supporting accurate spatial judgments in complex 3D environments [77].

Finally, Slice WIM adopts the strategy of interacting with 3D data that exists in a 3D space above the interactive surface. This is related to (and perhaps most accurately described as the inverse of) techniques that project data onto a physical prop held above a table (e.g., [81]). Our approach also builds upon prior successful strategies for linking a 2D display with a 3D display, such as linking pen-based interactions on a tracked tablet with a wall-size 3D display [82].

5.2 Slice WIM Overview

This section will first provide a general overview of the different visual elements of the Slice WIM metaphor, followed by a description of how these elements are used to navigate data sets. The visual elements and navigation are implemented using the ITW hardware described in chapter 3.

5.2.1 WIM Visuals and Slices

Slice WIM is composed of several visual components, each diagrammed in Figure 5.2. As in traditional WIMs, a miniature 3D copy of the world serves both as a map and an interaction space. In Slice WIM, this *miniature world* is positioned so that it appears to float above the table. As in many traditional WIMs, Slice WIM also provides a visual indicator within the miniature world of the user's current location. However, rather than displaying a miniature avatar of the user, it displays a miniature rendering of the physical display hardware. A small green rectangle (the *view window*) indicates

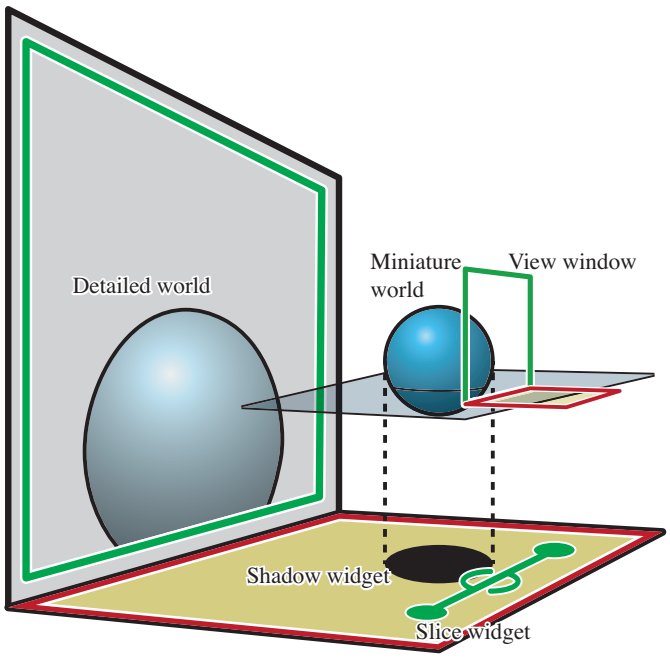

Figure 5.2: Visual components of Slice WIM. In this diagram, the 3D dataset being visualized is simply a blue sphere.

the position of the vertical display screen relative to the miniature world, and a small red rectangle does the same for the table surface. Each of these rectangles lies within a plane (one vertical, one horizontal), and the primary means of interacting with the WIM is manipulating these two planes (slices through the data). As the user adjusts these planes or moves the 3D data relative to the planes, as described in section 5.3, the three linked data views (2D widgets on the table, 3D miniature world, and 3D detailed world) update accordingly. The bottom edge of the view window is always constrained to rest on top of the horizontal slice through the data, which may be thought of as a ground plane, while the vertical slice through the data defines the plane of the view window. The large view of the *3D detailed world* displayed on the stereoscopic vertical screen is defined precisely by the position, orientation, and scale of the view window relative to the miniature world, an example of which is shown in Figure 5.3.

Through iterative design, we found that it is critical to establish a strong correspondence between the physical space of the display and the horizontal and vertical slicing metaphor. Thus, the interface itself uses a consistent color scheme in which all vertical

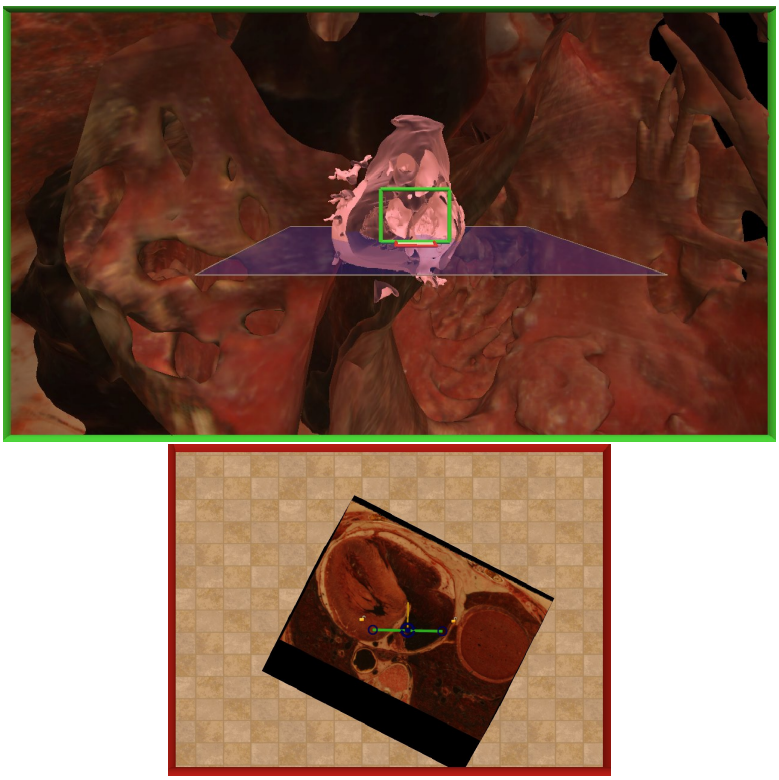

Figure 5.3: The large detailed world shows the geometry of the heart. The 3D miniature world provides contextual information about the orientation of the model. The "shadow" projected onto the table (here interpreted as a slice through the volume data) provides both context and a widget for controlling the 3D interface.

visual elements share one color and all horizontal visual elements share another distinct color. This includes both the rectangles that define the slices and the widgets that are utilized to adjust them. Although the gestures used to control the slicing can be recognized without such explicit visual aids, we found through testing that including specific visual handles and arrows on the slice widget make the interface more readily understood by novice users and facilitates a sense of direct manipulation of the widgets.

5.3 Navigating Datasets

Navigating through the dataset to view the volume and its projection on the table from different angles is supported by a multi-touch interface using two widgets that are not

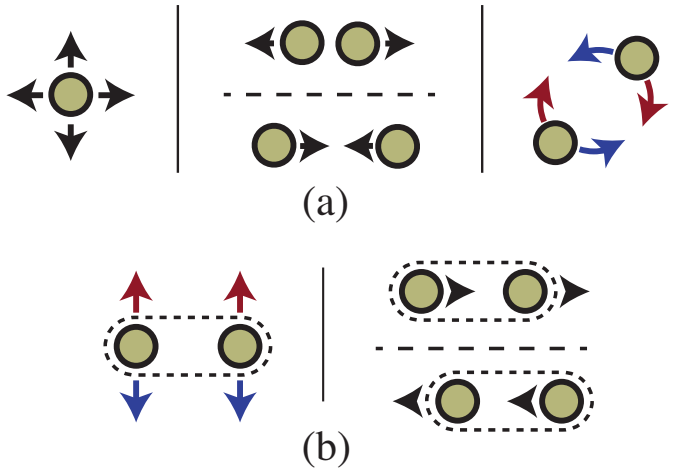

Figure 5.4: (a) Gestures for translating, scaling, and rotating the WIM in the plane of the table surface. (b) Gestures for tilting and rolling.

part of traditional WIMs: (1) a *shadow widget* displays a parallel projection of the 3D miniature world, and (2) a *slice widget* displays handles and controls that can be used to adjust both the vertical and horizontal slices described in the previous section. As both widgets refer to 3D objects the user sees floating above the table, a convenient way to understand these is to think of them as interactive shadows. We next describe each of the widgets in turn.

The shadow widget is a parallel projection of the 3D miniature copy of the world onto the table surface. When interacting with the environment, users quickly interpret the projection as a shadow. To reinforce this metaphor, we often render the projection to look precisely like a shadow, although we have also explored visualizing data on top of the shadow, as in Figure 5.3.

Touching and manipulating the shadow widget causes the 3D miniature copy of the world to move in a corresponding way. During this manipulation, the view window follows along with the 3D miniature world. Thus, the primary reason for performing this interaction is to translate, scale, and/or rotate the miniature world relative to the multi-touch table. Following the multi-touch gestures defined by Coffey et al. [127], these interactions are performed by touching the table with one or multiple fingers and then performing the gestures illustrated in Figure 5.4. As shown in part (a) of the figure,

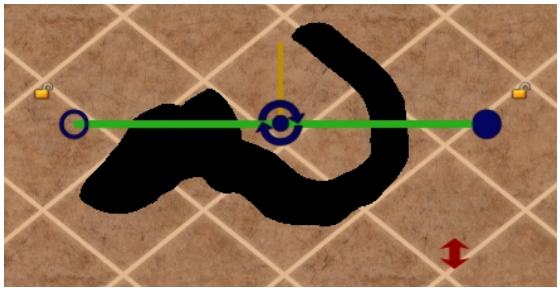

Figure 5.5: The slice widget. Handles (blue circles on both ends of the line) are used to position the view window inside the miniature world. The height of the horizontal slice is adjusted up or down by dragging the thumb near the red double arrow.

a single point of contact provides translation in the horizontal and depth dimensions (the height of the 3D object above the table remains fixed), and two points of contact are used to scale and to rotate about a vertical axis. Rotation around the other two axes in space is also supported. To do this, the gestures in Figure 5.4b are used, which require a slightly different constraint than the gestures in Figure 5.4a. In this case, if the view window were to remain in a fixed position relative to the miniature world, the view window would rotate so that the bottom slice would no longer be parallel to the table. Since this would create a mismatch in the orientation of the view window and the physical display, we decouple the view window from the rotation of the miniature world in this situation. Moving two points of contact together either left or right relative to the user in a direction roughly parallel to the plane of the vertical display rolls the miniature and detailed worlds around an axis perpendicular to the vertical display, and moving two points of contact together in the depth dimension (toward or away from the vertical display) tilts the miniature and detailed worlds toward or away from the user around an axis parallel to the vertical display.

The second important widget is the slice widget (Figure 5.5). The thick green line is the projection of the view window onto the table. A small yellow arrow also points in the viewing direction. The circular areas at each end of the widget are handles that can be grabbed and manipulated by the user. To adjust the view, the user touches and moves these handles. Each handle can be moved independently, or by using two fingers, both handles can be moved at the same time. As the handles on the widget are moved, the view window floating above the table moves along with it, slicing through

Measure	Size Ratio	Mean	SD
Time Taken (seconds)	Size Ratio 1:1.6	37.41	26.26
	Size Ratio 1:1.4	44.05	29.00
	Size Ratio 1:1.2	53.49	38.39
Accuracy	Size Ratio 1:1.6	1.00	0.00
	Size Ratio 1:1.4	0.95	0.09
	Size Ratio 1:1.2	0.93	0.10

Table 5.1: Summary of design study performance data.

the miniature world, clipping the 3D geometry to the plane of the view window as it goes, and updating the large detail view accordingly. Note that this interaction sets not only the position and orientation of the view window relative to the miniature world, but also its scale. As the two handles are moved closer to each other, the width of the view window shrinks relative to the miniature world. In the large linked view of the detailed world, this has the effect of zooming into the dataset.

As a shortcut, if the user wishes to keep the center point of the view window fixed and rotate the viewing direction around this point, this can be accomplished by touching the rotation icon at the center of the widget with one finger and moving a second finger placed anywhere else on the table around this point.

All of these interactions control the vertical slicing plane; it is also important to be able to adjust the height of the view window, which is done by adjusting the horizontal slice that acts as a ground plane for the view window. With at least one of the slice widget handles active, moving an additional point of contact, typically the thumb, vertically on the table changes the current height of the horizontal slice. The red double-sided arrow in Figure 5.5 reminds the user of this control and illustrates a typical location for a thumb touch, below and slightly to the left of the right handle, when the the right index finger is touching the right handle of the widget.

Together, this combination of slicing metaphors, shadow widgets, and multi-touch controls makes possible a particularly fluid style of navigating volumetric datasets.

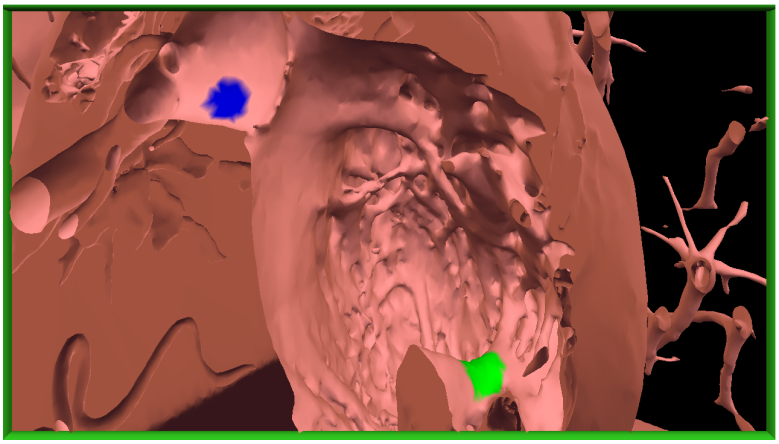

Figure 5.6: The design study search and size judgment task.

5.4 Design Study

We developed a synthetic data analysis scenario consisting of both visual search and size judgment tasks and conducted a design study to understand how novice users are able to learn and apply the multi-touch interface to perform tasks representative of scientific analyses. Eight compensated participants (7 male and 1 female, ages 20-27, drawn primarily from the university computer science population) took part in the study, which included time for training, use of Slice WIM, use of other multi-touch visualizations, and discussion. Specific visual search and size judgment tasks were developed to add structure to the sessions. In all, each participant completed 15 search and size judgment tasks using Slice WIM. During each task, an anatomical dataset similar to that described in section 6.2.1 was visualized with two surface features highlighted, as in Figure 5.6. Participants were asked to first locate the two features within the volume and then make a relative size judgment, reporting which feature (blue or green) was larger by touching a simple colored widget on the table. The two highlighted features were selected randomly from a set of 26 feature points. The surface area surrounding the first feature was colored green, the second blue, and the size differences were varied so that one of the features was always between 1.2 to 1.6 times as large as the other.

The quantitative results include the mean values for accuracy (96%) and for time taken to complete each task (45 seconds). Table 5.4 further breaks down these measures to include the means and standard deviations across the three ratios used for the

differences between the size of the feature points.

We also obtained valuable qualitative feedback from the design study. Most users adopted the following strategy to complete the task: First, they used the shadow widget to familiarize themselves with the shape of the anatomy and position and to orient the miniature world to their liking. Then they grabbed the slice widget handles with two fingers and began to quickly explore the data by slicing through it. As they moved the slice widget, they looked at both the detailed view window and the 3D miniature to locate the target points. If the target was not found, they adjusted the 3D miniature to view the environment from a different orientation, repeating the process until the target was located. Once found, each target point was interrogated in more detail by modifying the view plane to view the target with an optimal orientation and scale. Users also rapidly jumped between the two target points multiple times to facilitate size comparison.

In exit questionnaires, all users reported that the interface was well suited for the task at hand. Users were observed quickly understanding the relationship between the shadow widget and the miniature world. Users also quickly learned to utilize head tracking to view the 3D scene, including the WIM, from different vantage points. When asked what they would like to change about the interface, we received several comments related to improving the ergonomics of the interface. Two users reported that the widgets were sometimes difficult to reach on such a large table; they suggested limiting the interaction space to a smaller region. Three users commented that the technique required concentration and suggested that some form of tactile feedback (e.g., a clear physical widget placed on the table) might enable the widgets to be used more effectively when not looking at the table. One user reported that the miniature world was in the way, suggesting that moving it to the side may help. (Following this feedback, we implemented the WIM locking and return-to-home-position features described earlier.)

In the future, we are keen to conduct comparative studies of Slice WIM with alternative visualization methods; however, we recognize that the design of such a series of studies is a major research effort. One reason for this effort is that, when compared to tools used in current practice, such as the popular Mimics software [128], Interactive Slice WIM is a new system and way of working that combines several novel strategies (multi-touch input mapping 2D to 3D space, head-tracked stereoscopic viewing, and

volumetric WIM techniques). As described below, our application users have found the distinction (and potential advantage) of this new approach obvious compared to current practice. Thus, our evaluations have focused primarily on qualitative comparisons with current practice. For future studies, our recommendation is that the most interesting quantitative insights regarding the interface and visualization techniques that make up Slice WIM are likely to come from a series of studies designed to evaluate the relative importance of specific features (e.g., multi-touch input mapping 2D to 3D space, head-tracked stereoscopic viewing, volumetric WIM techniques) in different contexts (e.g., single-user mode, collaborative mode). The task used in our preliminary study may provide a useful starting point for this challenging and exciting future work.

5.5 Conclusion

We have presented Slice WIM, a multi-surface, multi-touch metaphor for overview+detail navigation of scientific volume datasets in virtual reality. Slice WIM extends successful desktop-based overview+detail visualization strategies (e.g., multiple linked 2D and 3D views) and the successful VR WIM metaphor to semi-immersive interactive visualizations of scientific data. The Slice WIM metaphor is targeted toward navigating large and complex single data instances through a multi-view strategy that allows for viewing data at different scales while maintaining a sense of context.

Chapter 6

Slice WIM Implementation, Applications, and Extensions for Enabling Virtual Engineering Design

The Slice WIM metaphor described in chapter 5 supports overview+detail navigation for exploratory analysis. However, enabling real design work requires interfaces that go beyond navigation, specifically to give users the ability to interrogate, select, and query big volumetric data. Effective interfaces for performing these tasks remain elusive in virtual environments, but are particularly important for supporting real science and engineering design workflow.

This chapter presents a specific implementation of the Slice WIM metaphor that not only supports the navigation style described in chapter 5, but also introduces a number of extensions, including several additional interaction techniques and a suite of GUI tools used for interrogating volumetric environments. We call this implementation and extensions *Interactive Slice WIM* because of its focus on highly interactive techniques that support virtual engineering design. In addition to the extensions described, we also provide a number of specific virtual engineering design applications of the Interactive Slice WIM that demonstrate the usage of this tool.

This chapter begins with a high-level description of five generalizable extensions that build upon the core Slice WIM metaphor and are used to interrogate volumetric data. Following this, four specific application categories are discussed that demonstrate a concrete implementation of the Interactive Slice WIM and specific instances of the generalizable extensions. After this, we introduce a new style of graphical user interface widgets called Shadow Icons that are used to perform common software operations. The Shadow Icons are specifically designed for use with the Interactive Slice WIM tool and ITW hardware. Finally, the chapter concludes with a discussion of design issues and domain-expert evaluation of the tool.

6.1 Interrogating Data

This section introduces a set of extensions to the core functionality of the Slice WIM interface to support interrogating volume data. A key challenge in supporting data interrogation is developing an effective mapping between the rich 2D input available via multi-touch hardware and the 3D data space. As argued previously in the discussion of related work for the Slice WIM, there is reason to believe that rich 2D input may actually be preferable to direct 3D input for some 3D data querying tasks if the right mappings between 2D input and 3D functions can be developed. To this end, Interactive Slice WIM introduces five generalizable strategies for 3D data querying. Each is described in general terms in the following sections, followed by additional details in the application case studies in section 6.2.

6.1.1 Specifying 3D Points, Curves, and Volumes

The first strategy extends the slice widget to support a new ability to specify 3D points, curves, and volumes relative to the current view of the data. The interaction technique utilized is inspired by the creative balloon selection interface introduced by Benko and Feiner [124]; we demonstrate how the bimanual balloon-style controls can be integrated with WIM interfaces and used to construct 3D curves and volumes from control points. The core interaction technique uses input from two fingers on a table to specify a 3D position. The first finger (in Figures 5.1 and 6.1, the index finger of the right hand) specifies the position of the 3D point in a horizontal plane parallel to the table. The

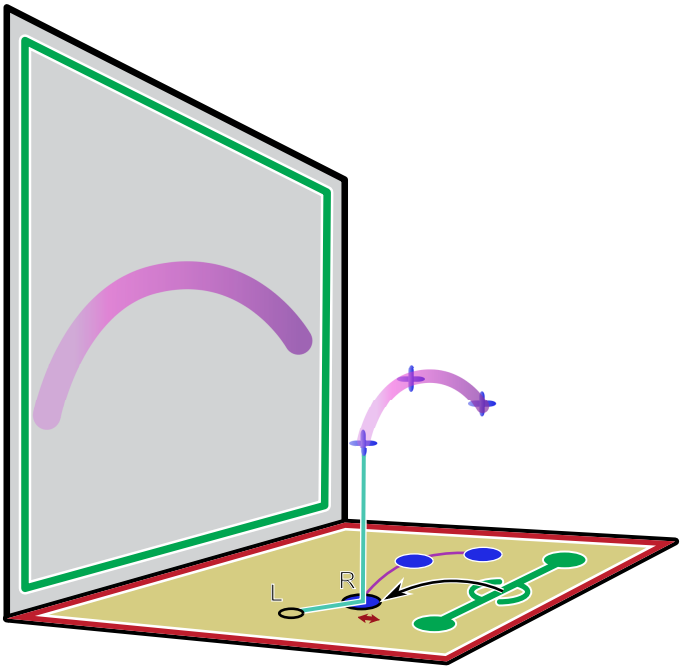

Figure 6.1: Points are specified in the 3D space relative to the miniature world via a multi-touch interface. Multiple points are combined to form curves, and curves may be expanded via an additional touch gesture to define a generalized cylinder of variable radius.

distance of the second finger (in the same figures, the index finger of the left hand) from the first specifies the height of the 3D point relative to the table. From the users' standpoint, a useful metaphor is to imagine the 3D point they are positioning is a balloon that floats above the table attached to a string of a fixed length that they are pinning down to the table with their two fingers; in the figures, this virtual string is depicted as a blue line.

This interface for specifying a 3D point is integrated into the slice widget, as depicted in Figure 6. The technique is activated by touching the center circle on the slice widget and dragging that finger in front of the widget. As the finger leaves the center circle, the 3D point is displayed via a crosshair icon. Placing and dragging a second finger on the display adjusts the height of the point above the table, as described above. The point is displayed both within the miniature world and the detailed world. After the point is created, an icon remains displayed on the table surface to indicate the original touch

point. The position can be edited later by touching this icon to reengage the balloon-style interface, and the set of active points can be cleared by touching an additional icon on the table.

The ability to precisely position 3D points within a complex volumetric space as guided by WIM interface is a useful building block for many additional data-querying operations. After a first point is placed in space, additional points may be constructed in the same manner. A ruler feature (Figure 5.1) is activated when two points are present in the view. When three or more points are added to the view, an interpolating spline is displayed. The arc length of the active curve is printed numerically on the table surface, which provides a natural display for quantitative information like this.

In addition to points and curves, volumes may be specified via an extension of the technique. Using a third point of contact similar to the horizontal height adjustment described earlier (see the double-ended red arrow in Figure 6.1), a radius can be associated with each point. (This input is similar to the balloon radius control in [124].) In the spline drawing mode, this input is tied to the radius of a tube rendered about the center of the spline, turning the interface into a technique for specifying a generalized cylinder of varying radius. A useful application of this technique is described in section 6.2.1.

6.1.2 Interacting with Multiple Persistent Slices

Further extending the core Slice WIM interface, we introduce a second generalizable strategy for interacting with 3D data using a WIM interface controlled via 2D input, specifically the ability to define, organize, and interact with multiple persistent vertical slices displayed on the table. Figure 6.2 illustrates how this strategy integrates with the shadow widget. A new view window is introduced in the front right corner of the table. This window duplicates the view shown in the vertical wall display (hence it is outlined in green) with the exception that the view shows only a vertical slice through the data at the position of the wall display, not a full 3D rendering of the volume. After navigating to a desired location, users turn this slice into a persistent interaction space on the table by touching it and dragging their finger to a new position on the table. This action is illustrated in Figure 6.2, which contains three persistent slices.

We call the slices persistent because they stay visible on the table and remain associated with the position and orientation in which they were originally created even

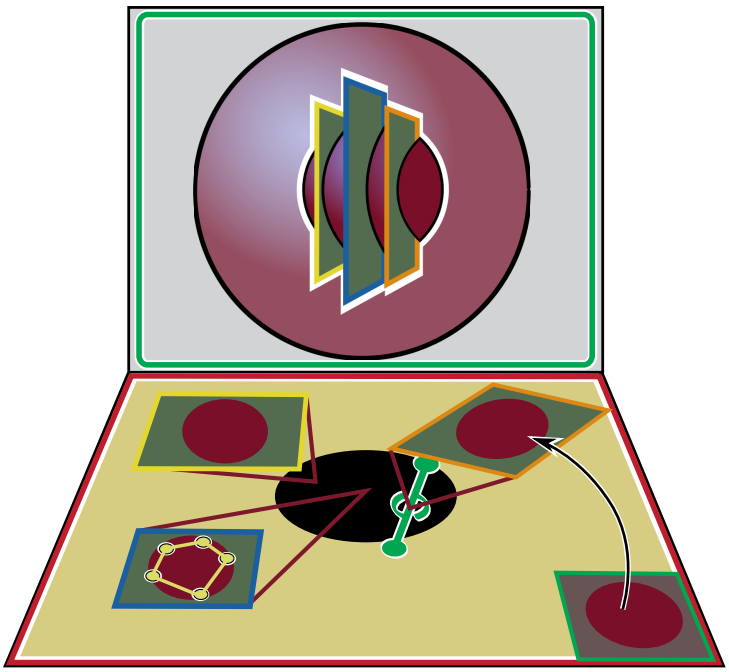

Figure 6.2: Persistent slices are created by dragging a copy of the vertical slice of the data pictured in the bottom right corner of the table to the middle of the table. These persistent slices provide a means of mapping 2D table input into multiple planes within the 3D visualization space.

after the user navigates to a new view. Two complementary rendering strategies are used to indicate each slice's location within the miniature world. Within the view of the shadow on the table, each slice is rendered as a callout box that originates at a location corresponding to the 3D center of the view window at the time the slice was created. In the miniature world, the slices are rendered directly in the 3D view. Consistent color coding is used to associate the callouts on the table with the 3D rectangles rendered in the miniature world. A mathematical mapping can be easily constructed between 2D points within the bounds of each 2D frame on the table and its associated 3D frame placed in the world. Thus, a touch within any of the 2D frames on the table uniquely identifies a 3D point within the 3D visualization space. This feature enables several powerful styles of interaction, as it makes it possible to quickly define many 3D points via multiple touches. A useful application of this strategy to a 3D fluid flow selection interface is described in detail in section 6.2.2. The user can rearrange each slice on the

table to organize the workspace using the standard multi-touch translate, rotate, and scale gestures that are now used in popular photo display and other applications.

It is important to note that for both these persistent vertical slices through the data and the horizontal slices that are central to the Slice WIM technique, it is critical for the visualizations to include sufficient out-of-plane contextual information so that users can correctly interpret the 2D data displayed on the slices. As described earlier, for horizontal slices, the design goes to great lengths via color coding and the shadow metaphor to provide this contextual information. For the vertical slices described in this section, contextual information is provided by the combination of color coding, 2D graphics on the table, and 3D representation of the frames of the persistent slices in the WIM (Figure 6.2).

6.1.3 Interacting with Multiple 3D WIMs for Comparison

In another extension to the base Slice WIM interface, we give users the ability to display and independently manipulate any number of 3D environments (i.e., multiple WIMs) within the space above the tabletop. The gestures described in the navigation-only Slice WIM allow for rotate-scale-translate (RST) control of only a single 3D environment at one time. By extending these gestures to respond to the location where the user touches on the table (e.g., on top of a shadow), the same gestures can be used to control more than one environment. This is shown in Figure 6.3, where three different environments, each with its own local coordinate frame, are positioned relative to a single global coordinate frame shown in black. If a user touches down to perform a gesture and at least one of the fingers lies on top of the shadow of an object (as shown in the red touch events), the RST manipulation will affect the local coordinate frame of the corresponding environment only. If multiple shadows intersect the touch location, the environment whose center point is closest to the touch location is selected. If both fingers do not intersect any shadows (as shown in the grey touch events), the global coordinate frame is modified (i.e., all environments move along with the global frame). A special case is when the user tilts or rolls the global frame, since tilting or rolling around the global frame would cause the local frames to go above or below the fixed height level of the floating WIMs. To avoid this, when the user performs these gestures, all of the local frames are tilted and rolled relative to their own frame (i.e., around their

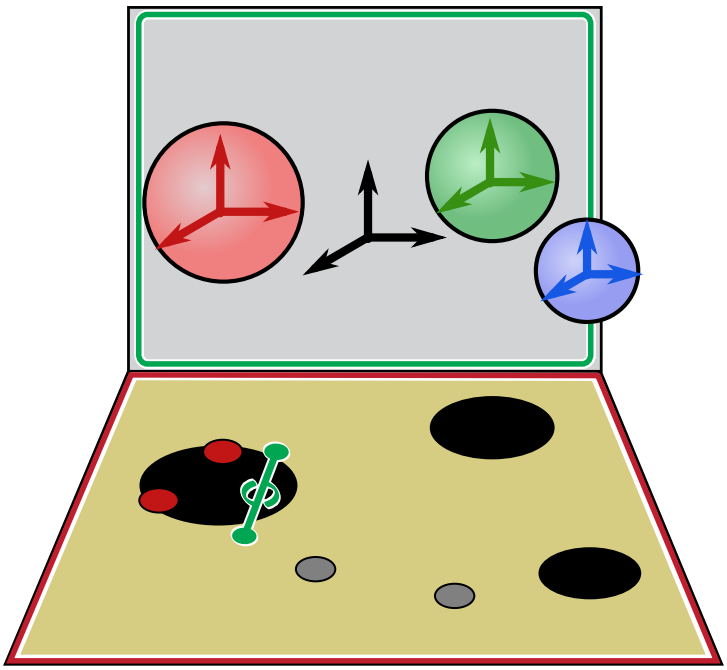

Figure 6.3: The core set of multi-touch gestures are extended to support manipulating multiple 3D environments based on the context of where the user touches. If one of the fingers intersects the shadow of an object, as shown in red, the local coordinate frame of the 3D environment is manipulated. If no shadows are intersected, as shown in grey, the global coordinate frame is manipulated.

center point).

This extension allows the user to perform comparisons across many different 3D environments simultaneously. For example, users can arrange the 3D WIMs in line with each other and rapidly move the view window between WIMs to quickly immerse themselves in the different environments in order to compare features between them. They can also expand the view window so that the zoom level encompasses multiple WIMs in a single view on the detailed world or overlay WIMs on top of each other for even more direct comparison. By touching off the shadows and modifying the global coordinate frame, all of the WIMs can be instantly grouped and modified as a whole.

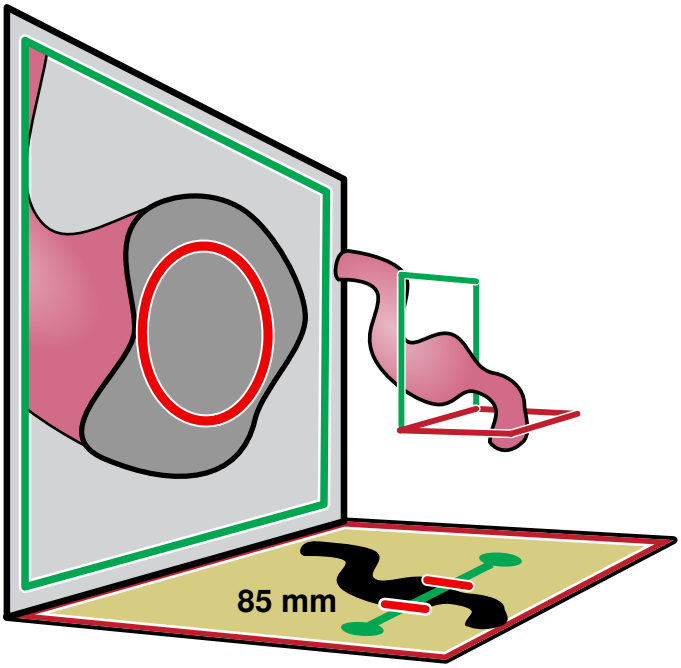

Figure 6.4: The user is able to rapidly measure a radial distance by dragging the red tabs on the slice widget to set the radius relative to the center of the view window. The corresponding circle is drawn in the detailed world on the surface of the screen.

6.1.4 Radial Geometry Sizing Relative to the Environment

In addition to the 3D measuring via curves described above, another extension to the Slice WIM interface allows for radial measurements relative to the center position of the view window (i.e., the 3D position that corresponds to the center of the detailed world screen). This works as shown in Figure 6.4. Two tabs are added to the left and right side of the slice widget and are able to slide from the center of the slice widget to the outside handle respectively (i.e., the left tab slides to the left handle). The distance between the two tabs set the diameter of a circle that is displayed in the detailed world on the surface of the vertical screen. The corresponding exact measure for this diameter is displayed on the table so the user can precisely set the size of the circle.

