Designing Effective Motion Visualizations: Elucidating Scientific Information by Bringing Aesthetics and Design to Bear on Science

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

David Allen Schroeder

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Doctor of Philosophy

Daniel F Keefe

November, 2014

Acknowledgements

The work in Chapter 3 was based upon work supported in part by the University of Minnesota Digital Technology Center's Digital Technologies Initiative and by the Washington Technology Center Research Technology Development grant RTD11 UW SB01.

The work in Chapter 4 was based upon work supported by the National Science Foundation under Grant No. IIS-1054783. This work was also partially supported by NIH/NIAMS grants T32 AR050938 and T32 AR056950.

We thank Mike Kirby and David Laidlaw for making available the vector field datasets used in Chapter 5.

The work in Chapter 6 was supported by the National Science Foundation under Grant Number IIS-1218058.

Abstract

The visual system is the highest-bandwidth pathway into the human brain, and visualization takes advantage of this pathway to allow users to understand datasets they are interested in. Recent scientific advances have led to the collection of larger and more complicated datasets, leading to new challenges in effectively visualizing these data. The focus of this dissertation is on addressing these challenges and enabling the next generation of visualization systems. We address these challenges through two complementary research thrusts: "Advanced Visualization Practice" and "Visualization Design Tools."

In our Advanced Visualization Practice thrust, we take steps to extend the process of interactive visualization to work effectively with complicated multivariate motion datasets. We present brushing and filtering operations that allow users to perform complicated filtering operations in a linked-window visualization while maintaining context in complementary views, including two-dimensional plots, three-dimensional plots, and recorded video. We also present the concept of "trends," or patterns of motions that behave similarly over a period of time, and introduce visualization elements to allow users to examine, interact with, and navigate these trends. These contributions help to implement Shneiderman's information seeking mantra (Overview first, zoom and filter, then details-on-demand) in the context of collections of motion datasets.

During our work in Advanced Visualization Practice, we realized that there were a lack of tools enabling visualization developers to rapidly and controllably create and evaluate these visualizations. We address this deficiency by our Visualization Design Tools thrust, introducing the idea of visualization creation interfaces where users draw directly on top of data in order to effect their desired changes to the current visualization. In an application of this idea to streamline visualizations, we present a sketch-based

streamline visualization creation interface, allowing users to create accurate streamline visualizations by simply drawing the lines they want to appear. An underlying algorithm constrains the input to be accurate while still matching the user's intent. In a second application of this idea, we present a Photoshop-style interface, enabling users to create complicated multivariate visualizations without needing to program. A colormap painting and dabbing algorithm allows users to create complicated colormaps by drawing colors on top of a colormap; an algorithm determines the desired locality of the user's input and updates the colormap accordingly. These interfaces show the potential for future interfaces in this direction to expand the visualization design process to include users currently excluded, such as domain scientists and artists.

Through these two complementary thrusts, we help to solve problems preventing newer datasets from being fully exploited. Our contributions in Advanced Visualization Practice solve problems that are impeding the visualization of motion datasets. Our contributions in Visualization Design Tools provide a blueprint for the creation of visualization interfaces that can enable all users instead of just programmers to contribute directly to the visualization design and creation process. Together, these set the stage for future visualization interfaces to better solve our biggest visualization challenges.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Visualization is the process of visually communicating data. Visual communication has been used for over 40,000 years starting with Paleolithic cave art and continuing through the modern day. During the Renaissance, Leonardo da Vinci produced what could be considered some of the first scientific visualization with his illustrations in his notebooks of topics ranging from anatomy to turbulent fluid flow. With the creation and development of computers, scientists could finally create visualizations where visual parameters (e.g., line width, color, texture) correspond directly to measured values without the potential inaccuracies involved in human image creation.

As technology has continued to become more powerful, larger and more complicated datasets have emerged, and new visualization techniques and designs are needed to take advantage of this data. In this dissertation, we have helped solve this problem via two complementary research thrusts: "Advanced Visualization Practice" and "Visualization Design Tools" (as shown in Figure 1.1. The Advanced Visualization Practice thrust encompasses the work we have done on creating new algorithms to allow the interactive visualization of complicated motion datasets. Based on the work we did in this thrust,

Figure 1.1: Our research aims to advance the field of visualization with two complementary research thrusts. The "Visualization Practice" thrust involves the development of new algorithms and visualization systems to examine larger and more complicated datasets more easily. "Visualization Design Tools" involves the development of visualization creation systems where visualization parameters can be intuitively and directly adjusted to better and more rapidly explore the entire design space.

we realized that the lack of appropriate visualization design tools is hindering the development of better visualization tools. Our second research thrust seeks to address this problem by designing tools to allow for interactive and direct visualization creation.

1.1 Advanced Visualization Practice

Currently, the field of visualization is struggling with difficult problems applying visualization to current datasets. Uncertainty visualization focuses on the task of bringing the information from a boxes-and-whiskers plot to highly dimensional data. Visualization at scale focuses on adapting current visualization techniques and creating novel techniques to address the problems facing users with ever-growing datasets. This can be compounded by datasets that are either very highly dimensional or that have a temporal component to them. Many existing visualization techniques fall apart with the uncertainty, scale, dimensionality, or temporal components of these datasets, and need to be fundamentally rethought. In this dissertation, I've focused on the difficulties of visualizing motion datasets, creating novel visualizations to help address the problems that temporal data bring to visualization.

Motion visualization requires nearly all visualization techniques to be fundamentally reconsidered when compared to a static-time visualization. In motion visualizations, not only must users be able to determine the state of what's being shown at one point in time, but they must be able to relate how what is occurring at the present relates to what has happened at any other point in time. This is related to but different from the problem of visualizing many instances of a dataset at once: with multiple instances, there is no 'before-after' relation that needs to be maintained between two different instances, nor is there a need to encode the (temporal or other) distance between two instances.

Motion visualization will continue to increase in importance in the future. Many motion datasets are already being generated in fields ranging from meteorology to astrophysics. However, with the continued advancement of technology, there is the possibility for motion data to be used where previously only single values were used before. As an example, currently when a patient is examined by their doctor to assess neck pain, a series of measurements are made of the angular range of motion of the neck in a few different planes. Currently, this is done because it is both easy to measure and easy to examine. Hardware advances (such as the Microsoft Kinect and Occulus Rift) could be used to capture this data, but in order to draw useful conclusions from this data, motion visualization is needed.

Motion visualization is a hard problem. Shneiderman's Information Seeking Mantra ("Overview first, zoom and filter, then details on demand"[5]) provides a general structure for visualization systems, but the application of these three steps to motion visualization is a significant research challenge. Providing an overview of an entire motion or an entire collection of motions requires either simplifying the data to non-temporal summary metrics or performing significant processing on the data to determine what

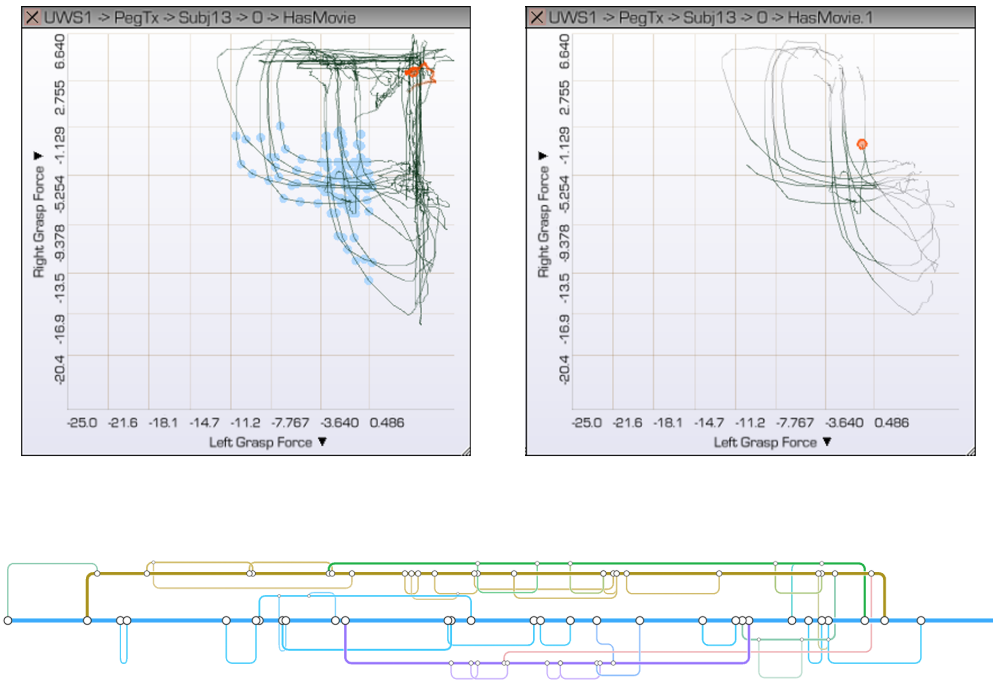

Figure 1.2: Novel visualization elements created to enable Shneiderman's Information Seeking Mantra in motion visualization tools. Zoom and filter is enabled by allowing the user to drag to create a selection (top left) and then automatically determine temporal context in the resulting filtered data (top right). At bottom, a subway map-style visualization shows the major patterns that emerge and change over time in a biomechanical motion dataset.

patterns exist in the data and need to be shown. Zooming and filtering can be performed as simply selecting a single motion instance or temporal sub-range to examine, but this prevents users from focusing on a non-temporally contiguous or non-spatially contiguous subset of the data.

We approach the problem of motion visualization by providing solutions to the hard problems in fulfilling Shneiderman's Information Seeking Mantra for motion datasets. In every case, our aim was to provide an interaction that maintained as much context as possible, so that users were able to continuously navigate through the dataset and

effectively filter without getting lost. In this dissertation, we present two key contributions that enable this information seeking process in motion datasets: an intelligent selection expansion algorithm for the 'zoom and filter' operation, and a time-varying clustering interface to provide an overview of a motion dataset.

The improved selection tool shown in Figure 1.2 (upper) and further discussed in Chapter 3 allows the selection of regions of data that are temporally related to a region of interest in a two-dimensional scatterplot. When a user makes a selection, such as that shown in Figure 1.2 (top left), and creates a filtered view, the system adds the data points directly added (Figure 1.2 top right) in a solid color, and adds the required context in a light gray. To do so, an algorithm finds temporally connected components within the user's selection and extends each component forward and backward in time until either a time limit is reached or until a discontinuity is detected in one of the data variables. With this selection technique, users are able to select regions of interest, and then examine that selection with the required temporal context. When added to a linked-window visualization system, this allows for the incredible power to see in a video view the context associated with particular patterns that emerge in a two-dimensional scatterplot. Without this technique, selections such as the one shown in Figure 1.2 would be impossible with a simple rectangular selection.

The subway map-style visualization shown in Figure 1.2 (lower) and further discussed in Chapter 4 shows a novel visualization built upon the concept of *trends*, and designed to provide users with an overview of potentially complicated motion datasets. Trends are groups of motions that behave similarly for a period of time. These trends can last for the entirety of the motion dataset, or can last for only a short portion of time. Motions may be a member of only one trend, or they can change their trend membership to reflect the way that motions can be similar to different groups at different points in time. By detecting and drawing all of the trends in a motion dataset, users get an

overview of the structure of a given dataset. In contrast to conventional clustering techniques, trends change over time, may not exist for the entire duration of a motion dataset, and motions can change their trend membership, more accurately capturing the way that motions can behave similarly for a stretch of time and then diverge. Users use this visualization to navigate through the dataset, and it is constantly on screen in order to allow users to maintain their context in complicated motion datasets.

Through both of the contributions outlined above, we advance the state of the art of motion visualization. Both of the techniques above help to address limitations in current tools for examining the larger and more complicated motion datasets that users are increasingly finding themselves working with.

1.2 Visualization Design Tools

Visualization tools are tools that produce a visual representation of a provided dataset. Interactive visualization tools allow users to adjust parameters as the system is running and to get updated visual results. Visualization design is the process of determining what visualization algorithms and parameters to use, and what user interactions to allow. Visualization design is similar to other design processes in that there are general rules about what works better than other options, but the resulting systems and resulting use cases are too complicated to determine a single optimal solution.

Tools that accelerate the visualization design process can have a profound effect on visualization design. All visualization tools go through a design process that to some extent relies on trial and error and iterative design. Visualization design tools have the promise of not only allowing current visualization researchers to explore and evaluate their ideas more rapidly, but also to allow entirely new classes of users to contribute to the process of visualization design. What visualization ideas would a world-class

Figure 1.3: A streamline visualization creation tool allows users to create visualizations by drawing the streamlines they want to appear. The tool then interprets the user's input in the context of the underlying data to determine the appropriate streamline to add to the visualization.

graphic designer be able to create, given the right tools?

Currently, visualization design is hampered by the long iteration times needed to evaluate different design decisions. Some decisions, such as choosing fonts, color ramps, and visual sizes of visualization elements can be tested and evaluated fairly rapidly. Others, such as which visual elements to use, where to place them, and what variables to show in which ways, take a much longer time to evaluate.

In order to attempt to ease the visualization design process, we have worked on tools that allow for visualization creation without requiring intimate knowledge of programming or of a visualization pipeline. Instead, we adopted interfaces and metaphors that have been used for decades (such as the layer-based interface used in Photoshop) or millennia (such as a sketching-based interface).

In this dissertation, we show that visualization design tools can help to allow users

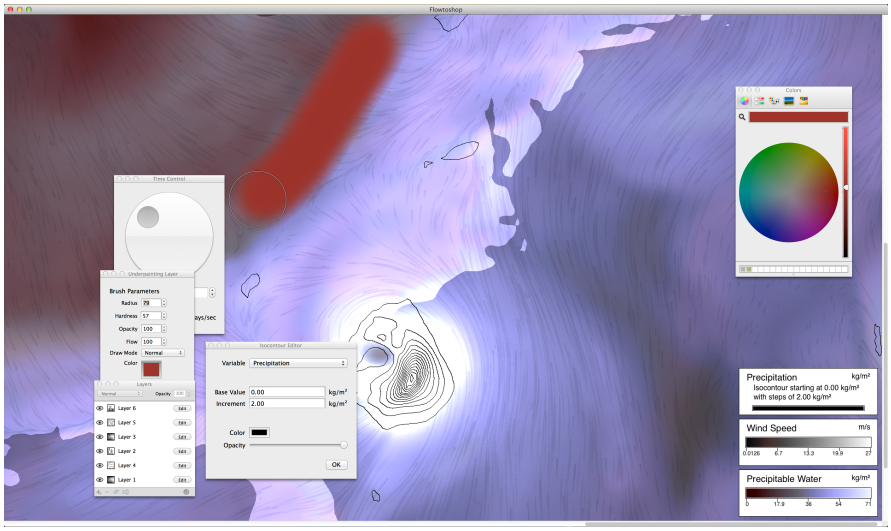

Figure 1.4: A Photoshop-style interface allows for the creation of complicated motion visualizations of two-dimensional scalar and vector fields. In this case, a weather visualization shows Hurricane Arthur from the 2014 Atlantic Hurricane Season.

to create accurate scientific visualizations. We present an interface that allows users to sketch a streamline visualization and have the computer automatically refine their input to be an accurate visualization (Figure 1.3). We also present a visualization design system where users can use a Photoshop-style interface to design sophisticated two-dimensional visualizations of time-varying scalar or vector fields (Figure 1.4). Perhaps more importantly, through these two tools, we identify the most promising next directions for research into the next generation of visualization design tools.

1.3 Thesis Statement

The central thesis for this dissertation is: "New, carefully designed, visualization techniques can improve our ability to analyze complex motion collections, and new visualization design tools, accessible to both artists and scientists, can provide more efficient and creativity-supporting workflows for developing these difficult-to-design visualizations."

1.4 Document Structure

The structure of this dissertation is as follows. Chapter 2 overviews work related to both the visualization practice and visualization design tools research thrusts. Chapter 3 presents a surgical training visualization tool that allows users to focus their attention on sub-sections of a motion instance. Chapter 4 presents a trend-based visualization tool that algorithmically performs time-varying clustering on a motion dataset. Chapter 5 begins the discussion of visualization design tools, and presents an interactive streamline visualization tools where users directly draw streamlines on top of the data and the system corrects these streamlines. Chapter 6 presents a multivariate, time-varying visualization tool for visualization design exploration and visualization-based storytelling. Chapter 7 concludes this dissertation and discusses where this research as a whole can head in the future.

Chapter 2

Related Work

The existing research related to the work in this dissertation falls into two large categories: Motion Visualization, and Visualization Design. Within both of these research categories, there are well-established techniques, but there are also unsolved problems whose resolution will allow for new datasets to be explored, understood, and used.

2.1 Motion Visualization

Motion visualization is a broad term that applies to visualizations that depict objects or data as they move and change over time. These datasets can be generated in fields ranging from fluid dynamics to sports medicine. Many techniques that are second-nature for visualizing static datasets are woefully inadequate for motion datasets, either requiring some changes to work for motion datasets, or requiring a completely new visualization strategy in order to allow analysis for these data.

2.1.1 Motivation

Motion visualization used in many fields and for many types of data. These data can be in the form of biomechanical motion trajectories, where evolutionary biologists are interested in the detailed relative motion of bones, such as in a pig's chewing cycle [6]. By contrast, some motion datasets are continuous scalar or vector fields that change over time, and can be explored by museum visitors [7].

Beyond these examples, motion visualization has been useful in medicine [8], sports [9], animation [10], migration studies [11], traffic analysis [12], and fluid dynamics [13]. For some of these datasets, there are multiple instances of a motion happening that need to be compared to find patterns of behavior. In others, there are multiple regions in space or time that need to be compared to determine how the system as a whole behaves. All of these tasks can be performed by simply watching and rewinding a video, but the amount of time, mental effort, and attention needed to draw useful conclusions makes this impractical for anything but the smallest datasets. Visualization techniques are needed that are tailored specifically toward motion visualization.

2.1.2 Motion Visualization Techniques

The simplest possible motion visualization is an animation of a process occurring over time. Despite its simplicity, this method has several advantages for sufficiently short motions: many regions of motion are readily apparent, the overall behavior of the system over time can be determined, and it is generally simple for novice users to understand the system. However, as soon as the system becomes too complicated to be shown in a single frame of an animation, or as soon as multiple motion instances need to be compared, this technique breaks down.

Even for motions where a single object is moving through space, an animated display

might not be appropriate. In visualizing the underwater behavior of humpback whales, Ware et. al. found that providing an annotated 3D trace of a whale's motion allowed, as an example, for easy visual identification of different foraging patterns [14]. This static visualization allows for spatial relationships to be drawn between samples at different times which would be invisible in a standard animation.

Animated motion displays additionally suffer a loss of information when the display is paused or stopped for any reason. A number of techniques have been developed to augment a stopped display with additional marks that show at a high level the motion of different parts of the dataset [15].

When only a small number of motions need to be compared to each other, techniques such as multiple miniatures or overplotting can be used. Multiple miniatures refers to displaying multiple, separate, full visualizations of each individual motion adjacent to each other. This allows direct comparison by looking back and forth between the views, but suffers from requiring the user to divide their attention and from the decreased size of each individual visualization. In overplotting, multiple motions are drawn on top of each other, typically semi-transparently. By doing so, motions can be compared directly in a shared space, but with an increasing number of motions, this display can get too complicated and cluttered.

More advanced techniques relying on various forms of clustering or filtering can allow motion visualization tools to both provide an overview of a dataset while also allowing users to drill down into the dataset. In [10], Bernard et. al. present a visualization system for navigating datasets of motion-captured human motions. Using clustering and dimensionality reduction, users can examine the dataset in multiple levels of detail. At the highest level, the dataset can be viewed as transitions between a few poses; at the lowest level, individual poses within a group of similar poses can be examined to determine what motions they are a part of.

Some recent research has focused on better describing the motion visualization design space. In [16], Coffey et. al. developed a taxonomy of motion visualization types, identifying two orthogonal axes: Time and Space. Both of these axes can be either static, animated, or user-controlled. Ware's whale visualization incorporates a static time representation (no animation occurs over time) and an interactive space representation (the user is free to navigate and move through the dataset).

2.1.3 Unsolved Motion Visualization Problems

Despite years of research, many unsolved problems make motion visualization design a difficult process. Motion datasets by their very nature can not be completely visible at a single instant, either requiring summary techniques to create a static representation, or requiring animation to advance through time. Depending on the dataset, even the process of creating a static representation or overview of a motion dataset can be an unsolved research problem. Without an overview visualization, users are less able to mentally model the dataset's structure, making further analysis more difficult and time consuming.

Once a static representation or other simplified representation of a motion dataset is found, the challenge then becomes enabling users to narrow their focus to a specific subset of the data. This filtering can be in extracting specific regions of a spatial dataset, specific instances of a motion being performed, or specific points in time where one or more motions behave in a particular way.

Finally, there is the unsolved question of how to allow users to easily perform a comparison between multiple subsets of a motion dataset. This is related to the problem of providing an overview of the entire dataset in that both require the depiction of a set of data at a time, and that both frequently need to summarize data in order for it to be easily representable. In contrast to depicting an overview of the dataset, comparison

requires ensuring that data that is *not* part of the subset being examined isn't visually prominent, and that significant differences between the data are visually apparent.

2.2 Visualization Design

Visualization design includes many aspects of a visualization system, including, among others, the interaction techniques provided, the choice of colors and textures to map variables to, and the positioning of elements on screen. In most visualization systems, making these decisions requires programming the decision result. This prevents users who have valuable insights (such as scientists, graphic designers, etc.) from contributing to the visualization design process.

2.2.1 Motivation

Visualization design is a difficult process, and current process of visualization design does not take proper advantage of the knowledge that resides both in users and in visual design experts. Users know what aspects of their dataset are interesting, what are uninteresting, and have some ideas on how they would like to see data presented. Visual design experts have years of experience in communicating visually, and the current design process fails to take advantage of this expertise.

Critique provided by visual design experts has been shown to mirror the results of a quantitative visualization study for 2D vector field visualizations.[17] These experts clearly have useful knowledge and experience to bring to the visualization design and evaluation process, and yet the current state of visualization creation tools precludes their involvement. While working on the visualization tool discussed in Chapter 4, we collaborated closely with a trained graphic designer. She was able to generate a wide range of visualization ideas, and yet all of her designs existed only as Adobe Illustrator

files, and not as visualizations tied to an underlying dataset.

How would the process of visualization design be changed if these users and experts were able to directly create visualizations instead of needing a programmer to implement their ideas?

2.2.2 Visualization Design Techniques

A number of different visualization design strategies are available. Some researchers have focused on how to bring artists' insights into various components of a pre-selected visualization environment, such as allowing artists to design their own 3D glyphs by creating a few exemplars[18] or by letting them specify the shading for a volumetric transfer function by providing images of shaded shperes[19]. Others have investigated the role that artists can have in the visualization design process with varying interaction techniques, ranging from VR drawing to conventional art supplies[18, 20]. Through all of these, artists and traditional artistic techniques were able to help the visualization design process, but the tools to enable this interaction were lacking.

When implementing visualizations, the most powerful visualization creation environment is a general purpose programming language paired with a graphics library such as OpenGL. In this environment, any visualization element can be created and new visualization elements can be devised. However, with this extreme power comes the requirement that users have sufficient programming experience and spend a large amount of time implementing a visualization system and the individual components they are interested in. Existing software libraries such as VTK[1] significantly simplify this process and implement common algorithms, but this visualization creation process still relies on programmers typing in code to create a visual result.

[1] Available at `http://vtk.org`

Tools such as Paraview[2] allow users to create visualizations without any programming experience, but it is an extremely technical tool that typical users would require significant training on. Other tools such as Tableau[3] allow users to interactively and intuitively create visualizations, but is aimed more at facilitating business analytics than scientific data visualizations.

Some researchers have implemented a more intuitive interactive tool where users can control a large number of parameters using sliders to attempt to find an optimal design solution.[21] The interface allows users to adjust the parameters of the visualization, but users are limited to adjusting only the explicitly included visualization elements, and are unable to, for example, add an additional colormap to the visualization.

2.2.3 Unsolved Visualization Design Problems

While tools exist that allow visualization creation without programming, these tools are either too specialized, or require users to use tools that are designed from a data processing standpoint, rather than from a visual design standpoint. Artists are able to come up with creative and potentially useful ideas, but require programmers as the ultimate gatekeepers to implement these ideas.[22] This limits the artists, and prevents them from rapidly iterating on their ideas, since a true evaluation requires a significant time investment by a programmer. A tool that afforded users all of the visual expressive freedom of tools such as colored pencils and paper, but allowed these designs to be directly tied to an underlying dataset would revolutionize the visualization design process.

[2]Built on top of VTK, and available at `http://paraview.org`
[3]http://www.tableausoftware.com

Chapter 3

Visualizing Surgical Training Databases[1]

This chapter begins the discussion of the "Advanced Visualization Practice" research thrust introduced in Section 1.1 with an interactive exploratory visualization tool using a database of surgical data at the scale of hundreds of task performance instances. We introduce several novel visualization techniques that have been iteratively designed to be responsive to the requirements of surgical data analysis: (1) A smart interactive brushing (selection) interface that automatically extends a selection in time so as to provide appropriate context to interpret the data after zooming. (2) A multi-view video overlay strategy for analyzing time-series data corresponding to coordinated bimanual actions. (3) An animated 3D tool trace visualization that encodes 14 data dimensions via variation in the texture, color, and form of 3D data glyphs. Finally, we present results and discussion of use case studies of the tool conducted with two groups of practicing surgeons.

The specific data examined is of hundreds of surgeons performing peg transfer tasks

[1]This chapter is based on work published in [23].

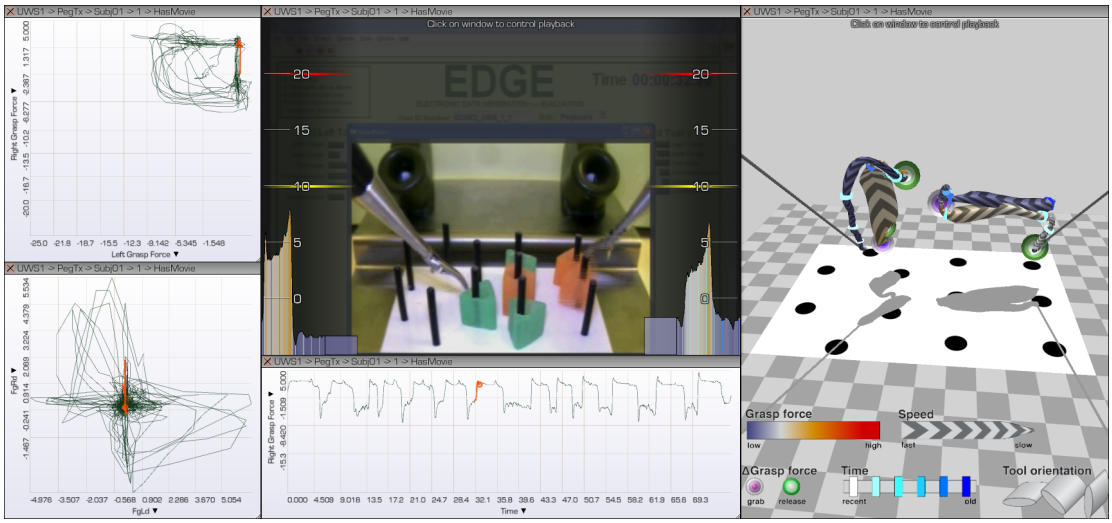

Figure 3.1: Visualizations to explore multidimensional data collected during laparoscopic surgical training exercises. 2D plots support analysis and filtering. Videos augmented with data overlays provide contextual information required to interpret trends, and 3D visualizations support analysis of multivariate tool path data within a spatial context.

– one of five standard tasks within the ubiquitous Fundamentals of Laparoscopic Surgery training module [24]. Previously, machine learning techniques have been applied to these quantitative performance data, for example, classifiers can be used to distinguish novice and expert surgeons (e.g., [25]). In order to bring visualization to bear on this data, we must begin to address specific challenges in the areas of organizing and querying databases of movies, forces, positions, high-level descriptions of tasks, and other data, and in recognizing that the results generated during data exploration are themselves useful data. A second core challenge is supporting visualization of coordinated complex bimanual activities based on multivariate, time-varying data. Scaling visualization applications to support aggregate analysis across hundreds or thousands of instances of complex time series data such as these remains a major visualization research challenge.

3.1 Related Work

In addition to the related work mentioned in Chapter 2, specifically Section 2.1, this work builds on work on visualizing surgical datasets. An overview of this work is presented below.

3.1.1 Surgical Visualization

Computer graphics and visualization have been applied previously to several problems in surgery. Fueled by advances in computer graphics, virtual reality, robotics, haptics, visualization, and related hardware technologies, surgical simulation tools have experienced tremendous growth recently and have reached new levels of realism [26]. To complement simulators, research in visualization has contributed many new tools to assist in understanding imaging and related data. Representative work can been seen in the results of the IEEE Visualization 2010 contest, which centered on a neurosurgical planning task [27]. Real-time clinical surgical visualization tools have also been explored both in research contexts [28] and more recently in commercial consoles such as the da Vinci surgical robot from Intuitive Surgical Inc. Although many simulator tools are able to report summary statistics at the end of a trial, we know of no previous exploratory visualization systems designed to examine aggregate multidimensional high-resolution surgical performance data.

With respect to the goal of scaling visualizations of time-varying data to work across a large set of motions rather than just a single motion, our work is similar to recent multi-view visualizations that have been applied to studying biomechanics in animals [29]. Surgical applications raise new questions in terms of incorporating video into multidimensional visualizations, developing abstract 3D representations to encode bimanual tool trace data, and designing visualizations that can convey appropriate feedback to

surgeons.

3.2 Surgical Data Visualization System

This section describes in detail the surgical data that we aim to visualize and the features of the exploratory visualization application we have developed.

3.2.1 Multidimensional Surgical Data and Block Transfer Task

The surgical data that we visualize were captured by the EDGE platform (an instrumented set of laparoscopic tools able to capture tool positions and forces) and consist of 300 laparoscopic block-transfer training tasks performed by more than 50 surgeons. The specific variables recorded are: Cartesian tool position, tool rotation around shaft axis, tool grasp force, tool grasper angle, and time. In all, 428,924 frames of multi-variate data were collected. The data were captured at 30Hz, and if watched in sequence, it would take almost 4 hours to watch all of the video collected during these trials. Additional quantities (e.g., velocity, block transfer events) can be derived inside of the visualization tool based on the input data.

The goal of the standard peg transfer training task captured in this dataset is to pick up and move a series of six blocks from the pegs on one side of a board to the pegs on the other and then move them back again. The blocks must be picked up by one hand's laparoscopic tool and then transferred in the air to the other hand's tool. Each tool has a gripping device on the end resembling a small alligator clip. The training task must be completed without errors (e.g., dropping a block) under a given time threshold in order to be considered successful.

3.2.2 Overview, Zoom and Filter, and Details-on-Demand for Surgical Data

Inspired by Shneiderman's information seeking mantra [5], Figure 3.1 illustrates how several complementary interactive data views are used to support an exploratory visualization process. Traditional 2D plots support graphing any dimension of the data against any other dimension. The plot in the bottom center displays right grasp force over time. The two plots on the left show left vs. right grasp force and the time derivative of these quantities. Additional displays play back video sequences recorded during the trial (middle window in Figure 3.1) and show detailed 3D tool trace information (right window in Figure 3.1).

All views are linked such that multiple windows can refer to the same set or subset of data and interactions in one view impact the others. Brushing techniques enable users to select a subset of data displayed in the current window and then filter the data based on this selection, activating new visualization windows to display just the relevant subset of data. These selections can be loaded and saved, allowing repeatable data analysis

3.2.3 Smart Brushing Interface

The basic interaction for controlling selection and data filtering is a click and drag operation. Based on the mouse motion, a selection rectangle is created, and any datapoints that fall within this window are selected. After a selection has been made, a right click activates a contextual menu, which contains options for creating new visualization windows tied specifically to the selected datapoints.

We developed several extensions to this basic brushing interface, used in many visualizations, in order to support the new application to surgical data. For example,

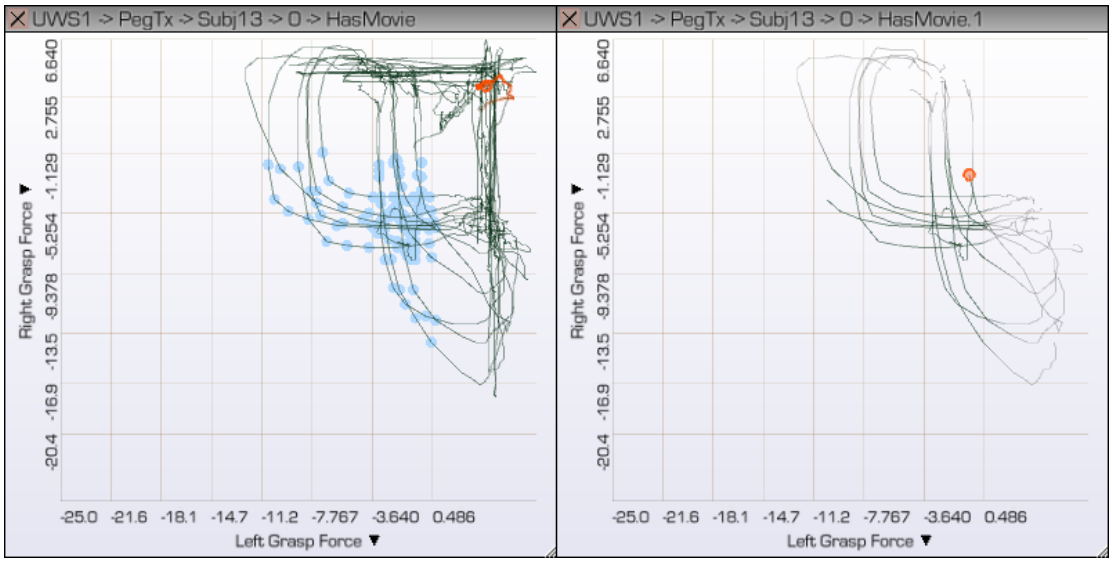

Figure 3.2: A brushing interface intelligently extends mouse selections forward and backward in time to capture appropriate context. The user highlighted the blue points in the left window; the curves in the right widow show the selection as interpreted by the interface.

Figure 3.2 (left) shows a plot of left grip force vs. right grip force. The user has selected an interesting subset of the data that usually corresponds to the sub-task of transferring the block from one hand to the other. To understand what is really happening at these points in time, the user would typically activate a video window, which would then play the sequences of frames that have been highlighted. The problem in this situation is that the frames that have been highlighted represent very small time windows – just the split second when the block is transferred from one hand to the other. Watching just these frames of video provides very little context to interpret the data.

Our solution, illustrated in Figure 3.2 (right), is to automatically extend the selection forward and backward in time to provide enough context to interpret the data. In the situation illustrated in Figure 3.2 (right), the selection shown would be impossible to make without the addition of this smart interface; without it the user would not be able

to select the appropriate subset without also selecting additional, irrelevant frames.

We have experimented with several techniques for determining how far to extend the user's selection in time. The simplest is to expand by a fixed time, e.g., 0.5 seconds. More data-driven approaches are also possible. In Figure 3.2, for each sequence of frames selected, the algorithm traverses the frames before and after to identify discontinuities in the data (e.g., a jump from high to low force) and the selection endpoints are set to correspond to these instances.

3.2.4 Video Overlays

Abstract data plots alone typically do not provide enough context to interpret the data. Videos, on the other hand, are particularly valuable for providing contextual information, but do not provide concrete quantitative data. One of our design goals is to tightly integrate these two complementary data displays.

With this motivation, we have overlaid animated data plots on top of video replays, bringing quantitative data displays as close as possible to the videos that help us interpret them. Figure 3.3 shows two videos, one corresponding to a trial performed by an expert, the other, a novice. The bar plots overlaid on the left and right of both windows depict the instantaneous grip force applied by the left hand (on the left side) and the right hand (on the right). To further illustrate patterns in the use of force, the display also includes an animated line graph, which captures the last several seconds of force measurements for each hand.

The augmented video visualization also includes aural and visual indicators activated when tool forces exceed either a warning or a danger threshold. The audio tones are localized to the speaker corresponding to the tool that exceeded the threshold. Using the same thresholds, the corresponding side of the video is tinted yellow (warning) or red (danger) to indicate excessive use of force. Currently, these features are used for

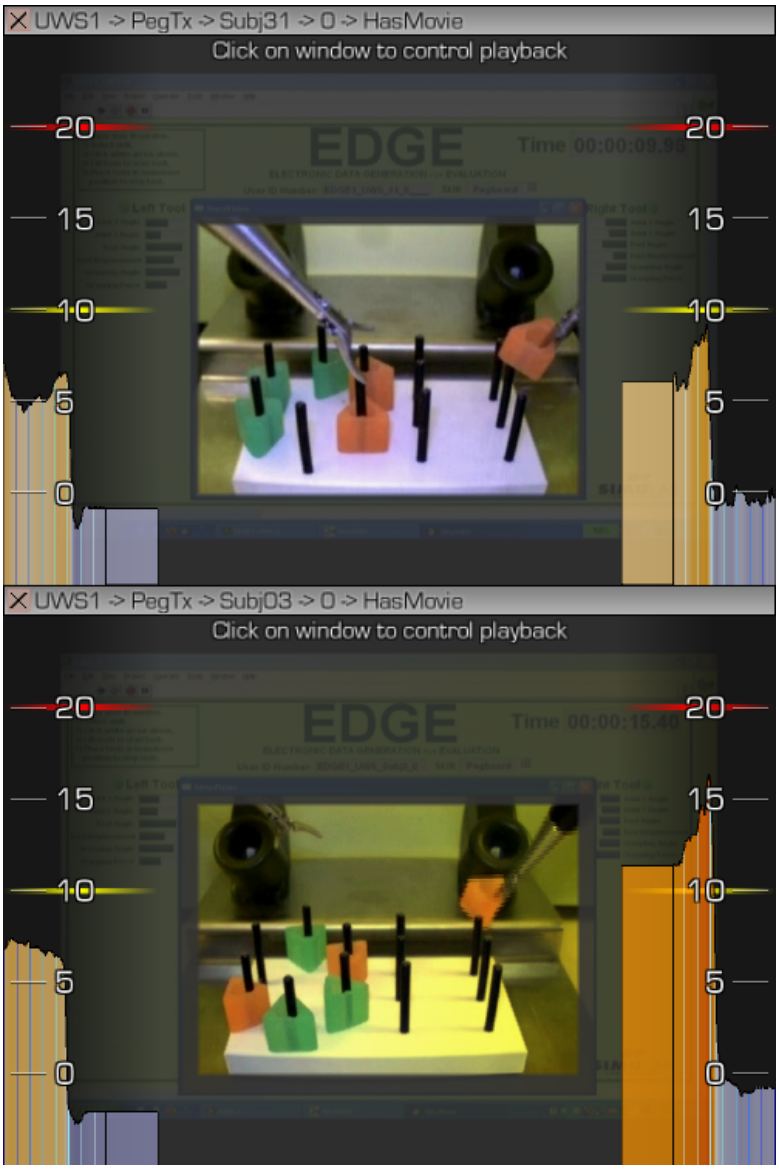

Figure 3.3: Video overlays show grip force for each hand. The display is animated and synchronized with video playback. The large bars on each side of the window indicate the current forces, and the history of change in forces is plotted as an animated graph extending off the screen from each bar. The high force visible in the bottom image relative to the top is indicative of a difference observed between novice and expert surgeons.

post-hoc data visualization, but we believe there is great potential to utilize similar visualization techniques for real-time formative feedback (see Section 3.3).

Together, these augmented video visualizations enable two new modes of analysis. First, patterns in the use of force from one hand to the other (i.e., bimanual coordination) can be observed by looking at the data displayed in a single video. Second, comparisons can be made across different tasks, either multiple tasks performed by the same surgeon or tasks performed by multiple surgeons.

3.2.5 Multivariate 3D Visualization

It is challenging to identify interesting correlations between multiple data variables and to understand how these variables change relative to 3D tool position. To facilitate this through visualization, we have developed a series of exploratory 3D visualization techniques. Our approach is inspired by the success of the textured and color-coded 3D glyphs introduced by Ware et al. for visualizing the diving and feeding activities of whales as they move through the ocean [14]. Our data are similar in representing motion paths through a 3D space, but a number of important differences have informed the design of the rendering algorithm depicted in Figure 3.4. First, the surgical tools often return to the same positions as the surgeon works, so path traces overlap each other frequently. Second, the data can be noisy, and there are portions of little motion. Finally, in this surgical application, we are interested in analyzing coordinated motions, that is, motions of two tools working in tandem.

The legend in Figure 3.4 describes our current mapping from data to visual. A textured ribbon traces out the 3D tool path. The ribbon's orientation, texture, and color are all tied to data variables, and additional glyphs are included in the space surrounding each ribbon. The color mappings match those used in the video overlay visualizations, making it possible to make associations between the two types of displays

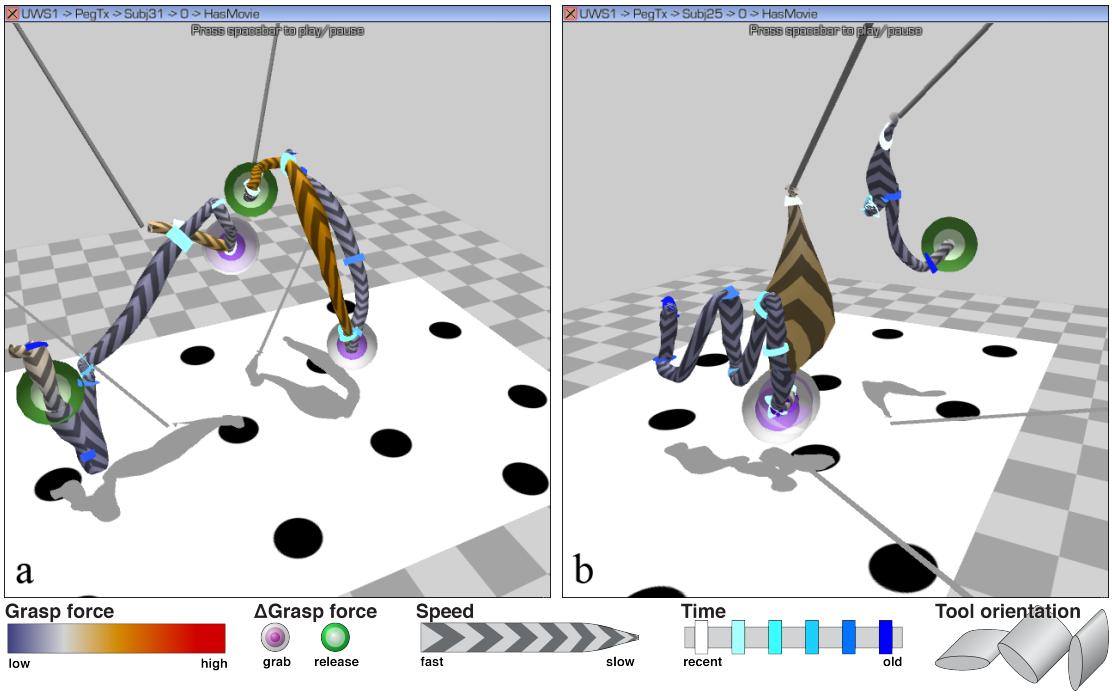

Figure 3.4: In (a), the surgeon is moving relatively quickly and decisively. He/she recently set down a block on the left of the screen, then picked up one near the middle, and have just transferred it in the middle of the screen. The blue colored rings allow temporal correlation, while the spherical glyphs provide a depiction of where the transfers occur. In (b), the surgeon is having trouble judging the depth dimension. The ground plane allows you to see where the block they are trying to pick up is, and the pickup glyphs show you where they eventually did pick up the block.

when animated synchronously.