This style of measuring is useful because it allows the user to rapidly size the circle while still being precise. It accomplishes this by reducing the complex 3D measurement down to a single dimension: a distance along the slice widget. The passive haptics

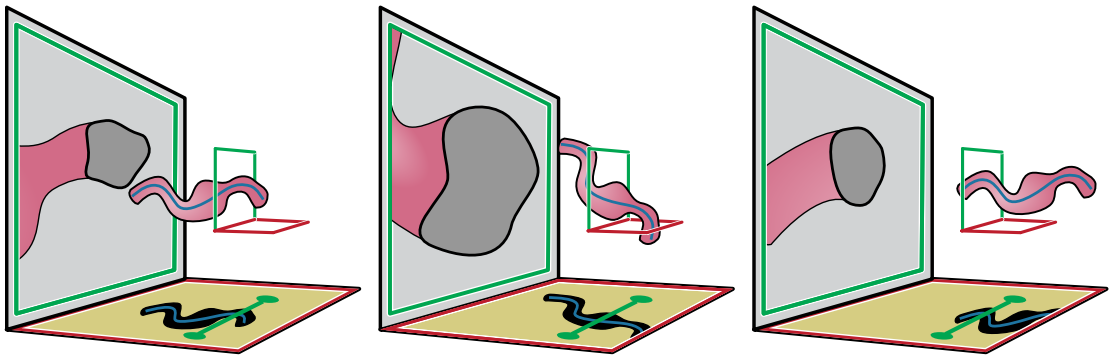

Figure 6.5: After a curve has been created, the user can initiate a fly-through. The view window is automatically centered and aligned to the curve. As the fly-through animates, the WIM is "moved through" the view window, creating the effect of a fly-through in the detailed world.

combined with the exact diameter distance displayed on the table allow users to be precise. Constraining the circle to lie on and be centered at the detailed world vertical screen enables several useful workflows. For example, users can navigate to the center of a cavity and then quickly expand the radius to the bounds of the geometry to measure the extents. User are also able to measure clearances for a known size. They can set the diameter to their desired measurement and then navigate through the environment and visually inspect the detailed world to determine fits and clearances.

6.1.5 Immersive Camera Fly-Throughs

In a final extension to the Slice WIM, we build upon the curve specification extension described in section 6.1.1 to perform interactive fly-throughs of the volumetric environments. After the user has created a curve using the balloon-style interface, he can initiate a fly-through using this curve as a camera path. This works as shown in Figure 6.5. The view window is first centered on the curve and aligned perpendicular to the direction of the curve at the intersect location. Once the fly-through starts, the WIM is rotated and translated such that it "moves through" the view window, keeping the curve centered and perpendicular to the view window. The resulting effect in the detailed world is that a user's view animates along the path. If desired, the user can also adjust the view window during path animation, as it remains stationary relative

to the table. This can be used to either adjust the tangent of the camera path or to change the scale of the detailed world during the fly-through. This extension is made more powerful when combined with the radial sizing extension described in section 6.1.4 in order to visually inspect clearances inside volumes as the camera animates.

6.1.6 Additional Considerations for Highly Interactive WIMs

In the original navigation-only Slice WIM interface, the 3D portions of the WIM (miniature world and view window outline) are displayed only when the user is interacting with the WIM. Three seconds after users lifts their finger from the table, this part of the WIM slowly fades out of view, becoming more and more transparent. When users next touch the table, the 3D portion of the WIM fades back into view. With the new ability to place points, curves, and volumes in space and to interact with persistent slices, we found that in addition to having the 3D WIM fade in and out, users often wished to either focus entirely on the 3D WIM, keeping it displayed at all times, or dismiss it entirely, using only the 2D portion of the WIM on the table while focusing on the detailed 3D view. We found it interesting that this feedback about the role of the 3D WIM came to light most prominently only after users were asked to perform data interrogation tasks instead of just navigation. To address this, we have implemented a small extension to the interface (the lock icons shown in Figure 5.5) to lock the 3D portion of the WIM in either a visible or hidden state.

In our design study, we received feedback suggesting that the most comfortable place to interact with the Slice WIM is in the front and center of the table (section 6.4.4). Although several users mentioned this, the resulting ergonomic concerns did not appear to slow them down when we asked them to navigate through volume datasets. In contrast, during testing of the 3D point creation interface described above, we observed that this ergonomic concern began to impact performance. Since users want to specify 3D points relative to the volume data displayed in either the miniature world or the detailed world, the interaction that specifies those points must happen in a space "beyond" the view window (because the view window clips any geometry in front of it). This means that to create these 3D points, users must reach to a point on the table that is farther away from their body than the slice widget. Often this turns into a long reach; in fact, in preliminary versions of the interface, users tended to default to interacting in front of

the slice widget because it was more comfortable and were confused as to why the 3D point they were specifying did not show up within the detailed view.

To help users default to a movement in which their hands naturally move into the portion of volume that is displayed in the 3D view, we implemented an extension to the interface that returns the slice widget to a home position close to the front edge of the table, as in Figure 6.1. Immediately after the user finishes interacting with the slice widget handles, the entire WIM reorients itself through an animated transition such that the slice widget returns to the home position. The key benefit is that as users initiate the 3D point placement feature, the most natural motion for them to make as they drag their finger off the center of the slice widget is to move their finger away from their body toward the center of the table, which places the 3D point nicely within view in both the miniature and detailed worlds.

6.2 Applications and Specific Extensions

This section describes the use of Interactive Slice WIM in several VR scientific visualization applications that are grouped into four different categories. The first category is data that consist of static high-resolution anatomical segmentations. The second category is data generated from high-resolution simulated fluid flow. The third category is dynamic, time-varying data. The fourth and final category is data composed of multiple 3D anatomies that require direct comparison. For each application category, we describe how the Interactive Slice WIM supports both navigation and data interrogation, emphasizing extensions to the core concepts introduced earlier that enhance these specific applications.

6.2.1 Visualizing 3D Static Anatomy

Two different datasets demonstrate this application category. The first is a heart model generated from 3D anatomical data collected via the visible human project [129], shown in Figure 6.6. The second is a throat model generated via a MRI scan and segmented into multiple anatomical parts, shown in Figure 6.7. In these applications, the primary focus is understanding details of the geometry, such as the specific shape of each of the cavities of the heart. These are representative of a broad range of potential medical imaging

Figure 6.6: Rendering the shadow widget as a true shadow (left) provides strong visual cues for the user controlling the WIM via touch gestures, but reinterpreting the shadow as a more sophisticated data display (right) is often preferable when positioning the view window relative to internal structures in the data is particularly useful.

applications. In both these types of data, high-resolution color information available in the dataset provides additional data channels to visualize beyond those available via MRI or CT modalities alone. In the visible human heart dataset, this color data comes from the photographic imaging modality and is displayed through 3D texturing on the isosurface. In the throat dataset, this color data comes from artist-created texturing designed after segmentation of the isosurface.

Navigating Data

A specific challenge that arises when navigating these data is maintaining a sense of context during detailed exploration. For example, consider viewing the scene in Figure 5.3 without the aid of the WIM, that is, just using the textured detailed world view shown in the background of the figure. This geometry is very difficult to interpret: spatial

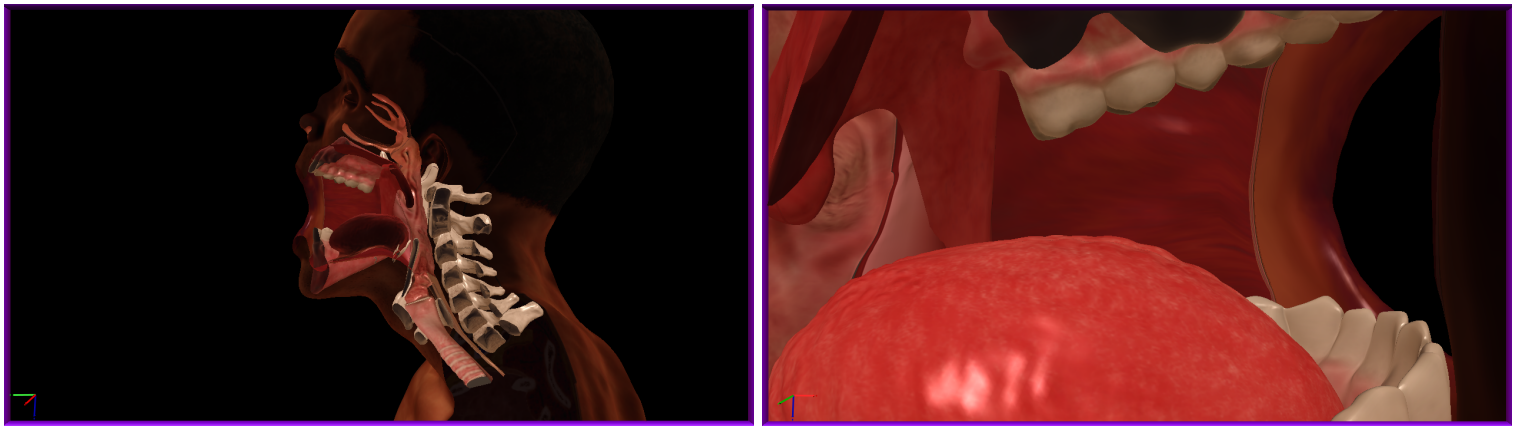

Figure 6.7: Different views as a user navigates and slices inside of a throat model composed of multiple anatomical parts extracted from medical imaging. Data courtesy Center for Research in Education and Simulation Technologies (CREST).

relationships are complex, the shape is organic and includes high variation in curvature, and the data include noise and anomalies that are distracting. When we zoom into these data, it is very easy to become lost. This confusion is minimized considerably by using the WIM as an overview visualization while exploring the anatomy in detail. The miniature world portion of the WIM provides the 3D geometrical context needed to interpret the global orientation of the dataset. The end result of this style of navigation is shown in Figure 6.7, where the users navigate at multiple scales in the detailed world. First, they obtain a zoomed out view of the entire anatomy via a cross sectional slice, then zoom into the mouth to examine the geometry in more detail.

This visible human heart application also explores an interesting adaptation of the basic interface. Rather than displaying a simple shadow of the miniature world on the table, the shadow is replaced here with color information taken from the current horizontal slice through the volume data. We implement this by 3D texture-mapping the original volume data onto a plane. Figure 6.6 illustrates the additional information that this strategy can add for this dataset compared to using just a shadow.

Interrogating Data

Figure 6.8 demonstrates a specific implementation and use of the ability to interactively specify 3D points, curves, and volumes described earlier. A series of points has been placed relative to the anatomy to capture the centerline curve of the aorta. The fact that this curve can be reliably positioned within the anatomy in this way provides some evidence of the control the user has over the interface. In the front left corner of the table, the arc length of the curve is displayed numerically, providing quantitative data to complement the immersive view. After placing the curve, the user is now starting to adjust the radius of a tube centered on the curve to match the extents of the surrounding anatomy, which will generate a volume that matches the form of the aorta. The position and radius of any of the control points that define the curve can be edited by touching the corresponding control point icon displayed over the shadow widget.

From a practical standpoint, this use case is relevant to engineers because they often have a need to make quick 3D measurements relative to volumetric imaging data without requiring time-consuming segmentation of the data. Using the curve construction feature, designers can perform path planning, determine a lead length needed to reach

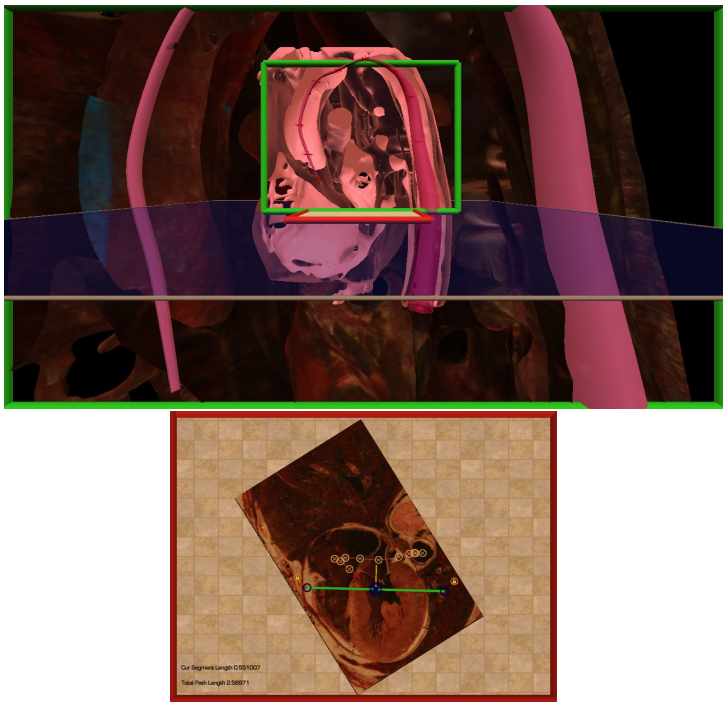

Figure 6.8: A 3D generalized cylinder of varying radius has been constructed interactively to make a volumetric measurement relative to the anatomy. Here the curve has been positioned to trace through the aorta and the user is beginning to interactively adjust the radius of the generalized cylinder to precisely match the aorta.

the target location, or measure curvature variations along a path. With the volume feature, designers can create approximations of anatomical objects on which FEA and CFD simulations could be run, measure clearances, and perform other operations. Since the imagery data are integrated into the visualization via slices, there is always a link back to the raw data. Thus, the interface can be used to accurately measure space even in situations where the 3D data view is only approximate (e.g., when the 3D isosurface displayed is constructed from just a quick approximate segmentation of a tissue). This has potential implications for shortening device design iterations.

6.2.2 Visualizing 3D Fluid Flows

The specific data for the second application category is dense 3D fluid flow data. This data comes from researchers who have developed a series of customized physiologically

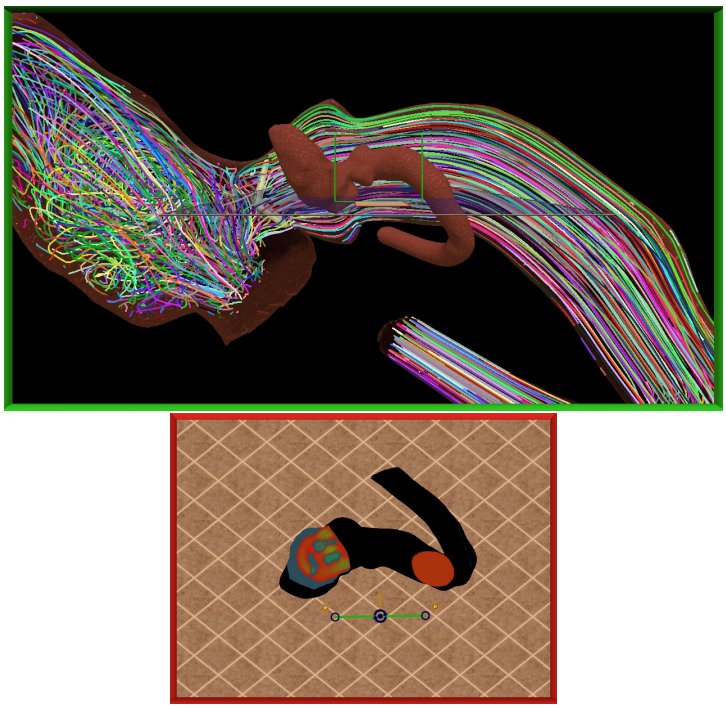

Figure 6.9: Visualizing the results of a fluid flow simulation. The shadow widget is augmented with a pressure map projected down from the horizontal slice and overlaid on top of a traditional shadow.

accurate supercomputer simulations of blood flow through a bileaflet mechanical valve positioned within realistic anatomical models of the left ventricle and aorta region of the heart [130, 131, 132]. In this application, understanding intricate details of the flow patterns formed as blood rushes past the valve and through the aorta is the primary focus, as a potential future application of the technology is virtual prototyping for surgery and medical device design.

Figure 6.9 illustrates the application of Interactive Slice WIM to this problem domain. The 3D scene displayed in the miniature world is a triangle mesh capturing the bounding anatomical geometry. The detailed world view also includes this mesh and, in addition, displays many streamlines (randomly colored in this example) to depict the details of the flow field. In this application, the shadow displayed on the table has also been augmented with additional data. A color map conveys the pressure within the flow at the location of the horizontal slice through the volume data. 3D texture mapping is

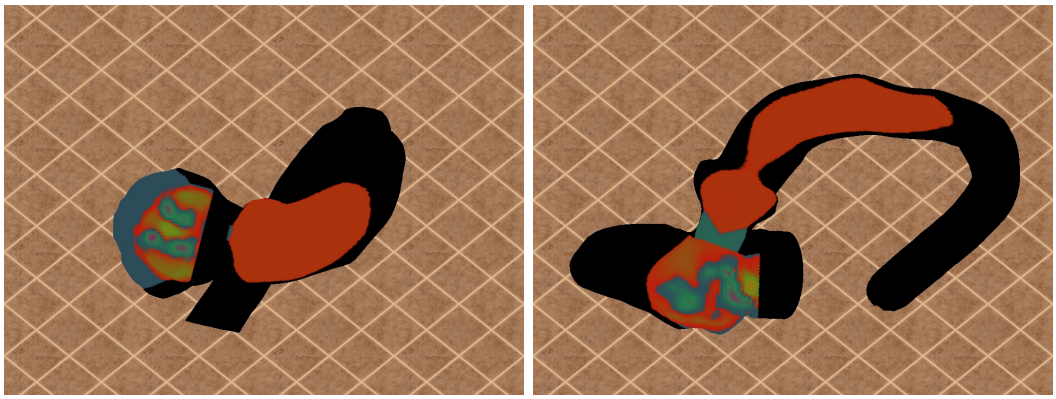

Figure 6.10: Since the planer slice that produces the pressure data does not pass through the entire geometry, it is rendered on top of the shadow of the entire geometry to provide a context for interpreting the data.

used to render the color map. Comparing the two applications, one important difference in the data displayed on the table surface here is that the pressure data are rendered on top of a true shadow (a parallel projection of the bounding triangle mesh onto the table). We found this was necessary because the geometry is such a strange shape that some slices intersect the geometry only in small areas. Figures 6.9 and 6.10 depict this clearly. The shadow is needed both to provide context for interpreting the pressure data and to reinforce the correspondence with the 3D miniature world.

Navigating Data

As seen in Figure 6.9, many detailed 3D flow structures can be captured in state-of-the-art computational fluid dynamics simulation results such as these. For scientists and engineers to interpret vortical structures and other flow features relative to the bounding anatomy and implanted devices, it is critical to be able to navigate within the data to visualize the flow at different scales. In more traditional visualizations, when the view is zoomed in enough to read small-scale features in detail, the larger flow features and bounding anatomy are no longer visible. By providing multiple coordinated views of the data at different scales, Interactive Slice WIM makes it possible to maintain the global context necessary to correlate local flow features with the larger environment.

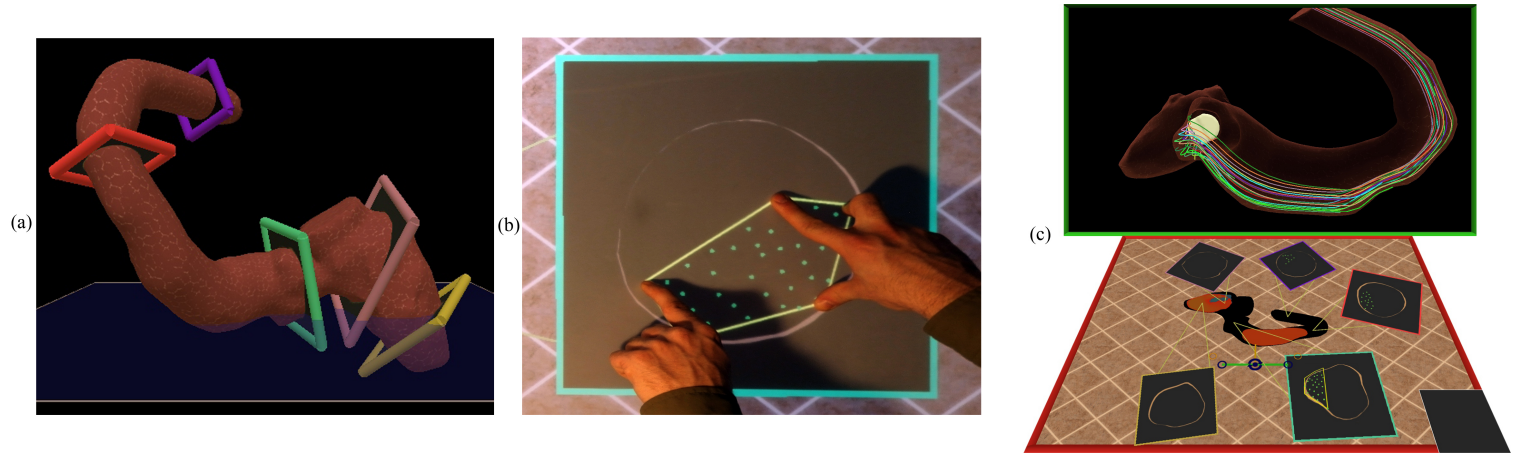

Figure 6.11: A multi-touch 3D selection technique using multiple persistent slices applied here to query streamlines. (A) Several persistent slices have been positioned strategically across the bounding geometry of the flow. (B) A selection is performed and quickly adjusted interactively via multiple touch points placed within one of the persistent slices on the table. (C) The selection is immediately reflected across all the persistent slices and the 3D detailed world.

Interrogating Data

Figure 6.11 describes a multi-touch 3D flow querying interface enabled by the multiple persistent slices feature described earlier. Several slices have been positioned throughout the flow volume to capture different cross-sections of the flow. Each is displayed on the table surface, and within each the cross-section of the bounding geometry is rendered along with a point for each streamline that passes through this cross-section. To select a region of flow, the user places three or more fingers on one of the slices. The system then computes the convex hull of the current touch points (Figure 6.11b). All of the streamlines that pass through this convex hull are selected and the rest are culled from the display. Each of the persistent slices is updated to reflect the currently selected streamlines, as is the 3D view of the detailed world. The selection is dynamic. Users can quickly move their fingers into a variety of shapes to select flow that passes down one side of the geometry or the other or flow that passes by just the hinging mechanism of the valve, for instance.

Several alternative strategies have been proposed in the literature for facilitating this type of 3D selection; for example, lassos and other gestures have been used to select flow lines, neural fiber tracts, and the like [133, 134, 110]. However, 3D selection within fluid flows remains a very challenging task. There are several potential advantages to the strategy we describe here. We are not aware of any other 3D fluid flow selection techniques that provide this level of fluid, dynamic adjustment of the selection. This is facilitated specifically by the multi-touch input technology. Although the input is technically 2D, the multiple points of contact make it an extremely rich 2D input. Using several fingers, a complex convex hull shape can be defined instantly and then adjusted very fluidly simply by moving one's fingers. Thus, rather than making a single selection and then stopping, this technique is most useful as an interactive tool, dynamically exploring the 3D space through iterative culling of streamlines. Another potential advantage of this strategy is that since it works across multiple persistent slices, the user can very quickly jump between and interact with slices that are positioned far away from each other in the 3D volume. In this way, the technique fits elegantly within a WIM-based metaphor, where the miniature environment often provides a simple interaction space for tasks that would be complex if performed directly in the detailed 3D world. For example, without the WIM, selections across multiple planes in 3D space

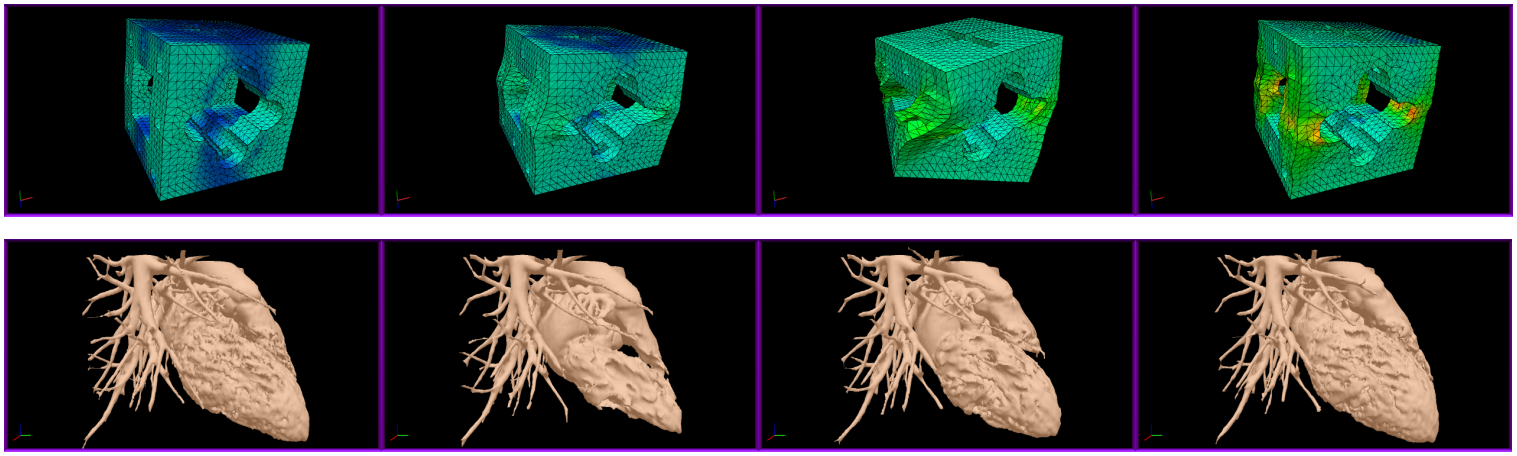

Figure 6.12: Visualizing dynamic temporal 3D environments through animation. Two different datasets are shown with 4 different snapshots in time. On the top, the deformation of a CAD part is animated over time while the color map displays the Von Mises stress at each time step. On the bottom, a beating heart is displayed over time as generated from 4D medical imaging. Data courtesy Boston Scientific Corp.

would require the user to first navigate through the world to reach these planes.

6.2.3 Visualizing 3D Dynamic Temporal Data

The third application category is the visualization of dynamic 4D environments that change over time. These data can be generated via simulation or acquired via imaging techniques. We present two specific datasets in this category. The first specific application is data generated from a FEA stress analysis of a CAD part, as shown in four different temporal snapshots at the top of Figure 6.12. The CAD mesh deforms over time as a result of a load applied to it, and the resulting Von Mises stresses are indicated by the color map. The second specific application is data acquired from 4D CT imagery of a patient's beating heart; the change in the heart geometry as the beating cycle completes is shown in the four snapshots at the bottom of Figure 6.12. These data are animated by playing back the temporal dimension in real time.

Navigating Data

Figure 6.12 shows a number of views that users are presented with after they have loaded the dynamic data and the temporal dimension is animated. All of the aforementioned navigation controls work, and the user is able to slice and scale using the same set of gestures as for static environments. In addition, following the conclusions from chapter 4, we give users interactive control over the temporal dimension, allowing them to control time via VCR style controls on the touch table. The animation can be automatically played, paused, sped up, slowed downed, or manually scrubbed through.

Interrogating Data

Powerful workflows are enabled by using the interrogative features described earlier with dynamic environments. The same measurement techniques are able to measure anatomy across the temporal dimension. For example, measurements can be made at one time step and how they change over time can be observed. Figure 6.13 shows how many of the extensions described earlier are combined to perform a powerful new workflow. First, using the curve specification feature, a path is plotted through the left ventricle of the beating 4D heart. Then, using the radial sizing feature, the diameter of a medical

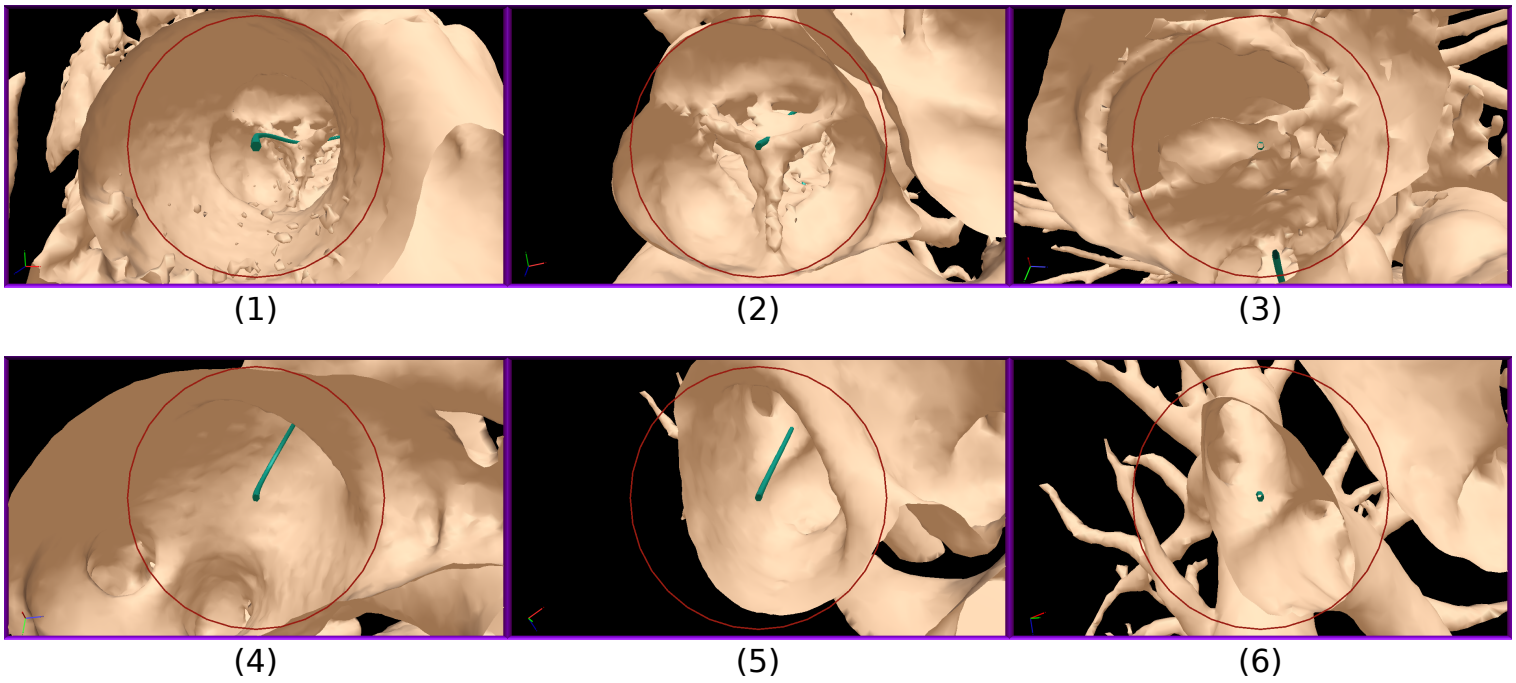

Figure 6.13: Using the radial geometry sizing extension combined with the camera fly-through extension to examine a path through a beating heart. After creating a path that traverses the heart and setting a radial measurement size, shown in red, the designer initiates an animation of the camera along the path. Six different snapshots are shown as the designer visually inspects the size of the anatomy at various features along the path (i.e., the aorta, the aortic valve, the mitral valve, the left atrium, and the pulmonary veins, respectively). Data courtesy Boston Scientific Corp.

device is set. Finally, the camera fly-through extension is initiated on the path. As the camera traverses the path, the fit of the device is visually inspected along the curve. This fit is performed relative to the beating heart, which dynamically changes in the immersive view of the world. This sizing operation is practical for both device design engineers and the medical doctors who implant the devices, as the patient-specific fit and path of deployment are critical to the success of the device.

6.2.4 Visualizing Multiple 3D Anatomies for Comparison

The fourth and final application category visualizes multiple related 3D anatomies in a single virtual environment. Comparisons can be performed across the multiple anatomies by independently navigating and interrogating them. We present two specific medical applications that demonstrate the capability to perform comparison workflows. The data for the first application come from medical imaging of the human ear, used to help hearing aid designers analyze the fit of patient-specific devices. Engineers must understand the effective patient-specific fit of their devices in order to optimize the designs for comfort. The data for the second application come from medical imaging of diseased hearts that suffer from atrial septal defects (ASD), where a hole in the septum exists between the left and right atrium of the heart. Clinicians need to understand the geometry of these holes in order to treat and diagnose patients. Figure 6.14 shows these data loaded in the Interactive Slice WIM.