Our design began with a rendering algorithm nearly identical to Ware et al.; we then refined the strategy over several iterations to work more effectively for this surgical application. Initially, the ribbon form displayed in the visualization was a fixed width, which has the potential advantage of facilitating depth judgments. We found, however, that this design was very sensitive to changes in position and orientation of the tool that resulted from noisy data or muscular jitter. These portions of the data created discontinuities in the ribbon form that ended up being more visually salient than other, more important regions of the data. Our solution first applies a lightweight smoothing filter to the data then maps the ribbon width to speed (using a non-linear mapping). This design has several advantages. It emphasizes the form of the ribbon in areas of decisive motion while diminishing its visual saliency in areas of little motion. It also helps to simplify the geometry in the display so that the forms of overlapping motion traces are easier to discern.

Several other important design decisions are reflected in the display in Figure 3.4. Shadows on a textured ground plane are used to convey spatial relationships. Rather than displaying the entire tool motion over the course of a trial, we display only the last five seconds of data. When the display is animated, this produces something like a motion blurring effect. These animated displays are often useful to establish a clear connection to the video data, however, the 3D visualizations are often easiest to interpret when the playback is paused and mouse-based camera controls are used to navigate around the scene. Two visual strategies are used to link the two tool path traces together, helping to understand how motion is coordinated between the two hands. First, a series of bands are drawn around each ribbon as tick marks indicating the passage of time. Second, glyphs are utilized to denote discrete events, such as discontinuities in the derivative of force data, which tend to correspond to grabbing or releasing a block.

We have found that including these discrete markers on both tool traces dramatically improves the viewer's ability to understand how the motion of one tool relates to the other both spatially and temporally.

3.3 Use Case Studies with Practicing Surgeons

To assess the utility and potential impact of the system, we conducted data analysis sessions together with expert practicing surgeons and the engineers embedded in their teams to develop new surgical training tools. Two groups were used for the evaluation, one group from each of the two institutions collaborating on our long-term project. Each group included one senior surgeon leader and two engineers. Both senior surgeons have a background in urologic surgery, including minimally invasive techniques. One of the groups also included a second surgical fellow with similar areas of expertise.

Our approach included structured preselected data analysis and interactive visualization tasks as well as opportunities for free-form exploration, discussion, and other qualitative feedback. Before each session, together with the collaborators, we brainstormed a set of data analysis questions, including several that are challenging or impossible to answer using current tools. Two of the most important questions were: How does the use of force change between more- and less-experienced surgeons? And, are there any skill-based patterns in the velocity used when approaching a target? These and other questions provided structure and a relevant "real-world" context for the use-based evaluations.

Then, during each data analysis session, which lasted one to two hours, we began with a 10-15 minute explanation of the tool. We then moved into an example analysis of the use of force, which we introduced by using the brushing interface to select subsets of data. We then turned to the question of velocity patterns, and finally, depending on

the interests of the surgeons, to other questions we had brainstormed previously and/or to analyses of new hypotheses generated during the session. While using the program, a think-aloud protocol was used to gather qualitative data. The next sections report on specific feedback collected and discoveries made.

3.3.1 User Feedback and Assessment of Potential Impact

Feedback on the tool was consistent between the two groups of surgeons. The clear high-level response was that the tool is a drastic improvement over current practice. One surgeon commented, "This makes objective assessment of surgical skills so much easier than going through the video." Similarly, both groups emphasized that the ability to filter through data and create (even traditional 2D) plots quickly is a drastic departure from current analyses that are conducted using tools such as Matlab or Excel. Here, the analysis is immediate. A surgeon also remarked, "The beauty is that I know what I want on the left and I [can go directly from these 2D plots to the linked video] on the right." Thus, we believe one critical advance in this system is the ability to link video data with interactive 2D and 3D plots, which enables a rapid style of data navigation that was not previously possible.

3.3.2 Discoveries Made and Trends Confirmed

One interesting discovery enabled by this tool is a phenomenon that we have so far described as "sympathetic grasp-release" – we found that when the surgeon applies force to one tool (i.e., picking up a block) there is often a small drop in the force applied to the tool controlled by the other hand, even though the second tool is not currently performing any task (Figure 3.5). Neither group of surgeons had heard of such a pattern before, but once discovered, we were able to within minutes find several (¿ 10) instances of the pattern within the dataset. This led to a series of follow-on

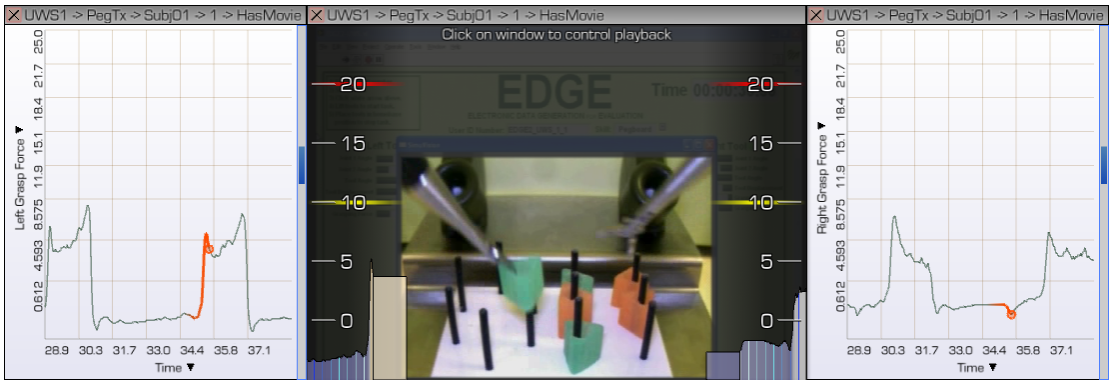

Figure 3.5: A sympathetic grasp-release pattern. When the left tool grabs on to the block to pick it up from the peg board, the force on the right tool is momentarily lessened. The left plot corresponds to the left tool, and vice versa. In the two two-dimensional plots, the orange circle corresponds to the current video frame, and the highlight shows the past few seconds of data. The large drop in the left plot is the grab; the small dip in the right plot is the sympathetic release

questions and hypotheses: (1) Does this pattern correlate with reported skill? (2) When might this be useful in surgery (e.g., one surgeon suggested that releasing pressure at one location on a tissue while grabbing another location might sometimes decrease the possibility of damage to the tissue)? (3) When might it be detrimental (e.g., another surgeon suggested that while cutting with one hand, it may be important to maintain a constant grasp force with the other in order to hold the tissue taught to make a controlled cut)? (4) Can the same pattern be seen in data for other tasks and in more realistic surgical situations (e.g., both groups wondered if the pattern would apply in suturing and other tasks)?

We are encouraged by this example result since it demonstrates that the visualization tool can be used to obtain valuable new insights. It also provides evidence that surgeons can relate patterns elucidated by the visualization with their own deep domain knowledge to generate new hypotheses – one of the most important goals of exploratory visualization. In this particular case, we first noticed the sympathetic grasp-release

trend while using 2D force-over-time plots together with a linked augmented video to look at individual transfer events. We observed that the force plot for one hand was varying at times when we could see in the video that it was the other hand that was actually picking up the block. Now that we know exactly what to look for, we are confident that we can find this pattern via more traditional data analyses, but without the coordinated linked data displays, we doubt that we would have ever discovered this pattern.

Other specific trends identified with the visualization include an initial spike in force when a grab is first made, followed by a decrease to a lower, more constant force. We noticed this pattern while developing the video overlay visualizations and assumed that it was representative of a bad practice. However, during the use case studies, the surgeons in one group provided us with a different interpretation. This trend may be reflective of an experienced surgeon practicing "respect for tissue" (i.e., the surgeon quickly reduces force applied once a successful grab has been made).

The problem of moving a surgical tool too far beyond an intended target is well known, but we found that the visualization system can provide new insight in understanding these situations as well. The depth dimension is typically the most difficult to align, and the 3D visualizations provide surgeons with new virtual viewpoints (e.g., a top-down view) to understand 3D tool path data.

3.3.3 Feedback on Specific Visualization Features

The 3D multi-variate visualization features described in Section 3.5 are the most experimental visual aspect of the system. To make these displays most easily understood, the tool also includes a simplified 3D tool trace visualization (Figure 3.6). These 3D views are valuable because they provide useful camera angles for understanding 3D tool paths.

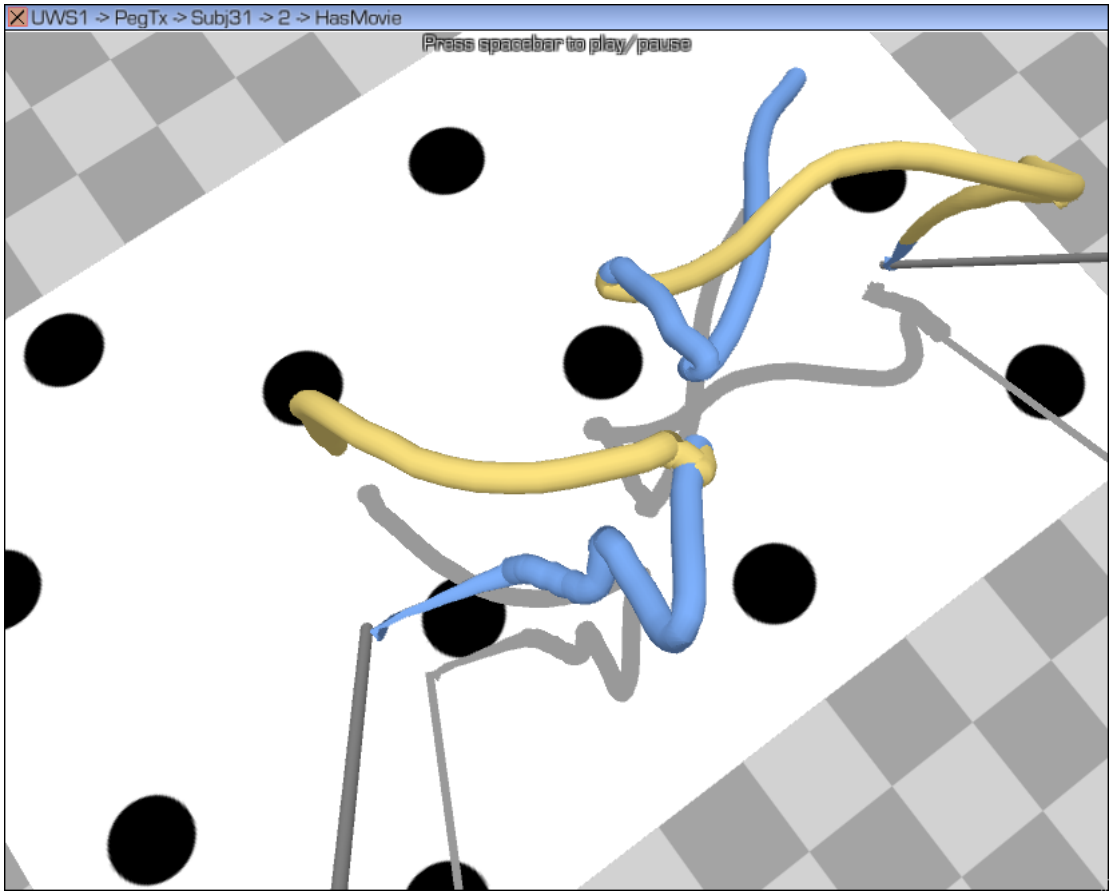

Figure 3.6: Simplified 3D tool path visualization. Yellow tubes indicate regions where the tool's grasp force is above the "grab" threshold and correspond to where the tool is grasping a block. Blue tubes indicate the tool's grasp force is below the grab threshold. This simplified visualization allows depth information to be more easily seen than the more sophisticated 3D visualization shown in Figure 3.4.

Beyond the simplified views, feedback on the glyph-based multi-variate visualization focused on new applications for this type of display. Both groups of surgeons recognized that all the information collected appears in these 3D views and that the mappings from data to form could potentially be adjusted based on importance. One surgeon suggested using this display for goal-oriented training, where a surgeon is presented with a readily understandable visual task, such as "reduce the red areas", then uses the multi-variate visualization to learn to adjust force, speed, etc. in order to achieve this goal.

The smart brushing interface (Section 3.2.3) for selecting subsets of data then calling up new data plots and videos for that subset is one of the most important features of the tool. All of the surgeons found this to be an improvement over current methods of examining this data. However, surgeons want this type of selection to work as naturally as possible. One surgeon wanted to be able to say, "I want 12 transfers", or perhaps even "I want the 6 left-to-right transfers for this participant." Although it is possible to select transfers like these using the current interface, there is room to improve the interface to make it more natural for a surgeon's mindset, as opposed to, say, for an engineer. This is an exciting direction for future user interface research.

3.3.4 Feedback Related to Iterative System Design

Several aspects of the current design of the system have been refined significantly based on collaborative iterative design. During the use case studies, we confirmed the improvement in these features.

The 3D visualization has been refined considerably to now include a pegboard image, which establishes a spatial context for the data. Shadows have been simplified to serve only as spatial cues rather than also encoding data. The glyphs used to represent "grab" and "release" events have also been refined to more concisely convey these events.

The 2D video overlay visualizations are the most immediately understandable views

in the system, thus we have been motivated to make these even more useful and to explore their potential for use in follow-on systems designed for real-time formative feedback. With this in mind, the surgeons and engineers in one group evaluated the tinting as an undesirable feature, in operating environments, the surgeon's visual channel is already nearly saturated, so feedback via that modality is likely not desirable. The audio feedback was judged by these users as a more promising avenue for future applications to real-time visualization for training.

3.4 Conclusions

This chapter presented several new visualization techniques in the context of a linked-window visualization tool, including a smart interactive brushing interface that provides context to interpret the data, a multi-view video overlay strategy, and an animated multivariate 3D tool trace visualization. This visualization tool was applied to surgical training data that included over 100 motion instances. Based on the results of discussions with practicing surgeons using this tool, these visualization techniques help to improve their ability to analyze complex motion collections such as surgical training datasets, supporting the central thesis of this dissertation.

Chapter 4

Trend-Centric Motion-Set Visualization[1]

Continuing in the "Advanced Visualization Practice" research thrust, this chapter introduces a new strategy for analyzing motion collections called *trend-centric motion visualization*. This strategy differs from previous motion visualization strategies (e.g., [31, 23, 6, 10]) in that it focuses on *trends rather than trials*. Through feedback from our collaborators (musculoskeletal biomechanics researchers studying the human spine) we know that a trend-centric strategy makes sense; for example, our closest collaborating domain scientist reports that basing his analysis on trends rather than trials is "absolutely correct" in terms of matching the mindset with which he would like to conduct his analyses. The technical challenge we address in this chapter is redesigning data processing and visualization algorithms to support trend-centric visualization.

Our work is motivated by the needs of scientists studying motion collections. Specifically, these scientists need to analyze how biokinematics differ in healthy people vs. people with injuries or people in various stages of disease, and they need to study how

[1]This chapter is based on work published in [30].

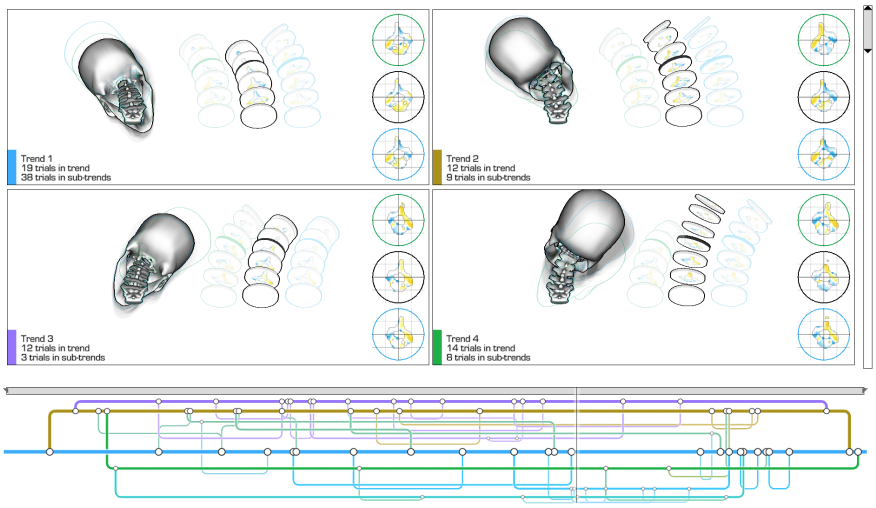

Figure 4.1: This trend-centric motion visualization helps scientists analyze human neck kinematic data in a new way. Rather than analyzing a single specific trial in detail, what scientists would most like to do with these data is to identify similarities and differences across the entire collection of trials in order to classify motions as healthy vs. non-healthy or suggest appropriate courses of treatment. Trend-centric motion visualization helps scientists accomplish this analysis by: 1. identifying the trends in a motion collection; 2. displaying the trends in the form of the 2D timeline shown at the bottom of the figure; 3. displaying the 4D anatomical context needed to interpret the trends, as shown at the top of the figure; and 4. combining all these elements into an exploratory visualization system that supports interactive selection and querying.

biokinematics differ pre- and post-treatment. Since a motion collection's trials all are based on the same type of exercise, it is common practice to use time-warping or sub-sampling to align all the trials to a common time axis [32]. Unfortunately, despite warping, the spatial complexity of the motions together with the large size of motion collections make it impossible to do a thorough comparative analysis of these data using current statistical or visual strategies.

This research makes several contributions. (1) We introduce an algorithm for detecting a motion collection's trends via time-dependent clustering. This algorithm does not simply cluster together whole trials that are similar; rather it identifies *trends* –

sequences of consecutive frames that are similar across multiple trials. For example, the algorithm might detect that a single trial follows a trend that is consistent with healthy neck motion during the first half of the trial, when the neck flexes, but follows a trend indicative of severe disc degeneration in the second half, when the neck extends. (2) We introduce a novel graphical technique inspired by transportation network maps for visualizing a motion collection's trends, including how trends come and go over time (Figure 4.1, bottom). (3) We contribute an illustrative graphical technique for visualizing a group of trials as a median with variance (Figure 4.1, top). (4) We demonstrate how widgets based on these two techniques can be combined into a single interactive tool, and apply this tool to analyze two different motion collections. The first is a set of 200 simulated motions of the human spine – this simulated data enables us to verify that the tool works as expected on known data. The second is from a recent cadaveric study of human spinal kinematics at different degrees of disc degeneration – we report on how our domain science collaborators used the tool to analyze these data. (5) Finally, we report on insights from our iterative design process, which included collaboration with both a traditionally-trained graphic designer and the domain scientists.

4.1 Related Work

4.1.1 Visualizing Motion

This work builds on previous work in motion visualization. An overview of related work in this field can be found in Section 2.1. In addition, this work builds upon the idea of visualizing whole collections of motions at once, as well as additional visualization techniques, such as illustrative rendering. Overviews of the related work is presented below.

4.1.2 Visualizing Motion Collections

Only a few recent visualization tools specifically address the problem of analyzing motion collections. Motion Explorer [10] combines a number of information visualization widgets (e.g., interactive dendrograms, node-link diagrams of motion graphs, a visual querying interface) with a pose-based clustering technique to help users analyze human motion capture databases as employed in computer game and movie animation.

We also use multiple interactive linked views, but our user requirements differ significantly. Motion Explorer helps users identify variations in stick figure poses based on motion capture data for about a dozen marker points distributed across the human body. Thus, the motion variations that it helps to detect are relative large (e.g., defensive vs. offensive boxing moves). In contrast, our scientists study detailed changes in the 3D spatial relationships, coordination of movement, etc. for multiple bones each of which have complex 3D shapes. This led us to analyzing trends rather than poses and utilizing detailed animated 3D views rather than thumbnail images.

Two recent visualization tools specifically help scientists understand motion collections. The first is applied to analyzing surgical training data [23], and the second to analyzing chewing motions in pigs, a topic studied by evolutionary biologists [6]. Like our work, both utilize multiple coordinated windows to convey multidimensional motion data. However, rather than manually selecting patterns in the data (e.g., via interactive brushing in a parallel coordinates plot), our tool first automatically classifies trends within the motion collection; then, these trends serve as the building block for interactive data analysis. Since individual trials can belong to different trends over time, this trend-centric strategy is a major departure from current clinical and research practice in biomechanics.

4.1.3 Other Related Visualization Techniques

Illustrative rendering and visual abstraction have been used to convey motion, for example, adding cartoon-like motion lines to show the trajectory of moving objects [33], see also [34, 35, 36, 37]. These and other illustrative visualization techniques (e.g., [38]) effectively use visual bandwidth by simplifying unneeded detail while emphasizing important detail. Our work adds two components missing from current illustrative renderings: depicting trends and variance across motion collections instead of a single motion, and driving these depictions with experimentally-collected motion data. Conceptually, visualizing differences between multiple motions is similar to uncertainty visualization [39, 40], which attempts to visualize not only the primary variables in a dataset but also the "error bars" around them. Our visualization techniques provide something similar to "box and whisker plots" for 3D motion data.