Navigating Data

Using the ability to load multiple environments at once enables many different styles of comparative navigation workflows. Once the multiple environments are loaded, by using the slice widget, users can change the view rapidly to perform these comparisons. For example, they can quickly zoom in on a feature in one environment, and then rapidly change the view to the corresponding feature in another environment. Alternatively, users can set the view window such that they can see all of the important features in a single view for more direct comparison, as shown in the hearing aid and ASD examples in Figure 6.14. Figure 6.14 also highlights the breadth of different comparison tasks that can be performed. In the hearing aid example, the multiple environments are used to duplicate the data so that different modes of data are displayed at once. In one

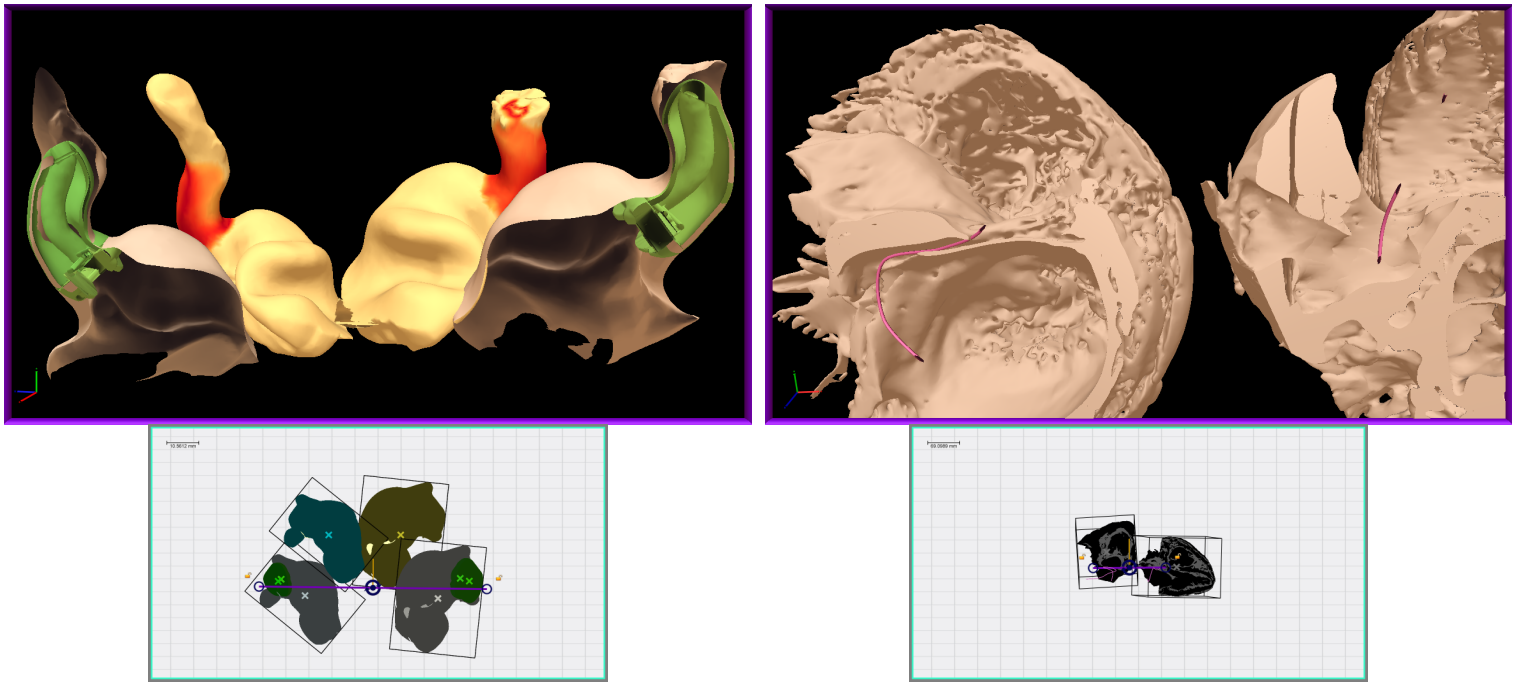

Figure 6.14: Visualizing multiple 3D anatomies within a single virtual environment for comparison. On the left, the geometries of the left and right ear canals are loaded twice, for a total of 4 environments. One canal shows the accurate model of the device geometry (shown in green), and one canal shows a color map of the distance between the device and the anatomy. On the right, two different patients' diseased hearts are shown, both suffering from atrial septal defects that result in a hole between the left and right atriums. A curve, shown in purple, is plotted through the two defects. The variation in the geometry between the two patient's defects is clear: the left elongated and narrow, and the right shallow and wide. Data courtesy Starkey Hearing Technologies (Ear) and Stephen Howard of the Visible Heart Lab (ASD).

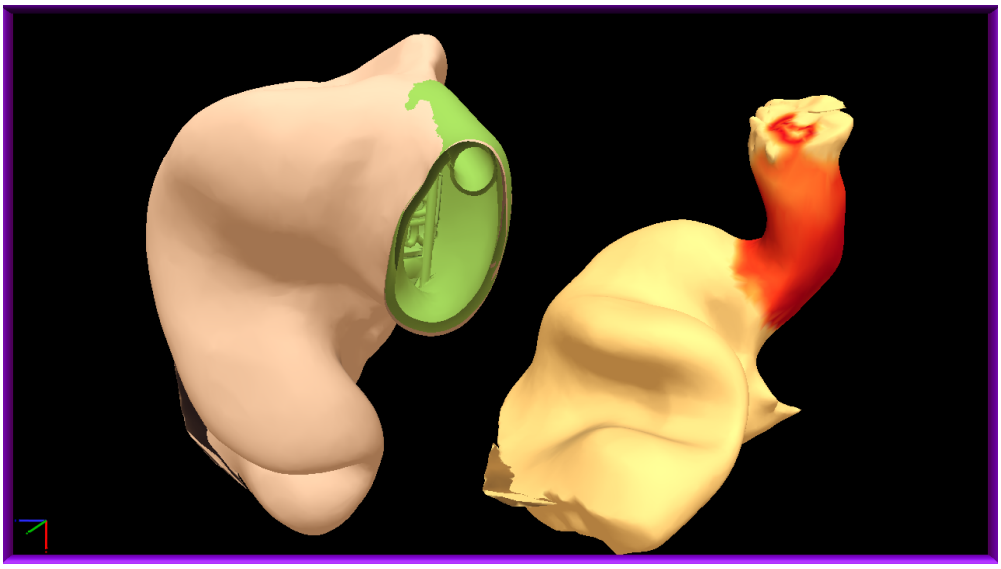

Figure 6.15: Comparison of the explicit hearing aid geometry fit to a derived distance map. On the left, the device's fit is shown by a cross section perpendicular to the length of the device. On the right, this explicit fit is compared against a derived distance field mapped to the outside of the canal. Data courtesy Starkey Hearing Technologies.

environment, the explicit geometry is shown, and in another, a derived quantity that represents the fit is shown. By navigating between these and comparing the two, the data can be correlated. In the ASD example, a different comparison task is performed. Two different hearts with the same diseased states are displayed at once, allowing the defect to be compared across patients to better understand how to diagnose and treat it. Many of these hearts can be displayed at once (as many as will fit on the screen), allowing the comparison to scale up and to categorize a large class of disease states.

Interrogating Data

Using the extension to the core multi-touch gestures described earlier, each of the loaded 3D environments is able to be independently manipulated (i.e., rotated, scaled, and translated) for precise control over the comparisons. For example, in the hearing aid example shown in Figure 6.15, the designer may rotate one of the ear canals to view the explicit geometry from a view perpendicular to the length of the device. The designer can then compare this view to the upright view of the derived distance map shown in

the other canal. Furthermore, using this same extension, the two environments could be manipulated such that they are co-located on top of each other and registered for the most direct possible comparison.

In addition to the ability to independently manipulate each of the environments, all of the aforementioned interrogative extensions are available to the multiple environments. For example, in the ASD data, the curve measurement tool can be used in a number of ways to characterize the defects. The total length of the defect can be measured, or the shape of the hole can be captured by mapping it with the curve tool. Similarly, the radial sizing tool can be used to measure the extents of the hole, and the camera fly-throughs tool can be used to examine the defects in detail. All of these interrogations can be performed across the multiple environments, creating the ability to perform interrogation of the full class of heart defects.

6.3 Graphical User Interface Elements: Shadow Icons

To perform real extended design workflows, the Interactive Slice WIM must be a fully featured software tool. Although the advantages of the new interaction and visualization techniques and extensions described in this chapter may be apparent, designers are often reluctant to adopt new ways of working if the tool is missing many of the basic features that their existing methods of work provide. For example, if the Interactive Slice WIM tool does not have an easy way to load in new 3D models from the hard disk, it is unlikely designers will employ it when their existing software tools easily support this functionality.

The challenge of supporting the basic functionality of existing software tools has plagued the development of virtual environments tools [112]. The traditional 2D GUI elements that use menus and toolbars do not translate well into a 3D space, mainly because the 3D virtual environment eliminates the concept of a window. 3D versions of these elements are often difficult to use or hard to understand. The use of a 2D touch surface in the ITW hardware is ideal for overcoming this problem, as it can be used to recreate many of the familiar 2D GUI elements. Despite the similarities, there are significant challenges related to using the 2D multi-touch surface as a GUI interface for the Interactive Slice WIM tool. These include the fat finger problem [135] that makes it

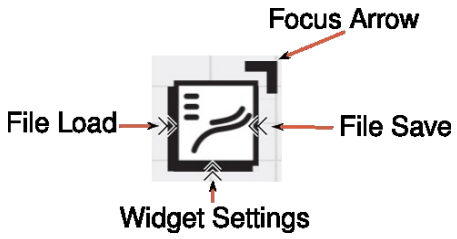

Figure 6.16: The Shadow Icon prototype. The square visual icon identifies the tool. The focus arrow is used to associate the tool with a virtual object on the table surface (e.g., the shadow of a loaded triangle mesh). Three windows can be expanded to select further options. The first two are located on the left and right of the icon and are used to bring up a file browser window that can either load in data or save data from and to a disk. The bottom expandable window contains additional settings related to the tool containing traditional 2D GUI elements, including check boxes, scalar fields, and drop-down lists.

difficult to be precise on touch surfaces, the lack of a keyboard to quickly enter text or other shortcuts, and the challenge of navigating many complex nested menus via touch input.

We have developed a set of new graphical user interface elements we call Shadow Icons that are designed to work with the Interactive Slice WIM tool. Shadow Icons build upon the core shadow metaphor used throughout the tool. Each GUI element is represented by a square icon that can be associated with a specific object or location inside the virtual environment by comparing its location on the table to the location of the projected shadows on objects on the table. For example, an existing virtual object is selected by dragging the icon on top of its shadow. A prototype for the visual layout of a Shadow Icon is shown in Figure 6.16. The focus arrow indicates the current location that with which this icon is associated. Additional arrows on the sides and bottom of the icon are expandable windows that control additional operations linked to this icon. The left and right expandable windows are used to export and import data to or from the disk space, and the bottom expandable window holds additional property settings the user may modify.

In the Interactive Slice WIM, the Shadow Icons are docked and arrayed on the right side of the table screen, creating the set of tabs seen in Figure 6.17. When the user pulls on a tab, it expands and its name is revealed. If pulled far enough, the icon sticks

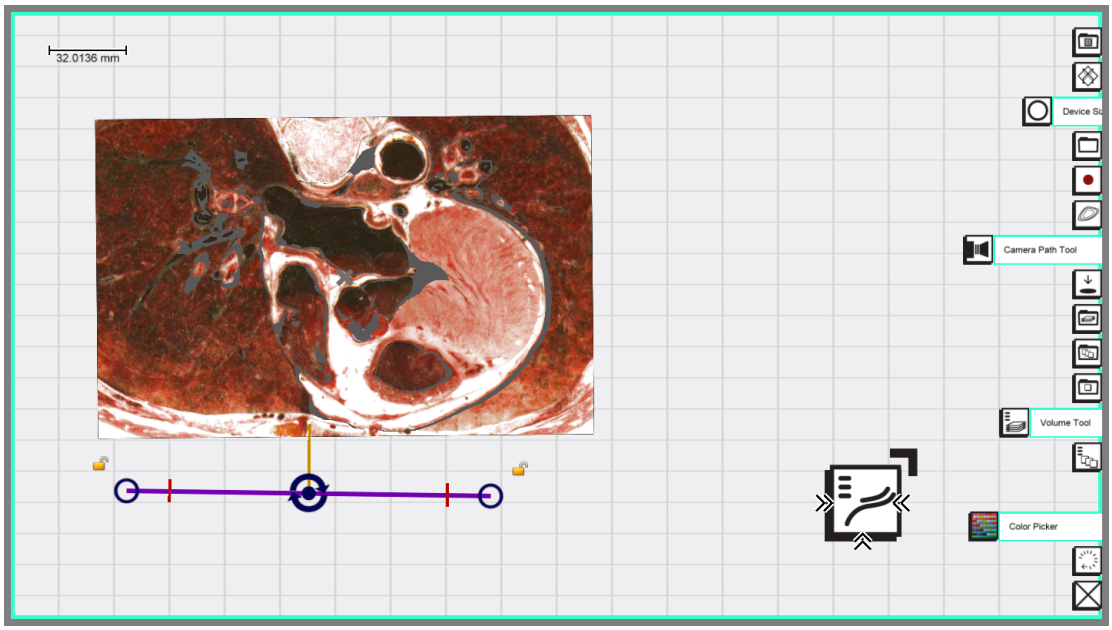

Figure 6.17: View of the table surface with Shadow Icons docked on the right side of the table. Tabs can be pulled out to reveal their name; if pulled far enough, they are undocked and free to move around on the surface.

to the finger and can be moved around the table. The user then releases the icon on top of a data shadow in order to link and activate the icon on the corresponding data. When finished, the user can redock an icon by simply dragging it to the right side and releasing it.

6.3.1 List of Shadow Icons

The following table describes all of the various Shadow Icons we have implemented for the Interactive Slice WIM tool. These GUI elements are used to configure and set up many of the interaction extensions described above. The set of features enabled by these icons make the Interactive Slice WIM a powerful tool in the hands of an engineer.

Name	Icon	Description
Program Exiter		This icon is used to close the software.

Color Picker		This icon is used to change the color an object is rendered in; both individual meshes and splines can have their color changed.
Camera Path Tool		is icon is used to animate a camera along a 3D spline. Once attached to a spline, the animation can be played, paused, reversed, sped up, and slowed down.
Delete Tool		This icon is used to remove objects from the virtual environment.
Device Sizing Tool		This icon is used to control the settings for the device sizing proxy geometry that allows diameters to be measured and visualized in the virtual environment.
Temporal FEA Loader		This icon is used to load and configure a custom file format that describes a 4D FEA mesh with multiple associated scalar fields.
Isosurface Extractor		This icon is used to extract an isosurface in real time from a prior loaded image stack.
Mesh Loader		This icon is used to load in triangle meshes.
Orthogonal Slice Viewer		This icon is used to enable and configure a set of 3 orthogonal axis-aligned 2D slice views.
Project Loader		This icon is used to save and load the current state of the software.
Screen Recorder		This icon is used to capture the current screen to a video file for playback at a later time.
View Reseter		This icon is used to reset all the virtual objects to a default view.

Spline Tool		This icon is used to save and load splines to disc and to configure a number of additional properties of splines (e.g., the ability to freeze them so they are immutable).
Animation Tool		This icon is used to control the animation of virtual objects once 4D datasets have been loaded. Timing information and playback controls are available.
Temporal Mesh Loader		This icon is used to load a set of meshes that are temporally linked (i.e., 4D data) from disk.
Volume Tool		This icon is used to configure the properties of a loaded volume. This includes settings that define the units of the volume and the dimensions.
Volume Loader		This icon is used to load image stacks (e.g. DICOM sets) from disk into volumes.

Table 6.1: List and description of the Shadow Icons used in the Interactive Slice WIM tool.

6.4 Discussion

6.4.1 Feedback from Application Users

We have collaborated closely with engineers and scientists working in the area of medical device engineering to review the design of the tool as it has progressed and to better understand its potential to impact this domain. Several design decisions come as a direct result of this feedback. Most importantly, the emphasis in the new work reported in this chapter on moving beyond navigation to also support interrogating data is reflective of

the needs of medical device engineers. In early stages of medical device design, estimation (e.g., of distances, forces) is often used, yet it is important to relate these estimates to realistic conditions, such as anatomical data collected from medical imaging. Interactive Slice WIM facilitates this because it makes it possible to very quickly navigate through data and take precise measurements. Currently, the measurement strategies include a 3D ruler, a curve tool, and a volume tool that follows a curve, although we believe many other extensions (e.g., fitting nurbs surfaces to anatomical features) are also possible via this framework. Moving beyond estimation, we are also excited by the potential to explore customized physiologically accurate simulations, such as in the application described earlier, which capture features at a level of complexity that is difficult or impossible to view via more traditional desktop-based visualizations. Thus, we envision that interfaces in this style can be useful not just as visualization spaces, but also as interactive design tools.

6.4.2 Alternative Designs Considered

An alternative WIM design that we considered departs from the shadow metaphor and instead positions the miniature world within the physical plane of the table, displaying a stereo view on the table surface. We were initially drawn to this design because we believed that using the table itself as a horizontal slicing plane would be a metaphor that would be readily understood by users. However, we determined that this design would present several problems: First, since the data that interest us are volumetric, we need to be able to see what lies inside the data. If the miniature world were placed on the table itself, it would be very difficult to see any of the internal structure of that world, since the view would always be limited to a bird's-eye perspective. In addition, the type of 2D data displays that we developed in both the applications to augment the shadow widget would not work in the configuration where the WIM is positioned directly on the table. Second, this configuration would suffer from the same perceptual issue mentioned earlier in the discussion of related work: when interacting with the table, the user's hand would obscure the display, ruining the stereo effect. For these reasons, we believe the current design offers a number of immediate advantages, including making it practical to include both 2D and 3D data displays within a WIM and making it possible to effectively utilize multi-touch input to interact with a 3D WIM.

6.4.3 Hardware and Ergonomic Considerations

Although it is possible that many of the Interactive Slice WIM techniques could generalize to other hardware configurations, several aspects of the current hardware configuration make it particularly well suited to the technique. One alternative that could be considered is to replace the large tabletop surface and move the input to a handheld tablet. This might have the advantage of enabling the user to walk around a large 3D display, such as a CAVE, or to sit in a chair. A disadvantage would be a loss of screen size, limiting the effectiveness of techniques such as the multiple persistent slices that require room on the table to organize data views. A second likely disadvantage is that the shadow metaphor may be less effective in a mobile environment where the touch surface is not fixed in a physical horizontal orientation.

Although the current hardware is limited to a single head-tracked view, we have held numerous successful multi-user sessions and discussions with 4 or more participants standing around the table. As a result, we now believe these collaborative scenarios will emerge as one of the most important use cases for Interactive Slice WIM. In these scenarios, users typically stand around the table, but in other cases, it may be preferable to sit on stools around the table.

As with any visualization technique developed for multi-surface display hardware, users' attention may potentially be divided between the two displays in a way that negatively impacts their workflow. Building upon the recent successes of multiple coordinated view visualization techniques in 2D environments (see section 5.1), we believe similar strategies can make a major impact in improving VR workflows. The challenge in VR is determining the right interface for integrating complementary (often 2D or slice-based) information with traditional VR displays. In this regard, we consider the addition of a second table display surface as potentially a extremely valuable addition to VR hardware. As in traditional VR techniques that make use of virtual toolbelts, handheld palettes, or menus floating in space, the table display requires users to look away from the primary 3D subject to access additional information; however, we have observed that users often fluidly change their focus between the 2D and 3D views even as they are performing interactions such as the fluid flow selection interface built on multiple persistent slices. Thus, we believe this multi-display framework can fit naturally into many workflows.

6.4.4 Perceptual Considerations

Another result from the design study was the finding that users felt uncomfortable when the miniature world appeared too close to their eyes. This can create a problem, since when users push the miniature world further away, this also moves the shadow widget away toward the back of the table, where it is less comfortable to reach. We determined that this issue can be easily solved by applying a slight offset and downscale to the miniature world, moving it farther away from users' head but keeping its shadow close to their hands. Scaling the miniature world down slightly relative to the shadow also helps. Our implementation uses an offset of 0.5 ft. and a scale factor of 0.75. After several months of use, we were surprised that no users noticed this offset and scale without being told about it. Thus, it seems to significantly improve usability without introducing a perceptual mismatch between the miniature world and its shadow.

Inspired by this finding, one avenue of research that we plan to investigate in the future is developing a better understanding of the perceptual issues surrounding manipulating interactive shadows of 3D objects. We believe this may be an area where the smart use of perceptual illusion can enhance 3D user interfaces.

6.4.5 Design Guidelines for Immersive Touch Workbenches

Through the implementation of the Interactive Slice WIM, we have arrived at a number of general guidelines that should be followed when developing techniques for use with Immersive Touch Workbench hardware. These guidelines are supported by the design decisions we arrived at through the implementation of the interactions that make up the Interactive Slice WIM tool.

Support New Complex Tasks via More Immediate Gestural Interfaces

One of the most exciting opportunities provided by emerging touch technologies is the ability to explore styles of input that have not previously been possible (e.g., using many simultaneous touch points). In developing new interfaces, we should strive to go beyond replicating existing interfaces with touch wrappers and to instead enable new complex tasks with fluid and immediate gestures. For example, consider how different the interaction described in section 6.1.1 would be to perform using 3D modeling software with a

mouse and keyboard. That process would take many iterations and multiple viewpoint adjustments to precisely create a curve. The immediate, real-time updates combined with physical action in the 3D touch interface are likely to dramatically decrease completion time and accuracy. Similarly, the volumetric selection in section 6.1.2 allows not only quick 3D lassoing, but also fluid, immediate adjustment of the selection, qualities that are absent in current 3D visualization interfaces.

Use Smart Metaphors and Hardware to Avoid Hand-Occlusion Issues

An important design challenge in immersive touch interfaces stems from the problem of breaking the stereo illusion when one's hands block the view of virtual objects beneath. Some hardware-based solutions to this problem are possible (e.g., the multiscreen approach of ITWs). Equally important is the use of smart interaction metaphors. For example, the slicing metaphors all use a shadow projection to ground the floating WIM to the horizontal table. Additionally, the visual elements and colors that depict the slice through the WIM are linked to the physical orientation of the screens, with red indicating a horizontal orientation and green a vertical orientation. The balloon technique employed in section 6.1.1 is also a good example of a metaphor that helps to work around the problem of hands occluding stereo touches. By interacting with a "string" attached to a 3D object above the table, this interaction technique avoids the situation where the hands and object are collocated and has the advantage that the mapping between the hands and balloon is grounded to the physical touch surface.

Combine 2D and 3D Inputs and Visuals

One potential strength of immersive touch workbenches is their ability to easily mix and match 2D and 3D inputs and visuals. Too often we unnecessarily limit ourselves to one or the other when each presents its own strengths and weaknesses, leading us to advocate for a more holistic approach. For example, the Shadow Icon GUI system shows how traditional 2D GUI elements are adapted to the ITW workspace for effective usage. Similarly, the visuals employed in section 6.1.2 make use of the tabletop as a rich 2D organizational space that is linked to a detailed 3D view.

6.5 Conclusion

We have implemented and built upon the core Slice WIM navigation metaphor to develop a software tool we call Interactive Slice WIM. The Interactive Slice WIM tool introduces a number of new generalizable techniques for interrogating and querying volumetric data. These include techniques for:

- specifying curves and volumetric selections,
- creating persistent interactable 2D slices on the table surface,
- manipulating multiple 3D environments that exist above the tabletop,
- performing radial measurements, and
- flying through volumes via animated camera paths.

These extensions offer many new possibilities for querying and exploring volume data, as demonstrated by the numerous specific scientific applications presented. We also introduced a new graphical user interface technique designed to work with the Interactive Slice WIM called Shadow Icons that allows for familiar software operations within this new hardware and software platform. Finally, we provided a discussion of the Interactive Slice WIM tool that includes expert evaluation and a set of design guidelines derived from the implementation. In the future, we are excited by the potential to expand upon these techniques and the new VR applications presented in this dissertation, specifically with the goal of providing not just views of data, but also new interactive engineering workflows.

Chapter 7

The Design by Dragging Framework: Direct and Natural Exploration of Simulation Design Spaces

This chapter introduces an exploratory visualization system targeted toward overcoming the second limiting measure of big data: its many data instances. Specifically, we focus on simulation-based design problems through a system we call Design by Dragging. Paralleling the earlier description of the Slice WIM metaphor and implementation, this chapter describes an abstract overview of the Design by Dragging framework. The subsequent chapter provides a detailed technical description of our implementation of this framework, along with numerous examples and expert evaluations.

Simulations (e.g., computational fluid dynamics [CFD], finite element analysis [FEA]) are poised to play an increasingly important role in design processes within engineering, science, entertainment, and art. Unfortunately, despite their clear importance to many disciplines, current frameworks for working with simulations do not complement (and in many cases inhibit) the human process of creative design. In the medical device industry, for example, the current consensus is that simulation-based engineering will be one of the most important keys to future innovation [136]. Nonetheless, working with

simulation tools often requires practicing engineers to adopt a completely different design process. The major problems with today's approaches to simulation-based design are twofold:

1. Simulation tools do not allow designers to understand, compare, and make decisions based upon alternatives in a large design space (i.e., a set of many related simulations).

2. Simulation tools require an often distracting focus on boundary conditions, parameters, and other low-level details that in many cases are tangential to the designer's goals and may be understood only by a trained expert in computational methods.

Today, these limitations make simulation more useful as a method for detailed validation of a single design than for creative exploration of alternatives throughout the design process.

Outside of engineering, designers in other contexts (e.g., entertainment) face many of the same challenges in trying to appropriately fit physically based simulations into creative processes that rely upon human input. On one end of the spectrum of design tools, traditional paper and pencil and quick, sketch-inspired 3D modeling systems provide designers with the immediacy, expressiveness, and human engagement needed to enhance creative design. On the other end of the spectrum, detailed simulations, often run in batch mode, provide one or more sets of physically accurate data. There are few computational tools between these that can facilitate creative design *along with* new insight from data-driven simulation.

In this chapter, we argue that these challenges can in large part be solved through new computer graphics interfaces. Our work shares a common motivation with the early work in design galleries in setting simulation parameters for computer graphics and animation [67] and with a series of recent results that focus on visualizing large sets (ensembles) of simulation results (e.g., [68, 72, 137]).

Here we introduce a direct manipulation framework we call Design by Dragging. Our goal is to enable the designer to focus entirely on the design problem rather than the underlying details of simulations and to react naturally to visualizations of simulation results. For medical device engineers exploring the space of possible biopsy device designs, this means that they will sometimes want to drive the exploration in a specific

direction by adjusting a geometric parameter of the device, which we call forward design. In addition, they will also want to react to the visualization of simulation results, asking, for instance, "What causes this region of high stress, and what would it take to move this region to another location or remove it entirely?" We call this inverse design, since the path it implies through the design space is specified in terms of simulation outputs rather than inputs.

One of the defining features of this framework is that it supports direct manipulation for both forward and inverse design strategies within the same interactive computer graphics display. For forward design, we accomplish this through interfaces in the style of successful direct manipulation 3D modeling tools (e.g., [138]). For inverse design, we navigate and interpolate within a set of pre-computed simulations, displaying the best match based on the user's input. To make the user interaction feel as direct as possible, we introduce the idea of directly manipulating data visualizations, such as scalar stress fields output from simulations, in order to create a continuous descriptive visual query. The style of the direct manipulation is reminiscent of recent shape manipulation interfaces based on dragging or tugging metaphors [139, 140].

The major contribution of this chapter is a novel generalizable interaction framework for exploratory visualization of simulation-based design problems using direct manipulation, which includes the following:

- A new accessible approach to simulation-based design for tuning parameters of computationally intensive simulations and for visually evaluating large sets of results.
- An overview of our work on the Design by Dragging framework in terms of the pre-computation data structure and the interactive exploratory visualization loop.
- A discussion of the design philosophy for systems implementing the Design by Dragging framework, including a set of design principles and guidelines to be followed when designing interactions in this framework.
- A description of the visual elements used in the framework and a specific example of a forward and of an inverse direct manipulation interaction.

7.1 Related Work

Our work builds upon recent results in direct manipulation interfaces, simulation-based design tools, and ensemble visualization.

7.1.1 Direct Manipulation Interfaces

We employ a natural, direct manipulation user interface for modifying geometric models included in the simulations. Our approach is inspired by Sketch [138] and other more recent direct manipulation modeling interfaces, such as iWires [139]. Several interfaces (iWires and others [141, 140]) include a real-time simulation and/or constraint engine to add physical realism to direct inputs that fit seamlessly into our framework. For inverse design, we utilize an As-Rigid-As-Possible Shape Manipulation algorithm together with a multi-touch interface to enable the user to specify rich visual queries more quickly and directly than can be accomplished with a single-point input device.

Given that a major focus of our work is inverse design, we are particularly interested in interfaces that control (or feel as though they control) an inverse process. A key challenge in creating such interfaces is balancing the goals of providing a direct, expressive interface for the user and of steering the direct manipulation toward a valid configuration. Some previous systems that do not include simulation nonetheless provide powerful examples of this idea. In Accessible Animation [142], the user navigates through a space bounded by valid keyframes either by using a control space widget (forward direction) or, in a more exciting way, by "tugging" directly on the animation (inverse direction). During a tug (similar to our inverse dragging), the system interprets the user's intent and morphs from one valid configuration to the next most logical choice. The new configuration may imply a change to more than one aspect of the animation. For example, if the user tugs on the head of a character to turn the character around on the page, this could also imply a change in the character's feet, which may also need to turn to maintain a valid pose. A similar approach has shown promise for navigating through a space of registered car models simply by tugging on line drawings of the cars [143]. An inverse-feeling direct manipulation interface has also been used for browsing videos [144].

When these interfaces work correctly, they can feel almost magical to the user -

there is no more direct way to query a video dataset for, for instance, the time when a car drives out of a parking lot than to simply grab the picture of the car in the video and drag it along. We believe one of our most important contributions is achieving this sensation for users working with simulation datasets, including FEA or CFD data used by engineers in real-world design problems.

7.1.2 Design with Simulations

Our work builds upon the motivation set forth in early work on Design Galleries, which introduced the general approach of intelligently sampling a design space via pre-computed simulations and then presenting the results to the user in the form of a gallery of keyframes [67]. The simulations that best fit this approach are defined by two characteristics that are shared by our applications: high computational cost and unquantifiable output qualities. Together, these imply that the simulation-based design problem cannot be solved via a computational optimization but rather requires human input in the design and usually human visual analysis of intermediate results.

Many-Worlds [69], a system for trajectory-based character animation, also shares a similar motivation but different approach and application. Because we target dense, continuous datasets, our preview bubbles and visualizations via morphing must provide much more information than simple trajectories. Also, our goal configurations are less well defined than trajectories, often requiring a trained engineer to interpret. Other approaches to simulation designs include refining and accelerating the simulation to the point that it can accept real-time inputs [145], using real-time simulation to suggest valid alternatives (e.g., in furniture design) [146], and developing simulation engines that can work toward a desired output based upon a detailed artistic specification (e.g., 3D curves created in Maya to show the direction a simulated flame should take) [147]. Our work differs from these examples in that it can be combined with existing simulation packages (e.g., Abaqus, Maya) and provides a new level of directness for user interactions. Although our intended contribution is not aimmed at real-time simulations, the approach could be applied to simulations that run in real time.

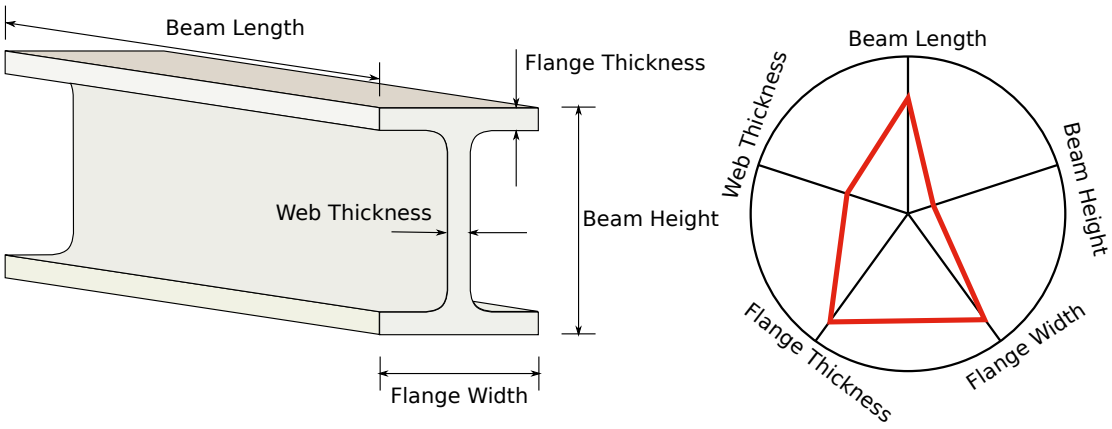

Figure 7.1: Left: A sample parameterized model of a canonical I-beam. Right: The design space implied by the parameters is shown as a wheel plot, with a single I-beam configuration corresponding to the 3D model highlighted in red.

7.1.3 Ensemble Visualization

The problem of visualizing sets of simulation results is an active area of research, including recent applications in engineering design [72, 73], visual effects [68], and disaster management [71, 137]. These systems build upon a design gallery approach and provide a number of significant enhancements to the visualization, including clustering, linked views, interactive brushing, timeline widgets, and queries by example. Many of these features could be combined with and complement our approach; however, our goal is to explore an alternative form of ensemble visualization with direct manipulation at its core. We argue that the directness of the interface makes it possible to do effective ensemble visualization in place, using essentially a single-view interaction and visualization space rather than visual layouts based on small multiples and multiple linked views.