Our work also has some similarity to the broad topic of ensemble visualization. Ensemble visualization has typically been applied to sets of simulations run on high-performance computing architectures, as in a parameter study for engineering design [41] or an exploration of what if scenarios for disaster management [42]. We share a common goal of visualizing a collection rather than a single instance.

4.2 Trend-Centric Motion Visualization

Our trend-centric motion visualization strategy begins by clustering the multidimensional data (e.g., distances, velocities, and bone orientations) in each trial. Clustering is first computed separately for each frame. Then, the clusters are connected to form coherent "trends" over time. This section begins by describing the trend-detecting algorithm in detail. Then, we present the series of 2D and 3D interactive visualization widgets developed to enable scientists to analyze trend-based motion collections.

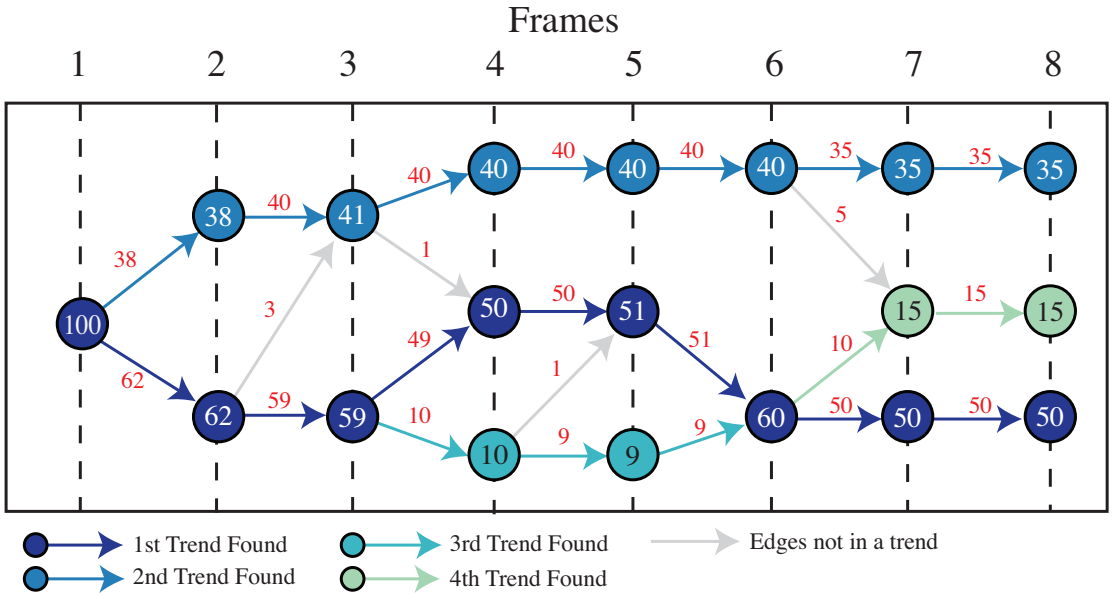

Figure 4.2: Trends are identified using a directed graph data structure and a longest path algorithm that gives preference to the strongest trends present in the data.

4.2.1 Detecting Trends in Motion Collections

Although clustering is well studied within the visualization community, the problem of identifying trends that come and go over time within a set of many related trials is different than typical applications of clustering. Our goal is to identify important sequences of similar consecutive frames that exist across multiple trials in the collection. An example scenario that we want to detect is one where many trials all follow a similar pattern up to a certain point in time (e.g., until the neck reaches close to a maximum angle of flexion) and only then do significant differences begin to emerge in the data (e.g., healthy necks follow one trend, moderately diseased follow another, and severely diseased follow yet another). To accomplish this, we require a localized approach to clustering that operates at the frame-to-frame level.

Our algorithm assumes that each trial in the motion collection is comparable along a common time axis. This is not a limiting assumption. Note that since the example

applications discussed later in the chapter deal with either synthetic data or *in vitro* data collected with the aid of a robotic device, each trial in these motion collections is already of the same duration. For *in vivo* studies, similar, temporally-aligned data can be derived using time warping, as is common practice in biomechanics analysis [32].

Feature vectors are defined at each frame of each trial, but the definition of the components of the vector depend on the dataset used. For example, in our *in vitro* application, the feature vector includes the components of the frame-to-frame helical axis that describes the movement of L4 relative to L5. Once feature vectors are computed for every frame of every trial, the values are normalized per component to lie in the range $[0, 1]$.

The algorithm's first step using these data is frame-wise trial clustering. Each frame is considered in isolation, and trials are clustered based on the similarity of their feature vectors. Our implementation uses a threshold-limited agglomerative clustering, and we found that it is a useful practice when working with, often noisy, experimental data to include neighboring frames for this step. For example, in our implementation, the feature vector at a given frame is calculated using a sliding window to include values at neighboring frames.

The result of Step 1 is a set of one or more clusters for each frame of the motion data, as illustrated in Figure 4.2. Here, each cluster is represented as a circle. Notice that just one cluster is detected at Frame 1, and this single cluster contains all 100 trials in the motion collection. At Frame 2, two clusters are detected; one contains 38 trials, and the other contains 62. At frame three, a few trials switch from the lower to the upper cluster. Then, at Frame 4, three clusters are detected, one containing 40 trials, one containing 50 trials, and one containing 10 trials. Note that, the same clustering threshold is used for all the frames, but depending upon the data, this may result in a different number of clusters identified at each frame.

The second step is to connect these clusters to form trends that exist across multiple frames by identifying correspondences between the clusters at Frame i with those at Frame $i+1$. A graph data structure is used to accomplish this. We treat the clusters identified in step 1 as the vertices of the graph and connect the vertices with weighted, directed edges, as shown by the arrows in Figure 4.2. Edges are added to connect every cluster detected at frame i to every cluster detected at frame $i+1$. The edge weights are set to the number of trials that are common between the two clusters, and zero weight edges are omitted. For example, in Figure 4.2, look at the edges connecting the clusters at Frame 3 to the clusters at Frame 4. The top cluster in Frame 4 contains 40 trials, and we determine (by simply comparing the id's of the trials contained in each cluster) that all 40 of these came from the top cluster in Frame 3. The middle cluster in Frame 4 contains 50 trials, and we determine that 49 of these came from the bottom cluster of size 60 in Frame 3. The bottom cluster in Frame 4 contains 10 trials, and these all came from the bottom cluster in Frame 3.

With this graph complete, the trends can be found in a way that gives preference to the strongest trends in the data by iteratively calculating the longest path on the graph. The overall longest path found is identified as the first trend. Then, the vertices and edges that belong to this trend are removed from the graph and a new longest path is found. This second longest path is the second trend, and so on. Since this graph is trivially topologically ordered by frame index, each longest path calculation can be completed in linear time.

Each trend is defined by three data attributes: (1) the index of the frame at which the trend starts; (2) the index of the frame at which the trend ends; and (3) for each frame that the trend is active, the set of trials that belong to the trend. Our algorithms sometimes also consider the "strength" of each trend. We define the *frame-wise strength* of a trend as simply the number of trials contained in the trend at that frame. By

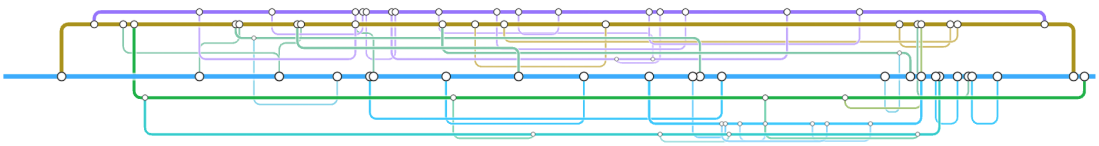

Figure 4.3: The trend timeline illustrates how trend split and converge over time by using a style visual style similar to transportation network maps. Time progresses horizontally, and trends are separated vertically.

extension, the *overall strength* of a trend is the sum of all its frame-wise strengths.

4.2.2 Visualizing Trends on a Timeline

Figure 4.3 shows trends visualized with the horizontal axis representing time and different trends separated vertically. A trend can branch into several weaker trends or several trends can join together to form a new, stronger trend. When a new trend is formed, we define its main parent as the trend contributing the most trials to the new trend; this is easily determined from the graph's edge weights. If the trend ends before the end of the motion sequence, this implies that the trials contained within it have all joined other trend(s); in this case, the trend to which the majority of the trials move is considered a second parent.

Each trend is rendered as a line with a unique vertical position. The first, strongest trend is assigned to the middle of the viewport. Then, a screen-space quality metric is minimized to find the best positioning of each successive trend. The quality metric $Q(y)$ is given below, where y is the proposed position, F is the set of frames for which the trend is active, T_f is the set of already laid-out trends that exist at frame f, y_t is the position of trend t, s_t is the strength of trend t, and ϵ is a small positive constant to avoid division by zero. We use $y \in [0, 1]$ to correspond to the height of the viewport,

and set ϵ to 0.05.

$$Q(y) = \sum_{f \in F} \sum_{t \in T_f} \frac{\epsilon \, s_t}{\epsilon + |y - y_t|} \tag{4.1}$$

In practice, we found that the overall trend strengths for the biokinematic datasets that interest us tend to follow a bimodal distribution, with a relatively small number ($\tilde{5}$) of strong *main trends* plus a much larger set of weaker *sub-trends* with not much in-between these two groups. We use this insight to our advantage in assigning colors and line weight to the trends.

We test each new motion collection algorithmically to see if the trend strengths do indeed follow a bimodal distribution. If they do, we divide the trends into the two classes: *main trends* and *sub-trends*.

Trend colors are then assigned based on strength and on the colors of a trend's parents (if they exist). All main trends are assigned a color in the CIELAB color space on the plane $L^* = 60$, and on a circle centered on $(a^*, b^*) = (0, 0)$ with radius 110[2]. These coordinates provide distinct, vividly saturated colors for the main trends. Then, all sub-trends with at least one parent are assigned a desaturated, lightened version of the average color of the parent(s).

The line weight (width) of each trend line is similarly based on the trend's strength and computed using the equation

$$w_i = (0.7\sqrt{s_i/s_1} + 0.3)w_{max}, \tag{4.2}$$

where i is the index of the trend, w_{max} a the maximum line width based on the size of the viewport, and s_1 is the strength of the strongest trend. The square root allows for differences between trend strength to be seen across the full range of strengths.

[2]After conversion to sRGB, negative components are clamped to 0

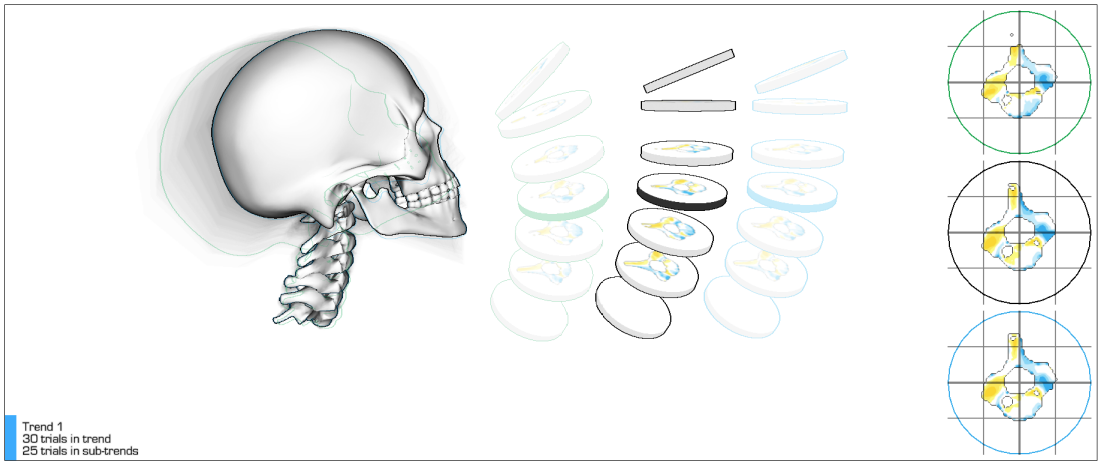

Figure 4.4: Trends are displayed using a 3D visualization technique that functions like a box-and-whiskers plot, enabling scientists to understand both the key characteristics of the trend's median motion as well as a sense of the variation that exists across all of the trials in the trend.

Trends are drawn at their assigned vertical position using the computed color and line weight and a horizontal line that spans all of the frames at which the trend is active. Finally, curved lines are added to connect each trend to its parent(s) when appropriate.

4.2.3 Visualizing Trends with Appropriate Anatomical Context

The trend timeline overviews a motion collection's trends, but interpreting the motions also requires seeing the anatomical context. Building upon existing motion visualization techniques, our 3D visualization includes techniques useful for analyzing individual trials (e.g., contour lines, and axes of rotation), but also must depict a set of related motions.

To address this problem, we introduce a median plus variance visualization technique that, as shown in Figure 4.4, juxtaposes three trials in the same 3D view. First, the most representative trial of the trend, its median trial, is rendered. Then, two additional trials are rendered, one plus and one minus a standard deviation away from the median.

The median trial is computed taking into account all the available data recorded for

each frame (positions, orientations, distances, speeds, etc.). First, the feature vector defined in Section 4.2.1 is computed for each frame of each trial in the trend. Then, the mean feature vector is computed for each frame. The median is computed by comparing each trial's sequence of feature vectors to the sequence of mean feature vectors. The trial with the minimum summed squared difference is considered the median. Note that although we compute an "average motion" during this calculation, at the request of our collaborators, we deliberately display the median motion rather than the mean, since the mean is not guaranteed to be physically possible, while the median is an actual observation.

The trials that are +/- one standard deviation away from the median are computed using the same feature vectors. A value for the standard deviation is calculated from the distribution of each trial's total distance from the mean trial; this distance is based on the feature vectors summed over all frames of the trial. With a standard deviation calculated, it is then possible to identify the subset of trials that fall within one standard deviation of the median. From this subset of trials, the two trials with the largest mutual dissimilarity are selected as the trials that are 'plus' and 'minus' one standard deviation away from the median.

Finally, an underpainting is added to the visualization to convey the full range of a trend's trials. All of the trend's trials are drawn in a flat, semi-transparent color. By analogy to a box-and-whisker diagram, the three 3D renderings of the bones are similar to the box in that they prominently display values for something like the first, second, and third quartiles in the data. The underpainting is similar to whiskers in that it displays in a less prominent manner the full range of the data, including any outliers. The final result is an animated rendering, as shown in Figure 4.4 that conveys not just the characteristic motion that defines the trend but also a sense of the variance within the trend.

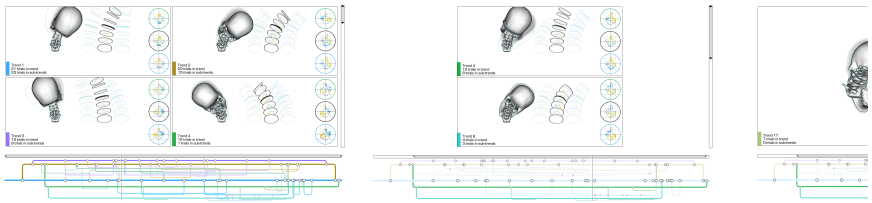

Figure 4.5: A typical analysis sequence begins with an overview of all trends in the motion collection (left). Then, scientists select a subset of interest (middle). Then, the range bar is adjusted to further focus the visualization by displaying just a subset of the selected trends in the 3D view; here, analysis is focused on a sub-trend of interest (right).

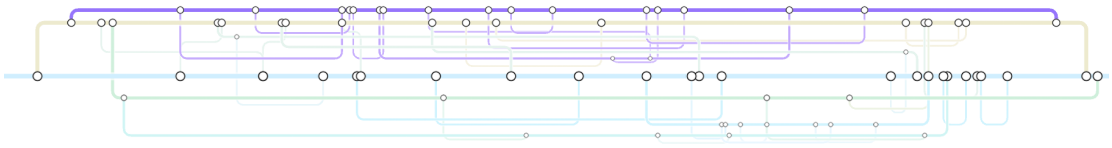

Figure 4.6: When a single trend is selected by clicking, the trend timeline visualization updates to convey the provenance of each of the trials in that trend. Here, a trend near the top of the motion sequence is selected – notice how line width is used to convey where the trials in this trend came from and are going "upstream and downstream" from the selected trend.

4.2.4 Interactively Exploring a Complete Motion Collection

Figures 4.1 and 4.5 illustrate how the trend timeline and the 3D trend visualizations are combined into an interactive tool. The trend timeline at the bottom of the screen not only provides an overview of how all the trends in the motion collection come and go over time, but also provides a way to drill down through the data to examine specific trend(s) in detail. Clicking on a trend selects the trend and its subtrends for further analysis. Each of the selected trends is then displayed in its own 3D visualization viewport on the top portion of the screen. A range bar on the right side of the visualization enables users to narrow the subset of trends to display in 3D windows, which is useful during the most detailed stages of analysis.

To complement these interactive zooming and filtering techniques, scientists use

mouse and keyboard interactions inside both the 3D and 2D windows to explore the data more closely. Scientists can rotate, pan, and zoom within the 3D views. By default, the median trial is fully shaded, but this can also be adjusted interactively by simply hovering the mouse over either the plus or minus one standard deviation renderings. Mouse hovering, in both the 2D and 3D views, also causes the timeline visualization to indicate the provenance of all of the trials in that trend, as in Figure 4.6.

4.3 Applications and Evaluations

Our research has been conducted in collaboration with a group of musculoskeletal biomechanics researchers. Both of the applications described in this section are derived from real data analysis problems faced in their research. In the next section, we describe a verification study, which uses simulated data that we created in order to verify that the visualizations work as expected under known conditions. Although we generated this data synthetically, the type of motion is motivated specifically by a recent study conducted by our collaborators. The second application is based on actual experimental data collected by our collaborators during a cadaveric study of 18 human lumbar spine specimens.

4.3.1 Verification Study

When patients with neck and/or back pain visit a doctor they are often asked to perform physical exercises from which measurements are made. The simplest is a flexion/extension exercise – the patient bends his or her neck as far forward and as far backward as possible, and the maximum angles reached without pain are recorded. In the biomechanics research community, there is now a push to incorporate data from

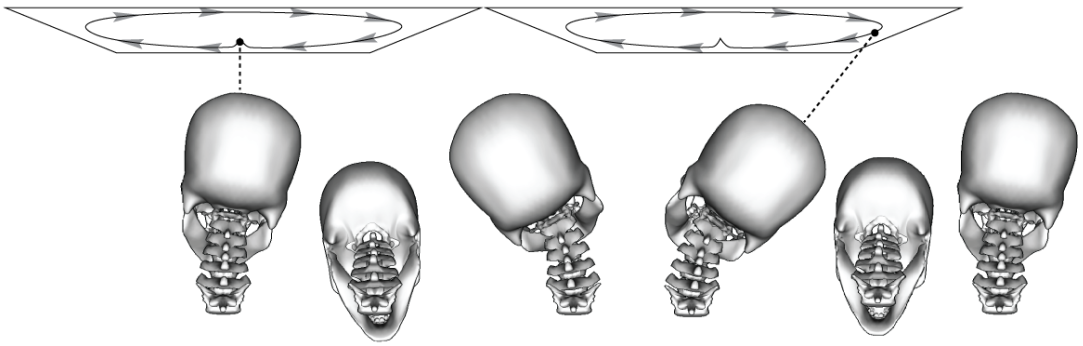

Figure 4.7: The simulated skull motion. The skull is constrained to point toward a point on a spline on a plane above the skull. The spline moves the skull forward, where it then rolls counterclockwise and returns to the resting position.

more complex exercises into clinical decision making. One such exercise is neck circumduction (Figure 4.7). As an example, a recent study measured head-to-torso kinematics using a lightweight linkage device. The superior end was worn on the head and inferior end was positioned over the T1 vertebra and snugly attached to the torso. The study contained over 100 patients, and each patient performed the neck circumduction exercise three times at two different sessions, for a total of 600 trials [43]. Researchers are excited about these datasets and their potential impact on healthcare because they capture much more information than current 2D descriptions of motion commonly used in current clinical practice. Even more exciting than the external tracking data described above is the ability to collect and simulate the internal motions of the vertebrae. These data can be collected experimentally using biplane fluoroscopy (e.g., [44]) or simulated using a neck kinematics model (e.g., [45]); however, since this type of data has only recently become available, there is not yet consensus on what patterns will be most useful for clinical decision making.

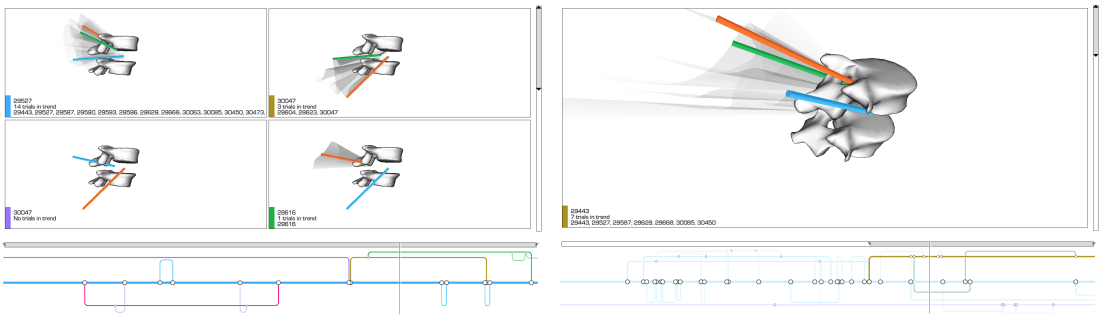

Figure 4.8: Two screen shots of the trend-centric motion visualizations used to analyze data from a recent *in vitro* study of 18 lumbar spine specimens. The visualizations highlight the helical axes computed for the motions and the 3D paths that these axes sweep out over time. The motion of the vertebra is more subtle in this example, as compared to the example pictured in Figure 1. Here, the variation in the helical axis orientation is the most important factor to analyze. So, in this example, we apply the median plus variance visualization technique described earlier directly to the helical axis data.