7.2 Definitions

We start by defining consistent terminology that will be used throughout this chapter.

Input parameters. A given simulation design problem contains a set of variable input parameters. The designer selects a value for each parameter and inputs them to the simulation to generate a result. In the simplest case, each parameter represents

a range of continuous scalar values, but this need not be the case. Any higher-level construct that the simulation understands may serve as an input parameter, such as enumerated types or 2D curves describing simulation-relevant behavior. The number of input parameters define the dimensionality of the design space.

Parameterized 3D model. Often one of the inputs to a simulation will be a 3D geometric model (e.g,. in the case of an FEA analysis, a meshed CAD model). Take, for example, the canonical example of an I-beam shown in Figure 7.1. A subset of the the input parameters represents the geometric properties of this parameterized 3D model and explicitly describes its shape; for the I-beam, these parameters are labeled at the left of Figure 7.1. In Design by Dragging, all of the variations for the specific design problem must be captured by the parameters specified for the model.

Configuration. A specific selection for each of the input parameters together define a configuration. A conceptual way to think of a configuration is to treat it as a vector of input parameters. We can visualize this vector, along with with the range of input parameters, using a wheel plot, as shown in the right of Figure 7.1. Although only the geometric parameters that describe the 3D model are shown in this example, in more complex examples, all of the input parameters would be represented on the plot. In this wheel plot, each spoke corresponds to a single parameter, with the minimum value at the center and the maximum at the extent. The red polyline that passes through each spoke indicates a unique configuration vector.

Output fields. As with input parameters, a given design problem has a set of output fields that are computed by the simulation. These fields also will vary in type, from simple scalar values (e.g., the buckling force for the I-beam), to fields across the entire 3D model (e.g., normal stress). A key advantage of the Design by Dragging system is that many outputs can be optimized and considered at once. For example, the I-beam FEA analysis may produce many fields associated with each node in the FEA mesh, along with any number of scalar values that characterize the performance of the design.

Result. Analogous to configurations, the result of a simulation describes a vector of specific output fields, as computed by inputting a configuration into the simulation. As with the configuration, the result can also be represented by a wheel plot. A key feature of the Design by Dragging system is that in terms of both visualization and interaction,

results are treated identically to those in configurations. In many ways, therefore, the differences between configurations and results are often in name only.

Design instance. A configuration combined with a corresponding result constitute a unique design instance. We can think of a design instance as a new vector created by appending the result vector to the configuration vector. The Design by Dragging system works by sampling many design instances and providing direct ways for the user to query and explore these instances via new visualization and interaction techniques.

Simulation codes. The simulation codes are the computer programs that generate results from configurations. There will be one or more codes for each design problem. The codes can range from simple approximation equations that are computed locally to complex coupled FEA and CFD multi-physics simulations that are remotely run on supercomputers.

Parameter and field distance metrics. Each input parameter and output field must have a distance metric defined for it. It is often most useful to use a normalized metric. In the case of scalar values, this corresponds to the normalized distance relative to the maximum and minimum values (i.e., the ranges on the spoke in the wheel plot). For more complex parameters and fields, different metrics can be used (e.g., in a FEA analysis, a sum squared distance (SSD) between all of the node's fields is one option).

Distance between instances. Using the parameter- and field-specific distance metrics, we compute a single distance metric between each design instance. By treating each instance as a vector, a convenient choice for this metric is simply a Euclidean distance composed of the individual distance metrics. By default, each parameter is weighted equally in the distance calculation, but it is possible to change the weighting based on input from the user. Thus, given any two design instances, A and B, each stored as an n-dimensional vector, the distance between the instances is computed as

$$d(A,B) = \sum_{i=1}^{n} \sqrt{w_i * (D(A_i, B_i))^2}, \tag{7.1}$$

Where $D(A_i, B_i)$ is the parameter and field specific function that returns the distance metric for the i-th input parameter or output field and where the weight of the i-th parameter or field, w_i, defaults to 1 if all distance metrics are normalized.

Instance morphing function. In the Design by Dragging framework, as a user navigates between design instances, a smooth transition is displayed. Thus, a function is needed that that linearly interpolates between any two design instances A and B, creating a morph between them. The morphing function must interpolate all of the configurations and results. In practice, there are often an infinite number of possible morphing functions. The ideal morphing functions should avoid discrete jumps and match the user's expectations for how A would transition to B in the most natural way. More complex input parameters and output fields require more complex ways to morph. For example, morphing the length of the 3D I-beam model is very straightforward, but morphing the resulting stress field displayed on the I-beam is much more complicated.

Design space. The design space is the space of all possible configurations. For instance, running a FEA or multi-physics simulation to evaluate the utility of a particular configuration would provide data for just one of an infinite number of possible configurations. What designers would like to do is intelligently fill in this large space, discovering through this process cross-interactions between the parameters and how the input parameters directly impact output fields.

7.3 System Overview

There are five preconditions for applying Design by Dragging to a new simulation-based design problem:

1. A description of the design space, including a list of input parameters and their range of possible values.

2. A parameterized 3D model.

3. Simulation codes that produce results for given configurations.

4. A distance metric for the entire design instance and for each input parameter and output field.

5. A morphing function that can be defined for each pair of design instances within the design space.

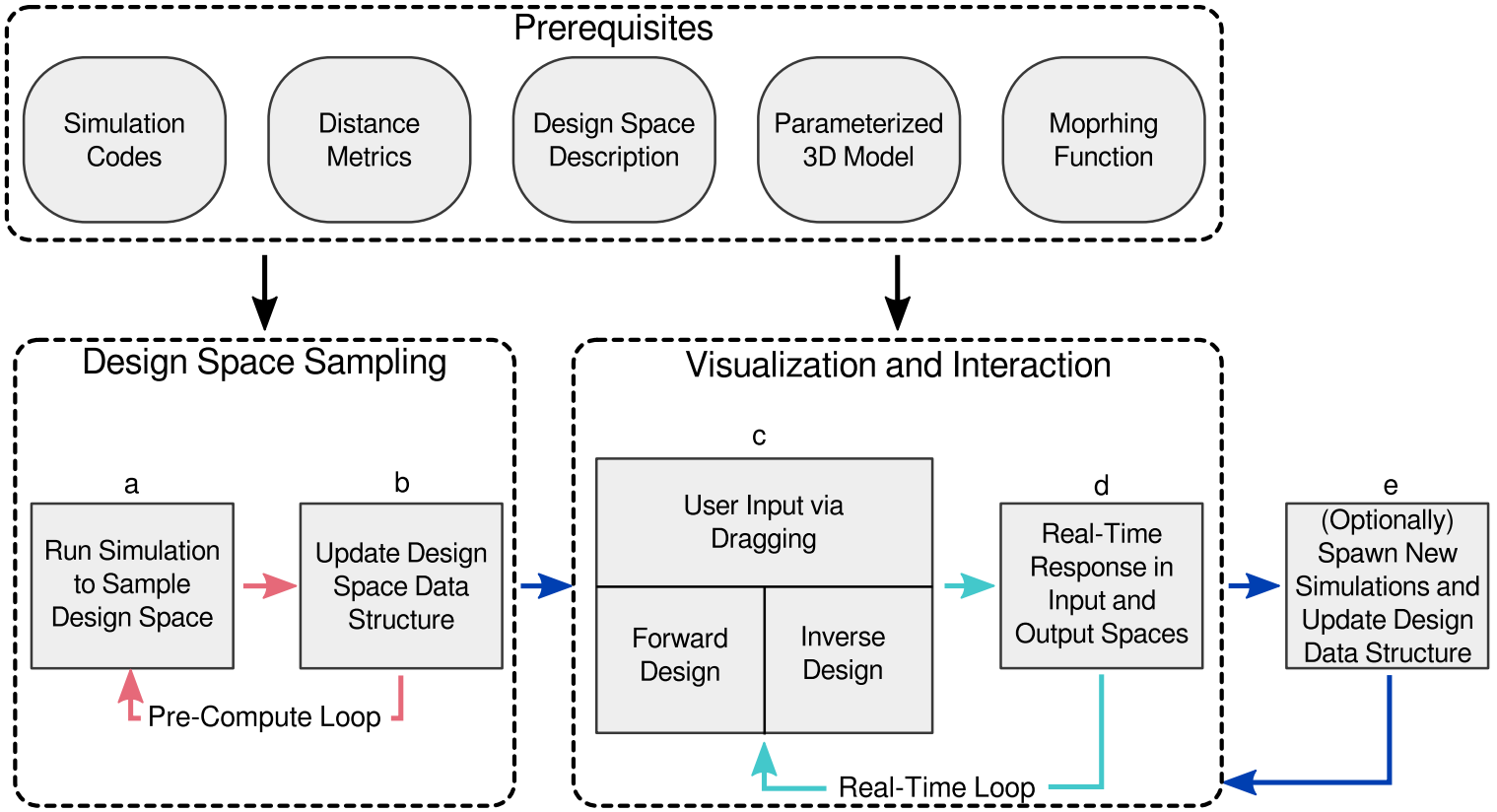

Figure 7.2: Design by Dragging pipeline.

Figure 7.2 provides an overview of the system. Given the above prerequisites, there are two phases of the Design by Dragging workflow. In the first phase, diagrammed in boxes (a) and (b), the design space is sampled by pre-computing a large number of design instances for a variety of configurations. As the design space is sampled, the relationship between each pair of design instances is also pre-computed. This includes both the distance between instances and the information needed to perform a visual morph between them. The second phase, diagrammed in boxes (c) and (d), represents the real-time interactive loop that enables exploration of the sampling data-structure via both forward and inverse direct manipulation queries. In the basic system outlined in this overview, any specific implementation of the Design by Dragging framework will provide data-specific means of performing these queries that depend on the types of input parameters and output fields the designer is exploring.

7.3.1 Design Space Sampling

The specifics of the sampling strategy employed during the first phase outlined above may vary depending on the domain specific design problem. Ideally, a uniform regular sampling of the entire design space would be used, although this is not feasible for higher dimensional design spaces that exponentially grow in size. In these cases, randomized strategies are needed to capture the breadth of the design space. A purely randomized strategy may be augmented with additional information to capture the critical areas of the design space. For example, if the designer has a priori knowledge of which input parameters are particularly important or sensitive, they can be sampled at a higher rate than the other input parameters. The distance metrics between design instances can also be used to guide the sampling strategy during computation to generate a diverse sampling that captures as much variation as possible while using as few samples as possible, similar to the sampling strategies employed in design galleries [67].

As simulation design problems become more complex and the dimensionality of design spaces increases, even advanced sampling strategies will not be able to adequately capture the entirety of the space due to the sheer number of samples required to do so. To address this, the system can optionally add the ability to generate new samples mid-interaction, as shown in box (e) in Figure 7.2. In this approach, the system further samples the area in the design space that the user will explore next. This area is

computed by using artificial intelligence techniques to predict the user's behavior. If the system is connected to high-performance computing resources (i.e., a supercomputer), this horsepower is able to densely sample the predicted region, allowing the user to explore the design space in great detail on demand and dramatically increasing the utility of the system.

7.4 Design Principles

As described earlier, the goal of the Design by Dragging framework is to provide novel ways to naturally explore large simulation design spaces via direct manipulation interactions. There are many possible ways to create these interactions, and implementations will vary largely depending on the input device used or the type of data being manipulated. For example, natural interactions that use mouse-based input will differ greatly from those that use multi-touch input. Similarly, natural interactions for manipulating scalar fields will differ greatly from those for vector fields. To capture the qualities of the Design by Dragging framework, we present a set of interaction guidelines that an implementation of the framework should follow when creating new direct manipulation interactions. We group these guidelines into three high-level categories that capture the most important characteristics of the Design by Dragging framework: directness, simplicity, and continuity. Together these guidelines can be thought of as constituting the design philosophy of the framework.

7.4.1 Design Guidelines

As Direct As Possible

1. *Interactions should always be performed in place.* Whenever a user manipulates a parameter or field, the interaction should be in place within or on top of the visualization of the design instance. For example, if users are manipulating a parameter of the 3D model, they should grab onto the model directly and drag to indicate how the grabbed feature should change or where they would like it to be placed.

2. *If an input parameter or output field has a spatial representation, it should be directly manipulatable relative to that representation.* Any parameter or field that is represented through a spatial element in the visualization should be directly manipulatable via that representation. In the simplest case, this means all the geometric parameters of the 3D model should be manipulated relative to the visualization of the model. More complicated cases should also follow this guideline. Consider the I-beam where one of the output fields is the deflection of the I-beam. Following this guideline, users should be able to grab onto the I-beam directly and indicate the desired deflection of the I-beam by dragging where the user would like the I-beam to deflect.

3. *Abstract input parameters and output fields that lack a spatial representation should have a relevant spatial representation imposed on them.* More abstract input parameters and output fields will often lack a direct spatial representation. In these cases, a relevant spatial representation may be imposed on them in order to maintain a more direct manipulation. For example, if one of the output fields was the manufacturing cost of the device, one possible spatial representation could be a stack of dollars. The user could drag the stack higher or lower to indicate a higher or lower manufacturing cost.

4. *There should be no distinction between the input parameters and output fields in terms of the interaction.* If an input parameter and an output field are of the same type, the interaction with them should be identical. In this way, the two become indistinguishable and designers are able to focus entirely on their goal, be it in the forward direction (manipulating input parameters) or the inverse direction (manipulating output fields). For example, if 2D graphs are used both as an input to the simulation and one of the output fields, the way the user directly manipulates all of the 2D graphs should be the same.

As Simple As Possible

5. *The visual presentation should be minimal and show only the most relevant information at any given time.* When dealing with simulation design spaces, the

amount of potential underlying information is very large (and often infinite). Presenting this vastness to the user often leads to information overload. The major limitation of the popular design galleries [67] approach is the information overload associated with presenting many design instances at once. Minimal visualization of only the most pertinent information is integral to the Design by Dragging framework.

6. *Only one active design instance should be displayed at any given time.* In line with the above principle, only one active design instance will be shown at a time so as to minimize information overload. The Design by Dragging framework instead relies on visualizing the change between design instances, as opposed to viewing many instances at once.

7. *Interactions should be self-revealing and match the user's expectations for how a particular input parameter or output field should be manipulated.* Users should be able to perform the manipulations with little or no prior knowledge of the interactions. This should be accomplished by relying on their expectations for how parameters and fields should be manipulated. For example, to change the length of the I-beam, users expect to be able to grab onto the model and drag it along the lengthwise direction.

As Continuous As Possible

8. *The displayed active design instance should always be either (a) a valid design instance from the sampling data-structure or (b) an interpolation between two or more valid design instances.* The system should display only valid design instances or "uncertain" design instances derived from interpolations between valid design instances. This results in a continuous representation of the design space in which unsampled regions are represented by a "best guess" via interpolation.

9. *When interpolating design instances while interacting, smooth interpolations that avoid discrete jumps should be chosen if possible.* Following from the above principle, the interpolations between valid design instances should be smooth, avoiding discrete jumps entirely if possible. For more complex parameters and fields, this

can be difficult, as a perfectly smooth interpolation may not exist. In addition, there may be multiple (sometimes infinite) ways to interpolate these parameters and fields. Of these, the interpolation that best minimizes discrete jumps should be chosen.

10. *All interactions should be rapid and reversible.* Users should be able to quickly explore design alternatives in real time by dragging. All of their actions should be reversible, encouraging exploration and branching that is easily undone. For example, as users drag to lengthen the I-beam, the closest matching design instance (or design instance interpolation) should be displayed in real time. If users decide to return to the prior I-beam length, they can reverse the operation mid-drag and return to the starting design instance.

7.5 Visual Components and Layout

This section will provide a general overview of the fundamental visual components and layout of the Design by Dragging framework. These components are derived from the design principles described in section 7.4.1. The three base-level components are illustrated relative to the I-beam example in Figure 7.3. In the following subsections, we will describe the basic visualization and interaction function of each of the visual components, with the understanding that the final implementation of the components will vary largely depending on the simulation problem domain (i.e., these components will look very different for a CFD example than for an FEA example like the I-beam).

7.5.1 Active Design Instance

The active design instance component represents the primary visualization in the Design by Dragging framework. This view will display the design instance that corresponds to the user's current location in the design space. It will display both the 3D model and the results of the current configuration output from the simulation codes, as shown by the I-beam and resulting stress field isocontours illustrated in Figure 7.3. As users explore the design space and their current active location changes, the visualization displayed will respond accordingly. As described in the design principles, the active

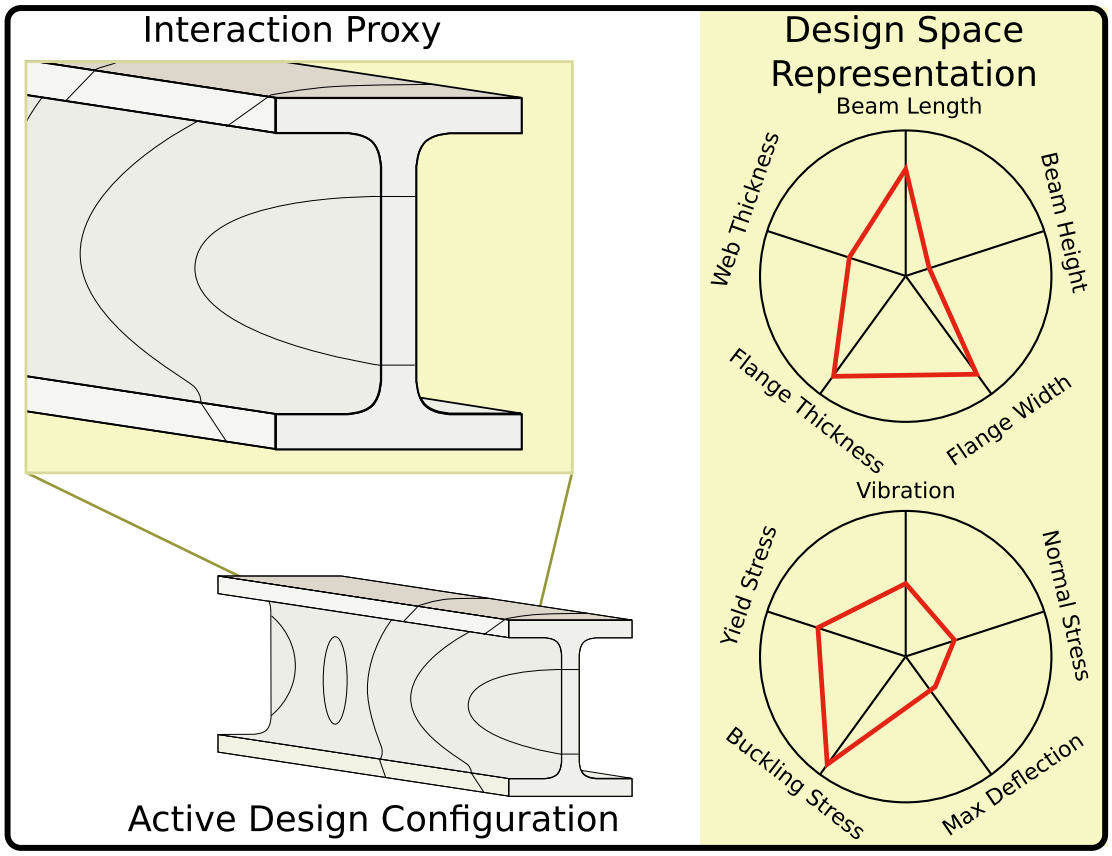

Figure 7.3: Visual components of the Design by Dragging system. Three separate components exist in the core visualization: the Active Design Configuration, the Interaction Proxy, and the Design Space Representation.

design instance will interpolate smoothly throughout this exploration. In this way, the active design instance provides users the ability to understand relationships between input parameters and output fields. The resulting change between configurations is displayed through the smooth interpolation of the design instances. For example, if users lengthen the I-beam, they are able to observe how this changes the resulting stress field by observing the morph of the isocontours.

7.5.2 Design Space Representation

The second fundamental visual component of the Design by Dragging framework is the design space representation, as shown on the right of Figure 7.3. This view provides the

user with an abstraction of the entire high-dimensional design space, using the wheel plots described in section 7.2. Two wheel plots are used here, the top representing the input parameters and the bottom the output fields. The red line in each of the wheel plots corresponds to the active design instance; as the user navigates the design space, the red line interpolates synchronously with the 3D visualization. These wheel plots provide users with the context to identify where they have been, where they are, and where they are going relative to the entire design space.

7.5.3 Interaction Proxy

The final visual component is the interaction proxy, shown in the top left of Figure 7.3. This component provides a secondary view of the 3D visualization representing the active design instance and serves two purposes. First, it uses a callout box to provide a zoomable view of the active design instance, allowing the user to view the 3D visualization in an overview+detail style. Second, and most importantly, it serves as an input space where the user performs direct manipulation interactions. The user drags on top of the 3D model shown in the callout box to initiate queries into the sampled design space data structure.

Performing interactions on top of the interaction proxy introduces a level of indirection to the interaction that appears to contradict the as-direct-as-possible design principle. Ideally, a more direct interaction could be performed on top of a single 3D visualization, but this becomes problematic for more complex parameterized 3D models. These difficulties arise because the user's searches through a sampled design space may cause many input parameters and output fields to change simultaneously. For example, a query requesting an increase in the I-beam's length may result in an increase to both the beam height and the web thickness. This is because a sample may not exist in the data structure that solely increases the beam length. This large change in the other parameters makes direct manipulation difficult to perform and interpret, as the drastic changes in the model will often render the manipulation meaningless when the cursor no longer lies on top of the desired model feature. The interaction proxy avoids this by permitting the proxy to desynchronize with the active design instance. As users interact with the visualization, only the parameter or field they are currently manipulating changes. Once the user stops interacting, the proxy is synchronized with the

active design instance by morphing the remaining input parameters and output fields to match it.

7.6 Example Direct Manipulation of Simulation Input Parameters and Output Fields

The above sections have described the Design by Dragging framework in detail, including the interaction philosophy described through a set of design principles and a conceptual visual layout for systems implementing this framework. Before concluding this chapter, we provide a more concrete yet still illustrative example that brings these concepts together and describes two different representative interactions that follow the Design by Dragging framework. These two examples are shown in Figure 7.4 relative to the I-beam example and demonstrate how similar queries are performed in both the (a) forward and (b) inverse direction. The two queries start with the same design instance and result in the same final target design instance, although how the user navigates between the the starting and target design instances differs largely between the two.

In the first interaction example, shown in (a) at the top of Figure 7.4, the user performs a forward manipulation of the I-beam to explore the design space. Users begin on the left by viewing a starting design instance with input parameters set as indicated by the wheel plot. They then grab the top feature of the beam and drag downward. The system interprets this as the users' requesting a decrease in the beam height and initiates a query through the sampled design space. The design instance distance metric is used to find the closest matching target design instance that exists in the pre-computed sampling data structure. As the user continues to drag downward, following the path indicated by the purple arrow, the system interpolates between the starting and the target design instances. Not only the 3D parameterized model but also the output field showing the stress on its surface so interpolates. This sequence of actions allows the user to ask questions such as, "What happens to the stress field if I decrease the beam height?"

At the bottom of Figure 7.4, (b) shows how a similar query is performed in the inverse direction. The users begin with the same starting design instance, but instead of grabbing onto a geometric feature, they grab onto an isocontour of the stress field, which

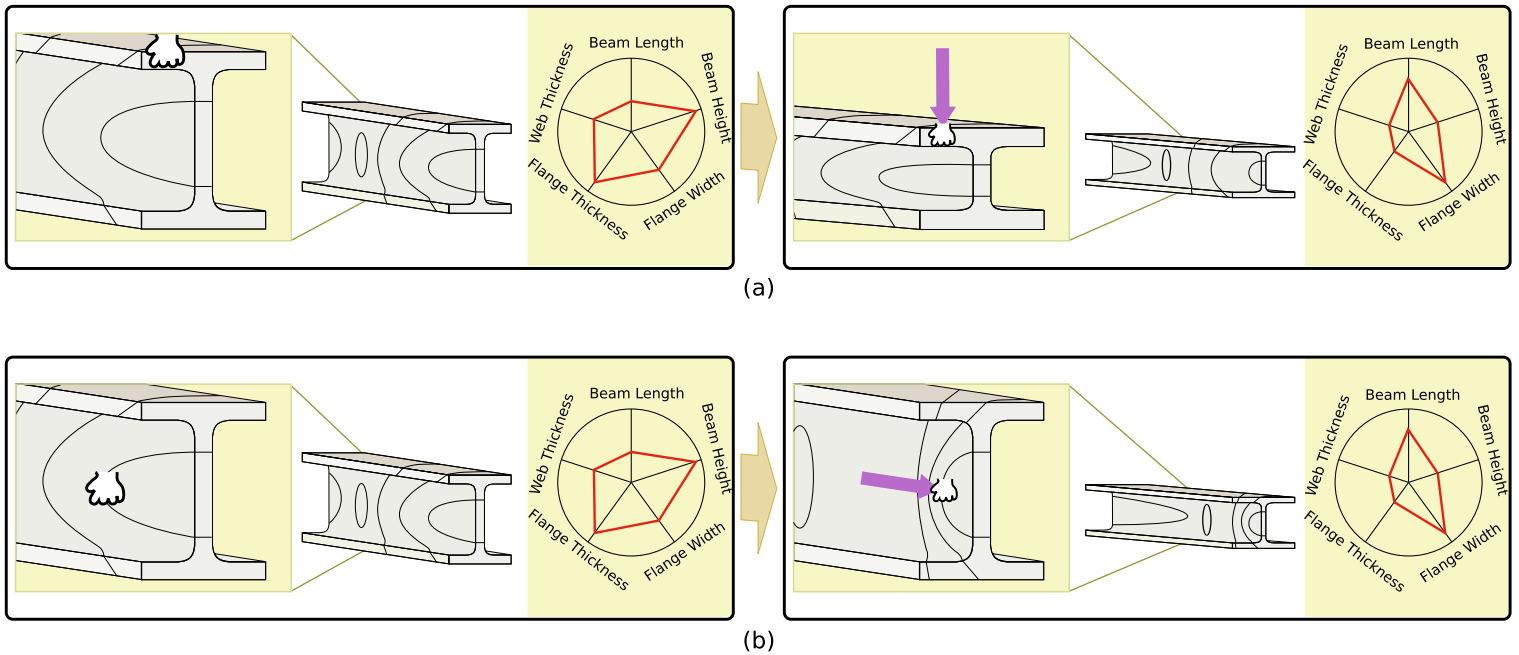

Figure 7.4: Example of forward and inverse direct manipulation.

they then drag to indicate the desired target location of this isocontour. The system searches through the sampling data structure again and finds the target design instance that best matches their manipulation. As the users drag the contour, as indicated by the purple arrow on the right, the two design instances are again interpolated until they arrive at the same target design instance shown in (a). This inverse interaction style allows the user to ask questions such as, "What input parameters are needed to produce a stress field where the isocontour moves here?"

The interaction proxy in both the forward and inverse direct manipulations highlights how the proxy desynchronizes with the active design instance. As shown in the right side of Figure 7.4, only the input parameter or output field that a user is currently manipulating changes in the interaction proxy. In (a), the beam model's height is decreased while everything else stays the same (including the stress field), and in (b), the stress field is moved while the 3D parameterized model remains static. Despite this difference between the interaction proxies, the active design instance is identical and follows the same interpolation throughout the manipulation in the two different examples. The red line in the wheel plot also interpolates along with the active design instance.

7.7 Conclusion

In this chapter, we introduced Design by Dragging, a new framework used for data-intensive exploration of simulation-based design spaces. This framework builds upon direct manipulation, a type of natural user interface, to explore the relationships between input parameters and output fields in complex design problems that make use of simulation. We described a system overview that lays out the basic workflow of a system implementing the framework and the components that are needed to apply it to a given design problem. We provided a description of the design philosophy of the framework through a set of interaction principles and a set of visual components that follow this design philosophy. Finally, we ended with a specific illustrative example of how the Design by Dragging framework would be applied to interact with an I-beam simulation model. This example covers the two types of interactions a user performs with Design by Dragging, forward design and inverse design. In the next chapter, we

will describe a concrete implementation of the Design by Dragging framework and a number of real-world sample applications.

Chapter 8

Design by Dragging Implementation, Applications, and Extensions for Medical Device and Visual Effects Design

In chapter 7, we described Design by Dragging, an interaction framework that enables exploratory visualization of very large simulation design spaces, focusing on the high-level description of the framework and design philosophy that defines it. In this chapter, we provide a detailed implementation of the Design by Dragging framework, including the technical details of the implementation and a number of real-world simulation design problems that have been applied to the implementation.

We are particularly interested in the medical device design process, which is currently poised to take advantage of simulation-based design processes. For example, our implementation applied to a biopsy tissue device can be seen in Figure 8.1. As described in the previous chapter, we support two different types of exploration. In forward design, the designer will sometimes want to drive the exploration in a specific direction by adjusting a geometric parameter of the device (e.g., changing the length of the tissue cutting window, as in Figure 8.1 left). In inverse design, the designer will also want to react to the visualization of simulation results, asking questions such as What causes

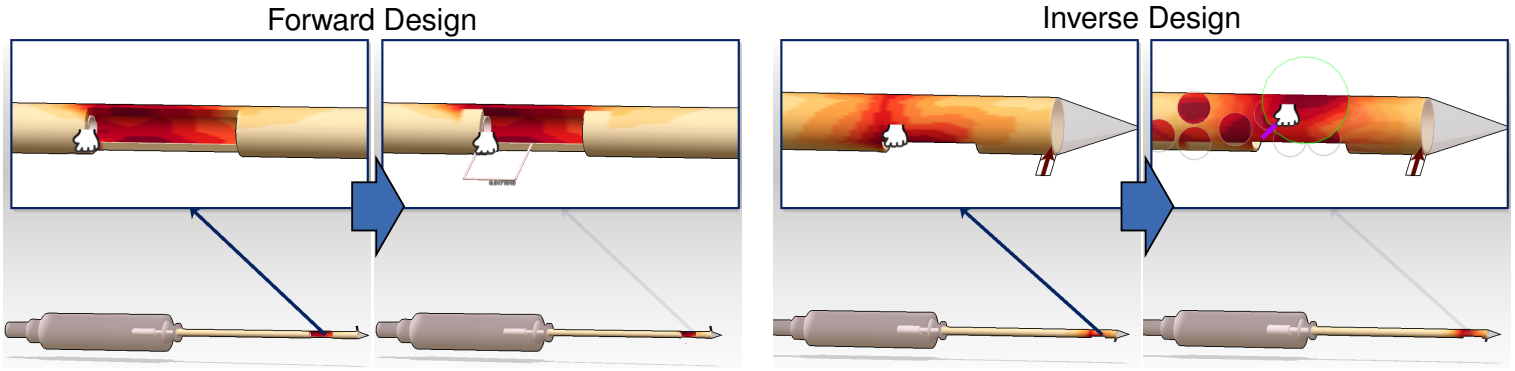

Figure 8.1: In Design by Dragging, designers navigate through a space of hundreds of simulation outputs using direct manipulation interfaces on top of data visualizations. In this example, medical device engineers refine the design of a mechanical biopsy device with two concentric cylinders (cannulas) that slide against each other to cut tissue. In forward design (left), drag operations change the value of simulation inputs, such as the length of the cutting window. In inverse design (right), drag operations directly change the simulation outputs. Users bend and reshape stress fields and the system responds continuously, displaying morphs between the pre-computed simulation results.

this region of high stress, and what would it take to move this region to another location or remove it entirely?

The contributions of this chapter include the following:

- The technical, visual, and interaction details of a specific implementation of the Design by Dragging framework.
- A strategy and data structure to establish correspondences between arbitrary scalar fields output from FEA and CFD simulations and to morph between these fields.
- Algorithms for interpreting the intent of users' drag operations relative to parameterized models and simulation outputs.
- An in-place interactive data visualization for large sets of related simulation results.
- An extension of the core Design by Dragging interface that uses multi-touch input in combination with an as-rigid-as-possible shape manipulation to support rich visual queries.
- A long-term focused evaluation of the approach via an engineering design application to a mechanical biopsy device in collaboration with expert mechanical engineers.
- A second, more exploratory application to visual effects design using a physically based flame simulation to demonstrate how the approach can work with more irregular simulation outputs.

We begin by introducing an sample problem, the tissue biopsy device shown in Figure 8.1, that is described throughout the chapter to explain the Design by Dragging interface in more detail. We then describe the specific morphing function that is used to interpolate between design instances of the tissue biopsy device. Following this, we describe the interactions and visuals used within the Design by Dragging interface, including both forward and inverse direct manipulations and the details of a wheel plot widget that is used to help guide the designers' exploration through the design space.