Synthetic Data

With this understanding of the type of motion datasets that investigators are now collecting and struggling to analyze, we developed a verification study based on our own simulated dataset of neck motions. Although our simulation method is inspired by models published in the biomechanics community [45], we must begin our discussion by emphasizing that our goal in this application was *not* to develop a new physically-accurate neck kinematics model. Rather, we set out to generate a synthetic dataset of physically-plausible motions of the vertebrae with programmatically controlled variance. This enables us to verify that the visualization techniques work as expected on synthetic data.

The synthetic data simulate the motion of the spinal column from the skull to the T1 vertebra. We use the Bullet physics engine, and each bone is modeled as a rigid body connected to neighboring bones via springs. To calculate vertebra positions and orientations, we fix the position and orientation of T1 and apply a constraint to the

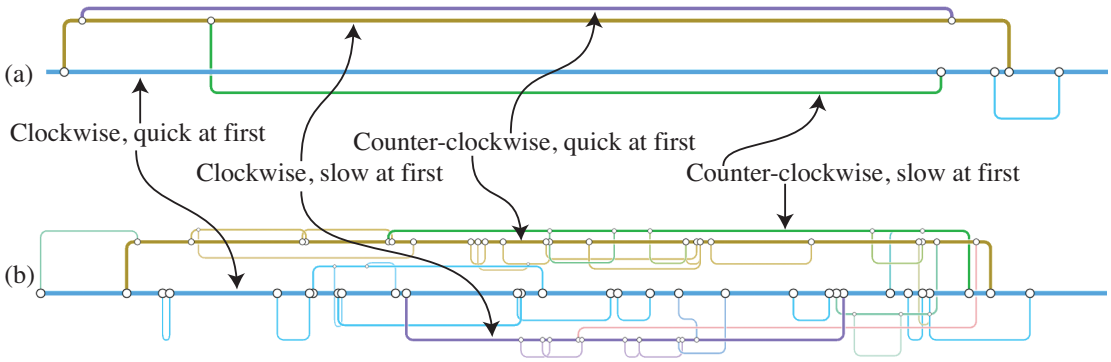

Figure 4.9: Clusters identified in two motion collections differing only by the amount of noise present in the data. The trials in (b) have 4 times as much noise applied. This has a clear impact on the trends detected, in particular a number of new "sub-trends" are detected; however, the major expected patten in the synthetic data is still visible: all trials start following a single trend branch into 4 main patterns with some noise added, then come back together to follow a single trend.

skull to force its "up" vector to point at a moving target located on a plane 1.3 meters above the skull. The spring constraints then adjust the position and orientation of the skull and the vertebrae between the skull and T1, producing a physically plausible pose for each bone. The characteristics of each simulation can be adjusted by adjusting the moving target's path. To create an initial path, we captured an example motion experimentally using an optical motion tracker, and then simplified the data to convert it into a spline with 11 control points. By adjusting the spline path programmatically, we generate motion collections where each motion is slightly different.

We generated two different synthetic motion collections using this strategy. Table 4.3.1 summarizes the characteristics of each. Each contains 100 trials, which we break into four subgroups based on relatively significant changes in the spline path, such as changing the direction of the movement (clockwise vs. counter-clockwise) and the speed profile (quickly at first vs. slowly at first). Additional variation is then added by applying a Gaussian noise function to the positions of the spline's control points

Table 4.1: Simulated motion datasets.

Dataset	Num. Motions	Direction and Speed	SD of Noise	Speed Variation
Dataset 1	40	clockwise, quickly at first	0.1 m	2%
	20	clockwise, slowly at first	0.1 m	2%
	27	counter-clockwise, quickly at first	0.1 m	2%
	13	counter-clockwise, slowly at first	0.1 m	2%
Dataset 2	40	clockwise, quickly at first	0.4 m	8%
	20	clockwise, slowly at first	0.4 m	8%
	27	counter-clockwise, quickly at first	0.4 m	8%
	13	counter-clockwise, slowly at first	0.4 m	8%

and applying random variations to the speed at which the moving target traverses the spline. Since the moving target is positioned on a plane 1.3 meters above the skull, it sweeps out a path that covers a relatively large area (about 3 x 4 meters). In Dataset 1, the Gaussian noise added to the control points has a standard deviation of 0.1 meters, so significant variation can be observed. In Dataset 2, the standard deviation is 0.4 meters. The variations in Dataset 2 are significant enough that some trials include major changes from the expected pattern, such as a "hitch" in the motion or a short sequence where there is a reversal in the direction of rotation.

Visualization Results and Analysis

Figure 4.1 shows an example trend-centric visualization of these data, and Figure 4.9 shows a detailed comparison of the trendlines that are generated for each of the two synthetic datasets. The feature vector used for clustering in these examples was composed of four physiologically meaningful quantities for the motion of the skull relative to T1: (1) the amount of forward/backward bending, (2) the amount of left/right bending, (3) the positional speed, and (4) the rotational speed. All of these were normalized to lie on the range (0,1) to ensure equal weighting.

As shown in Figure 4.9 (top), the trend line produced for Dataset 1 closely matches the trends we expect to discover given the properties of the data described above. For

the majority of the frames, four distinct trends are clearly identifiable. If we look at the composition of these trends, we find that each trend corresponds to one of the subgroups in Table 4.3.1. All of the trials also cluster into a single trend at both the beginning and end of the entire sequence. Each simulated exercise begins from the same neutral upright pose, then the head lowers so that the eyes would be looking straight down. Then, the rotational motions begin. The timing of the branching point observed in the trend line matches the transition to rotational movement. Likewise, at the end of each exercise, the head returns to the same neutral upright pose; thus, each trial is very similar at this point in the sequence.

Figure 4.9 (bottom) shows a trendline produced with the same clustering threshold, but this time for Dataset 2, which includes far more substantial variation between trials. This provides an interesting point of comparison. The major trends in the data are still identified, but additional "sub-trends" are also identified as the variation causes individual trials to sometimes join or detach from a trend over time. These sub-trends do not cross over major boundaries. For example, notice that there is a major branch early on in the sequence that divides the trials that run clockwise from the trials that run counter-clockwise. New trends and sub-trends come and go after this point, but clockwise and counter-clockwise trials are never characterized as belonging to the same trend until the end of the motion sequence at which time all of the trials follow a similar pattern of returning to the neutral upright pose.

We interpret these results, augmented with our observations from interactive use of the tool, as a verification that the visualization system functions as expected with well-understood data, displaying major trends clearly and merging into a single trend when appropriate. When the variation across the trials increases, the complexity of the trends identified also increases, but major patterns remain visible and smaller patterns provide useful pointers to interesting nuances of the data.

4.3.2 Experimental Study and Expert User Feedback

The second application is to an *in vitro* biomechanics study conducted by our collaborators. The study compares motion of the lumbar spine for 18 human spine specimens donated through an anatomy bequest program. All musculature was removed and the spine was mechanically tested using a Spine Kinetic Simulator, which is a six-axis servo hydraulic testing apparatus capable of reproducing spinal motion for *in vitro* specimens. The superior (L3) and inferior (Sacrum) vertebral bodies bodies were fixed to the device and pure moments were applied in all bending directions: flexion/extension, left/right lateral bending, and left/right axial rotation. The motion of the two vertebrae of interest for this study (L4 and L5) were then tracked optically using a marker-based Vicon motion capture system, calibrated to record motion data at 100 Hz with an accuracy of 0.02 mm.[43]

Results and Analysis

Figure 4.8 shows the visualization results achieved for the left/right lateral bending trials recorded for the 18 specimens, each selected because it is representative of a different degree of disc health. The helical axes between L4 and L5 were of primary interest to our collaborators. The previously presented 3D visualization strategy is maintained: the helical axes of the median trial and the two "+/- one standard deviation" trials are shown in distinct colors, and helical axis sheets for all trials are shown in the background.

The results both confirm some expected patterns in the data and suggest new directions for further analysis. In Figure 4.8 (left) notice that there are two regions where almost all of the trials are grouped together into a single trend. These groupings make sense given the investigators' current hypotheses and the analyses they have conducted to date using other methods. Moving from left to right across the trendline, the spines

begin in a neutral position, then bend all the way to the right, then all the way to the left, then return to neutral. When the trials come together into a single trend, this corresponds to the sections of motion when the spines are transitioning from being unloaded to loaded in the other direction. The likely explanation of these groupings is that during these sequences, the spine is going through the "neutral zone" – it is not fully loaded. The investigators hypothesize that the trials will be most similar during these transitions, and this hypothesis is supported by the trend-centric analysis. In contrast, as seen in Figure 4.8 (left), at other frames, many distinct trends are detected. What we believe the trend-centric analysis is demonstrating in this case is that the investigators did an excellent job of picking a wide range of specimens for their study. In this study, the specimens were specifically chosen to cover a wide range of disc health, age, and other factors that may impact the motion. The analysis suggests that these factors do, indeed, yield different motion patterns but only in a loaded state at the end ranges of motion.

Together with our collaborators, we then explored the logical follow-on question, "how do the trends change if we adjust our notion of how similar the trials must be to be included in the same trend?" By decreasing the clustering threshold so that clusters form more readily, the three most prominent trends at one point in the data match the scientists hypotheses, for example, the two specimens that are considered "most healthy" are clustered together in one of these trends. At this point, it is less clear what to make of the two other trends, but the scientists are excited to follow up on this analysis. For example, perhaps the other trials that were clustered together share a similar pattern of disc degeneration that could be detected with MRI.

User Feedback

High-level feedback from the domain scientists was positive. Since few visual data analysis tools in this style exist, our first concern in evaluating the tool was to understand whether the metaphors adopted in the visualization, especially the notion of motion trends that may change over time, actually fit the way these scientists think about their analyses. The response from scientists was that the approach is "Absolutely correct on separating into groups."

Another high-level indication of the success of the tool is that our collaborators have repeatedly told us that they are going to change the way that they collect their data in the future in order to better facilitate similar visualization efforts. Since this will bring visualization into the earliest stages of their investigative process, we take this as an important indication of the potential value of this tool.

Several lower-level points of feedback also emerged during discussion and tool use together with our collaborators. The visual aesthetic and overall display of the data was appreciated. We believe this increases engagement with the tool; moreover, we believe the presented design has a significant usability advantage over traditional multi-view visualization systems that require manual window positioning. The ability to adjust the 3D viewpoint was evaluated as absolutely critical, and different views (e.g., top down) were preferred depending on the particular question being discussed. There was much discussion about the trend visualization and how well it did or did not capture characteristics of the data. During this discussion, we realized how critical it is to support the ability to interactively adjust the threshold used in clustering, which is possible for this dataset. (The synthetic datasets, which are larger, require around one second to cluster; however, exploiting per-frame clustering's inherent parallelism could speed this up dramatically and allow real-time interaction.)

4.4 Iterative Design and Insights

Our trend-centric visualization techniques were designed through an iterative approach involving domain scientist collaborators plus a traditionally trained and experienced graphic designer. This work builds upon recent research that seeks to understand how best to integrate visual artists into designing and evaluating sophisticated data visualizations [18, 17, 46, 47]; researchers and practitioners working in any area of data visualization can adapt our collaborative design process.

This section describes our team's iterative design process and lessons learned, and is oriented around three specific aspects of trend-centric visualization: (1) design of the trend lines; (2) design of the discs; (3) design of the median + variance visualizations. For each of these aspects, the graphic designer developed, literally, hundreds of "design sketches", often using traditional graphic design tools, such as Adobe Illustrator. This sketching was the key activity in our design process's ideation phases. Several sketches are pictured in Figures 4.10 and 4.11. Each of the sketches, along with additional inspirations (e.g., Figure 4.12), were evaluated by the team using critique. The graphic designer and visualization researchers participated in these critique sessions once a week, for nearly a full year, with the domain scientists joining the group once every 4-8 weeks. Based on the insights generated during critique, the graphic designer refined the design and/or handed the design off as a specification to the team's programmers. As full-featured data visualizations were developed through the programming process and programmers added their own creative insights, they were likewise critiqued. The three specific examples described next illustrate the types of visual design insights that resulted from the process and justify some key design decisions we made.

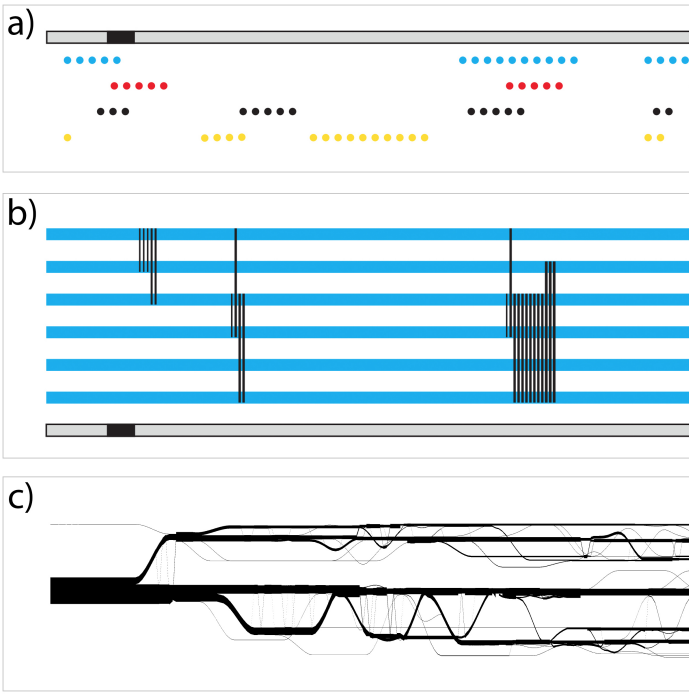

Figure 4.10: Three design sketches used to prototype and evaluate alternative representations for a the trend timeline.

4.4.1 Design of the Timeline

Our trend timeline designs started with visuals conveying the framewise similarity between multiple trials.

Figure 4.10a displays four trials with colored dots arranged below the timeline at the display's bottom to denote specific frames of each trial that have similar characteristics. The visual idea of representing each frame as a colored dot was evaluated as an interesting way to support detailed, frame-level analysis on a timeline, but the idea was evaluated as limited in its applicability to large datasets since it would necessitate many small dots (or a scrollable timeline) for trials with many frames.

Figure 4.10b illustrates an example follow-on design. Here, the timelines have a continuous visual representation (a solid blue horizontal line), which supports large datasets well. To convey similarity between the trials a black vertical line connects them at the frames where they are most similar. The main strengths of this design are the continuous representation for the trials and that the viewer's eye is immediately drawn to the strong visual links between the trials. The critical weakness is that this vertical line strategy does not work unless the trials are sorted in some way so that the links only need to reach adjacent trials and do not need to skip over lines; thus, this design was judged as impractical for use with any realistic data but was useful for pushing the team toward visuals that provide strong visual links between the trials.

Figure 4.10c shows the result of a critical innovation in the design. The idea demonstrated here switches from representing individual timelines for each trial to representing multiple trials together in a single "trend". This was evaluated as such an exciting concept that we advanced well beyond design sketches with this idea and actually produced a full implementation – Figure 4.10 is a result for the same dataset as used in the validation study described earlier. The main strength of this design is that bundling the

trials into trends provides the clearest depiction yet of the similarity of multiple trials over time. We evaluated using line width to encode the strength of each trend, and deemed it too difficult to implement the gradual transitions that can be seen as a trend branches off. We implemented several alternatives but could not develop a satisfactory answer to the question of how these trend lines should be drawn when several sub-trends branch in and out from each other during transition periods. Another weakness is that it is unclear exactly where one trend starts and another ends. This makes it difficult to design a good visual strategy for linking the trend line display with the 3D visualizations at the top of the screen.

The final trend line design pictured throughout this chapter solves both of these problems. We were inspired in this final design by Roberts' studies of transportation network maps [48], borrowing the visual concepts of: explicitly representing junctions; using clearly distinguishable colors for main lines and similar, but less saturated, colors for branch lines; and constraining angles when lines branch and join, thereby increasing readability and making implementation easier. We reinterpreted each of these visual concepts within our own driving applications in order to advance from the designs pictured in Figure 4.10 to those seen in Figures 4.1 and 4.3.

4.4.2 Design of the Discs

The design decision to use a set of simplified cylindrical discs to better convey the orientation and distances between the vertebrae is one that we think we would not have reached without an interdisciplinary collaborative design process. Figure 4.11 shows three design sketches produced during refinement of the idea. As computer graphics researchers, we found that our tendency was to create visualizations based upon the most detailed and realistic 3D models of the vertebrae that we could acquire. The sketch in Figure 4.11a, builds on this approach, exploring a visual technique for conveying

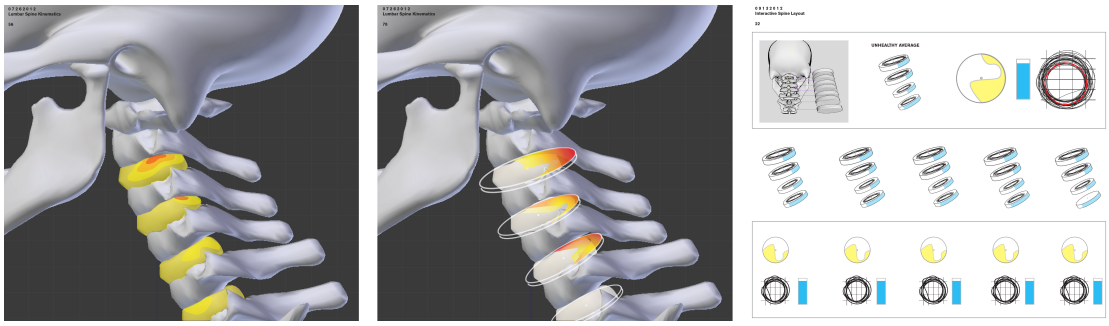

Figure 4.11: Three design sketches used to prototype and evaluate visual techniques for conveying the motion of individual vertebrae and the distances between the vertebrae; this type of ideation led to the offset-simplified-discs visualization technique pictured in Figures 4.4 and 4.1.

the distance between the vertebrae though color mapping directly on the vertebrae. However, this region is so tight that it is difficult to read the data. Figure 4.11b is the first of many sketches introduced by the graphic designer based on her idea to simplify the vertebrae by representing them as cylinders, making it easier to read the data displayed on them. As shown here, the first of these ideas were based on a two-pass rendering approach: draw 3D vertebrae with a traditional 3D rendering and then draw the disc geometries on top. The design was refined through many iterations (e.g., Figure 4.11c) to include offsetting the discs to the right of the accurate 3D rendering and several additional visual strategies for facilitating reading data off the discs. Our critique-based evaluations of this idea identified several strengths of the approach. First, the disc motion is easily seen because the geometry is simpler than the spine's geometry and occlusion is minimized. Second, it is much easier to read visual widgets drawn on discs (flat tops and bottoms and uniformly curving sides) than ones drawn on bone surfaces.

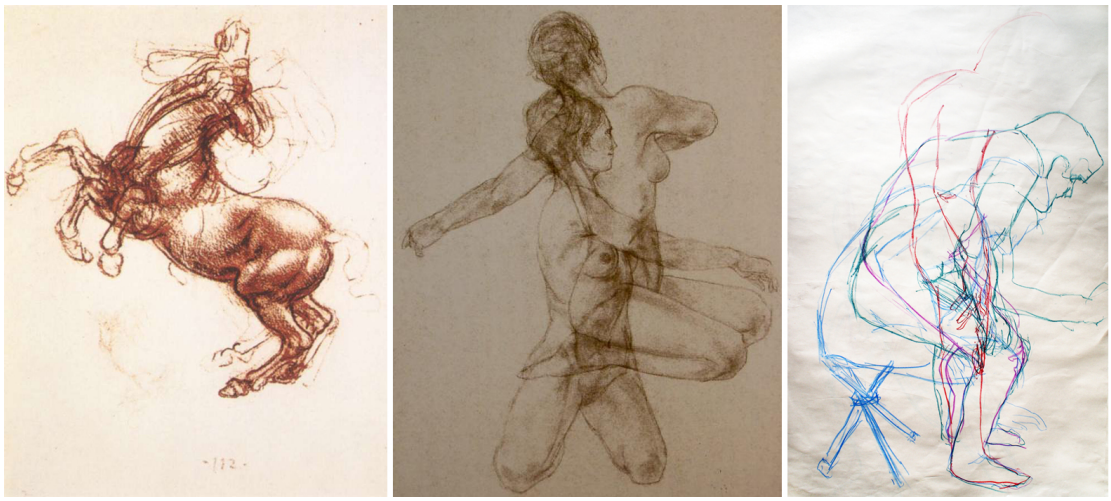

Figure 4.12: These three drawings demonstrate the effective illustrative technique of juxtaposing multiple poses in artistic studies of motion. These motivate the median + variance rendering strategy developed. Center image copyright Gary Kaemmer, used with permission. Right image copyright Aleksandra Kulecka, used with permission.

4.4.3 Design of the Median + Variance Visualizations

The visual technique used in the median + variance 3D visualizations is inspired directly by a common illustration technique used in traditional drawing: artists, to study and convey motion, juxtapose two to three different subject poses in one drawing. Figure 4.12 shows three examples. On the left, Leonardo daVinci emphasizes one pose while also suggesting several others that are drawn with less detail. Like daVinci's horse, our design aims to highlight one primary pose while simultaneously conveying a more complete sense of related poses using a less detailed rendering style. We accomplish this through the underpainting layer. In the center, contemporary artist Gary Kaemmer juxtaposes two detailed poses, demonstrating that even detailed renderings of human anatomy in action can be effectively understood when a small number of poses are juxtaposed together. Like Kaemmer's example, we include a small number of detailed renderings of the anatomy when these poses are critical to understanding the

motion. We use three rather than two poses, one for the median trial and one each for the trials that are plus and minus one standard deviation away from the median. On the right, another contemporary artist, Aleksandra Kulecka, employs color to distinguish the multiple poses depicted in this quick gesture sketch. Like Kulecka's work, we utilize color as a cue to distinguish the different poses.