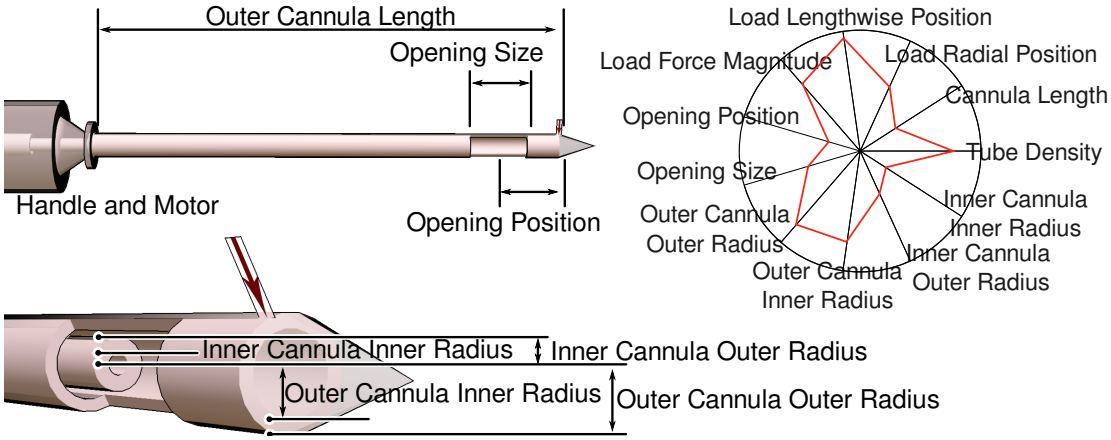

Figure 8.2: Left: A parameterized model that is used as input to Design by Dragging. Right: The design space implied by the parameters, with a single device configuration highlighted in red.

Finally, we end with a number of sample applications, along with a domain-expert evaluation of the interface.

8.1 Sample Problem

The medical device pictured in Figures 8.1 and8.2 is a mechanical system for performing biopsy surgeries. The distal end of the device has a sharp point and a tissue collection window near the tip. It also has an inner cutting cannula with a sharpened distal edge that is intended to slice though tissue pulled through the collection window by vacuum pressure. Two motors located inside the hand piece drive the inner cannula so that it both rotates and translates as it moves across the collection window. The most important geometric parameters of the device are labeled in Figure8.2. The right side of Figure8.2 illustrates the geometric parameters of the device along with additional simulation input parameters, such as the location and magnitude of a load applied to the cannula, using a wheel plot as described in chapter 7. In Design by Dragging, all of the variation within the design space must be captured by the parameters specified for the model.

Balancing the tradeoffs inherent in designing the mechanical biopsy device is challenging. The problem cannot be solved via a strict optimization routine, and the design

space is too large to sample exhaustively. In the specific example of the biopsy tissue device, simulation is used to explore scenarios where a load is applied perpendicular to the main axis of the device. This is an important case to consider because during a breast biopsy operation, a doctor may apply such a load by pushing the device against the ribcage in order to insert the device into difficult-to-reach regions. If the load is too great, the cutting action or another aspect of the design may fail. On the other hand, if the cannulas and/or motors are strengthened, the device may become too obtrusive, leading to a more invasive procedure, to a device that is difficult for the surgeon to balance in his hand, or to other problems. Our goal is to enable medical device designers to gain a better understanding of these tradeoffs and to make better, more informed design decisions. To this end, the biopsy example includes FEA simulations of a number of different scalar fields, including normal stress, Von Mises stress, shear stress, and deflection that characterize the behavior of the device during use.

8.2 Morphing Function Between Design Instances

As described in chapter 7, a morphing function is needed in order to interpolate both the input parameters and output fields. For the input parameters, this is fairly straightforward, since the parameterized 3D model of the biopsy is explicitly defined in terms of the geometric input parameters. Given a function that generates a 3D model from the input parameters, two 3D models are morphed by computing a linear interpolation of each of the geometric input parameters between the two models and by outputting a new 3D model with these interpolated values. In addition, simple rotation, scaling, and translation parameters can be similarly morphed as well. For example, the offset position of the inner cannula may be used as input to a tissue cutting simulation and can be morphed between instances by interpolating the coordinate frames of the parts.

Morphing the output fields of the biopsy device is more challenging. We treat each of the scalar fields as a 2D grayscale image that wraps around the cylinder of the outer cannula. What is needed to morph between multiple 2D images are image warpings, specifically non-rigid deformations that align one scalar field into another. Given these warpings, an image morph is performed by simultaneously deforming the two images

toward each other, while also cross-dissolving the two. This is a challenging problem because the data include both structured quasi-rigid deformations and aggressive large deformations. A variety of algorithms could be used for this purpose, including application-specific algorithms such as a warping algorithm that has been developed to work specifically with imagery of fire [148]. Our implementation uses a single algorithm that generalizes to many different 2D scalar fields, as described in detail in the following section.

8.2.1 Implementation of Scalar Field Morphing

The implementation is based on the elastix [149] image registration framework. A two-step stochastic optimization is used to register the images [150]. In the first step, a non-rigid Euler transformation is applied. A non-rigid B-Spline transformation is then applied using a three-tiered image pyramid to capture image features at multiple scales. Mattes Mutual Information (MMI) [151] is used as the optimization metric, as we found it preferable to the SSD metric for images containing large differences in intensity. We also found that it is important for the warps to be invertible (no foldovers), a property not guaranteed by the B-spline transformation. To account for this, a regularization factor based on "bending energy" is added to the metric to penalize sharp deviations in the transformation. It is also important that the warps between two images, A and B, be inverses of each other (i.e., that $A \rightarrow B$ is as close to the inverse of $B \rightarrow A$ as possible), since a cross-dissolve between both warps is displayed during the morph. In practice, one of the warps is usually better (based on the MMI metric), so the warps are computed first in both directions and the best warp is kept. The other is then recalculated using information from the first in an optimization that computes a warp close to the inverse [152].

In our examples, the procedure takes 15-30 seconds for a single image warp. This is expensive, but given that it is a pre-computation step, we have not yet made a significant effort to optimize the implementation (which is trivially parallelizable). With this said, even after optimization, we believe it may still be useful to be able to reduce the number of warps required so as to save time and disk space when working on large problems. Thus, we introduce a sampling strategy. To reduce the number of required warps, the algorithm shifts from using a fully connected graph of simulation results to a random

regular graph. The random regular graph is ideal because it ensures that each node is connected via a bounded max path length that scales well even with low degree graphs. By computing warps only along the edges of this graph, it is possible to reach every node, with the caveat that multiple warps may need to be composed to do so. After the warps are pre-computed, the algorithm then computes the all pairs shortest paths. In this step, it is important that the weights on the edges are defined such that they account for the quality of the morphs so as to minimize information loss when the warps are composed. We use $e_i = \text{MMI}(e) + c$, where $\text{MMI}(e)$ is the MMI metric for the edge and c is a constant that is larger than sum of the MMI metrics for the longest path in the graph. This ensures that paths of minimum length are selected; in a tie, the path with minimum MMI will be chosen. For time-varying simulations, additional edges are added to the graph to connect results from consecutive timesteps to ensure that continuous navigation through the timesteps is always possible.

8.3 Forward Direct Manipulation of Geometric Parameters

The first form of direct manipulation supported by Design by Dragging is forward design. This enables users to drive the design in a new direction by clicking and dragging on an appropriate location on the parameterized model to specify alternative values for simulation input parameters. To enable this interaction, the system must be able to interpret the intent of users' drag operations, which requires a mapping of the features of the parameterized model to the simulation input parameters. Features such as face, edge, cutout, or other geometric properties of a model can be defined differently depending upon the application. For input parameters that do not have a natural direct mapping to the model geometry, it is often possible to introduce a proxy geometry, such as using a red arrow to indicate the location and direction of a load applied to the geometry during simulation.

Figure 8.3 shows the mapping from features to simulation input parameters for the biopsy device example. Each feature is associated with one or more simulation input parameters that can be manipulated interactively. To discover the user's intent, the system first identifies the feature upon which the user has clicked using raycasting.

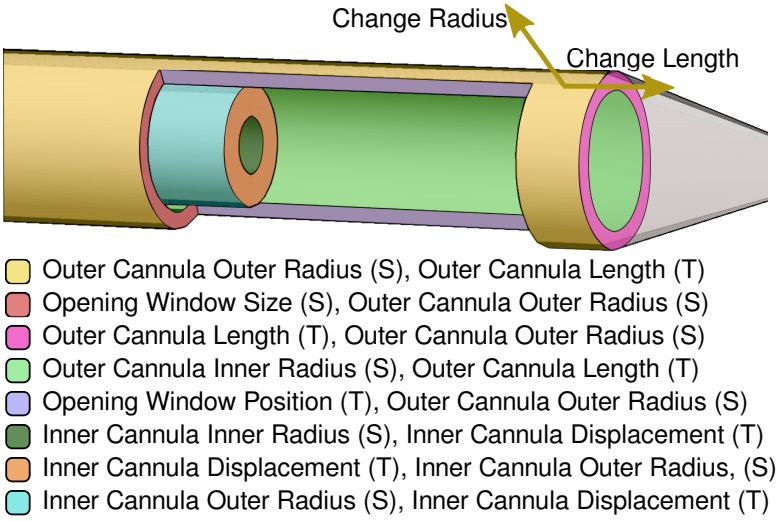

Figure 8.3: Each model has a set of logical features that users can select to manipulate input parameters. Since several parameters might be associated with a single feature, the direction of the dragging motion is used to select the specific parameter to modify (indicated above for a click on the outer cannula).

Then, once the user's cursor movement exceeds a threshold (e.g., 10 pixels) away from the initial click, the parameter to manipulate is set by finding the closest match between the user's motion vector and the ideal motion vectors associated with each parameter, as illustrated in Figure 8.3. For example, after clicking on the outer cannula of the biopsy device, the user might drag along the direction of the axis of the cylinder to lengthen or shorten it, implying a change in the outer cannula length, or drag perpendicular to this axis to imply a change in the outer cannula outer radius. Once the input parameter is identified, the display begins to update continuously in response to the dragging motion, as illustrated in Figure 8.1a-b. The current implementation supports a variety of geometric manipulations (e.g., changing a length, radius, scale, or offset) for features in the model. The features are currently identified manually for each new design problem.

Internally, the system always maintains a notion of a starting design instance, which is the instance that is displayed at the beginning of a user interaction. After the parameter that the user is dragging has been identified, the system then calculates a target design instance, which is the closest instance (based on the Euclidean distance metric

discussed in chapter 7) for which the value of the parameter being dragged matches the direction of the user's movement. For example, if the value of the outer cannula length parameter is 80mm in the starting instance and the user drags in the direction that implies increasing this length, the system will identify the closest instance with an outer parameter length that is greater than 80mm and mark this as the target. Let us assume it finds an instance with length 82mm. The display will update as the user drags to morph between the starting instance and the target instance, so when the user's cursor position indicates a length of 81mm, the visualization will display a morph that is 50% of the way between the starting instance and the target instance. It is important to note that this interpolated visualization involves not just an interpolation of the parameter that is being actively modified but also of all other input parameters and output fields. When the user's cursor finally reaches the position of the target instance, this becomes the new starting instance and drag operations can continue from this point, even without releasing the mouse button.

In some extreme cases, it is possible that the user's input might imply a large change to the geometry of the model. As introduced in the description of the visual layout in chapter 7, the Design by Dragging interface makes use of an interaction proxy to overcome this. The interaction proxy model is integrated into the callout window that displays a zoomed-in view of the visualization (Figure 8.1). This proxy model includes a notion of settling: when the user completes an interaction, the proxy model gradually morphs into the current working model to rectify any changes.

In the mouse-based implementation, all of these forward manipulations are controlled using mouse click and drag events within the zoomed-in view of the results, and inverse manipulations (described next) are controlled via right mouse clicks. Unicam-style camera manipulations [138] are controlled using clicks and drags with either button within the main visualization window.

8.4 Inverse Direct Manipulation of 2D Scalar Fields

The second method of direct manipulation design in Design by Dragging is to tug directly on the visualization of simulation results to navigate within the design space. As introduced in the related work section, this aspect of the interface is inspired by

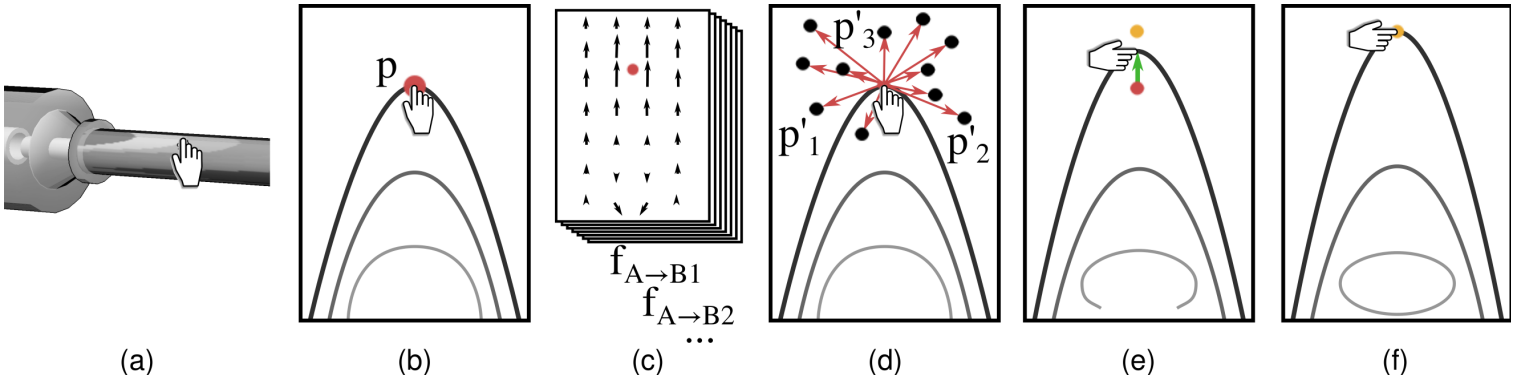

Figure 8.4: The interactive algorithm for transitioning from one instance to the next as the user drags a cursor consists of a series of steps. The 3D cursor position is first projected into the 2D image space of the simulation scalar output fields. Then the user's cursor movement is compared to the visual change that would occur in the data visualization during a morph from the starting instance to each possible target instance. When the best match is identified, the visualization itself is drawn as a morph controlled interactively by the user's cursor.

systems that exhibit an almost magical sense of direct manipulation. We introduce a style of as-direct-as-possible interaction that is similar but uses direct user manipulations on top of data visualizations.

8.4.1 Transitioning Between Instances

Our goal is to smoothly transition from one valid instance to the next most logical choice as the user interactively drags his cursor on top of the data visualization to tug it in a new direction. The strategy we introduce is to compare the motion of the user's cursor on top of the data visualization to the visual motion the visualization would undergo if it were to morph from the current instance to another instance. As the user drags the cursor, the algorithm repeatedly calculates the closest match between the user's motion and motion implied by the visual morph needed to reach possible target instances. The best target is identified, and the visualization updates in real time as the user continues to drag. The key to enabling this interface is knowing the transition from each simulation output to the others. As described in section 8.2, we know this information from the pre-computation data structure that stores the image warps between all pairs of output fields.

Given two images A and B, an image warp can be thought of as a function $f_{A \to B}$ that maps each 2D pixel location p in an original image to a new location p' in a second image, such that the transition between image A and B looks as smooth and natural as possible. We can write this as

$$p' = f_{A \to B}(p). \tag{8.1}$$

Note that image warping algorithms compute f at *every pixel location* in the image. Thus, f tells us where every pixel in image A will move during a smooth visual transition from A to B.

Figure 8.4 diagrams the algorithm for using image warps to transition between instances. Moving from left to right within the figure, in (a), the user has just clicked on a point of interest within the 3D visualization. From (a) to (b), the location of the cursor has been projected from the 3D geometry into to same 2D image space where the scalar field warps have been computed. The image space pixel that is closest to the cursor's position is then identified and labeled p. This is the cursor's pixel position at

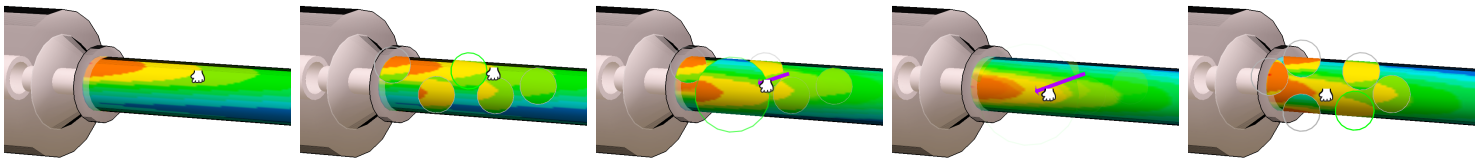

Figure 8.5: Inverse design using a single point manipulation: (a) the user begins by clicking on a point of interest, (b) preview bubbles are displayed, (c) the user drags toward an interesting preview, (d) the new simulation result morphs into place, and (e) a new set of preview bubbles is generated.

the *start* of the drag operation. At (c), the database of pre-computed image warps is used to evaluate the warping function at pixel p for every possible target instance. In other words, given the current starting instance, A, the function $f_{A \to B}(p)$ is evaluated for all possible target instances $B_1, B_2, B_3, ...$ within the design space. At (d), notice that this results in a set of new pixel locations $p'_1, p'_2, p'_3, ...$ that describe how the pixel under the user's cursor would move according to each possible image warp. Continuing with the user interaction, at (e), the user has begun to drag the cursor, and the cursor's movement vector (shown in green) is projected into the image space. The best possible target instance (highlighted in yellow) is identified using a formula that compares the green cursor movement vector to the red vectors shown in (d). Finally, at (f), notice that as the user continues to drag the cursor toward the target p', the visualization updates so that when the cursor has reached the target p', the data display has morphed to display the data for the target instance.

There are some important aspects of the formula used at stage (f) in Figure 8.4 that are worthy of note. In addition to picking a target instance for which p' is in line with the cursor's motion vector, it is also important to give preference to target instances that are close to (i.e., in the local neighborhood of) the starting instance. To make sure this is the case, the target instance is selected by finding the instance for which the dot product of the two vectors pictured in Figure 8.4 is greatest *and* the "distance" (calculated using Equation 7.1) between the starting instance and the possible target instance is smallest. Each of these two considerations is weighted equally.

The algorithm is also tuned experimentally to include some hysteresis. During a drag operation, once a target instance has been identified using the strategy above, the user must make a definitive move toward another instance in order to snap the interface away from the current target and orient toward a new target.

The result of this strategy is an interface with immediate feedback. Users can drag toward one target, see via the morph what this would do to the simulation outputs, and then, if they wish, change their mind, back up, and drag in another direction. As in forward dragging, once the cursor comes within a reasonable radius of the target, the target instance becomes the new starting instance and the algorithm proceeds fluidly. When using a mouse, the button does not need to be released and re-clicked to continue dragging from this new starting point, so from the user's standpoint, the transition is

seamless.

8.4.2 Single Cursor Manipulations

When interacting using a mouse or single touch, Design by Dragging makes use of a technique we call preview bubbles to provide a view of many possible design alternatives before beginning to drag. The benefit of these previews is that they provide an immediate visual suggestion of possible dragging operations, something that can encourage exploration of the design space and perhaps suggest new alternatives to the designer. The bubbles are displayed only in the callout window so that an uncluttered visualization of the output field is always visible in the main display area.

The preview bubbles fade into view immediately after the initial cursor down event to provide an in-place visualization of how the output scalar field would change if the user dragged in the direction of the bubble and released the cursor at the its center. The algorithm first identifies possible target instances as described above, then ranks them based upon the Euclidean distance metric, with the closest target instances first. Preview bubbles are then placed on the screen in the order of the ranked list. For each target instance, a bubble is placed at the screen space projection of its p' point as long as the bubble would not overlap with any bubbles that have already been placed. The specific number and size of the bubbles is adjustable via a parameter that we have set empirically for our applications.

Figure 8.5 shows an example of the interface, including the visual feedback during the drag operation. Notice that the target bubble enlarges and then gradually fades from the view as the user's cursor moves toward it and the underlying visualization morphs into a complete display of the target instance. When the cursor reaches a position within a small threshold of the center of a bubble, the display snaps to the new target instance, and the interface is restarted with a new set of preview bubbles. One interesting aspect of the interface is that since the radius of the bubbles is constant in screen space coordinates, the user can adjust the number of bubbles to display by zooming in or out.

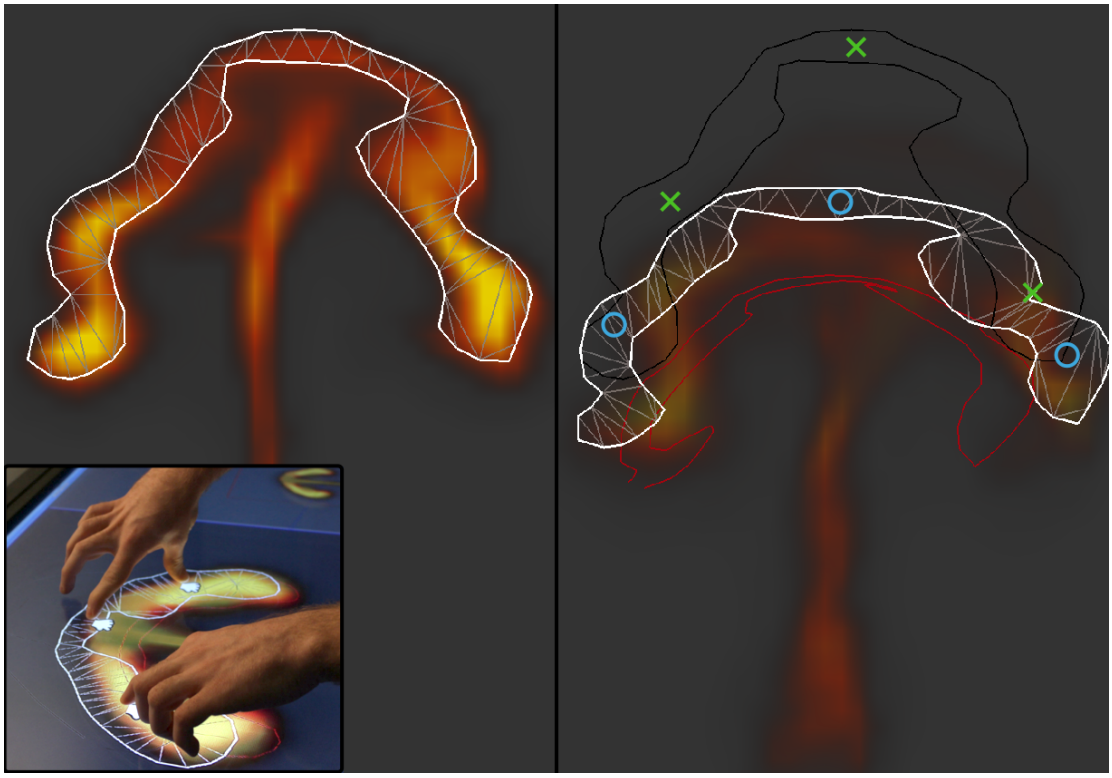

Figure 8.6: Left: In a multi-touch inverse design, a control mesh is generated interactively when the user selects an iso-contour of the simulation data. Right: The mesh is reshaped based on simultaneous input from multiple fingers to create a detailed inverse query into the result space. This example comes from the flame simulation described in section 8.6.2.

8.4.3 Multi-Cursor Manipulations

Multi-cursor (i.e., multi-touch) manipulations are most useful for global adjustments to the simulation outputs. Multiple cursors provide a richer input stream that enables more complex manipulations of the output space. Since these manipulations operate over a larger area, it is important to have specific feedback on how the control mesh that covers this area is moved, and this is the key visual feedback shown to the user within the interface. Preview bubbles are not used in this case; they would be confusing here because a separate set of preview bubbles would be needed for each cursor.

The manipulation begins with an initial touch and release on the scalar field visualization displayed in the callout window. The scalar value at this location is used

to extract an iso-contour from the scalar field. The inside of the largest connected component of the contour is then triangulated to create a control mesh, as shown in Figure 8.6. This mesh is manipulated via natural multi-touch interactions using the as-rigid-as-possible shape manipulation algorithm [153, 140], which uses a real-time optimization to intuitively preserve the shape of the mesh given the deformation constraints implied by each of the control points at the user's fingers. As usual, the system continuously updates and displays an interpolated result during the manipulation.

During multi-touch manipulations, the target instance is selected using the same strategy of comparing the cursor's movement during the drag operation to the image warps. The difference is that rather than a single cursor with a single motion vector, each vertex of the control mesh is treated as a virtual cursor. So, if there are 100 vertices in the control mesh, $f_{A \to B}$ will be evaluated 100 times for each candidate target instance B, and these values will be compared to the 100 motion vectors that describe how each vertex of the control mesh moved when the as-rigid-as-possible algorithm was applied.

The result of this strategy is that with a few fingers, the user can specify a much more sophisticated change in the shape of the scalar field than can be done with a single mouse cursor. This change is propagated intuitively beyond the reach of the fingers because the as-rigid-as-possible algorithm makes the control mesh deform in an intuitive manner. Since the target instance is selected based on the motion of the entire control mesh, this process can take into account even global changes to the scalar field. This gives the user a powerful way to quickly navigate the design space.

8.5 Wheel Plot Widget

As introduced in chapter 7, a wheel plot is used to represent the design space. Beyond this representation, the wheel plot also serves as a widget that can be used to guide the interface. Figure 8.7 shows our wheel plot widget implementation. At its core, this approach, in which we visualize the simulation design space and one spoke is used per input parameter, is very similar to that Bruckner and Möller [68]. In the following sections, we first describe the visuals used in the wheel plot. Following this, we will describe how the user is able to interact with the widget to guide his navigation. Finally, we end with an extension to the wheel plot widget that enables it to represent

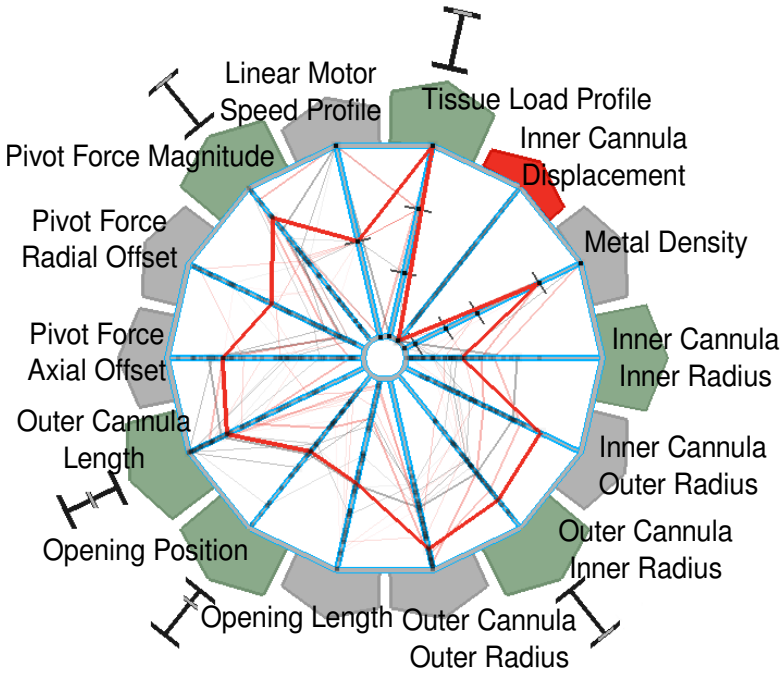

Figure 8.7: A parameter space wheel widget enables designers to guide the interface by pinning or weighting parameters.

multiple levels of detail for simulation design problems that make use of different scales of computation.

8.5.1 Visualization of Design Space

The length of the wheel plot represents the range of each parameter, with the maximum at the radius of the wheel and the minimum at the center. A bold red polyline indicates the user's current location in the design space. In addition to this line, we add an interactive visualization of the local neighborhood of the design space; when the user first begins a dragging operation, a set of black polylines are displayed to indicate the closest neighboring instances. Transparency is applied to make the closest instances stand out and those that are farther away fade into the background. Likewise, the history of the design space exploration is also visualized using transparent polylines, this time in red. Small dots along the axes are placed at every value for which a sample exists in the design space, and tick marks are used for parameters that take only discrete

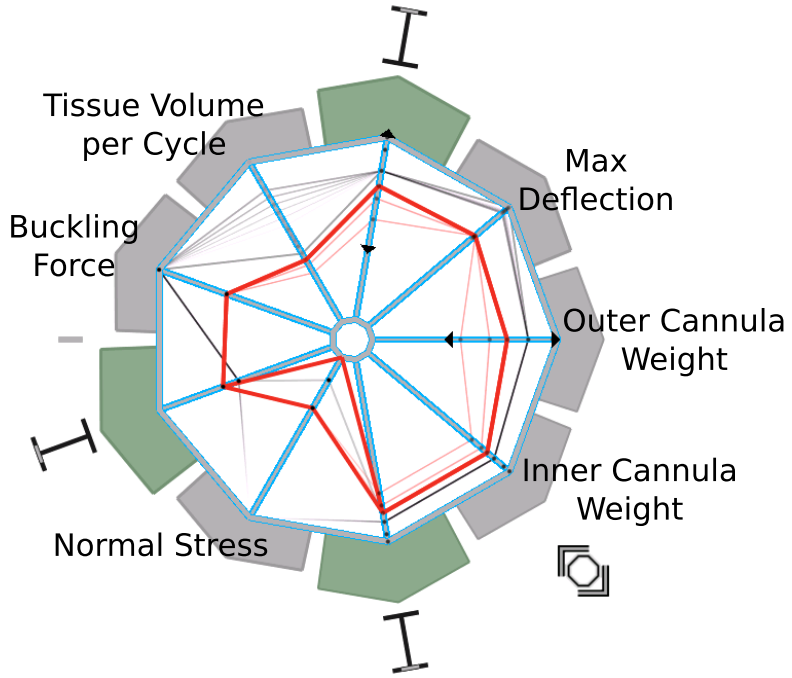

Figure 8.8: Output field wheel widget.

rather than continuous values, such as wire that comes in only certain gauges.

The output fields can also be represented by a wheel plot widget. In our implementation, a second wheel plot is added that represents all of the output fields. The simulation result of the current design instance is shown using a bold red line as well. This works easily for output fields that are scalar values (e.g., the volume of tissue extracted per cycle), but often there are output fields that are not directly representable by a single scalar value. For example, the stress fields used in the biopsy device example represents a set of many scalar values. Different representations can be used to show this as a spoke in the wheel plot, but the one we use is to show the minimum and maximum values of the current stress field by splitting the red polyline into two as it passes through the spoke corresponding to the stress field, as shown in Figure 8.8. This figure also highlights one final capability of the wheel plot widget. Using the mouse wheel, the user zooms into a subwindow on an individual spoke, as indicated by the arrowheads that bookend two of the spokes. This is useful for differentiating close samples when outliers extend the range of the spoke.

8.5.2 Control for Guiding the Interface

Recall that the following Euclidean distance metric is used to find the closest design instances by comparing two instances A and B, where each is thought of as an n-dimensional vector:

$$d(A, B) = \sum_{i=1}^{n} \sqrt{w_i * (D(A_i, B_i))^2}, \tag{8.2}$$

Where $D(A_i, B_i)$ is a parameter and field-specific function that returns the distance metric for the i-th input parameter or output field and where the weight of the i-th parameter or field, w_i, defaults to 1 if all distance metrics are normalized.

The sub-elements of the vectors in this metric may be weighted to guide the navigation. These weights can be changed using the wheel plot widget with a tab located at the end of each spoke. Each tab has three quick states that set the values for w_i used in Equation 8.2. These states are pinned (the weight is infinite, so the parameter will not vary during any interaction), normal weighting (the parameter is weighted 1, equal to all others), and free (the parameter has zero weight; it might vary dramatically even in results that are considered close to each other). Alternatively, a small slider at the end of each tab can be activated via a double-click, and a weight between 0.0 and 1.0 can be assigned using the slider. In practice, non-uniform weights are used most often in the quickset mode to limit the design space for constrained exploration. For example, by pinning the parameters for the load applied to biopsy device, the designer can explore the subset of design possibilities that have been simulated for a specific loading condition.

8.5.3 Levels of Detail for Scales of Computation

The wheel plot widget can be extended to overcome one limitation of the Design by Dragging interface. This limitation relates to the continued acceleration of computing horsepower and simulation complexity. Designers of simulation-based applications are always going to need to work with the most state-of-the-art simulation. These simulations will continue to advance and will always push the boundaries of the current class of supercomputers. The Design by Dragging interface works by densely sampling the design spaces, which is inherently impossible with these advanced simulations because

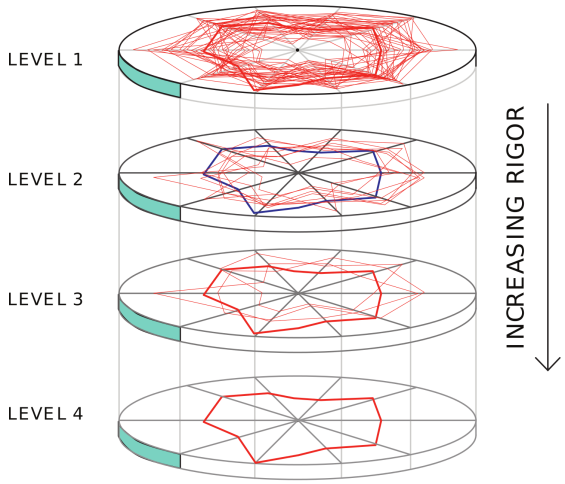

Figure 8.9: Different levels of detail, or computation rigor, are encoded into 3D by turning the wheel plot on its side and extending it into a cylinder. At one extreme, computationally cheap design instances are densely sampled. At the other end, computationally intensive design instance are sparsely sampled.

of their expensive computation cost. The Design by Dragging interface can overcome this limitation and work with state-of-the-art, computationally expensive simulations by introducing the concept of levels of detail into the design space.