4.5 Conclusion

This chapter presented trend-centric motion visualization – a new strategy for improving the way that scientists analyze motion collections. The key conceptual advance is to base analysis on motion trends rather than trials. To implement this strategy, we developed a series of visualization techniques, including a time-dependent motion clustering technique, a 2D trend lines visualization inspired by transportation network maps, a 3D illustrative visualization technique for depicting a cluster of trials as a median with variance, and interactive techniques for linking these visualizations together to create an exploratory visualization tool. Domain experts confirm that this trend-based visualization design matches their existing mental model, and that automatic trend detection improved their ability to see the overall patterns in the dataset. This work supports our central thesis by showing that a carefully designed visualization system can improve this complex analysis process. However, even for datasets of this size, designing the visualization took over a year. In the coming chapters we will examine how new visualization design tools can facilitate this process.

Chapter 5

Drawing with the Flow: A Sketch-Based Interface for Illustrative Visualization of 2D Vector Fields[1]

This chapter begins our discussion of the "Visualization Design Tools" research thrust, introduced in Section 1.2. Based on the previous chapters, there is a need for scientific visualization design tools that are accessible to scientists and artists alike. This chapter presents "Drawing with the Flow," a 2D sketch-based streamline visualization creation interface designed to be accessible to users ranging from visualization researchers to domain scientists to artists. Users sketch new streamlines directly on top of the underlying flow data; an "ink-data settling" algorithm maps the user's input to an accurate streamline.

[1]This chapter is based on work published in [49].

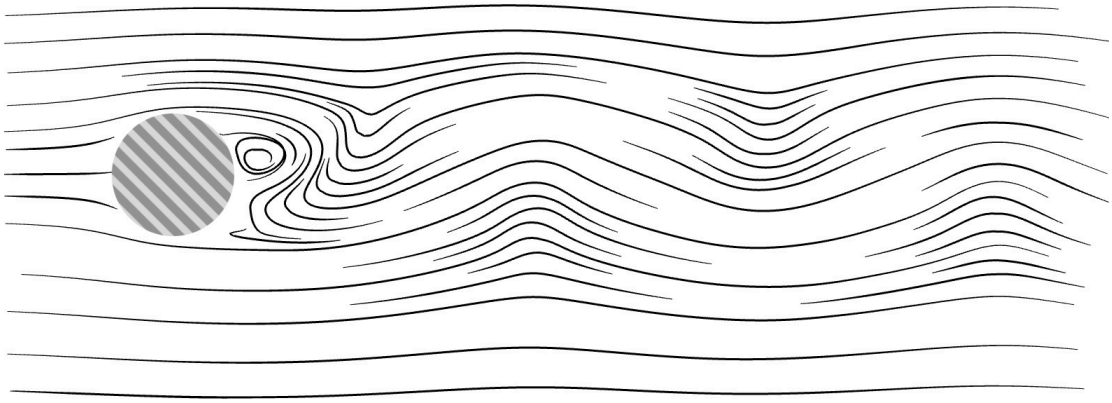

Figure 5.1: An illustration of simulated 2D fluid flow past a cylinder created using Drawing with the Flow.

New visualization and interaction techniques are needed to facilitate this interaction. In this chapter, we present a method for interpreting hand-drawn marks relative to an underlying vector field, including an ink-data settling process that subtly maintains an image consistent with the underlying data, a variable-density interactive streamline seeding algorithm tied to sketch-based input, and a set of sketch-based interactions to control the system. As a result, users are able to create accurate scientific visualizations while imparting their personal style and highlighting specific portions of the dataset (See Figure 5.1.)

The remainder of the chapter begins with a discussion of relevant related work. Then, the key advances in the Drawing with the Flow system are described. This is followed by an analysis of several illustration results created using the system.

5.1 Related Work

In addition to building on the work discussed in Section 2.2, Drawing with the Flow connects to and builds upon three main threads of related research described in the

following sections: illustrative visualization techniques, streamline seeding algorithms, and applications of sketch-based interfaces to visual design and visualization.

5.1.1 Illustrative Visualization

A consistent recent theme within the visualization literature has been using art-based or illustrative techniques to describe complex data. Some techniques mimic the appearance of brushstrokes and encode data in the color, width, and other characteristics of the stroke [4, 50]. These techniques can encode much information in a single image and they often result in engaging, aesthetically pleasing visualizations [51]. Other examples mimic specific pen-and-ink styles of illustration, including methods for increasing emphasis in certain portions of an image by adding detail via additional pen strokes [52]. Many other examples build on related illustration+visualization themes, e.g. [53, 34].

Somewhat surprisingly, given the wealth of recent research in this area, there are only a few examples of research in art-inspired visualization that include real artists and illustrators in the process. Notable examples include the work of Donna Cox [54] and collaborative work at Brown University and the Rhode Island School of Design, which has demonstrated the potential of using illustrators for expert critiques of scientific visualizations [55] and several virtual reality interfaces for applying artistic 3D modeling to visualization design [56, 57, 18]. Our work builds on these recent results and attempts to connect research in illustrative visualization to sketch-based interfaces for illustration and design. Ultimately, the aim of our work is to provide the right interface to enable artists and illustrators to contribute to solving visual design problems in scientific domains, such as flow visualization.

5.1.2 Optimized Streamline Placement for Flow Visualization

Streamlines and their derivatives (streaklines, pathlines) are one of the most fundamental features that can be used to describe a fluid flow or other vector field. Since streamlines are constructed by integrating through a vector field starting from a seed point, an infinite number of streamlines can be created for a given flow field by starting from different seed points. However, including too many streamlines in a single image can clutter and confuse it, thus, the topic of picking the best set of streamlines to show in order to most accurately depict the important features of a flow is one that continues to receive much attention in the literature.

Turk and Banks introduced an image-based technique to create images with an even density of streamlines by low-pass filtering the image generated and optimizing streamline placement to reduce variation in the image [58]. The visual results from this work were reproduced by Jobard and Lefer using a more deterministic algorithm with better runtime characteristics. This algorithm starts with an initial streamline and iteratively adds new streamlines at a specified distance from existing streamlines until the image is filled, ensuring a consistent density [59]. We introduce an extension to this algorithm to support variable density automatic streamline seeding. Variations in density can be useful for illustrating flows. Automatic algorithms have attempted to leverage this by seeding streamlines in specific patterns around critical points [60] or by only displaying streamlines that are quite different from each other [61]. Both of these algorithms are motivated by illustration techniques, in that they seed streamlines non-uniformly. Our interactive tool can be used to create similar results, where streamlines are placed precisely and deliberately, but through an interactive process controlled by the illustrator.

5.1.3 Sketch-Based Interfaces in Visualization and Design

Our work builds upon a number of recent advances in sketch-based interfaces, most notably the 3D modeling and design system ILoveSketch [62], which includes a notion of ink drying that has a similar visual aesthetic to our ink-data settling procedure. In general, our approach strives for a similar, fluid sketching, user experience. As in the Lineogrammer system [63], many of the marks drawn by the user in our system are interpreted based upon an underlying constraint, however, in our case, these constraints come from underlying 2D vector fields.

Sketch-based interfaces have been applied to visualization applications before [64, 65]. Most closely related to our work is the exploratory flow visualization system created by Isenberg et al. [66], which also supports drawing on top of fluid flows, but with the aim of exploring flow datasets through coupling animation with loose, freehand drawing. In contrast, our work strives to enable artist-refined, illustrative visualizations to convey information to other viewers. Our current implementation works with steady flows (static 2D vector fields) and creates a single image as a result. In the future, an interface similar to ours might be used to create animated illustrations, which may be particularly useful in describing how unsteady flows evolve over time or making steady flow visualizations even more engaging.

5.2 Drawing with the Flow

Drawing with the Flow aims to make it as easy as possible for an illustrator to explore a variety of illustration styles using sketching, a natural mode for design work [67], while also creating visualizations that are truthful in their representation of the underlying vector field data.

There are three main components to Drawing with the Flow, which are described

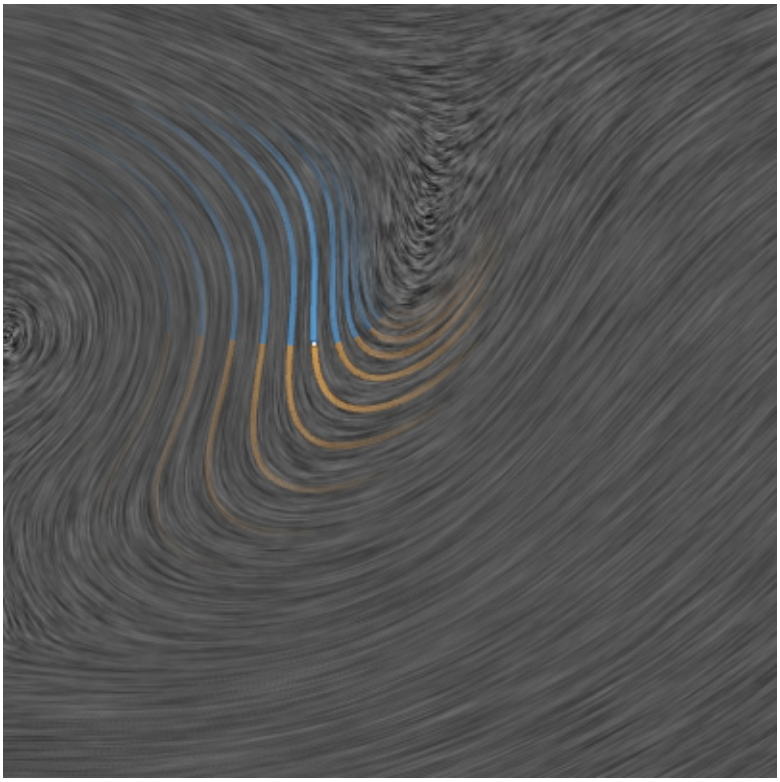

Figure 5.2: A sense of the underlying flow data is provided to the illustrator using an underpainting of the flow field and a local flow preview widget (blue and orange curves) that updates interactively as the stylus moves over the display.

in the following sections: 1. a sketch-based interface, 2. ink-data settling, and 3. illustration with automatic streamlines.

5.2.1 A Sketch-Based Interface for Illustrators

Drawing with the Flow utilizes a pen-based interface. The current implementation uses a 21-inch Wacom Cintiq tablet display. When the application first starts, it displays a subtle line-integral convolution (LIC) [68] visualization of the vector field as an underpainting on the display in order to provide the illustrator with a visual cue for the underlying data, see background image in Figure 5.2. The LIC underpainting provides

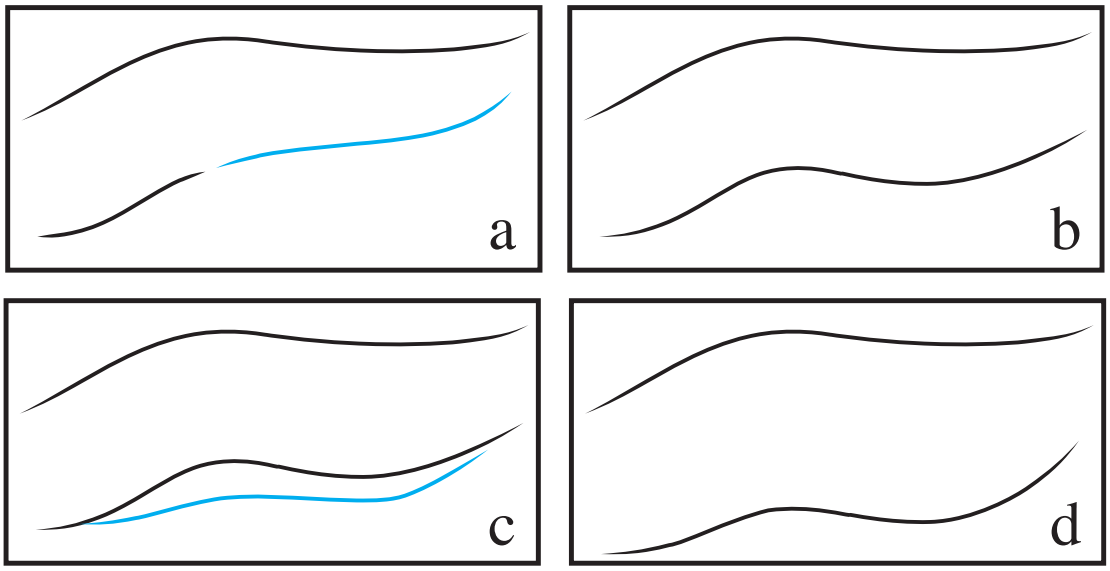

Figure 5.3: The illustrator can refine existing strokes in two ways. In a, the illustrator draws an extension (in blue) to the existing stroke, which the ink-data settling algorithm refines to the image shown in b. In c, the illustrator indicates that the bottom stroke should take a different path. The program first combines the original stroke and the refinement stroke, and then settles the stroke to match the underlying data.

information for every point in the flow without including sharp, emphasized flow lines, which makes it well suited to the task of providing the illustrator with a visual cue for the underlying data without dominating the illustration that is being designed. In general, LIC has been shown to be an effective visual style for evoking the sense of a flow, although other visualization methods are preferred for many data analysis tasks [69].

To augment the underpainting display, a local flow preview widget was developed to provide the illustrator with a finer sense of the flow immediately under the stylus, in the style of a traditional magic lens. This widget draws a small number of streamlines seeded in the vicinity of the stylus point and updates interactively as the stylus is moved on the display. The streamlines in this widget are colored to indicate the direction of the flow (orange=forward flow, blue=backward flow), providing information not readily conveyed in LIC-style visualizations. A number of other important design decisions are

included in this widget. The seeding positions of the streamlines are arranged such that the direction from one seeding point to the next is perpendicular to the flow direction, ensuring that the streamlines adequately capture twists and turns in the flow. The size of the widget also changes in reaction to the local speed of the flow; slower speeds are represented by shorter streamlines.

Given some understanding of the structure of the underlying vector field provided by the underpainting and the local flow preview widget, the illustrator is ready to begin designing a custom streamline visualization. This is done through sketching. When the illustrator draws a stroke, it is interpreted as either a gesture, a new stroke to add to the visualization, or a refinement of an existing stroke. A refinement stroke can be either an extension of an existing stroke or a re-routing of an existing stroke, as demonstrated in Figure 5.3. An extension of an existing stroke is indicated by drawing a line off of the end of the existing stroke, while a re-routing is indicated by overdrawing.

Several important gestures are recognized by the system, as shown in Figure 5.4. A stroke drawn through an existing mark in a direction roughly perpendicular to the existing mark is recognized as a crop gesture and divides the stroke at the point of intersection, deleting the smaller half. A scribble-out gesture, as used in a number of previous sketch-based interfaces (e.g. [62]) is useful for quickly deleting marks. Finally, "more" and "less" operations that automatically add or delete streamlines in the illustration (described in section 5.2.3) are activated using clockwise and counterclockwise loop gestures.

Strokes that are not recognized as gestures are interpreted either as new streamlines to add to the illustration or as modifications to existing streamlines. In both cases, the strokes drawn by the illustrator may not exactly match the underlying vector field data. Drawing with the Flow reconciles these differences through a procedure we call ink-data settling, described in the following section.

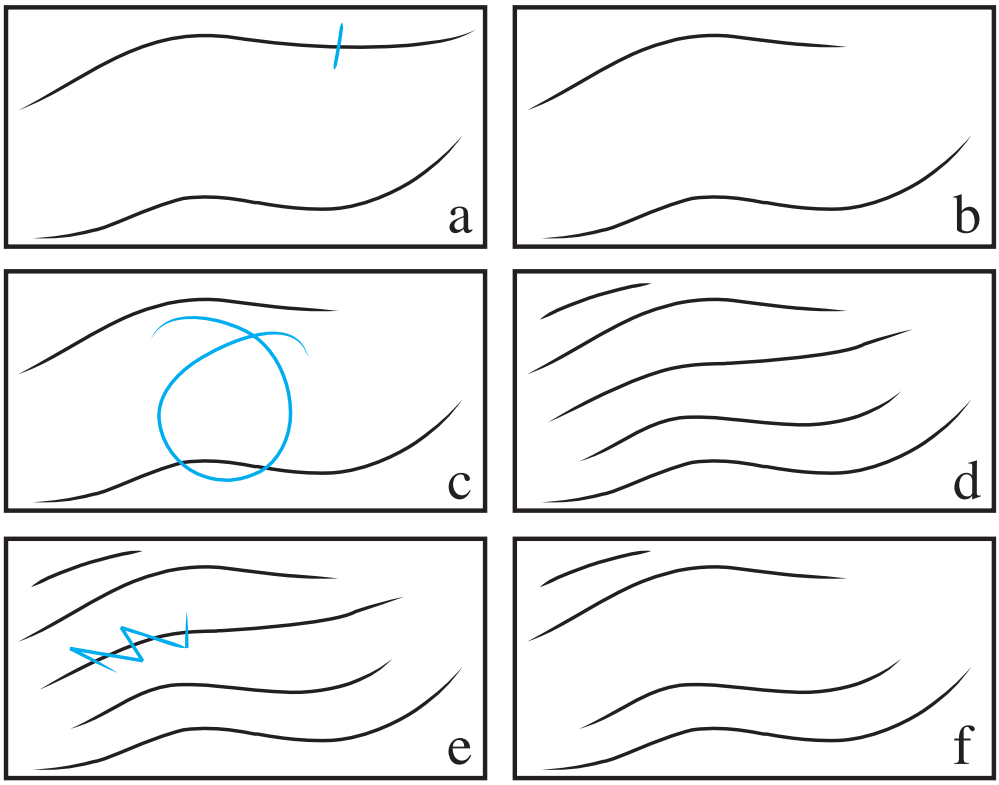

Figure 5.4: Three gestures are supported: crop, more, and delete. In a, the illustrator draws a crop line (in blue) through the top stroke, which gets cropped to the result in b. In c, the illustrator performs the more gesture, which instructs the program to add more streamlines to the illustration, resulting in d. In e, the user scratches out a stroke, deleting it from the visualization, shown in f.

5.2.2 Ink-Data Settling

Since the visualizations produced by the technique are intended to be used for flow analysis, it is important to ensure that the visualization accurately represents the underlying data. Thus, when the illustrator draws a stroke that represents a streamline, the stroke is interpreted relative to the underlying data and then refined to match these data.

The first step in this process is to identify a correspondence between the illustrator's

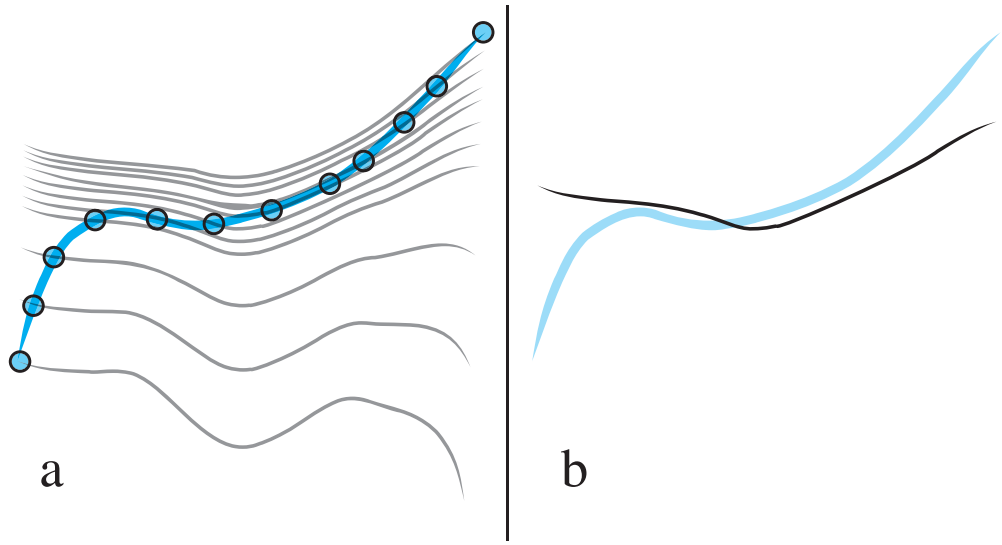

Figure 5.5: Fitting a stroke to a streamline. In a, several sample points (blue circles) are chosen on the illustrator's stroke (blue stroke). Through each of these points, a streamline (gray line) is generated based on the vector field. A metric is evaluated to determine which streamline is most similar to the original stroke. Part b of the figure shows the best fit in this case. The input stroke will "settle" into this streamline.

stroke and candidate streamlines. This is done by sampling a set of candidate streamlines in the vicinity of the drawn stroke and then selecting the best streamline based upon a similarity metric. First, the illustrator's stroke is resampled to achieve an even distribution of twenty samples along its length. Then, from each of these sample points, a streamline of the same length as the illustrator's stoke is generated according to the underling data. (Figure 5.5 shows an example.) Each of the new candidate streamlines is compared to the original stroke using a similarity metric computed as follows. For each point on the original stroke (p_1), the nearest point on the candidate streamline is found (p_2). At each of these points, the (normalized) stroke direction is found ($\vec{d_1}$ and $\vec{d_2}$, respectively). The similarity metric S for this point is then calculated as follows.

$$S = w_{dist} * \text{distance}(p_1, p_2)^2 + w_{dir} * \left(1 - |\vec{d_1} \cdot \vec{d_2}|\right)$$

The variables w_{dist} and w_{dir} determine the relative importance of the distance and direction in determining similarity. In our implementation, the values $w_{dist} = 10$ and $w_{dir} = 1$ are used with a coordinate system that ranges from zero to one across the image.