Many simulation-based design problems are computable at different levels of detail or rigor. This is often possible by making simplifying assumptions about the problem domain. The simulation may be run at a lower resolution, or even use explicit equations to approximate the solutions. For example, in the tissue biopsy device, one way this is possible is by using cylindrical beam equations to approximate the deflection and normal stress on the outer cannula.

The wheel plot widget is extended to accommodate these levels of detail by turning it on its side and extruding it into a cylindrical representation in 3D, as shown in the concept sketch in Figure 8.9. At the top of the cylinder, the low-cost (often taking only seconds) computations are densely sampled, and at the bottom, the high-cost (often taking days or weeks) computations are sparsely sampled. Using this representation, the designer is able to explore the design space fully at lower levels of detail so as to explore hypotheses before drilling down into the higher levels of detail results to confirm their findings.

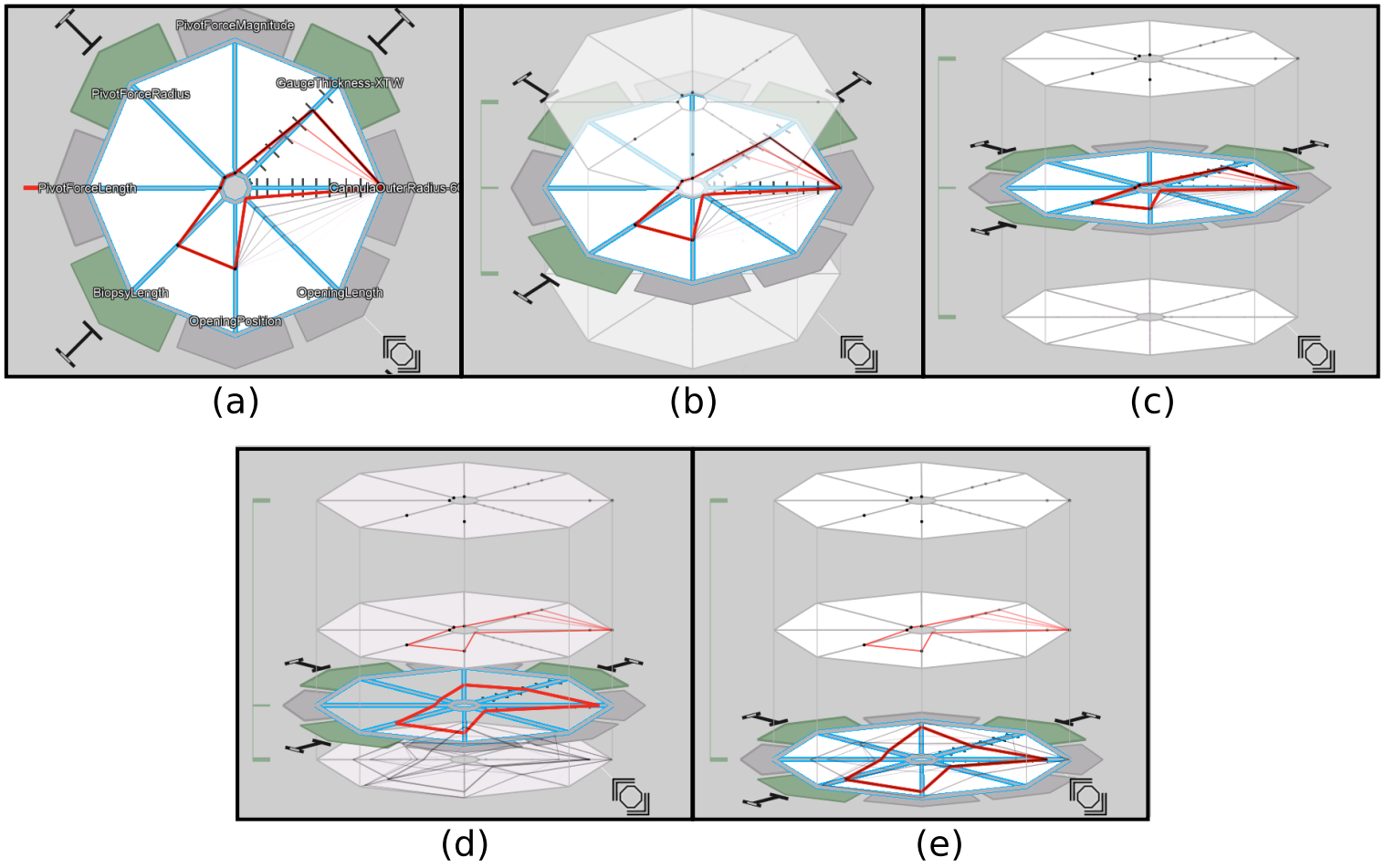

Figure 8.10: Wheel plot extended to represent different levels of simulation detail.

Figure 8.10 shows the levels of detail extension implemented in Design by Dragging. The vector representing the design instance is extended by one element to include the "level" of a specific instance. Similar to the other elements of the distance metric, the level may also be weighted to control the navigation across levels. As shown in the Figure 8.10 (a), users start in "flat" mode, with the weight associated with the level set to infinite so they cannot navigate between levels of detail. In (b) - (c), users rotate to 3D mode, exposing the additional levels available and triggering an automatic change in the weight that allows them to navigate across levels. In (d) - (e), after the widget is in 3D mode, the user grabs the left handle and drags down to manually change the level, triggering a query and interpolation between two design instances, each belonging to a different level of detail.

8.6 Applications

We have applied the Design by Dragging system to two separate domains. The first is the medical device design domain that has been used throughout the chapter, specifically the highlighted biopsy tissue device. The second domain is artistic visual effects design, specifically fluid simulations of fire. The following sections detail our development of a number of applications for both of these domains. Table 8.1 presents a summary of all the relevant statistics for each of the datasets that will be described in the following sections.

8.6.1 Application to Medical Device Engineering

The research described here has been performed collaboratively with a team of mechanical engineers who focus on medical device development. The senior mechanical engineer on our team has more than 40 years of experience in mechanical engineering and has consulted for more than 50 companies on mechanical and product design. The specific biopsy device application described here is a real problem from his past experience, and thus we are confident that it is accurately representative of the types of design problems that engineers struggle to address using today's current design tools. The interface and application have been iteratively evaluated by the team of engineers in regular weekly design meetings for more than 2 years.

Dataset	Input Parameters	Resolution	Configurations Sampled	Graph Degree	Warps Computed
Biopsy Beam Eq., Normal Stress	14	100x200	1,500	22	33,000
Biopsy FEA, Von Mises Stress	14	100x200	200	100	20,000
Biopsy Tissue FEA, Von Mises Stress	4	150x300	1596(76x21)	75	119,700
Flame Point Emitter, Temperature	10	40x60	600 (60x10 timesteps)	100	60,000
Flame Ring, Temperature	12	100x130	240 (30x8 timesteps)	240 (fully connected)	57,600

Table 8.1: Summary statistics for datasets used in both applications.

In this section, we provide three separate applications that use the biopsy device problem. These three applications increase in complexity sequentially, and each highlights different aspects of the Design by Dragging system. In addition to providing real-world applications of the system, these examples also serve as an evaluation of the various components of the system and their capacity to handle the complex design challenges associated with these design problems. Beyond these applications, we also report on direct evaluative feedback regarding this work. Our goal in working in this domain has not necessarily been to create a radically improved design for this specific device but rather to use a familiar problem to demonstrate the potential to design in a way that is fundamentally different than current approaches.

Biopsy Device with Load Beam Equation

The first application analyzes the stress distribution on the outer cannula under the loading condition described in section 8.1. The force load location is parameterized and is free to both rotate around the biopsy and change its offset distance from the tip. In addition to the forward manipulation of the geometric input parameters, the user is able to grab the stress field and manipulate it in the inverse direction, as shown in data displayed in Figure 8.5. This example uses an approximation of the load-induced normal stress computed via a cylindrical beam equation that does not account for the window of the biopsy device. Due to the low computation cost of the beam equation, the normal stress field is able to be calculated at run time.

This application example is the simplest of the three presented for the biopsy device. The beam equation produces a stress field that is very structured, resulting in few singularities and similar features that exist among all design instances. As a result, there is typically a single morph for each pair of stress field images that can be considered the "correct" morph. This makes this example ideal for evaluating the correctness of the non-rigid deformation techniques, as all of the computed warpings should match the expected deformation. Figure 8.11 shows one of these image morphs in more detail. The stress field highlighted in green belongs to the starting design instance and the one highlighted in blue to the target design instance. The first two columns show four steps of these two fields as they are warped toward each other. The far right column shows the combination of these two warps with cross-fading to compose the final morphed

image. We can also see that the morph is working as intended in the interaction shown in Figure 8.5. Notice that in part (a) of this figure, the users have clicked right on a contour edge between the yellow and green regions of the scalar field. As they begin to drag and the preview bubbles appear, they can see a number of possibilities for where they could drag this edge in order to arrive at a new instance with a different scalar field.

In order to make this example work, we had to extend the Design by Dragging tool to handle wrapping the output stress field around the 3D geometry of the device. Since the outer cannula is cylindrical, there should be no visible seam where the edges of the wrapped image meet on the 3D geometry. To accomplish this, during the warp calculations, the scalar field image is mirrored and tiled on both sides of the dimension that is wrapped around the cylinder to create a larger image. This allows the warps to be computed smoothly across the edges of the original scalar field. The image is then cropped down to the original size before texture mapping it onto the cylinder for visualization. During drag operations, drags that cross the texture boundary are identified and handled as a special case. You can see the resulting circularity in Figure 8.11; as the high stress feature moves across the image boundary, it wraps to the other side in a continuous warp.

The specific details and statistics of this dataset are summarized in the first row of Table 8.1. This dataset used a random sampling strategy of valid configurations based on the following geometric input parameters for the outer cannula: inner radius, outer radius, length, cutting window position, and cutting window length. In creating the datasets for these applications, there is an obvious tradeoff between the pre-computation time required and the size of the dataset, including completeness of the graph data structure used. When composing image warps together in sequence, a loss of information is associated with each individual warp. Thus, the quality of the final image morph is entirely dependent on the quality of the warps that compose it. In this application, the more structured nature of the field resulting from the beam equation allows a very low graph degree to be used, as very little information is lost with each individual warp. For the computation of this data structure, we decided we were willing to wait about 1 day to compute the data, which we used as a general guideline for setting the number of configurations to sample and the degree of the regular random graph. The

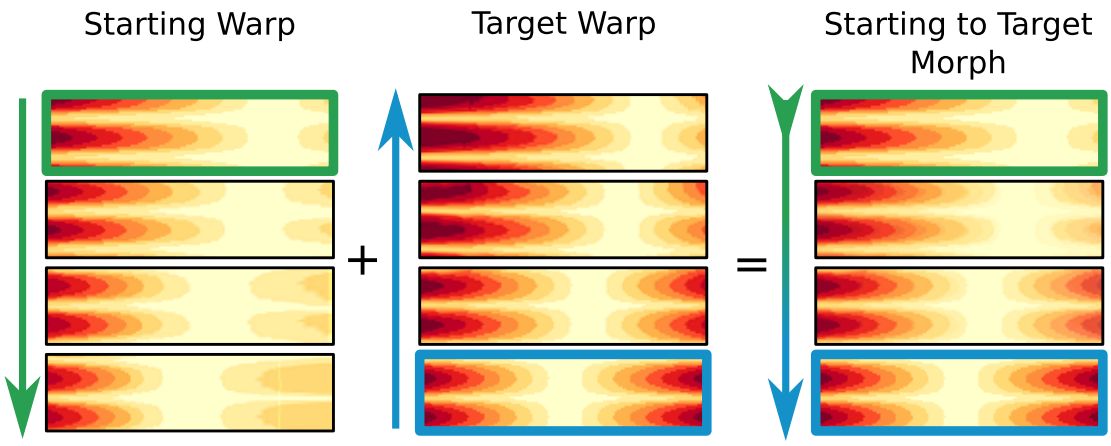

Figure 8.11: An sample image morph between two normal stress fields that result from the computation of a beam equation representing a perpendicular load applied to the biopsy device. Note the circular aspect of the morph; as the image is deformed across the boundary, the image wraps due to the cylindrical nature of the biopsy device

pre-computation took roughly 24 hours to complete on a Linux machine with 24 cores.

Biopsy Device with Load FEA Analysis

The second application is a more advanced and accurate model of an analysis similar to that described in the first application. While it also computes a stress distribution caused by the same perpendicular load, the stress field results from a full FEA simulation of the load condition on the biopsy device. The Von Mises stress is computed and the resulting mesh is resampled to produce a 2D image that is wrappable around the outer cannula. The location of the force load is fixed at the tip of the biopsy device to target a more constrained subset of the design space than that used in the first example.

A snapshot of the complete graphical user interface of Design by Dragging with FEA Von Mises stress data loaded in it is shown in Figure 8.12. In addition to providing a more realistic use case, this example also demonstrates the successful application of the system to a more complicated output field. The FEA analysis of the biopsy device highlights the area of high stress near the collection window. Stress fields in this example are more unstructured than those resulting from the beam equation, and thus represent a more challenging image morph to compute. Despite this, the non-rigid deformation technique described in section 8.2.1 is able to produce accurate warpings that transition

between all pairs of scalar fields, as shown on the right of Figure 8.1. Here users grab the high-stress area near the corner of the collection window. Following this, the set of preview bubbles pop up and give them a sense of the local region in which this stress field is capable of moving. Finally, after dragging toward one of the bubbles, the user is able to inspect a device where the area of high stress has moved away from the corner of the collection window on the backside of the cannula.

This example also highlights an extension to the basic Design by Dragging system: the ability to use 2D plots as additional inputs and outputs. In this case, these plots are used to estimate the linear motion of the inner cannula as it cuts through the tissue sample. These plots are visible on the bottom of Figure 8.12, where the arrow pointing to each plot indicates what geometric component it is associated with. Two of the plots, which can be experimentally gathered, define the motor force and tissue cutting characteristics. A third plot is outputted by integrating through the two input plots, representing a motion profile for the displacement of the inner cannula as it cuts through the tissue. This motion profile is in turn used to compute additional characteristics of the design, including the time to complete one cutting cycle and whether the motor stalls in the middle of its cycle. In keeping with the core design philosophy of the tool, these 2D plots are directly manipulatable and the user is able to grab the curves and drag them around to explore alternatives choices.

A detailed summary of statistics for this dataset is shown in the second row of Table 8.1. We generate a random sampling of 200 valid configurations. The Von Mises stress for each configuration is simulated using the Dassault Systmes Abaqus solver and a tetrahedral mesh of the outer cannula. Due to the more unstructured nature of the resulting scalar field, a high graph degree is used to ensure a short minimum path length for composing multiple warps. As with the first dataset, these numbers were chosen to set the amount of pre-computation time. It took roughly 24 hours to complete the pre-computation of this dataset on a Linux machine with 24 cores.

Biopsy Tissue Cutting Analysis

The third application explores a different type of analysis, focusing on the tissue cutting aspect of the device rather than the loading condition on the outer cannula. It is the most realistic and complicated model of the three biopsy device models presented in

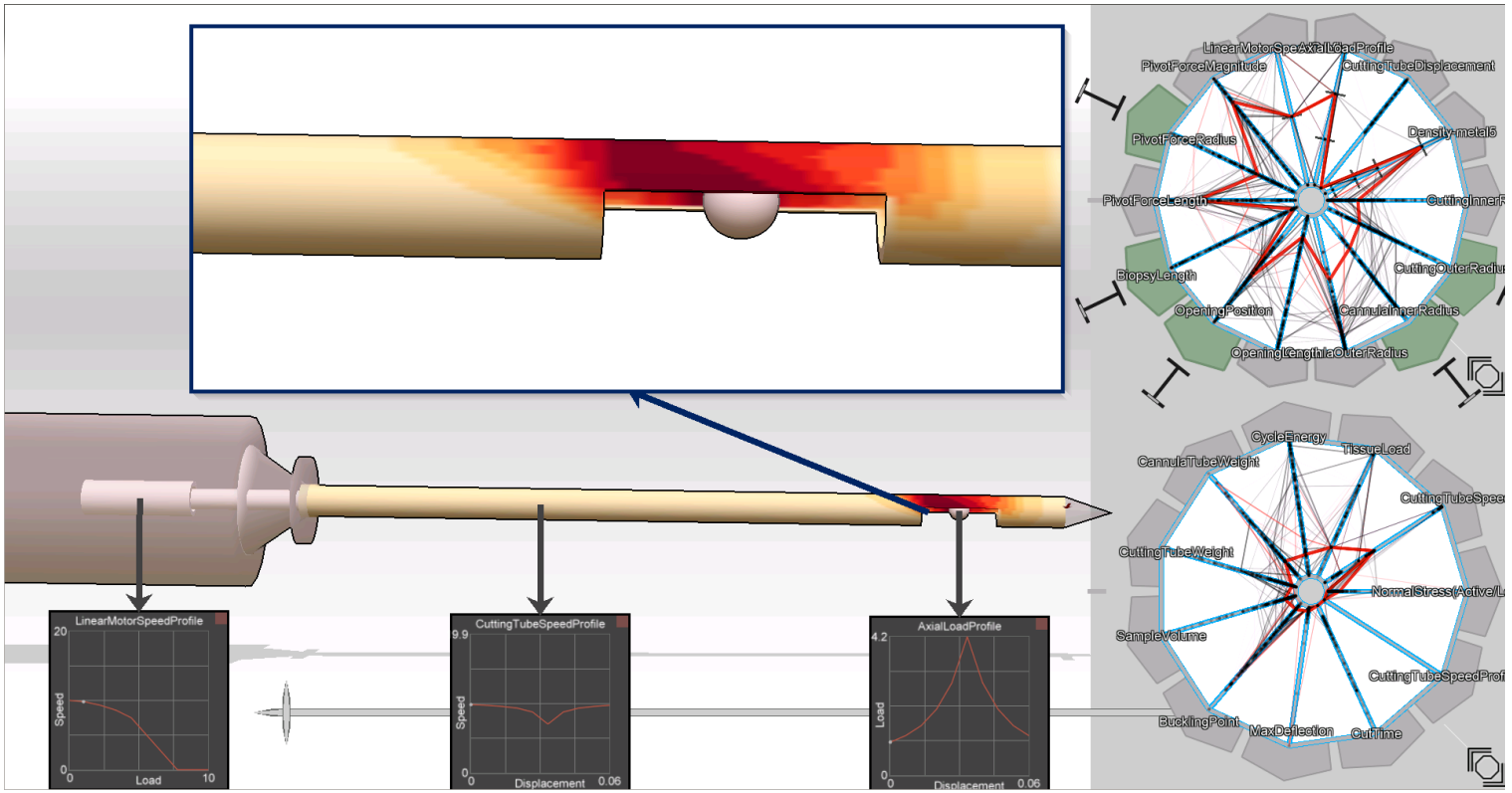

Figure 8.12: Complete graphical user interface of Design by Dragging, loaded with data resulting from a perpendicular loading condition used to run a FEA simulation of the resulting Von Mises stress distribution.

this section. In this model, a meshed 3D volume of tissue is placed inside the collection window of the device. The rotation and translation motion, driven by different motor parameters, of the inner cannula is prescribed over time. As the cannula interacts with the tissue, the resulting cutting of the tissue is simulated by evaluating a breaking condition on the elements. This is shown by the cutting process captured through time in Figure 8.13. In addition to analyzing the Von Mises stress on the tissue during cutting, additional output fields are computed, including the volume of tissue extracted and the strain on the motors.

This application makes use of the levels of detail functionality of Design by Dragging described in section 8.5.3. Three different scales of computation are used to sample the design space represented in the tissue cutting model. The first level uses approximations and estimations to predict the strains exerted on the motors and the cutting force and torque. It represents the top level of the design space, as it is the cheapest and least accurate computation, lacking the volumetric data shown in Figure 8.13. The lower two levels focus on FEA analysis of the tissue volume. The middle level uses a meshed volume of tissue containing 10,104 elements. This resolution isn't high enough to guarantee an accurate solution, but is useful gaining insight into the potential effect of input parameters. The lowest level uses a meshed volume of tissue contained 729,600 elements, a resolution that is high enough to accurately capture the tissue cutting process.

A summary of all the inputs parameters and output fields at the different levels for this specific example are shown in the Tables 8.2 and 8.3. As seen in the first table, the estimation level uses a different set of parameters for defining the tissue properties. They are defined in terms of two parameters, the "fracture toughness" and "friction factor". The two FEA levels use a more sophisticated model to define the elasticity properties of the tissue elements as well as their breaking conditions. These properties are set for two different types of tissue including an adipose, or fatty, tissue and a fibroglandular, or muscular, tissue.

Care was taken to select the most valid ranges and sets of samples for the three levels. Perhaps the most critical parameter that was identified in the design space was the "slice-push ratio" that sets the ratio of rotary speed to linear speed during the cutting operation. This parameter was expected to have a large impact on the cutting action.

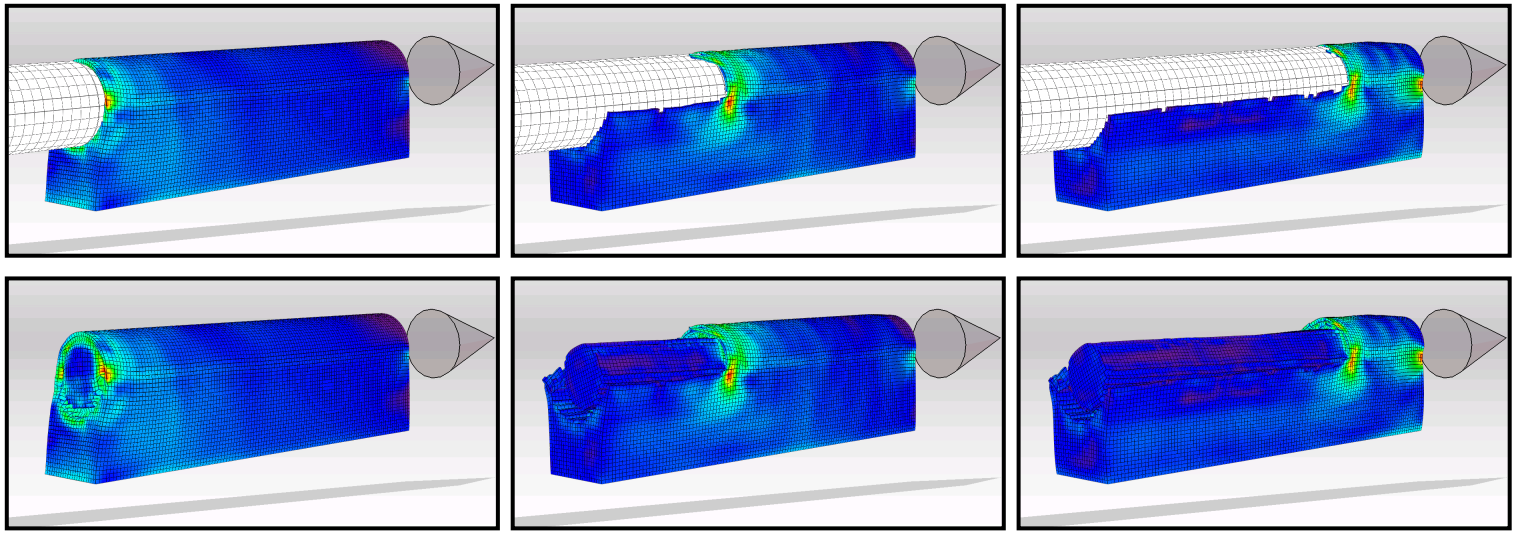

Figure 8.13: The volumetric meshed tissue sample as it is cut over time with the resulting Von Mises stress displayed as a color map. The top row displays the inner cannuala position, and the bottom row displays the cannula removed to avoid occlusion of the tissue.

As such, it was the most finely sampled of the input parameters, with 19 total values: 10 uniformly sampled from the range 0.1-1.0, and 9 from range of 2.0-10.0. 5 different motors were selected as possible candidates and their specifications were entered, as indicated in the tables. For the top level (i.e. estimation), the remaining input parameters (i.e. linear speed, fracture toughness, and friction factor) were uniformly sampled in their listed range 4 times. As a result, the total number of samples for this level was 6,080 (19x5x4x4x4). In the middle level (i.e. coarse FEA), 2 linear speeds were sampled at the high end and low end. Together with the 2 possible tissue types, this led to a total of 380 samples (19x5x2x2). The FEA results were independent from the motor selection, so there were 76 different FEA solutions represented in the set of 380 samples, each of which had 21 timesteps. The samples for the lowest level were generated by fixing the linear speed to be high and fixing the tissue type to be adipose. The selection of these parameters was guided by an exploration of the middle level. The slice-push ratio was again sampled for the same 19 values used in the middle level, leading to a total of 95 design instances (19x5x1x1).

The lower two FEA levels of this application support inverse manipulation on the Von Mises stress field of the tissue volume. The image morphing techniques discussed in this chapter are limited to 2D domains. Thus, to apply them to this specific application, a 2D domain is generated by resampling the volume using a cutting plane aligned with the center line of the tissue volume and a second cutting plane perpendicular to the first aligned to the lower edge of the inner cutting tube. A coplanar rectangle that fits the extents of the deformed tissue is defined and resampled for every timestep. The relationships between the pairwise 2D cutting planes are computed using the image registration techniques described in section 8.2.1. The inverse drag operation on one of these cutting planes is shown in Figure 8.14. Summaries of the morphing datastructures statistics for this example are shown in the third row of Table 8.1.

The tissue cutting example represents the most complete demonstration of the Design by Dragging implementation. In addition to being driven by the most realistic simulation model of the three, it also brings together all of the concepts of the system, including the levels of detail feature, into a single holistic example. The design space is complex and contains many design tradeoffs for the large set of output fields, but

		Level of Detail		
Variable Input Parameters	**Values/Range**	**Estimation**	**Coarse FEA**	**Detail FEA**
Tissue Type	Adipose or Fibroglandular		X	X
Fracture Toughness(N/mm)	0.02 - 0.1	X		
Friction Factor	0 - 10	X		
Linear Speed(mm/s)	50 - 100	X	X	X
Slice-Push Ratio	0.1 - 10	X	X	X
Rotary Motor Choice(Enum)	Motor 1 - 5	X	X	X
Motor Torque Constant(Nmm/A)	0.782, 1.15, 9.17, 10.9, 10.7	X	X	X
Motor Rated Current(A)	0.72, 1.66, 0.403, 0.84, 0.253	X	X	X
Motor Thermal Resistance 1(K/W)	46, 39.8, 21.3, 15.8, 29.8	X	X	X
Motor Thermal Resistance 2(K/W)	14, 5.1, 10.5, 4.0, 5.5	X	X	X
Motor Thermal Time Constant(Sec)	5.18, 1.51, 11, 15.4, 3.53	X	X	X
Motor Weight(g)	17, 13, 33, 159, 21	X	X	X
Motor Price($)	102.63, 288.88, 72.25, 181.88, 54.38	X	X	X
Fixed Input Parameters	**Values/Range**	**Estimation**	**Coarse FEA**	**Detail FEA**
Cutter Outer Radius(mm)	2.095	X	X	X
Cutter Inner Radius(mm)	1.715	X	X	X
Cutter Length(mm)	150	X	X	X
Window Length(mm)	150	X	X	X
Initial Cutter Gap(mm)	2	X	X	X
Cutter Material Density(g/mm)	0.0078	X	X	X

Table 8.2: List of input parameters used for the tissue cutting simulation, with an indication of each level of detail to which it is applicable. The estimation level uses a simpler tissue cutting model that relies on the fracture toughness and friction factor. The motor choice enumerated parameter is used to look up all the specific motor parameters dependent on its selection.

		Level of Detail		
Output Fields	**Type**	**Estimation**	**Coarse FEA**	**Detail FEA**
Motor Electrical Load(Amp)	Graph (vs Cutting Time)		X	X
Max Motor Electrical Load(Amp)	Scalar	X	X	X
Motor Overload Factor	Graph (vs Cutting Time)		X	X
Max Motor Overload Factor	Scalar	X	X	X
Min Motor Offer Time	Scalar	X	X	X
Max Motor On Time	Scalar	X	X	X
Total Operation Time for 5 Samples(Sec)	Scalar	X	X	X
Total System Weight(g)	Scalar	X	X	X
Motor Stall	Boolean	X	X	X
Final Undeformed Tissue Volume(mm^3)	Scalar	X	X	X
Undeformed Tissue Volume(mm^3)	Graph (vs Cutting Time)	X	X	X
Final Deformed Tissue Volume(mm^3)	Scalar		X	X
Deformed Tissue Volume(mm^3)	Graph (vs Cutting Time)		X	X
Cutter Displacement(mm)	Graph (vs Cutting Time)		X	X
Tissue Volume with Von Mises Stress	3D Hexahedral Mesh		X	X
Center Clipping Plane with Von Mises Stress	2D Image		X	X

Table 8.3: List of outputs used to evaluate a device in the tissue cutting model, with an indication of each level of detail each output is applicable to. The estimation level is missing the more detailed graphs over time (instead just showing estimated maxes) and any 3D volume data.

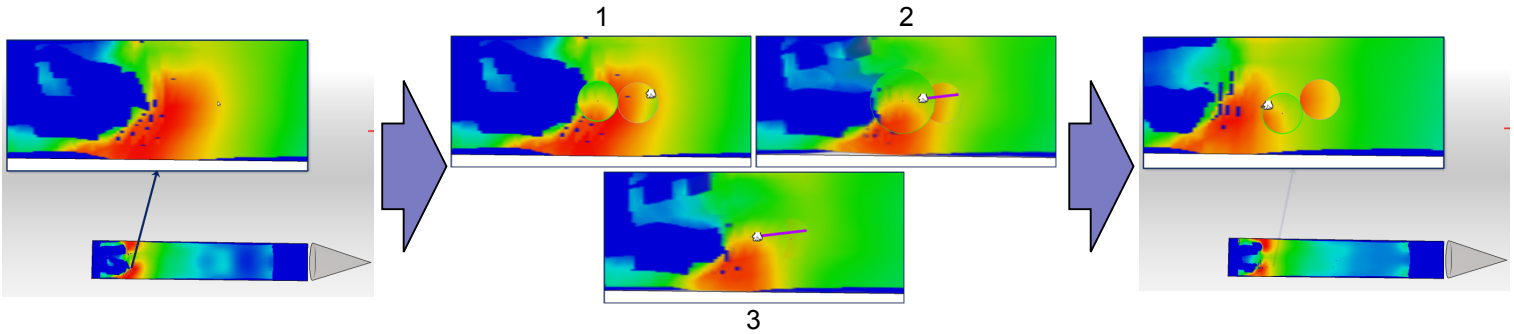

Figure 8.14: Inverse drag operation on the leading edge of a high stress area in the center of the tissue volume. The area is "shrunk" to return a design instance with a more concentrated area of stress, leading to a better cutting operation.

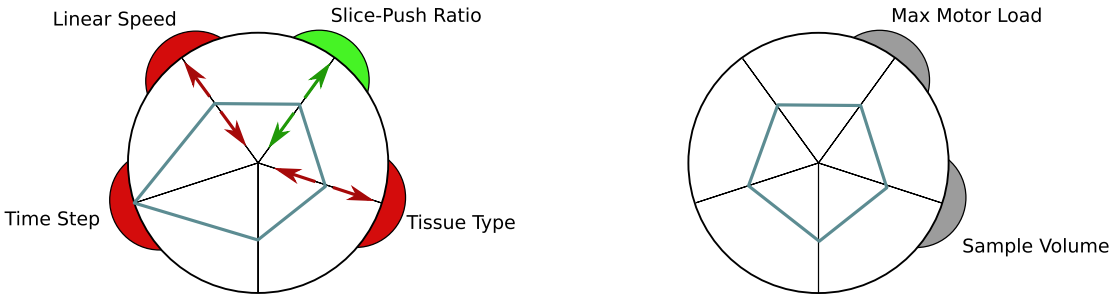

Figure 8.15: First use case interaction: evaluating FEA results

constrained enough to be solvable and relevant to a real world problem. The next sections provides a detailed description of how domain experts in medical device design both used and evaluated the Design by Dragging system, particularly in regards to this example.

Domain Expert Use Cases

In order to understand how Design by Dragging is used to solve design problems we invited our collaborators to use the system and gathered observations and feedback during use. Specifically, the mechanical engineer who developed the biopsy FEA models used the system for 15+ hours to explore the tissue cutting model described in section 8.6.1. This included several hours for training and many additional hours for unsupervised individual use. He had previously viewed several of the individual design instances while developing the models, but sitting down with the Design by Dragging tool was his first time viewing the holistic set of simulations. Based on logged notes taken by the engineer during use, we explain a number of specific forward and inverse interactions he performed, and the insight he was able to gain through each of the interactions.

Figure 8.15 summarizes the first interaction the mechanical engineer performed. In this figure, the colored tabs indicate the weighting applied to a particular input parameter or output field (i.e., red = pinned, green = normal weighting, grey = free weighting) and the arrows indicate the parameters or fields that were changed. The engineer reported that the purpose of this first interaction was to gain a basic understanding of the effect of each input on the FEA results, specifically the volume of the tissue sampled and the maximum load on the motor. To accomplish this, he dragged the slice-push

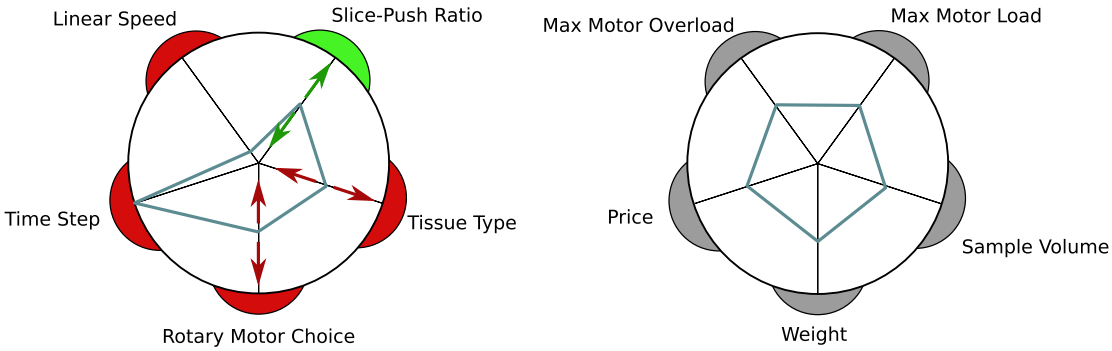

Figure 8.16: Second use case interaction: evaluating motors in low linear cutting speed.

ratio parameter while locking the linear speed and tissue type in place. He performed this drag operation for each combination of linear speed (high vs low speed) and tissue type (adipose vs fibroglandular). Based on this interaction, he reported the followings insights:

- On average, higher linear speeds result in relatively high tissue sample volumes.
- The effect of slice-push ratio on sample volume is not clear, but a lower slice-push ratio seems to contribute to a relatively larger volume.
- In general, the motor load increases when the slice-push ratio increases, as the rotation speed increases.
- Cutting glandular tissue generates smaller motor load.

In the second interaction, the engineer focused on understanding the effect of rotary motor choice during a low linear cutter speed. As seen in Figure 8.16, the linear motor speed was locked into its low state, while the motor choice and tissue choice were varied through all of their combinations. For each combination, the slice-push ratio was dragged through its range of values and the effect on the outputs were observed. The following insights were reported by the engineer:

- Motor 1 and 2 were ideal choices, as they both have very light weight at the cost a higher price.
- Motor 1 and 2 can be overloaded at slice-push ratio's greater than 4, which can cause both overheating issues and long operation times.

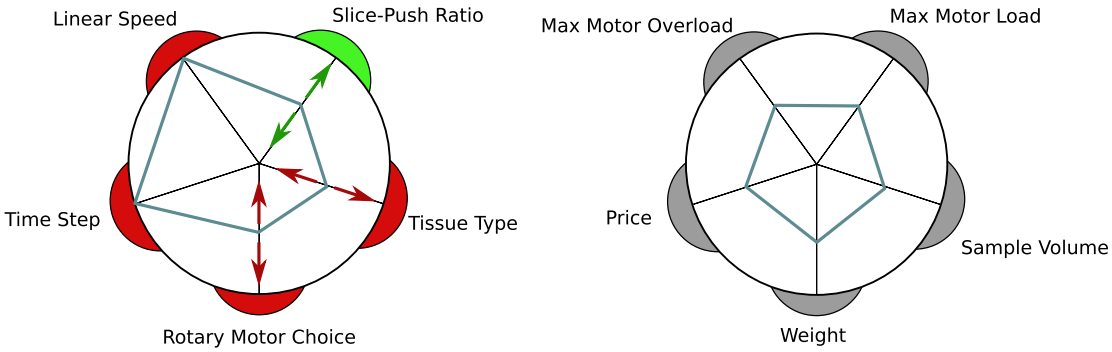

Figure 8.17: Third use case interaction: evaluating motors in high linear cutting speed

- Slice-push ratio has a large effect on motor overload factor.
- Slice-push ratio has a smaller effect on sample volume per time. The effect is more significant on motor 2.

The engineer then repeated the same interaction as described above, but changed the linear speed of the cutter to its high value, as summarized in Figure 8.17. He reported the following insights from this interaction:

- Motor 1 and 2 can be overloaded at slice-push ratio's greater than 4, which can cause both overheating issues and long operation times.
- The slice-push ratio has a large effect on motor overload factor.
- In general, the size of tissue sample volume is much greater in the high linear speed instances.

Using the insights gained from the aforementioned interactions, the engineer's final goal was to select an optimal design that maximizes the tissue sample volume while minimizing the motor weight and price. To do so, he equally weighted each of these three criteria, as indicated by the green tabs in Figure 8.18. The motor weight and price were dragged to their minimum values. After these two were set, the sample volume was dragged to increase its value, allowing the system to search for designs that have low motor weights and prices, while still resulting in large tissue sample volumes. Using this interaction, he was able to converge on a single design instance that was noted as a potential candidate for best design.

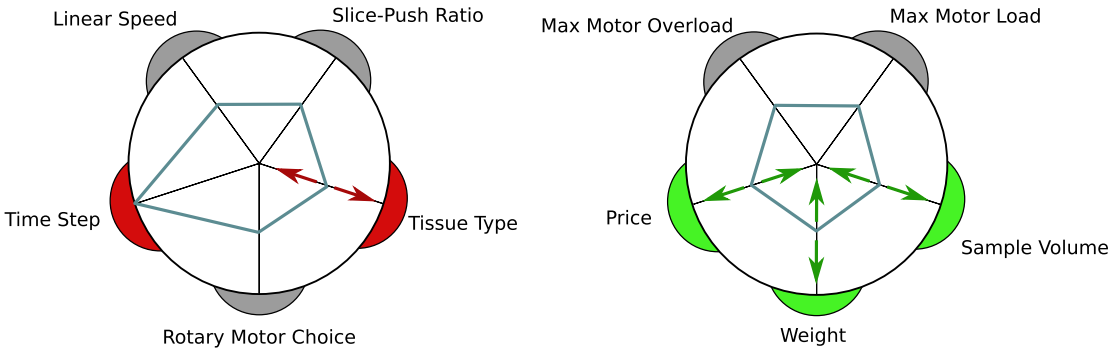

Figure 8.18: Fourth use case interaction: finding optimal design solutions

The engineer also performed several inverse manipulations of the stress field to optimize characteristics of the tissue data. The first such interaction, shown in Figure 8.14, was motivated by the knowledge of the engineer that a more concentrated area of high stress near the cutter tip is indicative of a better cutting action. On the left of Figure 8.14, a design instance is displayed that is particularly bad in this regard, with a diffuse distribution of stress. The engineer was able to grab the leading edge of this distribution and drag back toward the cutter to indicate the desire to "shrink" the boundary of this area. As the engineer drags toward the center of the preview bubble, it becomes clear that the target configuration contains the desired result, and after arriving at the center the system displays a new cutplane with a much more concentrated area of stress. In this particular example, the design instance changed from a slice-push ratio of 0.4 and a high linear speed, to a slice-push ratio of 0.6 and a low linear speed.

Figure 8.19 shows a second inverse manipulation interaction the engineer performed. The ending location of the leading edge of the tissue sample was identified as one characteristic related to the success of the cutting operation. If the tissue ends near the tip of the outer cannula, it indicates that the tissue was largely "pushed" rather than "sheared". This figure demonstrates how the cause of this effect is explorable via inverse manipulation. In it, the leading edge of the tissue is grabbed and preview bubbles pop up to indicate that the edge is able to be dragged either forward or backward. By dragging towards the preview bubbles in both directions, the engineer was able to observe that slice-push ratio has a very large effect on this characteristic of the tissue model.

In the final inverse manipulation interaction, shown in Figure 8.20, the engineer

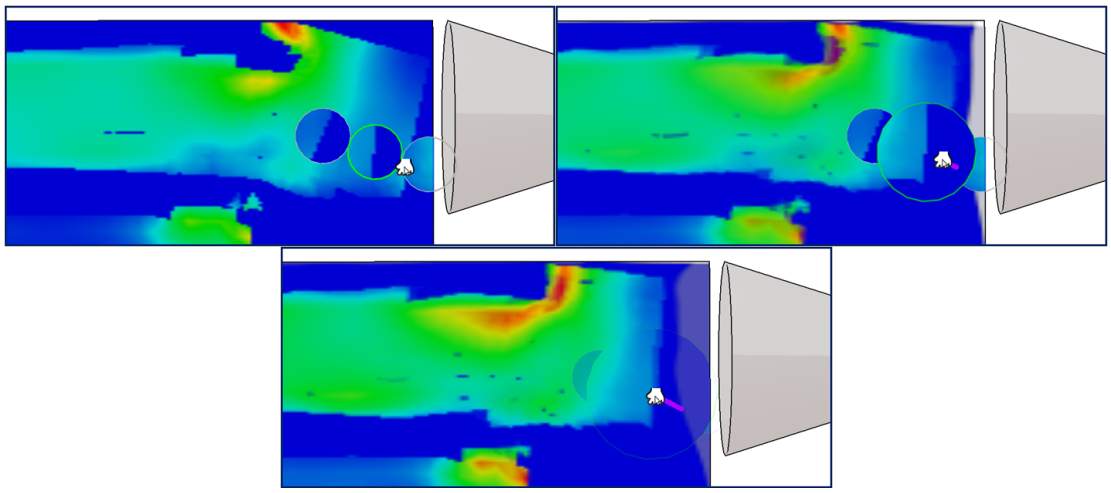

Figure 8.19: Adjusting the ending location of the tissue sample via inverse dragging. The ending location of the tissue sample is indicative of how well the cutting operation proceeded. The user grabs the leading edge of the tissue and drags toward the handle end of the device to search for results that are not pushed as far up the cannula.

explores a specific a failure case of the device called a "dry tap". A dry tap occurs when the tissue is not fully separated after the cutting operation concludes. It results from the cutting action being dominated by tension as opposed to shearing. In the left of Figure 8.20 a design instance with the dry tap issue is displayed. The engineer uses a inverse manipulation to remove the dry tap issue and find a design instance where the tissue is fully separated. This interaction is particularly interesting because it involves a very discrete jump in the stress field, the area of high stress near the connected tissue essentially vanishes. In regards to the quality of the image morph, this would typically be considered a "bad" morph because of the discrete jump. However, in the context of the engineer's workflow the preview bubbles enable a particularly interesting engineering outcome.

Domain Expert Evaluation

Since medical device engineers have played an integral role in the development and evaluation of the tool, we are confident that this approach not only matches the mindset they bring to this design problem but also fills a gap in current tools for working with simulations. The immediacy and directness it offers is fundamentally different than

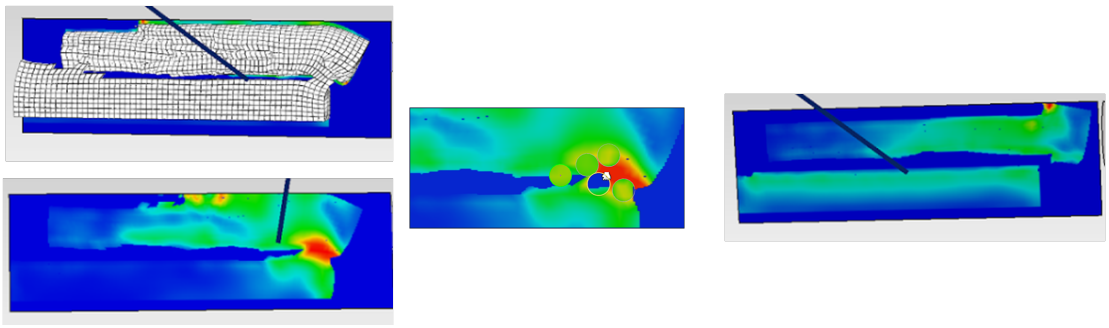

Figure 8.20: Resolution of the the "dry tap" issue via direct manipulation of the stress field. A "dry tap" occurs when the cutting operation is dominated by tension instead of shear and the tissue sample does not fully separate, as shown on the left. Using the inverse dragging operation the designer grabs the region with connected tissue and drags to the preview bubble that shows a separation of the tissue.

approaches in available commercial and research tools, which do not include direct manipulation, inverse design, and morphs between simulation results.

Specifically, the engineering team members report: "The current practice for this type of problem shies away from using computational tools as they are not efficiently connected [for example, CAD design, simulation, and visualization often require separate software packages] and there is not a way to optimize one or more parameters in the FEA model while at the same time optimizing geometry or material selection parameters... Although there is some inverse functionality beginning to creep into software tools for mechanical design, this approach leapfrogs that. To be able to do inverse [design] for materials, geometry, and FEA is just a huge breakthrough. On top of that you can visualize it."

A specific point of comparison is with the tools available in current commercial packages for organizing parameter studies. About these, the engineering team members report, "An example of such a tool is iSight (used in the medical device industry because it drives Abaqus). iSight provides quantitative analysis tools and a FEA execution engine. Although the quantitative analysis tools do help gain information for making design decisions, the number of parameters optimized is usually limited to about three. Further, the user still has to go back to the post-processing module of the original FEA package to view the simulation results one by one."

To our knowledge, no existing tools support Design by Dragging's style of in-place ensemble visualization, where the user can view many results from within the design space simply by dragging in different directions. Further, none support user interactions on top of visualizations of simulation output fields to do either forward or inverse design. Each of these has been evaluated by the collaborating engineers as not just a useful feature but as providing a major breakthrough in the way that they can approach medical device design.

Several aspects of the interface (e.g., integrating 3D modeling with data visualization, depicting design history, presenting many simulation results in the same display) are a direct result of feedback from the domain experts. In terms of criticism of the tool, the current limitation they identified is that the designer cannot make a drastic change in the geometry of a device unless this can be captured by the design parameters. In mechanical engineering terminology, the system thus is most useful for the detail design phase than for the conceptual design phase. We believe that it may be possible in the future to couple Design by Dragging with a quick, sketch-based 3D modeling interface that could facilitate use for conceptual design as well.

8.6.2 Application to Visual Effects

The second application that we explore is the design of visual effects. This application does not include the same level of long-term collaborative development and evaluation. Thus, we view it as a secondary, more exploratory application that provides an initial data point for how Design by Dragging might be applied to additional simulation-based design problems outside of engineering. We picked a motivating example of a visual effects artist designing a physically realistic flame animation to include in a movie or game, for two reasons. First, there is prior work in the visualization community on design with pre-computed flame simulations [68]. Second, the scalar fields that result from our flame simulations tend to be much more irregular (i.e., less structured) than those calculated in the biopsy device FEA simulations; thus, they provide a challenging case for verifying the reliability of the dragging operations.

Two fire simulations were generated using Autodesk Maya's fluid animator with input parameters including velocity noise, viscosity, temperature emission, fluid emission, and density emission. Dataset 1 is a set of solutions generated using a point-based

flame emitter, as pictured in Figure 8.6. Dataset 2 contains solutions for a ring of fire (Figure 8.21). Again, the warps for each dataset were computed in roughly 24 hours on a 24-core Linux machine. The detail statistics for these two datasets are shown in rows 4 and 5 of Table 8.1.

These two flame datasets are explorable by both single-cursor and multi-cursor manipulations. Figure 8.21 shows how single-cursor inverse design is used to explore the flamering dataset. Artists can search for a flame of a specific height or with wisps of a particular shape fanning out of the ring.

Observations and Testing

Because this application is targeted toward artists, it nicely illustrates what we believe is one of most powerful aspects of the approach. Although artists may not have an intuition for the impact of a simulation input parameter with a name such as "density emission," they are likely to have a strong intuition for how the shape of a flame should change in order to produce a desired effect. We have observed that the preview bubbles are particularly effective in this regard. In Figure 8.21, for example, notice that the previews sometimes identify potential target instances that are significantly different (e.g., have a much higher density of fire) than the starting instance. Users can rapidly explore these different alternatives by dragging toward and away from the various previews.

Another aspect of the interface that is particularly well suited to this application is the multi-touch inverse design strategy. We believe the reason for this is that the simulation datasets capture much more global change in the flame and that the multi-touch interface enables the user to more easily indicate major high-level changes to the output in addition to tweaking the output within a local region.

One useful insight from our iterative design and testing relates to the best way to implement the multi-touch technique using the as-rigid-as-possible shape manipulation algorithm. In early implementations, we applied as-rigid-as-possible shape manipulation to the entire scalar field by using a regular grid over the entire image as a control mesh. To the user, this feels like manipulating a rubber sheet. The current strategy creates a much more compelling and expressive interface by first extracting an iso-curve in the region of interest indicated by the user and then using the shape defined by this curve as the control mesh.

8.7 Conclusion

Design by Dragging introduces a new scalable natural user interface approach for as-direct-as-possible design with simulations. The system demonstrates how new interface algorithms for both single and multi-cursor environments can be integrated with data visualizations and how user input can be interpreted relative to data to support intelligent inverse queries. In addition to providing new capabilities for design, the tool also provides a new form of in-place interactive ensemble visualization. The examples provided demonstrate the applicability to both engineering design and less constrained artistic design, both supported through a method of calculating correspondences and morphs in a result space of hundreds of 2D scalar fields' output from simulations. Perhaps the most important qualitative contribution of the work is the almost magical feeling the interface can provide of directly manipulating data, even reshaping it with one's hands, during inverse design. Because this makes design with simulations so much more accessible to users of varied backgrounds, we believe Design by Dragging has the potential to be applied broadly to many design problems and perhaps open up new application areas for creative design with data-intensive simulations.

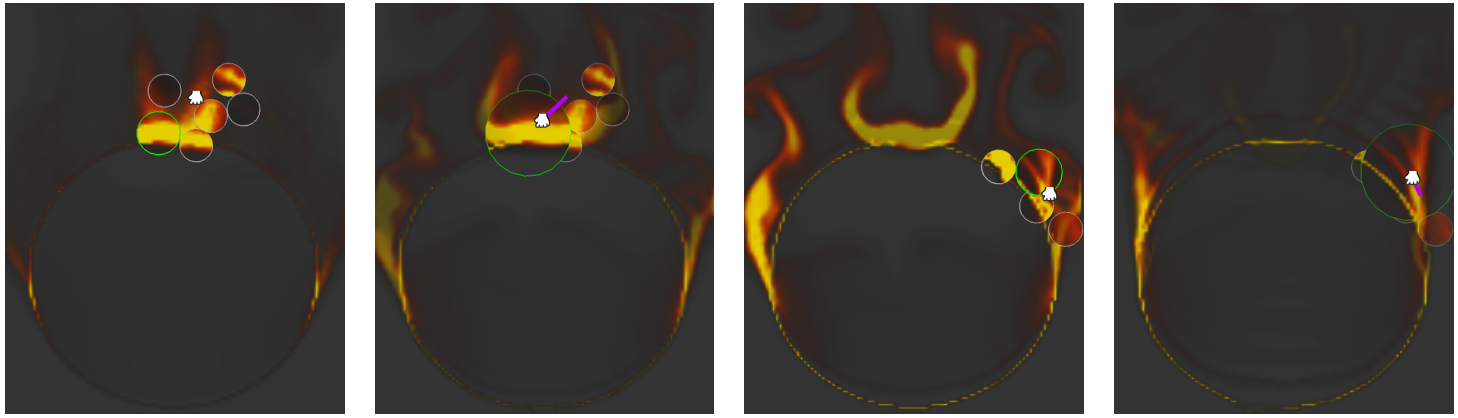

Figure 8.21: Inverse design of a flame effect. (a) Preview bubbles around an initial click show that the flame shape can be dragged to transform into a more vibrant flame. (b) As the user drags, the new flame expands and morphs into place. (c-d) Clicking and dragging at a different location on top of this new flame allows the user to further refine the shape of the flame.

Chapter 9

Discussion and Conclusions

This chapter concludes the dissertation and presents a summary discussion of its findings and their implications, including possible directions for future research and a review of this study's primary contributions and conclusions.

9.1 Future Directions

The work presented in this dissertation opens many exciting avenues for future research. Many of these directions are suggested by limitations in the existing literature and may provide powerful new ways to expand the capabilities of the tools described in this dissertation.

9.1.1 Expanding the Interrogative Features of Slice WIM

The Slice WIM tool presented in this dissertation provides not only a way to navigate complex volumetric datasets, but also a number of features for interrogating, measuring, and querying these datasets. These features enable many powerful workflows for medical device engineers and designers. There remain many additional extensions, each with significant research potential, to investigate in the future that will provide additional capabilities to the Slice WIM tool. In this section, we will look at a number of the most promising and exciting ways to expand the features and functionality of the Slice WIM tool.

Slice WIM provides a number of features for authoring new content inside volumetric environments (e.g, plotting 3D curves). This could be further expanded and made more powerful by enabling more detailed levels of analysis and simulation directly embedded within the tool by connecting it directly to external analysis tools (i.e., FEA and CFD simulation packages). The rich input space of Slice WIM excels at defining the complex and often 3D boundary conditions that are needed to set up these detailed analyses. The boundary conditions could be input to the external packages and simulated, the results of which could be returned to the Slice WIM tool and visualized to create a seamless visualization-interaction loop. For example, consider the following potential use case. A medical device designer trying to understand stent deployment in the heart creates a 3D curve to define the ideal path a catheter carrying the stent should follow. This curve, combined with the anatomy of the heart itself, are input as boundary conditions to an FEA solver, and the insertion process of the stent is simulated over time. The deformations and stresses resulting from the simulation are in turn visualized in Slice WIM, allowing the engineer to evaluate the process based on those results.

One of the features of Slice WIM enables the ability to compare and view multiple 3D anatomies in a single virtual environment. This feature could be improved by scaling this functionality to handle large databases of related anatomies. If these databases are composed of anatomies with metadata (e.g., patient age, sex, and disease states), novel user interfaces within Slice WIM can be created to give users targeted access to a huge class of anatomies available at their fingertips. For example, a database composed of a wide range of healthy and diseased hearts could be queried and accessed on the fly to understand how a replacement heart valve would fit across a broad spectrum of patient-specific hearts. Furthermore, a user could use the interrogative capabilities of the Slice WIM tool to measure and further classify the anatomies contained in the databases.

A critical workflow of the Slice WIM tool involves device placement and sizing relative to patient-specific anatomy. The tool currently supports this by allowing the user to co-locate virtual objects on top of each other through free-form manipulations. These manipulations work well for quick tasks that do not require high degrees of precision. In engineering workflows, however, higher degrees of precision are often required. One exciting direction for future work is to provide this precision through smart geometric-aware constraints and to design new touch interfaces that allow users to intuitively

specify these constraints. For example, when placing a device inside an anatomy, a designer could use the surrounding features of the anatomy to align or snap the device to fit in the best way. Combining this with large databases of anatomies, the same constraints could be prescribed to a large class of geometries automatically, allowing designers to rapidly view their device fittings across a wide range of anatomical conditions.

9.1.2 Increasing the Applicability and Computing Power of Design by Dragging

A number of exciting possible future directions would build upon the capabilities of Design by Dragging. These directions focus on two main goals. The first is improving the applicability and ease of use of the tool to facilitate easy integration of existing design problems into the tool. The second is increasing the raw computational power of the system by connecting it to high-performance computing (HPC) resources.

To apply the current tool to a new problem, a manual hurdle must be overcome, as the tool relies on a pre-visualization step of defining the geometric parameters and mapping them onto the features of the 3D model. In the future, we believe it will be possible to take existing CAD models and perform geometric analyses to automatically extract these parameterizations. Existing algorithms (e.g., [139, 154]) may provide an excellent starting point for automatically identifying the key geometric parameters that can be adjusted within the model, and then this parameterization could be input to Design by Dragging. Furthermore, these algorithms could also be used to interactively derive new parameters or features within the tool. For example, users could grab a model and indicate a new 3D feature they are interested in adding to the model, and the system could then automatically define the relevant parameters needed to sample this additional feature in the design space.

One of the most significant advances to the Design By Dragging tool could come from increasing the computational power of the system by connecting it to HPC resources. This opens up a number of possibilities. First, as described in chapter 7, the strategy for sampling the design space could be augmented by generating new samples mid-interaction. By analyzing the user's design history, the tool could intelligently spawn new simulations in real time. For this strategy to be effective and feasible, an interactive connection to HPC resources is needed. The computing power afforded by

this connection would allow the system to either densely sample low-cost simple simulations or sample a targeted handful of high-cost, complex simulations. Furthermore, the HPC connection would allow the computationally expensive morphing functions between pairs of design instances to be computed in real time. If the tool was augmented in this way, it would provide significant strides toward eliminating the batch mentality associated with simulation-based design workflows.

9.1.3 A Unified Grand Vision

This dissertation presents our steps toward accomplishing a grand vision for how scalable and natural user interfaces can be used for designing with simulations. One of the next logical steps toward completing this vision is to combine the two tools presented: Slice WIM and Design by Dragging. The Immersive Touch Workbench hardware used in the Slice WIM tool could serve as the main input and output platform for performing the direct manipulations in the Design by Dragging tool. We have already shown how the multi-touch capabilities can be used to perform inverse queries, but similarly, the touch interaction could also be used for performing forward manipulations in place of mouse-based interactions. Additionally, the rich 2D space provided by the table could be used to display the various wheel plots and 2D graphs used in the tool. The Slice WIM metaphor itself could also be applied to the Design by Dragging tool. The current GUI for Design by Dragging uses a call-out window approach for showing the interaction proxy and the active design configuration. These two components could be mapped to the WIM and the detailed world used in the Slice WIM metaphor. For example, the biopsy device could float above the tabletop, and users could perform their direct manipulations relative to it to initiate queries into the design space. The detailed world could show the result of these queries. This unified grand vision would provide a single workbench and system that the simulation-based design process could use for all aspects of its workflows.

9.2 Summary of Primary Contributions and Conclusions

In this section, we provide a summary of the main contributions and conclusions of the four research thrusts laid out in the introduction. Following this, we provide conclusions

regarding the entire body of work.

9.2.1 A Hardware Platform for Data-Intensive Exploratory Visualization

This dissertation began with the introduction of a new hardware platform we call the Immersive Touch Workbench, which is intended to provide the hardware foundation for data-intensive exploratory visualization. The main contributions of this platform include the following:

- The design of the Immersive Touch Workbench Hardware platform.
- A discussion of the implementation details and design decisions that went into creating the ITW.
- A representative interaction technique and user interface metaphor for creating software designed to work on this platform.

This workbench combines multi-touch technology with immersive virtual reality technology in a novel configuration. It is composed of two surfaces joined together at a right angle in an L-configuration. The first surface, a horizontal table, is a multi-touch-capable surface, providing the user's input to the system. The second surface, a large vertical rear-projected screen, provides an immersive, detailed 3D view to the user. This technology offers a number of potential benefits as a platform for enabling data-intensive exploratory visualization. First, the multi-touch input source provides many simultaneous points of contact to the system, allowing for a rich and high-dimensional input device. The passive haptics of pressing against the touch surface also has the added benefit of increasing precision when interacting. Second, the combination of virtual reality technology via an immersive view allows for natural viewing of complex 3D environments at large scales. Viewing data in this way often leads to new insights. Finally, the system is designed to be low-cost and to make use of existing open standards. Using off-the-shelf commodity hardware results in a low entry point that is critical to getting the platform out of the research lab and into the hands of engineers and designers.

The utility of the ITW platform is coupled with the utility of the software that runs on it, specifically the user interfaces and interaction metaphors that it makes possible. To this end, we have presented a representative 3D user interface technique called Shadow Grab that provides one effective way to design interfaces for the ITW platform. Shadow Grab allows manipulating objects that appear to float above the tabletop by grabbing onto their shadows and includes a set of multi-touch gestures that allow for full 6-DOF manipulation of floating objects.

The strengths of the ITW hardware and the Shadow Grab metaphor make them ideal for our data-intensive exploratory visualization techniques. We believe the ITW has the potential to be the workbench of the future, providing a single platform for designers working in the context of big data to use throughout their workflows. The remaining contributions of this dissertation all build upon this hardware platform.

9.2.2 Theoretical Evaluation of Design Choices for Data-Intensive Exploratory Visualization

Following the introduction of the ITW hardware platform, we presented theoretical foundations for developing data-intensive exploratory visualizations. This includes a visualization-interaction taxonomy and an evaluation of this taxonomy on this platform. The main contributions of the theoretical foundations include the following:

- A taxonomy of fundamental design decisions for depicting time-varying data on the ITW hardware platform.
- A user study and quantitative evaluation of the tradeoffs between the design decisions captured by the taxonomy.
- An application of the taxonomy to a real-world biomechanics problem and a qualitative assessment conducted with domain-expert scientists.

The taxonomy explores the fundamental design decisions available when creating exploratory visualizations. It focuses on two dimensions of a visualization that a visualization designer must choose how to implement: how to present the spatial dimension and how to present the temporal dimension. We argue that an analogous choice must be made across both of these two dimensions, as each can be depicted via either animated,

interactive, or static presentation. This leads to the motion taxonomy in which each category of visualization is created by combining one choice for space and one for time, leading to a total of 9 fundamental categories of visualization.

Using this design decision taxonomy, we created an implementation for 8 of the 9 visualization categories on the ITW hardware. The implementations use motion data (i.e., rigid bodies moving through time) as a representative data source. With these implementations, we ran a controlled quantitative user study for a collision task between two rigid bodies in order to evaluate the tradeoffs between the two dimensions of the taxonomy in terms of error rate and task completion time. The most significant result of this use study was that designs that included an interactive time element were most accurate. Given the value of accuracy in most scientific applications, visualization designers should strive to include interactive control over time.

To further evaluate this taxonomy, we performed a follow-on qualitative analysis using domain expert evaluations. Our collaborators in biomechanics served as the experts for this evaluation. We applied the 8 categories of the taxonomy to their datasets of motion-captured spinal vertebrae in patients with neck pain. The results from this evaluation were in line with the quantitative evaluation, namely that interactive control over time is the most critical element. However, the experts' feedback also suggested that visualizations using static space could be beneficial in medical domains.

The theoretical results and foundations summarized here, combined with the ITW hardware, were used to inform the design of the two exploratory visualization systems that followed. Namely, they built upon the high-level guideline that having interactive control over these two dimensions is critical. The following systems closely followed this guideline and explored how providing direct and expressive ways to interactively control these two dimensions may benefit the visual-interactive feedback loop.

9.2.3 Overview + Detail Visualization of Volumetric Data

Building upon the theoretical and hardware foundations described above, we presented Slice WIM, a software system that overcomes the difficulties associated with viewing high-volume datasets of rapidly increasing size and complexity. The main contributions of our work on the Slice WIM system are the following:

- A new multi-surface, multi-touch metaphor for overview+detail visualization of volumetric datasets in virtual reality.
- A user evaluation study of the metaphor using a representative search task.
- The details of an implementation of the Slice WIM metaphor plus a set of 5 core features and their specific applications to the metaphor that enable new generalizable techniques for interrogating and querying volumetric data.
- A graphical user interface technique called Shadow Icons for interfacing with the ITW hardware to control the Slice WIM implementation.
- A discussion of the design guidelines derived from the implementation, along with a expert user evaluation study .

The Slice WIM metaphor uses the ITW hardware to enable an overview+detail navigation style within scientific volume datasets using virtual reality. These volumetric datasets, often captured via medical imaging technologies, are becoming increasingly complex as their accuracy continues to improve. Slice WIM handles these complex environments by using a world-in-miniature (WIM) technique to provide multiple levels of detailed depiction of the volume. This allows a designer to explore a volumetric environment using immersive VR technology while maintaining a sense of context provided by the WIM.

Following the description of the core Slice WIM metaphor, we introduced a specific implementation of the metaphor that we call Interactive Slice WIM. In addition to detailing and discussing the various design decisions made during the implementation, we also introduced a set of new generalizable techniques for interrogating and querying volumetric data in this implementation. This set includes techniques for:

- specifying curves and volumetric selections,
- creating persistent interactable 2D slices on the table surface,
- manipulating multiple 3D environments that exist above the tabletop,
- performing radial measurements, and

- flying through volumes via animated camera paths.

These features greatly improve the applicability and generalizability of the Interactive Slice WIM tool, as demonstrated by the numerous scientific and medical use case scenarios we presented.

Finally, in addition to providing a new tool that allows designers to explore and interrogate volumetric environments, we learned a great deal as a result of the development and implementation of this tool. This learning included domain experts' qualitative evaluation of the system from the perspective of medical device design engineers analyzing the strengths and weaknesses of the tool. It also included a discussion of high-level design guidelines developed throughout the implementation that are generalizable to any software tool being developed for use on the ITW hardware.