This metric has the effect of selecting streamlines that have significant overlap with the drawn stroke. We found this approach to be more intuitive than an initial version of the metric based only upon minimizing the distance between closest points, which can lead to selecting a streamline that is, on average, close to the drawn stroke, but does not often pass through the drawn stoke.

Once the match between drawn stroke and streamline has been established, an animation is used to create the visual effect of the ink settling onto the data. The animation is driven by a linear interpolation between points in the illustrator's stroke and the fitted stroke. The timing of the interpolation is set to produce an ease in and out effect.

The length and density of the streamlines are critical to the overall style of the illustration, thus these parameters are not interpreted relative to the underlying data and are instead left for the illustrator to specify directly through drawing.

5.2.3 Illustration with Automatic Streamlines

While generating an illustration in this style, one tedious task that can arise is filling in a region of the drawing with "similar" streamlines. This section describes an algorithm to assist the illustrator in this task. The technique is controlled via the "more" and "less" gestures shown in Figure 5.4. The intent is to utilize the context provided by the illustrator's recent actions to determine a region and style (length, density) for new streamlines to automatically add to the illustration. From the illustrator's viewpoint, the interface should feel intuitive, e.g. "give me more of what I just did."

Review of Constant Density Streamline Seeding

Our algorithm extends Jobard and Lefer's constant density automatic streamline seeding algorithm [59]. The original algorithm is reviewed briefly here; our extensions are described in detail in the next section. The algorithm iteratively attempts to add new streamlines to gaps in the image given the constraint that new streamlines must be a specified distance (d_{sep}) away from all existing lines. The algorithm begins with a single original streamline added to a queue, then iterates as follows:

- Pop the next streamline S from the queue.
- For each sample point along S, find the candidate seed points p_1 and p_2 that are a perpendicular distance d_{sep} from the streamline on both sides of the streamline.
- For each point p_1 and p_2, if the distance from the point to any point on another existing streamline is $< d_{sep}$, discard the point.
- Otherwise, create a new streamline passing through the candidate point, integrating both forward and backward through the vector field until the streamline either exits the flow or comes too close to an existing streamline. (Too close is usually defined as within $0.5 * d_{sep}$ of another streamline.)
- Add the newly created streamline to the queue.

Extension to Support Variable Density Seeding

We introduce an extension to this algorithm to support variable spacing between streamlines, as it is an important aspect of many hand-drawn flow illustrations. Rather than a global d_{sep}, our approach varies d_{sep} throughout the image. Since streamlines can be long and travel through a large portion of the image, setting the separation distance at the streamline level is not sufficient, thus, we store a local value for d_{sep} for each sample

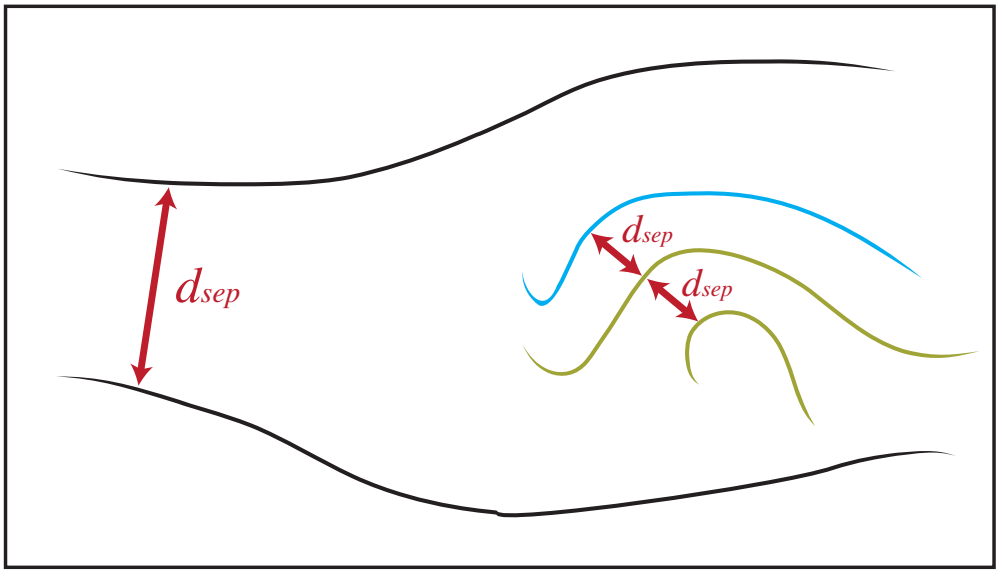

Figure 5.6: Automatically generated streamlines are seeded by examining the local separation of streamlines in the illustration. Here, the user has already drawn the black streamlines and the green streamlines. The green streamlines were drawn most recently, so new streamlines will be added close to these lines. In this region of the image, the separation of streamlines d_{sep} is relatively small. Thus, when a new streamline is added in this region (the blue line), this small, local separation value is used to seed the line.

in each streamline. This local value is used to find the candidate points p_1 and p_2 in the original algorithm.

To drive this variable density seeding strategy, the local d_{sep} for each sample on each streamline needs to be computed. This value should be representative of the local separation of neighboring streamlines, thus, the algorithm sets the local d_{sep} value for each streamline sample to be the distance from that sample to the closest streamline.

Automatic Streamline Generation

Figure 5.6 demonstrates the use of the variable density streamline seeding algorithm within a flow illustration. Several streamlines have already been drawn by hand and are shown in black. The illustrator then performs a "more" gesture, as described earlier,

which initiates the automatic seeding and displays a small circular widget to indicate the gesture was recognized. Now, as the stylus continues to move in a circular pattern around the widget, a single new streamline is automatically added to the image for each quarter revolution of the pen around the circle. Since just a single streamline is added at a time, there is no need store a queue of streamlines as described in the original seeding algorithm. Instead, drawn streamlines and sample points are selected at random until a new successful candidate streamline is found.

The automatically generated streamlines can be removed by simply reversing the direction of rotation around the circle widget. Returning to the forward direction of rotation will begin adding new streamlines again, this time picking new candidate seed points. This allows illustrators to scrub back and forth using the circle widget to explore different seeding possibilities.

As shown in Figure 5.6, recently drawn lines are used to infer attributes for the automatically generated streamlines. The length of new streamlines is constrained to the average length of the last two lines that were drawn by the illustrator. This constraint is removed if the last two drawn lines were longer than half of the total width of the image. The system interprets this as meaning that the illustrator intends to draw long lines that likely exit the flow field on both ends.

Automatically generated lines are also constrained to appear within a region of recent activity, if such a region can be identified from the illustrator's recent drawing. If the last two drawn lines are within a circular region with a radius of twenty percent of the image size, only candidate points inside this region will be considered.

5.3 Results

Figures 5.7–5.9 show result illustrations created using Drawing with the Flow on an Intel Core 2 Duo 2.66 GHz Mac Mini with a GeForce 9400 graphics card and a Cintiq 21UX Wacom Tablet display.

Figure 5.7 shows two illustrations that together demonstrate how 2D simulated flow past a cylinder changes depending upon the Reynolds number used in the simulation (Re=100 vs. Re=500). Each illustration was completed in approximately twenty minutes. The illustrative style adds detail to selected areas in each illustration to highlight the differences between the two cases.

Figure 5.8 demonstrates the variety of illustrative styles that can be achieved, even when working with the same dataset. The minimalist style on the left is similar to what a fluid mechanics professor might draw on paper to illustrate this canonical flow case to a student. The short lines style utilized the "more" gesture to quickly populate the background of the image. Each of these illustrations took from five to ten minutes to create. The final two illustrations demonstrate additional styles made possible by selectively including or excluding detail from the images. These took approximately twenty minutes to create.

Figure 5.9 shows illustrations of the type of 2D vector fields that are often used to evaluate automatic streamline seeding algorithms. Each of these fields contains at least one critical point. A smart selection of streamlines should make these critical points easy to identify, while a poor selection of streamlines can make them difficult to find. Each of these visualizations took approximately ten minutes to complete; the illustrator decided which streamlines to draw in each of these cases based on the goal of making critical points in the flow easy to identify.

5.4 Conclusion

In this chapter, we presented a sketch-based visualization design tool focused on streamline visualizations of 2D vector fields. In order to support this system, we presented an ink-data settling algorithm that refined users' input to be accurate, a variable-density streamline seeding algorithm to match a user's general style, and a set of sketch-based interactions to allow the user to control this interface. Although the system supports a great deal of artistic freedom, it also maintains the constraint that the marks displayed are accurate to the underlying vector field data. This visualization system allows users to visually design effective and accurate visualizations in an interface accessible to both artists and scientists. In the next chapter, we will build on this idea to allow analysis of much more complicated and multivariate datasets.

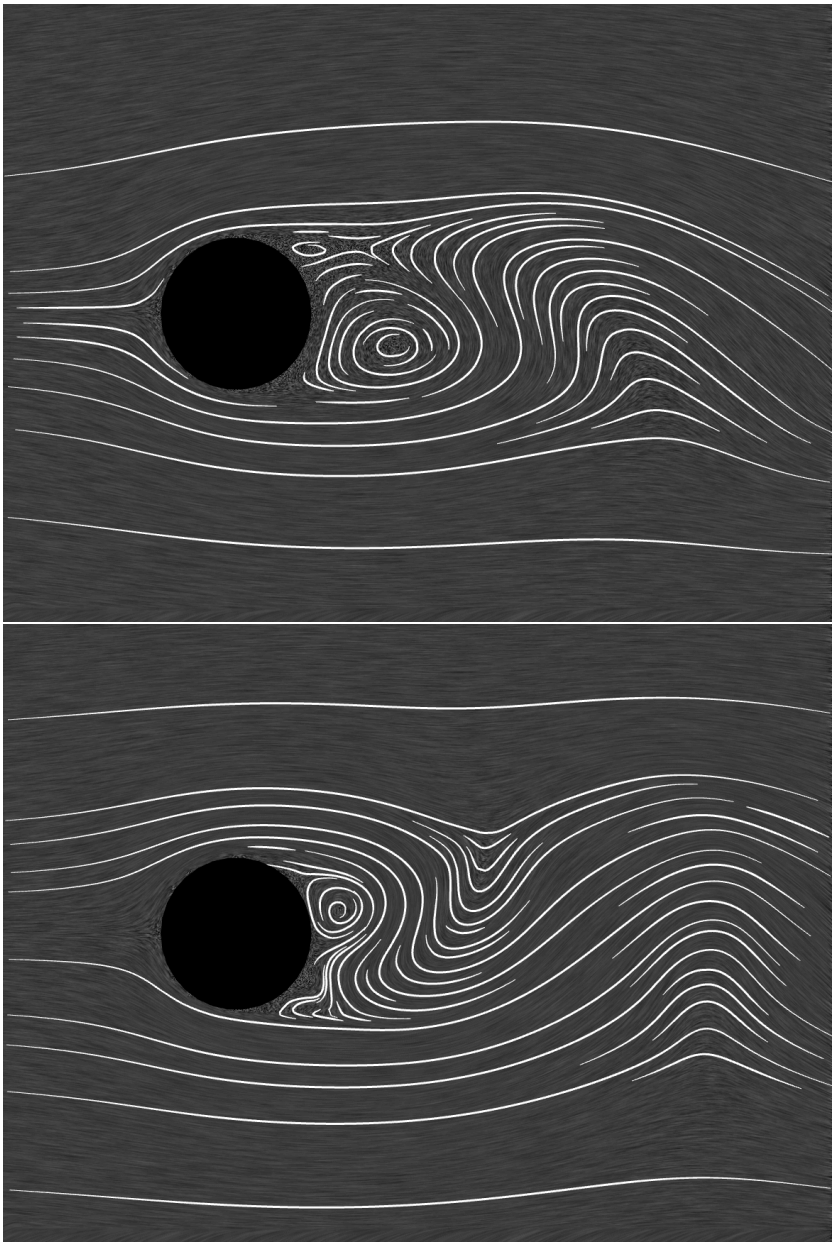

Figure 5.7: Two illustrative visualizations comparing flow past a cylinder with Reynolds number 100 (top) and 500 (bottom). The illustrator has attempted to emphasize the differences between the two flows.

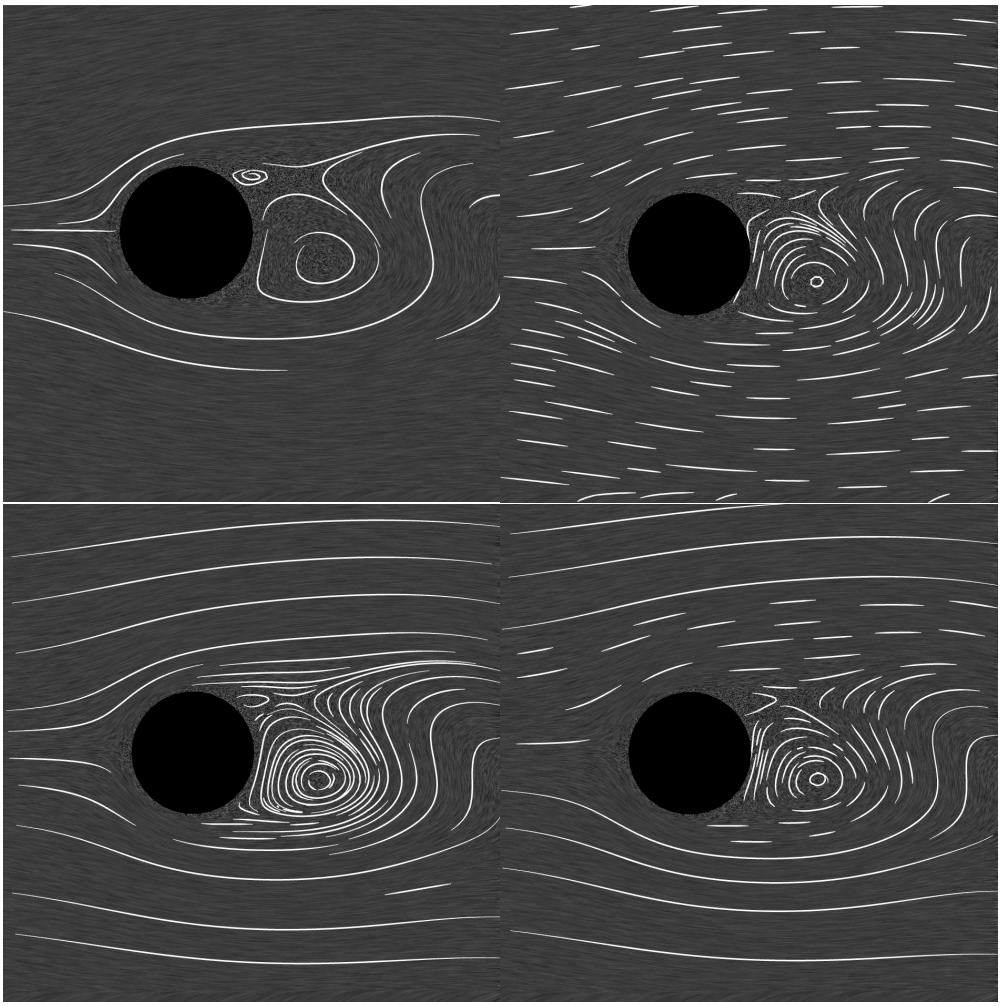

Figure 5.8: Illustrative visualizations of the same flow in four distinct styles, varying streamline density, length, and positioning.

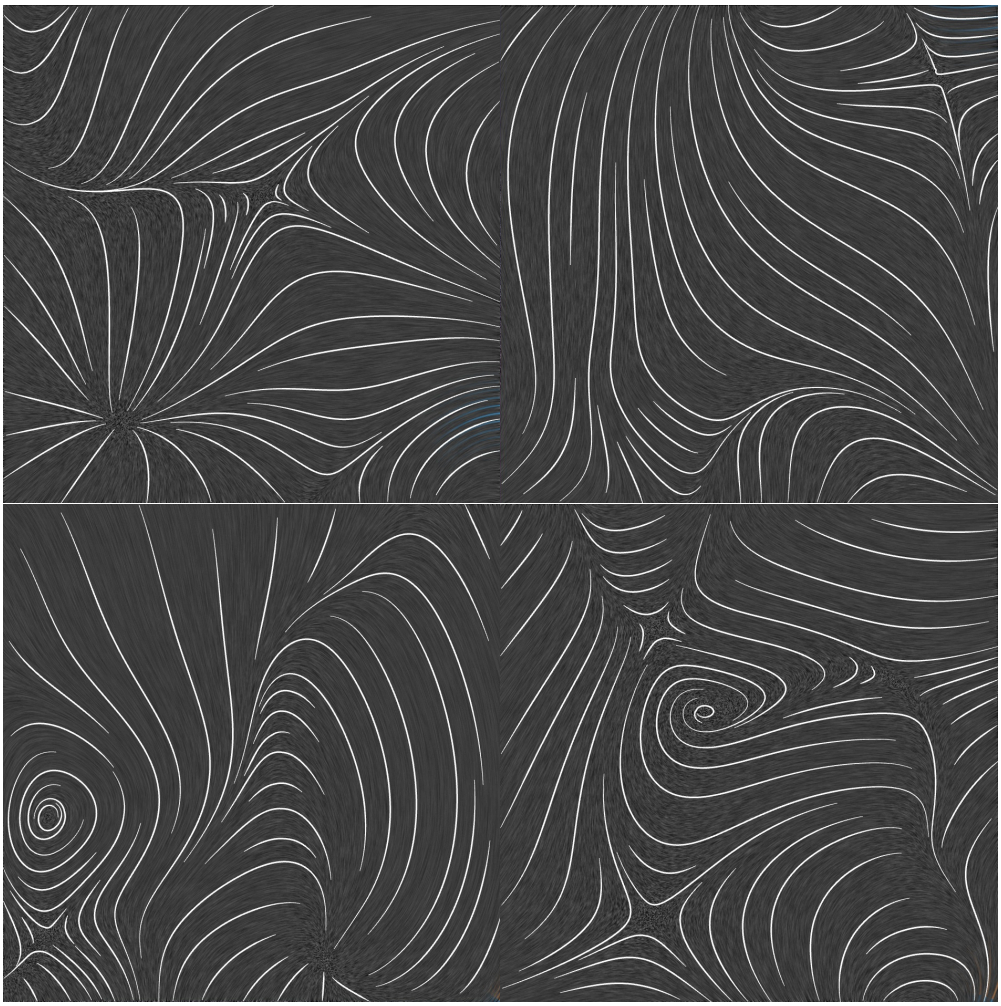

Figure 5.9: Illustrations of 4 fields containing different sets of critical points. Detail was added to highlight critical points, whereas areas of little change are left relatively open.

Chapter 6

Flowtoshop: An Artist's Interface for Visualization Creation

In this chapter, we present a visualization design tool that reimagines the interaction style of the previous chapter to time-varying multivariate data. As seen in the previous chapter, visualization design tools can be developed that produce accurate visualizations while providing a direct interface to enable users to interact directly with the data. We implement a Photoshop-style interface to allow the creation of complicated visualizations of datasets containing time-varying scalar and vector fields.

Flowtoshop is an interactive visualization design tool that encourages an iterative approach to visualization design where the user is able to repeatedly and intuitively adjust visualization parameters to arrive at what the user feels is a better design. Tools similar to this allow users to design a 2D flow visualization by, for example, adjusting a set of 22 preset sliders [21]. Flowtoshop takes a different approach where users are able to add and remove visualization elements, and wherever possible, adjust those elements directly rather than through the layer of indirection provided by sliders or other interface elements.

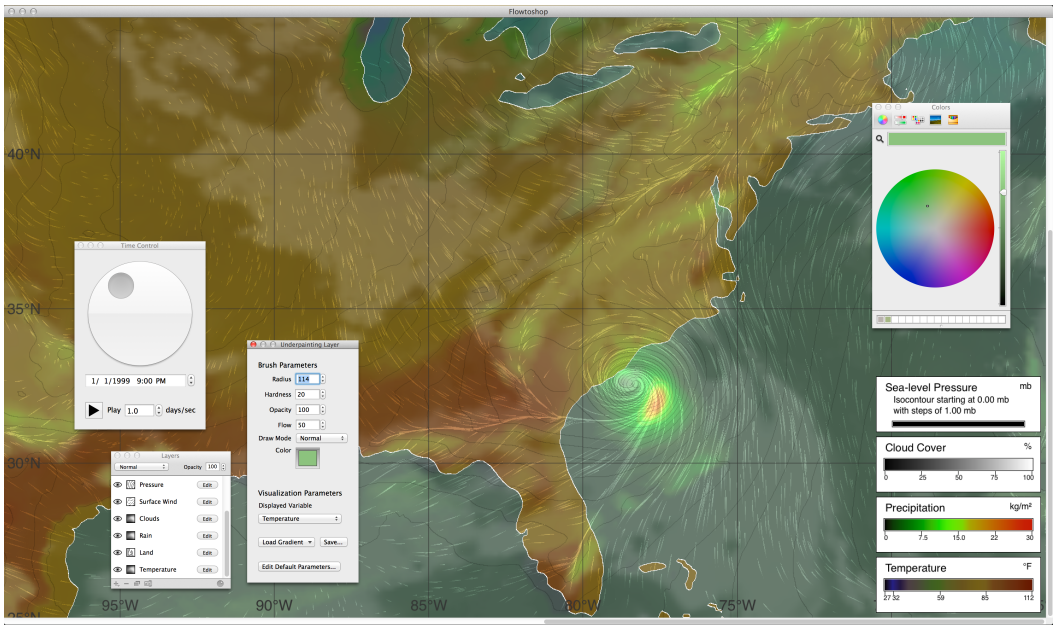

Figure 6.1: Flowtoshop allows users to create novel visualizations for two-dimensional scalar and vector fields. Here, Flowtoshop is visualizing the North American Mesoscale forecast (NAM) dataset, showing Hurricane Arthur from the 2014 Atlantic hurricane season. A combination of visualization elements allows for temperature, cloud cover, precipitation, wind speed and direction, and sea-level barometric pressure to be displayed.