9.2.4 Natural Exploration of Simulation Design Spaces

The Interactive Slice WIM tool is aimed at overcoming the difficulties associated with exploring single dataset instances of increasing size. The second measure of big data that presents challenges for exploratory visualization is handling datasets composed of many instances. To address this issue, we presented Design by Dragging, a software system that overcomes these challenges for the particular problem domain of simulation-based design. The main contributions related to the Design by Dragging system are the following:

- A new accessible approach to simulation-based design for tuning the parameters of computationally intensive simulations and visually evaluating large sets of results.
- An overview of this approach via the Design by Dragging framework, including the data structures used and the design principles to be followed when implementing the interactive visual feedback loop.
- The details of a specific implementation of the Design by Dragging framework for working in domains composed of scalar field outputs from FEA and CFD simulations.
- A number of applications in multiple domains, including medical device engineering and visual effects design, along with a long-term evaluation of the tool by

expert mechanical engineers.

The Design by Dragging framework is used for data-intensive exploration of simulation design spaces. It builds upon a class of natural user interfaces called direct manipulation interactions to explore the relationships between input parameters and output fields in complex design problems. We described a general overview of the framework, including a set of visual components and a list of interaction and visual design guidelines that should be used when developing a system that implements Design by Dragging.

Design by Dragging supports two different classes of direct manipulation interactions to explore simulation design spaces. The first of these, called forward design, is used to directly manipulate input parameters and to view the effect this change has on the output fields. For example, this allows a medical device designer to ask questions such as, "What happens to the stress field if I increase the diameter of my device?" The second type of interaction, called inverse design, is used to directly manipulate the output fields and to in return find the input parameters that best produce that manipulation. This allows a designer to ask questions such as, What input parameters produce a stress field with the area of maximum stress moved over here? Combined, these two classes of direct manipulation interactions enable a rich and natural way to explore simulation design spaces.

To demonstrate the utility of Design by Dragging, we created a specific implementation along with a set of applications. This implementation was targeted primarily toward medical device designers and the simulation design spaces they use in their engineering workflows. The applications we presented were built upon the example of a medical device used for performing a tissue biopsy. This device includes a set of geometric input parameters that are manipulable in the forward direction, and finite element analyses of stress fields are used to evaluate the effectiveness of a particular design. Using non-rigid deformations, the correspondences between all pairs of design instances' stress fields are computed, enabling smooth transitions and inverse manipulations of the stresses. Using these examples, we evaluated the tool with our domain expert collaborators in mechanical engineering. Their feedback indicates that Design by Dragging enables a fundamentally new style of human-in-the-loop design with simulations that differs dramatically from the current way engineers perform design work.

9.2.5 Capstone Conclusions

This research investigates the potential of using new scalable natural user interfaces in data-intensive exploratory visualization design workflows. These workflows present significant challenges to a designer trying to gain insight from data that stem from the increasing size and complexity of individual data instances and from integrating many of these data instances. Natural user interfaces, including multi-touch, virtual reality, and direct manipulation interaction, are employed because they allow a designer to focus on their immediate goals using intuitive and high-bandwidth inputs and outputs. Our numerous successful workflow and application examples demonstrate that scalable natural user interfaces are able to increase the effectiveness of data-intensive design workflows.

The big data paradigm will become increasingly central to the future of computing. As such, it is critical that we discover the most effective means of doing real work in this paradigm. Natural user interface technologies have reached a point where they are used everywhere in our daily lives but typically only for simple interactions. The advances presented in this dissertation demonstrate that these technologies are capable of being used in real-world engineering design workflows and that data-intensive exploratory visualizations built upon them are able to overcome the difficulties associated with designing in the context of big data. These results have the potential to shape the future of computing and change the default interfaces currently used by designers. The contributions of this dissertation, including the specific tools and interfaces it has developed, are an important step toward enabling innovative workflows on both a novel workbench of the future and existing platforms that are re-envisioned to work in new, exciting ways.

References

[1] K.M. Tolle, D. Tansley, and A.J.G. Hey. The fourth paradigm: Data-intensive scientific discovery. *Proceedings of the IEEE*, 99(8):1334–1337, 2011.

[2] Charles D. Hansen, Chris R. Johnson, Valerio Pascucci, and Cludio T. Silva. Visualization for data-intensive science. In *The Fourth Paradigm*, pages 153–163, 2009.

[3] Cory Doctorow. Big data: Welcome to the petacentre. *Nature*, 455:16–21, September 2008.

[4] Felice Frankel and Rosalind Reid. Big data: Distilling meaning from data. *Nature*, 455(7209):30, 2008.

[5] Clifford Lynch. Big data: How do your data grow? *Nature*, (7209):28–29, 2008.

[6] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *, IEEE Symposium on Visual Languages, 1996. Proceedings*, pages 336–343, 1996.

[7] D. Keim, H. Qu, and K. Ma. Big-data visualziaton guest edtitor's introduction. *Computer Graphics and Applications, IEEE*, 33(4), 2013.

[8] Johanna Beyer, Markus Hadwiger, Ali Al-Awami, Won-Ki Jeong, Narayanan Kasthuri, Jeff W. Lichtman, and Hanspeter Pfister. Exploring the connectome: Petascale volume visualization of microscopy data streams. *IEEE Computer Graphics and Applications*, 33(4):50–61, 2013.

[9] Teng-Yok Lee, Xin Tong, Han-Wei Shen, Pak Chung Wong, Samson Hagos, and L.Ruby Leung. Feature tracking and visualization of the madden-julian oscillation in climate simulation. *IEEE Computer Graphics and Applications*, 33(4):29–37, 2013.

[10] C. Johnson. Top scientific visualization research problems. *IEEE Computer Graphics and Applications*, 24(4):13–17, 2004.

[11] Chris Johnson, Robert Moorhead, Tamara Munzner, Hanspeter Pfister, Penny Rheingans, and Terry S. Yoo. Nih-nsf visualization research challenges report. Draft October 2005, October 2005.

[12] Jarke J. van Wijk. The value of visualization. In *Proceedings of IEEE Visualization 2005*, pages 79–86, 2005.

[13] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[14] Thomas Butkiewicz, Wenwen Dou, Zachary Wartell, William Ribarsky, and Remco Chang. Multi-Focused geospatial analysis using probes. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1165–1172, 2008.

[15] Daniel Keefe, Marcus Ewert, William Ribarsky, and Remco Chang. Interactive coordinated Multiple-View visualization of biomechanical motion data. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1383–1390, 2009.

[16] Andries van Dam. Post-wimp user interfaces. *Commun. ACM*, 40(2):63–67, February 1997.

[17] Steven C. Seow, Dennis Wixon, Ann Morrison, and Giulio Jacucci. Natural user interfaces: The prospect and challenge of touch and gestural computing. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 4453–4456, New York, NY, USA, 2010. ACM.

[18] A.G. Erdman, D.F. Keefe, and R. Schiestl. Grand challenge: Applying regulatory science and big data to improve medical device innovation. *IEEE Transactions on Biomedical Engineering*, 60(3):700–706, March 2013.

[19] Steve Bryson. Virtual reality in scientific visualization. *Computers & Graphics*, 17(6):679–685, November 1993.

[20] Wolfgang Kruger, Christian-A Bohn, Bernd Frhlich, Heinrich Schth, Wolfgang Strauss, and Gerold Wesche. The responsive workbench. *IEEE Computer Graphics AND Applications*, 14:12—15, 1994.

[21] W. Kruger, C.-A. Bohn, Bernd Frohlich, H. Schuth, W. Strauss, and G. Wesche. The responsive workbench: a virtual work environment. *IEEE Computer*, 28(7):42–48, 1995.

[22] A. Van Dam, A. Forsberg, D.H. Laidlaw, J.J. LaViola, and R.M. Simpson. Immersive VR for scientific visualization: a progress report. *IEEE Computer Graphics and Applications*, 20(6):26–52, 2000.

[23] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the cave. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 135–142, 1993.

[24] Ivan Sutherland. A head-mounted three dimensional display. In *Proceedings of the AFIPS Fall Joint Computer Conference*, volume 33, pages 757–764, 1968.

[25] Colin Ware, Kevin Arthur, and Kellogg S. Booth. Fish tank virtual reality. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pages 37–42, New York, NY, USA, 1993. ACM.

[26] Song Zhang, C. Demiralp, S. Zhang, Demiralp, D. F. Keefe, M. Dasilva, P. J. Basser, C. Pierpaoli, E. A. Chiocca, T. S. Deisboeck, D. H. Laidlaw, and B. D. Greenberg. An immersive virtual environment for DT-MRI volume visualization applications: a case study. In *In Proceedings of IEEE Visualization 2001*, pages 437–440, 2001.

[27] Andrew S. Forsberg, David H. Laidlaw, Andries van Dam, Robert M. Kirby, George E. Karniadakis, and Jonathan L. Elion. Immersive virtual reality for visualizing flow through an artery. In *Proceedings of the conference on Visualization*

'00, pages 457–460, Salt Lake City, Utah, United States, 2000. IEEE Computer Society Press.

[28] Bernd Hentschel, Irene Tedjo, Markus Probst, Marc Wolter, Marek Behr, Christian Bischof, and Torsten Kuhlen. Interactive blood damage analysis for ventricular assist devices. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1515–1522, 2008.

[29] Oliver Kreylos, Gerald Bawden, Tony Bernardin, Magali I. Billen, Eric S. Cowgill, Ryan D. Gold, Bernd Hamann, Margarete Jadamec, Louise H. Kellogg, Oliver G. Staadt, and Dawn Y. Sumner. Enabling scientific workflows in virtual reality. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, VRCIA '06, page 155162, New York, NY, USA, 2006. ACM.

[30] B Frischer, D Favro, P Liverani, S De Blaauw, and D Abernathy. Virtual reality and ancient rome: The UCLA cultural VR lab's santa maria maggiore project. In Juan Barcelo, Maurizio Forte, and Donald Sanders, editors, *British Archaeological Reports*, volume 843, pages 155–162. Archaeopress, 2000.

[31] Khairi Reda, Alessandro Febretti, Aaron Knoll, Jillian Aurisano, Jason Leigh, Andrew Johnson, Michael E. Papka, and Mark Hereld. Visualizing large, heterogeneous data in hybrid-reality environments. *IEEE Computer Graphics and Applications*, 33(4):38–48, 2013.

[32] Andries van Dam, David H Laidlaw, and Rosemary Michelle Simpson. Experiments in immersive virtual reality for scientific visualization. *Computers & Graphics*, 26(4):535–555, August 2002.

[33] Daniel F. Keefe. Integrating visualization and interaction research to improve scientific workflows. *IEEE Computer Graphics and Applications*, 30(2):8–13, 2010.

[34] D.F. Keefe and T. Isenberg. Reimagining the scientific visualization interaction paradigm. *IEEE Computer*, 46(5):51–57, 2013.

[35] Ben Shneiderman and Catherine Plaisant. *Designing the user interface: strategies for effective human-computer interaction*. Pearson/Addison Wesley, Boston, 2004.

[36] Alex Endert, Lauren Bradel, and Chris North. Beyond control panels: Direct manipulation for visual analytics. *IEEE Computer Graphics and Applications*, 33(4):6–13, 2013.

[37] Jaegul Choo and Haesun Park. Customizing computational methods for visual analytics with big data. *IEEE Computer Graphics and Applications*, 33(4):22–28, 2013.

[38] Petra Isenberg, Sheelagh Carpendale, Tobias Hesselmann, Tobias Isenberg, and Bongshin Lee. Proceedings of the Workshop on Data Exploration for Interactive Surfaces-DEXIS 2011. Rapport de recherche, May 2012.

[39] 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.

[40] Tobias Isenberg, Maarten H. Everts, Jens Grubert, and Sheelagh Carpendale. Interactive exploratory visualization of 2D vector fields. *Computer Graphics Forum*, 27(3):983–990, 2008.

[41] Chi-Wing Fu, Wooi-Boon Goh, and Junxiang Allen Ng. Multi-touch techniques for exploring large-scale 3d astrophysical simulations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2213–2222, New York, NY, USA, 2010. ACM.

[42] Lingyun Yu, Pjotr Svetachov, Petra Isenberg, Maarten H. Everts, and Tobias Isenberg. FI3D: Direct-Touch interaction for the exploration of 3D scientific visualization spaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6)(November/December 2010):1613–1622, 2010.

[43] 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 2012*, GI '12, pages 181–184, Toronto, Ont., Canada, Canada, 2012. Canadian Information Processing Society.

[44] Lingyun Yu, K. Efstathiou, P. Isenberg, and T. Isenberg. Efficient structure-aware selection techniques for 3d point cloud visualizations with 2dof input. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2245–2254, 2012.

[45] Nicole Sultanum, Sowmya Somanath, Ehud Sharlin, and Mario Costa Sousa. "point it, split it, peel it, view it": Techniques for interactive reservoir visualization on tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '11, pages 192–201, New York, NY, USA, 2011. ACM.

[46] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys (CSUR)*, 41(1):1–31, 2008.

[47] Patrick Baudisch, Nathaniel Good, Victoria Bellotti, and Pamela Schraedley. Keeping things in context: A comparative evaluation of focus plus context screens, overviews, and zooming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 259–266, New York, NY, USA, 2002. ACM.

[48] Ken Perlin and David Fox. Pad: An alternative approach to the computer interface. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 57–64, New York, NY, USA, 1993. ACM.

[49] George W. Furnas and Benjamin B. Bederson. Space-scale diagrams: Understanding multiscale interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 234–241, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[50] Roger A. Sayle and E.James Milner-White. Rasmol: biomolecular graphics for all. *Trends in Biochemical Sciences*, 20(9):374 – 376, 1995.

[51] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. Guidelines for using multiple views in information visualization. In *Proceedings of the working*

conference on Advanced visual interfaces, pages 110–119, Palermo, Italy, 2000. ACM.

[52] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: Detail and context smoothly integrated. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pages 173–176, New York, NY, USA, 1991. ACM.

[53] G. W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '86, pages 16–23, New York, NY, USA, 1986. ACM.

[54] George G. Robertson and Jock D. Mackinlay. The document lens. In *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, UIST '93, pages 101–108, New York, NY, USA, 1993. ACM.

[55] Lujin Wang, Ye Zhao, K. Mueller, and A. Kaufman. The magic volume lens: an interactive focus+context technique for volume rendering. In *Visualization, 2005. VIS 05. IEEE*, pages 367–374, Oct 2005.

[56] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 83–91, New York, NY, USA, 1992. ACM.

[57] Matthew Plumlee and Colin Ware. Zooming, multiple windows, and visual working memory. In *In Proceedings of AVI 2002*, pages 59–68. ACM Press, 2002.

[58] K. Sims. Interactive evolution of dynamical systems. *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artifical Life*, pages 171–178, 1992.

[59] Hiroaki Nishino, Hideyuki Takagi, and Kouichi Utsumiya. A digital prototyping system for designing novel 3d geometries. In *6th Int. Conf. on Virtual Systems and MultiMedia (VSMM2000)*, pages 473–482, 2000.

[60] Hiroaki Nishino, Hideyuki Takagi, Sung-Bae Cho, and Kouichi Utsumiya. A 3d modeling system for creative design. In *Information Networking, 2001. Proceedings. 15th International Conference on*, pages 479–486. IEEE, 2001.

[61] J. McCormack. Interactive evolution of l-system grammars for computer graphics modelling. *Complex Systems : from Biology to Computation*, pages 118–130, 1993.

[62] Gustavo Patow and Xavier Pueyo. A survey of inverse rendering problems. *Comput. Graph. Forum*, 22(4):663–688, 2003.

[63] John K. Kawai, James S. Painter, and Michael F. Cohen. Radioptimization: Goal based rendering. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 147–154, New York, NY, USA, 1993. ACM.

[64] Chris Schoeneman, Julie Dorsey, Brian Smits, James Arvo, and Donald Greenberg. Painting with light. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 143–146, New York, NY, USA, 1993. ACM.

[65] J. Thomas Ngo and Joe Marks. Spacetime constraints revisited. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 343–350, New York, NY, USA, 1993. ACM.

[66] Zicheng Liu, Steven J. Gortler, and Michael F. Cohen. Hierarchical spacetime control. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 35–42, New York, NY, USA, 1994. ACM.

[67] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: a general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 389–400, 1997.

[68] S. Bruckner and T. Moller. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1468 –1476, 2010.

[69] Christopher D. Twigg and Doug L. James. Many-worlds browsing for control of multibody dynamics. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[70] Jürgen Waser, Raphael Fuchs, Hrvoje Ribicic, Benjamin Schindler, Günther Blöschl, and Eduard Gröller. World Lines. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1458–1467, 2010.

[71] H. Ribicic, J. Waser, R. Gurbat, B. Sadransky, and M.E. Groller. Sketching uncertainty into simulations. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2255 –2264, 2012.

[72] Kresimir Matkovic, Denis Gracanin, Mario Jelovic, Andreas Ammer, Alan Lez, and Helwig Hauser. Interactive visual analysis of multiple simulation runs using the simulation model view: Understanding and tuning of an electronic unit injector. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1449–1457, 2010.

[73] H. Piringer, S. Pajer, W. Berger, and H. Teichmann. Comparative visual analysis of 2d function ensembles. *Comp. Graph. Forum*, 31(3pt3):1195–1204, 2012.

[74] Martijn J. Schuemie, Peter Van Der Straaten, Merel Krijn, M. Sc, M. Sc, Charles A. P. G, Van Der Mast, and Ph. D. Research on presence in vr: a survey. *ACM Virtual Reality Software and Technology (VRST*, 4:202, 2001.

[75] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, Seattle, WA, USA, 2005. ACM.

[76] M St John, M B Cowen, H S Smallman, and H M Oonk. The use of 2D and 3D displays for shape-understanding versus relative-position tasks. *Human Factors*, 43(1):79–98, 2001.

[77] Melanie Tory, Arthur E. Kirkpatrick, M. Stella Atkins, and Torsten Moller. Visualization task performance with 2D, 3D, and combination displays. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):2–13, 2006.

[78] Joëlle Coutaz, Christophe Lachenal, and Christophe Lachenal. Ontology for Multi-surface Interaction.

[79] R. Ajaj, F. Vernier, and C. Jacquemin. Navigation modes for combined Table/Screen 3D scene rendering. In *ACM international conference on Interactive Tabletops and Surfaces*, pages 141–148, 2009.

[80] Dzmitry Aliakseyeu, Sriram Subramanian, Jean-Bernard Martens, and Matthias Rauterberg. Interaction techniques for navigation through and manipulation of 2D and 3D data. In *Proceedings of the workshop on Virtual environments 2002*, pages 179–188, Barcelona, Spain, 2002. Eurographics Association.

[81] Martin Spindler, Sophie Stellmach, and Raimund Dachselt. PaperLens: advanced magic lens interaction above the tabletop. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 69–76, Banff, Alberta, Canada, 2009. ACM.

[82] 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*, pages 189–196, Barcelona, Spain, 2002.

[83] Joseph LaViola, Daniel Acevedo, Daniel F. Keefe, and Robert Zeleznik. Hands-free multi-scale navigation in virtual environments. In *Proceedings of ACM Symposium on Interactive 3D Graphics 2001*, pages 9–15, 2001.

[84] Martin Kaltenbrunner, Till Bovermann, Ross Bencina, and Enrico Costanza. Tuio - a protocol for table based tangible user interfaces. In *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005)*, Vannes, France, 2005.

[85] Martin Kaltenbrunner. Tuio implementations, 2010.

[86] Dane Coffey, Fedor Korsakov, and Daniel F. Keefe. Low cost vr meets low cost multi-touch. In *Proceedings of the 6th International Conference on Advances in Visual Computing - Volume Part II*, ISVC'10, pages 351–360, Berlin, Heidelberg, 2010. Springer-Verlag.

[87] Kenneth P. Herndon, Robert C. Zeleznik, Daniel C. Robbins, D. Brookshire Conner, Scott S. Snibbe, and Andries van Dam. Interactive shadows. In *Proceedings of the 5th annual ACM symposium on User interface software and technology*, UIST '92, pages 1–6, New York, NY, USA, 1992. ACM.

[88] Elizabeth L Brainerd, David B Baier, Stephen M Gatesy, Tyson L Hedrick, Keith A Metzger, Susannah L Gilbert, and Joseph J Crisco. Xray reconstruction of moving morphology (XROMM): precision, accuracy and applications in comparative biomechanics research. *Journal of Experimental Zoology Part A: Ecological Genetics and Physiology*, 313A(5):262–279, June 2010.

[89] G. Elisabeta Marai, Cindy Grimm, and David H. Laidlaw. Arthrodial joint markerless cross-parameterization and biomechanical visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1095–1104, September 2007.

[90] Daniel F Keefe, T. M. O'Brien, David B Baier, Stephen M Gatesy, Elizabeth L Brainerd, and David H Laidlaw. Exploratory Visualization of Animal Kinematics Using Instantaneous Helical Axes. *Computer Graphics Forum*, 27(3):863–870, May 2008.

[91] Daniel F Keefe, Marcus Ewert, William Ribarsky, and Remco Chang. Interactive coordinated multiple-view visualization of biomechanical motion data. *IEEE transactions on visualization and computer graphics*, 15(6):1383–90, 2009.

[92] Kenneth S. Breuer Sharon M. Swartz David H. Laidlaw Jian Chen, Daniel K. Riskin. Bookstein coordinate-based shape analysis of bat wing kinematics. In *Integrative and Comparative Biology*, volume 49, pages E30–E30, 2009.

[93] V Interrante, H Fuchs, and S M Pizer. Conveying the 3D shape of smoothly curving transparent surfaces via texture. *Visualization and Computer Graphics, IEEE Transactions on*, 3(2):98–117, 1997.

[94] Colin Ware and Peter Mitchell. Visualizing graphs in three dimensions. *ACM Transactions on Applied Perception (TAP)*, 5:2:1–2:15, January 2008.

[95] Taylor Sando, Melanie Tory, and Pourang Irani. Effects of animation, user-controlled interactions, and multiple static views in understanding 3D structures. In *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization - APGV '09*, page 69, New York, New York, USA, 2009. ACM Press.

[96] B. Tversky, J.B. Morrison, and M. Betrancourt. Animation: can it facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, 2002.

[97] Hans Rosling. TED Talks: Hans Rosling Shows The Best Stats You've Ever Seen, February 2006.

[98] George Robertson, Roland Fernandez, Danyel Fisher, Bongshin Lee, and John Stasko. Effectiveness of Animation in Trend Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1325–1332, November 2008.

[99] R Sekuler, S N J Watamaniuk, and R Blake. Perception of visual motion. *Stevens Handbook of Experimental Psychology,*, 1, 2002.

[100] Lyn Bartram, Colin Ware, and Tom Calvert. Filtering and integrating visual information with motion. In *In Proceedings on Information Visualization*, pages 66–79. Society Press, 2001.

[101] V Interrante, B Ries, and L Anderson. Distance Perception in Immersive Virtual Environments, Revisited. In *Virtual Reality Conference, 2006*, pages 3–10, March 2006.

[102] F Steinicke, G Bruder, K Hinrichs, and P Willemsen. Change blindness phenomena for stereoscopic projection systems. In *Virtual Reality Conference (VR), 2010 IEEE*, pages 187–194, March 2010.

[103] C Ware. Designing with a 2 1/2-D attitude. *Information Design Journal*, 10(3):258–265, 2001.

[104] Alark Joshi and Penny Rheingans. Illustration-inspired techniques for visualizing time-varying data. In *Visualization, 2005. VIS 05. IEEE*, pages 679–686. IEEE, 2005.

[105] J. Woodring and H.W. Shen. Chronovolumes: a direct rendering technique for visualizing time-varying data. In *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, pages 27–34. ACM, 2003.

[106] Johannes Schmid, Robert W Sumner, Huw Bowles, and Markus Gross. Programmable motion effects. In *ACM SIGGRAPH 2010 papers on - SIGGRAPH '10*, page 1, Los Angeles, California, 2010.

[107] M Tory, A E Kirkpatrick, M S Atkins, and Others. Visualization task performance with 2D, 3D, and combination displays. *IEEE Transactions on Visualization and Computer Graphics*, pages 2–13, 2006.

[108] Arin Ellingson, Craig Schulz, Gert Bronfor, and David Nuckley. Helical Axis Approach to Aberrant Motion Identification in Patients with Chronic Neck Pain. In *American Socitey of Biomechanics 2011 Annual Meeting*, 2011.

[109] S.L. Delp, F.C. Anderson, A.S. Arnold, P. Loan, A. Habib, C.T. John, E. Guendelman, and D.G. Thelen. Opensim: Open-source software to create and analyze dynamic simulations of movement. *Biomedical Engineering, IEEE Transactions on*, 54(11):1940 –1950, nov. 2007.

[110] 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, April 2004.

[111] Randy Pausch, Tommy Burnette, Dan Brockway, and Michael E. Weiblen. Navigation and locomotion in virtual worlds via flight into hand-held miniatures. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 399–400. ACM, 1995.

[112] 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.

[113] Ramn Trueba, Carlos Andujar, and Ferran Argelaguet. Complexity and occlusion management for the World-in-Miniature metaphor. In *Smart Graphics*, pages 155–166. 2009.

[114] Chadwick A. Wingrave, Yonca Haciahmetoglu, and Doug A. Bowman. Overcoming world in miniature limitations by a scaled and scrolling WIM. In *Proceedings of the IEEE conference on Virtual Reality*, pages 11–16. IEEE Computer Society, 2006.

[115] Patrick Baudisch, Nathaniel Good, and Paul Stewart. Focus plus context screens: combining display technology with visualization techniques. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 31–40, Orlando, Florida, 2001. ACM.

[116] Mark J. Flider and Brian P. Bailey. An evaluation of techniques for controlling focus+context screens. In *Proceedings of Graphics Interface 2004*, pages 135–144, London, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[117] Harald Piringer, Robert Kosara, and Helwig Hauser. Interactive Focus+Context visualization with linked 2D/3D scatterplots. In *Proceedings of the Second International Conference on Coordinated & Multiple Views in Exploratory Visualization*, pages 49–60, 2004.

[118] Jr Joseph J. LaViola, Daniel Acevedo Feliz, Daniel F. Keefe, and Robert C. Zeleznik. Hands-free multi-scale navigation in virtual environments. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 9–15. ACM, 2001.

[119] Daniel Wigdor, Chia Shen, Clifton Forlines, and Ravin Balakrishnan. Table-centric interactive spaces for real-time collaboration. In *Proceedings of the working conference on Advanced visual interfaces*, pages 103–107, Venezia, Italy, 2006. ACM.

[120] Yinggang Li, Chi-Wing Fu, and Andrew Hanson. Scalable wim: Effective exploration in large-scale astrophysical environments. *IEEE Transactions on Visualization and Computer Graphics*, 12:1005–1012, 2006.

[121] Kenrick Kin, Maneesh Agrawala, and Tony DeRose. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In *Proceedings of Graphics Interface 2009*, pages 119–124, Kelowna, British Columbia, Canada, 2009. Canadian Information Processing Society.

[122] Eva Hornecker, Paul Marshall, Nick Sheep Dalton, and Yvonne Rogers. Collaboration and interference: awareness with mice or touch input. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 167–176, San Diego, CA, USA, 2008. ACM.

[123] Gabriel Robles-De-La-Torre. The importance of the sense of touch in virtual and real environments. *IEEE MultiMedia*, 13(3):24–30, 2006.

[124] H. Benko and S. Feiner. Balloon selection: A Multi-Finger technique for accurate Low-Fatigue 3D selection. In *IEEE Symposium on 3D User Interfaces, 2007. 3DUI '07.*, 2007.

[125] Tovi Grossman and Ravin Balakrishnan. The design and evaluation of selection techniques for 3D volumetric displays. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 3–12, Montreux, Switzerland, 2006. ACM.

[126] D. Valkov, F. Steinicke, G. Bruder, K. Hinrichs, J. Schöning, F. Daiber, and A. Krüger. Touching floating objects in projection-based virtual reality environments. In *Proceedings of the 16th Eurographics Conference on Virtual Environments*, EGVE - JVRC'10, pages 17–24, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[127] Dane Coffey, Fedor Korsakov, and Daniel F. Keefe. Low cost VR meets low cost multi-touch. In *Proceedings of International Symposium on Visual Computing*, pages 351–360, 2010.

[128] Mimics software. http://www.materialise.com/mimics.

[129] The national library of medicine's visible human project. http://www.nlm.nih.gov/research/visible/visible_human.html.

[130] Iman Borazjani, Liang Ge, and Fotis Sotiropoulos. High-resolution fluid-structure interaction simulations of flow through a bi-leaflet mechanical heart valve in an anatomic aorta. *Annals of Biomedical Engineering*, 38(2):326–344, February 2010.

[131] Iman Borazjani and Fotis Sotiropoulos. The effect of implantation orientation of a bileaflet mechanical heart valve on kinematics and hemodynamics in an anatomic aorta. *Journal of Biomechanical Engineering*, 132(11):111005, November 2010.

[132] Trung Le, Iman Borazjani, and Fotis Sotiropoulos. A computational framework for high resolution simulations of patient-specific left heart hemodynamics with aortic valve prothesis. In *Euromech 529 - Cardiovascular Fluid Mechanics from theoretical aspects to diagnostic and therapeutic support - CVFM2011*, June 2011.

[133] 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. ACM.

[134] D. F Keefe, R. C Zeleznik, and D. H Laidlaw. Tech-note: Dynamic dragging for input of 3D trajectories. In *Proceedings of the 2008 IEEE Symposium on 3D User Interfaces*, 3DUI '08, page 5154, Washington, DC, USA, 2008. IEEE Computer Society.

[135] Katie A. Siek, Yvonne Rogers, and Kay H. Connelly. Fat finger worries: How older and younger users physically interact with pdas. In *Proceedings of the 2005 IFIP TC13 International Conference on Human-Computer Interaction*, INTERACT'05, pages 267–280, Berlin, Heidelberg, 2005. Springer-Verlag.

[136] FDA. FDA regulatory science report. http://www.fda.gov/AboutFDA/CentersOffices/OfficeofMedicalProductsandTobacco/CDRH/CDRHReports/ucm274152.htm, 2011.

[137] Jurgen Waser, Raphael Fuchs, Hrvoje Ribicic, Benjamin Schindler, Gunther Bloschl, and Eduard Groller. World lines. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1458–1467, 2010.

[138] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: an interface for sketching 3d scenes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 163–170, 1996.

[139] Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. iWIRES: an analyze-and-edit approach to shape manipulation. *ACM Trans. Graph.*, 28(3):33:1–33:10, 2009.

[140] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 1134–1141, 2005.

[141] Takeo Igarashi and Jun Mitani. Apparent layer operations for the manipulation of deformable objects. *ACM Trans. Graph.*, 29:110:1–110:7, 2010.

[142] Tom Ngo, Doug Cutrell, Jenny Dana, Bruce Donald, Lorie Loeb, and Shunhui Zhu. Accessible animation and customizable graphics via simplicial configuration modeling. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 403–410, 2000.

[143] Randall Smith, Richard Pawlicki, István Kókai, Jörg Finger, and Thomas Vetter. Navigating in a shape space of registered models. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1552–1559, 2007.

[144] Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowitcz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. Video browsing by direct manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 237–246, 2008.

[145] Jernej Barbič, Funshing Sin, and Eitan Grinspun. Interactive editing of deformable simulations. *ACM Trans. Graph.*, 31(4):70:1–70:8, 2012.

[146] Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.*, 31(4):86:1–86:11, 2012.

[147] A. Bangalore and D. H. House. A technique for art direction of physically based fire simulation. In *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, CAe '12, pages 45–54, 2012.

[148] Lei Zhu, Yan Yang, S. Haker, and A. Tannenbaum. An image morphing technique based on optimal mass preserving mapping. *IEEE Transactions on Image Processing*, 16(6):1481 –1495, 2007.

[149] Stefan Klein, Marius Staring, Keelin Murphy, Max A. Viergever, and Josien P. W. Pluim. elastix: A toolbox for intensity-based medical image registration. *IEEE Transactions on Medical Imaging*, pages 196–205, 2010.

[150] Stefan Klein, Josien P. Pluim, Marius Staring, and Max A. Viergever. Adaptive stochastic gradient descent optimisation for image registration. *Int. J. Comput. Vision*, 81(3):227–239, 2009.

[151] P. Thevenaz and M. Unser. Optimization of mutual information for multiresolution image registration. *IEEE Transactions on Image Processing*, 9(12):2083 – 2099, 2000.

[152] Coert Metz, Stefan Klein, Michiel Schaap, Theo van Walsum, and Wiro J. Niessen. Nonrigid registration of dynamic medical imaging data using nd+t b-splines and a groupwise optimization approach. *Medical Image Analysis*, pages 238–249, 2011.

[153] Takeo Igarashi and Yuki Igarashi. Implementing as-rigid-as-possible shape manipulation and surface flattening. *J. Graphics, GPU, and Game Tools*, pages 17–30, 2009.

[154] Niloy J. Mitra, Yong-Liang Yang, Dong-Ming Yan, Wilmot Li, and Maneesh Agrawala. Illustrating how mechanical assemblies work. *ACM Trans. Graph.*, 29:58:1–58:12, 2010.