Flowtoshop can create complicated visualizations, such as those shown in Figure 6.1. These visualizations are created using a Photoshop-style layer interface to visualize two-dimensional multivariate time-varying datasets, such as those that are created by climate models or fluid flow simulations. These data are usually positioned on a regular grid over some geographic frame of reference, and the quantities vary smoothly through both time and space. Flowtoshop distinguishes between vector quantities (such as wind velocity) and scalar quantities (such as temperature). Aside from this, no special consideration is given to any data variable.

6.1 Related Work

Many tools exist that aid in the visualization design process, but visualization design tools for non-programmers are either too limited or have an overly complicated interface. Software libraries can help the visualization design process, either by facilitating the creation of general two-dimensional or three-dimensional graphics (such as most window system toolkits, OpenGL, Processing, Matlab, etc.) or by implementing a wide range of visualization algorithms (such as VTK). Software tools can be designed to allow users to create a wide range of visualizations (such as Paraview), or to create a limited set of visualizations but with a much more accessible interface (such as Tableau). However, there's the challenge of enabling non-programmers to implement visualizations while not being required to learn a foregin interface.

Artists and visual thinkers have inuition that is comparable to the results obtained from quantitative user studies.[17] These experts are able to generate novel ideas, but require a team of programmers to have these ideas realized.[22] These existing visualization design tools are not built to take advantage of this insight and experience.

Flowtoshop also builds upon the visualization community's recent exploration into

the role that visualization tools can have in the storytelling process. Several researchers have identified the lack of storytelling tools to be a major area of future work [70, 71]. There are different aspects of visualization storytelling that are of interest: how storytelling can help a single user examine their own data more effectively, how storytelling can enable scientists to present their data and results to others who lack the requisite background knowledge to understand the data and its context, and how storytelling can create a closer attachment to the data, leading to a stronger social impact.

Many visualization tools have rudimentary support for displaying animations of their data that could be used to tell stories. Tools including ParaView, AVS/Express, and EnSight allow users to define keyframes that can be played back to show a series of views of a dataset. Keyframe animation is powerful, but can be difficult and time-consuming to use, so tools such as AniViz[72] have explored how new interaction techniques can allow users to more easily create the animations they intend to create, and to show what they want to in the underlying data.

6.2 Flowtoshop

Flowtoshop was designed to enable direct interaction with data, and we present two key techniques that enable this direct interaction. Direct visualization creation is perhaps best exemplified by our first key contribution, directly paintable colormaps and the differentiation between painting and dabbing. Our second key contribution is a direct particle design interface where users are able to adjust many parameters of a particle visualizations. Additional layer types allow richer visualization design possibilities, even if these layers themselves don't support the same degree of direct interaction. In order to support these contributions, two other areas needed special attention. First, the user interface was carefully designed to mimic Photoshop wherever plausible and to be usable

on graphics tablets, even for tasks such as directly scrubbing through a large dataset. Finally, typical graphic tool blend modes were implemented in order to allow users to combine layers in the ways they expect.

6.2.1 Paintable Color Maps

Color maps are effective at conveying the values in a scalar field, allowing for both local comparisons and for determining the value at an arbitrary location[73]. Conventional color mapping interfaces will often either have users choose from a preset list of gradients or will use a gradient editor where users specify colors at control points along a horizontal line. This interface can make it difficult to perform some adjustments as these may either require adjusting multiple control points in a coordinated manner or adding multiple new control points.

Instead, Flowtoshop adopts the metaphor of users drawing *directly on top of the data*, and the system determining how to best adjust the colormap to fit the user's intent. This means that if a user wants to create a colormap that is yellow in a particular region of the image, they don't need to manually adjust the gradient. Instead, they simply need to draw over the area, and in less than half a second the system will interpret the user's input to adjust the colormap. In mere seconds, users can create colormaps that begin to capture their ideas.

Users use a tool similar to the paintbrush tool in Photoshop to draw on top of the colormap visualization. As the user is drawing a single stroke, the stroke is drawn on top of the image with the specified brush parameters. Once the user finishes drawing the stroke, the system interprets their stroke and animates to the new colormap.

To control painting, users can choose the radius, hardness, opacity, flow, color, and blend mode of a radially symmetric brush. During drawing, the entire stroke is drawn into a mask in memory, and this mask then determines how to draw the stroke onto the

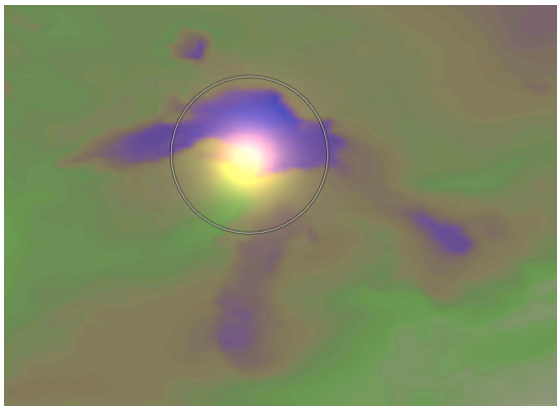

Figure 6.2: A local brush preview widget shows both the approximate size of the currently active brush in an outline, but also provides a preview of what the result of applying the brush over the background will be. Here, a blue brush with the "divide" blend mode is being previewed over a background gradient.

canvas. The radius of a brush defines the distance from the center of an individual brush mark to the point where the transparency of the brush is zero. The hardness controls how sharp of an edge the brush has; a hardness of zero is a Gaussian falloff with σ set to 1/4th the brush radius, and a hardness of one is a constant value within the radius. The opacity of a brush determines a scale factor applied to the mask, where an opacity of zero means the brush stroke can't be seen, and opacity of one means it is drawn at normal transparency. The flow of a brush determines a transparency scale factor applied to the individual brush image before blending into the mask buffer. Setting a low flow produces an "airbrush" style effect, while setting a high flow results in behavior similar to other computer painting programs. Color and blend mode change how the brush stroke affects the colors it is drawn on top of. Individual brush marks are drawn into the stroke buffer with a spacing equal to 15% of the brush radius.

It is difficult for a user to choose appropriate brush parameters to achieve their desired end result. To help solve this problem, Flowtoshop provides a brush preview widget, shown in Figure 6.2. This preview shows the apparent size of the current brush

as well as a preview of what the brush's effect will be on the underlying layer based on its current color, blend mode, opacity, and flow. Since a low hardness leads to a brush with a very narrow area of high intensity and a large area of minimal intensity, the preview radius ranges from 50% of the brush radius up to 100% of the brush radius as hardness ranges from 0% to 100%. Previewing the brush's effect on the gradient allows the user to use difficult to control yet powerful blend modes such as "divide" where the apparent color of the brush is the RGB complement of the actual brush color.

Flowtoshop supports two different methods of interpreting a user's brush strokes, termed "painting" and "dabbing," and will intelligently interpolate between the results of these two interpretations. "Painting" is a high-level, global operation where the user's contribution is determined as a result of the percentage of the visible pixels with a given data value the user paints over. "Dabbing" is a more localized form of input where the user is adjusting a specific local minimum or maximum, and where the user's contribution is stronger than merely painting would allow. Mixing between these two modes allows the user to have control over both global and local adjustments.

To determine the degree of "painting" versus "dabbing," and to determine the resulting gradient, Flowtoshop first resamples the gradient to have 20 equally spaced control points.

Each pixel on the screen, inside the bounds of the layer, has an associated data value that results from trilinear interpolation (bilinear interpolation in image space; linear interpolation between timesteps). In an evenly sampled gradient, each pixel on the screen therefore has color contributions from either one gradient control point, or a blend of two adjacent control points. We represent this as having a function $w_i(p)$ that gives the blend weight assigned to the ith control point for pixel p. For each pixel, at most two values of i will give a non-zero value for $w_i(p)$.

Similarly, we define two additional functions, $s(p)$ and $h(p)$. $s(p)$ is the value of the

brush stroke mask at pixel p, and ranges from 0 (fully transparent) to 1 (fully opaque). $h(p)$ is used to differentiate between painting and dabbing, and is the result of bandpass filtering the scalar data. The lower bandpass length is set to 1/4th the brush's radius (in terms of the dataset's coordinates), and the upper length is the brush radius. The values of $w(p)$ range from -1 to 1.

Computing a bandpassed image is computationally intensive, and can cause a delay in the user interface. As soon as the user selects a brush radius and zoom factor, a background thread begins computing the bandpassed image. If the parameters change such that different bandpass parameters are needed, the background thread is terminated and a new one is started. Additionally, since the bandpassed image isn't required until the user is finished drawing a stroke, Flowtoshop is almost always able to finish computing the bandpassed image, resulting in no additional delay.

Once the bandpassed image is computed, we compute arrays, "total", "selected", and "weights" as the sums of values defined at each pixel.

$$\text{total}_i = \sum_p w_i(p) \tag{6.1}$$

$$\text{selected}_i = \sum_p w_i(p) s_i(p) \tag{6.2}$$

$$\text{weights}_i = \sum_p w_i(p) s_i(p) h_i(p) \tag{6.3}$$

Then, the "weight influence" is calculated as

$$\text{weight influence} = \left| \frac{\sum_i \text{weights}_i}{\sum_i \text{selected}_i} \right| \tag{6.4}$$

A weight influence close to one indicates the user is performing an operation on a

local maximum or minimum and thus a "dab" operation, and a value close to zero indicates the user is performing an operation on a flat or smooth area, indicating "painting."

The influence of a brush stroke on a given gradient index is then[1]

$$\text{dab weight}_i = \begin{cases} \frac{|\text{weights}_i|}{\text{selected}_i} & \text{if selected}_i \geq 0.03 \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

$$\text{influence}_i = (1 - \text{weight influence})\frac{\text{selected}_i}{\text{total}_i} + (\text{weight influence})(\text{dab weight}) \tag{6.6}$$

Then, given a function $\text{blend}(a, b, \alpha)$ that corresponds to the blend function used for the current layer, as given in Table 6.1, and given b corresponding to the color of the brush, and c_i corresponding to the color in the existing gradient at the location of the ith control point in the new gradient, the resulting color at index i is

$$\text{color}_i = \text{blend}(c_i, b, \text{influence}_i) \tag{6.7}$$

After the new gradient is computed, a brief fade animation happens between the view with the stroke bring drawn and the view with the updated gradient. This entire computation process takes under a half a second for images with over 2.5 gigapixels.

6.2.2 Particle Design Interface

The particle design interface was designed to allow for a large degree of artistic freedom while encouraging effective visualization design. Some design parameters are fixed, based on existing literature. Particles are constrained to have a single color, to be

[1] Clamping dab weight to zero if less than 0.03 prevents division by zero inflating the value. Values less than 0.03 correspond to less than 1/256, and thus this difference is virtually unnoticeable.

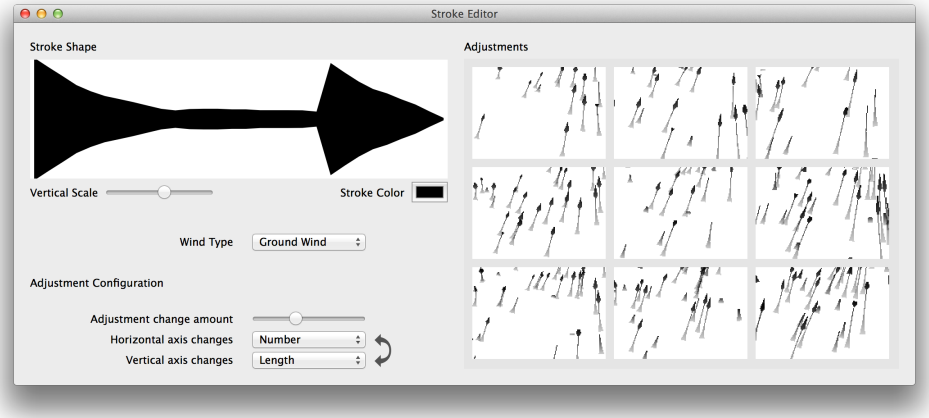

Figure 6.3: The stroke editor interface allows users to adjust the display parameters of particles that follow the underlying flow. By drawing over the stroke in the upper-left area, users can define new stroke shapes. By choosing from the differing view parameters in the view on the right, users can adjust the particles' density, length, lifetime, and speed.

symmetric about the direction of motion, and to fade out opposite their direction of motion to a user-controlled end opacity, ranging from fully transparent to fully opaque [21, 1, 74]. Users are able to freely adjust the profile, length (number of samples), speed, lifetime, color, and number of the strokes, as well as which vector field is being shown.

The particle design interface is divided into two regions. The first region allows the user to design the shape profile of the stroke, and is in the upper-left of Figure 6.3. The user can adjust the profile by drawing a line on top of the existing stroke. The stroke size at each particle sample point crossed by the user stroke is set to the distance between the stroke centerline and the drawn line. The user can adjust the editor's stroke scale to allow either narrow or thick strokes to be created.

The second particle design interface region allows the user to adjust the global particle properties. Four of these parameters (speed, number, length, and particle lifetime) are adjusted in the design grid on the right hand side of figure 6.3. In this

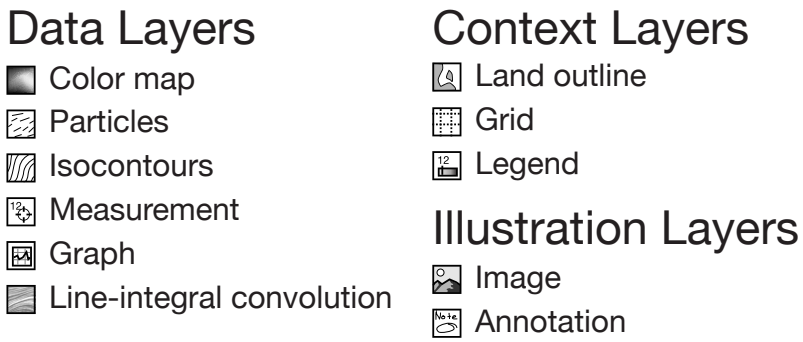

Figure 6.4: Flowtoshop supports a wide range of layer types, broadly grouped into three categories. Data layers depict the underlying scalar and vector data. Context layers provide a connection between the data and the geospatial context. Illustration layers allow the user to insert non-visualization elements, either to achieve a certain style or to bring emphasis to a particular region.

grid, the current visualization settings are shown in the center of the view, and two parameters vary either horizontally or vertically. Clicking on one of the cells chooses the associated parameters and updates all of the cells accordingly. This interaction allows users to see the parameter change's effects *before* applying those changes.

In the display window, the particles are seeded randomly in space and time. Each particle is integrated through the vector field using Runge-Kutta integration for a configurable number of timesteps. Particles are seeded within a region centered on the visible area on screen, but expanded by 20% in each direction to ensure that particles do not appear to be sparser near the perimeter of the screen.

6.2.3 Additional Layer Types

Flowtoshop supports a number of additional layers beyond the previously mentioned color map layer and particle layer, shown in Figure 6.4. These layers fall into three broad categories: data layers, context layers, and illustration layers.

Data Layers

Data layers directly depict data from the underlying dataset. The previously mentioned color map and particle layers are in this family, as well as an isocontour layer, a measurement layer, a graph layer, and a line-integral convolution (LIC) layer.

The isocontour layer allows users to add isocontours showing any data variable. Users are able to choose data parameters, such as the variable being displayed, and the spacing between isocontours and the 'zero point'. For example, a user can have isocontours displayed for atmospheric pressure at 1,000 millibars, with additional contour lines every 3 millibars. Users can also adjust visual parameters, such as the line color and thickness. The isocontours are computed on the GPU with a filter ranging from 5 to 13 samples, depending on the line width.

The measurement layer allows users to choose locations in there data where the current scalar value will be displayed as text. Users are able to place, move, and remove locations by directly clicking and dragging on the visualization. Users can control the units used to display these data by using Flowtoshop's units manager, shown in Figure 6.11. Users can control the font size and color, as well as the color, opacity, offset, and blur amount of an optional drop shadow.

The graph layer displays a small graph of a variable either over time or with respect to another variable, and is shown in Figure 6.5. These graphs are anchored to display at a given position on screen, and to show the data from a given point in the data. Users are able to adjust the foreground and background colors, as well as how many datapoints are plotted, what variables are plotted, and the size of the graph.

The line-integral convolution layer provides either a static or animated visualization of the velocities in a vector field, and is shown in Figure 6.6. Users are able to control the line integral length, the number of samples along the line, the colors used to display

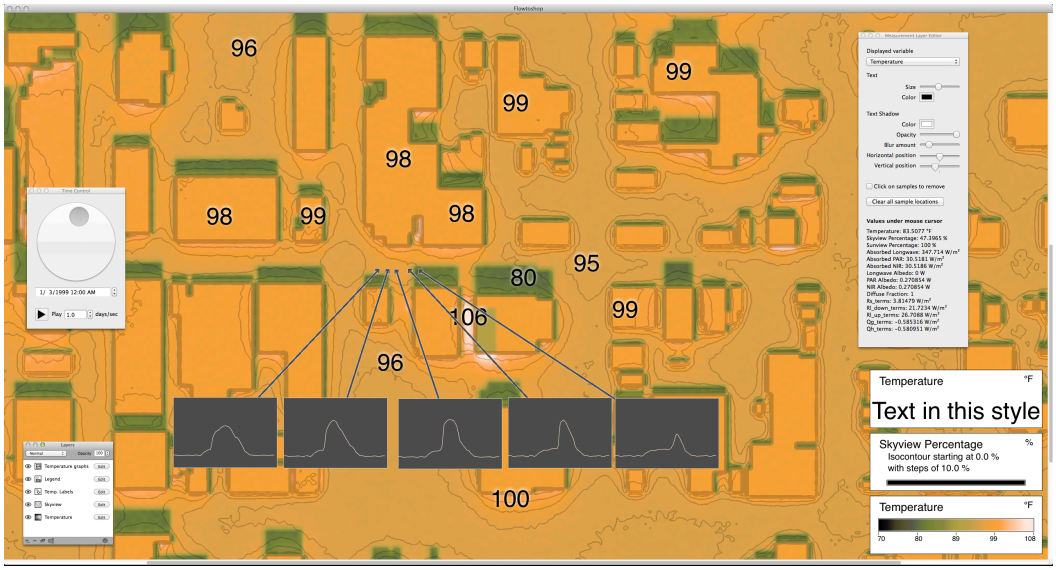

Figure 6.5: Flowtoshop can be applied to any regularly gridded 2D scalar and vector dataset. Here, flowtoshop is visualizing the results of a city-scale energy transport simulation on Salt Lake City. The colormap and displayed numbers show the computed temperature, the isocontours show the skyview percentage (the percentage of the sky hemisphere visibile from a given location), and the graphs show the temperature trends over an entire diurnal cycle.

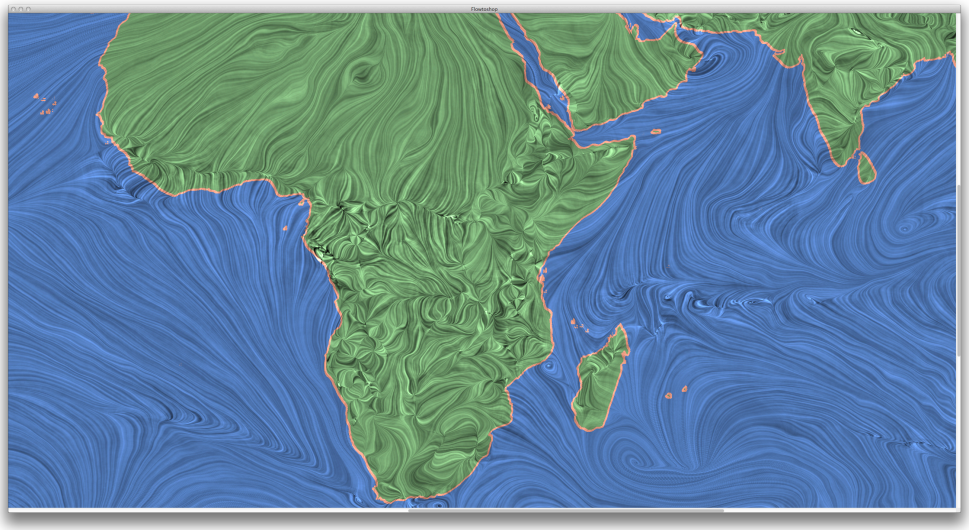

Figure 6.6: Line integral convolution (LIC) visualizations provide a dense view of the velocities present in a vector field. Flowtoshop supports LIC rendering, allowing users to control the LIC's appearance. A land outline layer is used to clearly distinguish between land and water.

the result, and the amount of contrast present in the resulting view. Users can choose to animate the LIC, which is accomplished by convolving the sampling envelope with a time-varying sine wave. To avoid sources or sinks in the vector field causing the resulting LIC to have a large region of a single color, the sampling envelope is further multiplied by the velocity magnitude at each sampled point, causing sources or sinks with a very small velocity to have nearly no effect on the resulting rendering. The LIC rendering is performed using Runge-Kutta integration inside of a GLSL shader, and is capable of rendering full-screen LICs with 60 integration steps per pixel at interactive framerates.

Context Layers

Context layers provide information about the visualization as a whole, and do not display specific data variables. These layers include a land outline layer, a grid layer, and a legend layer. The land outline layer allows the user to draw an outline of the

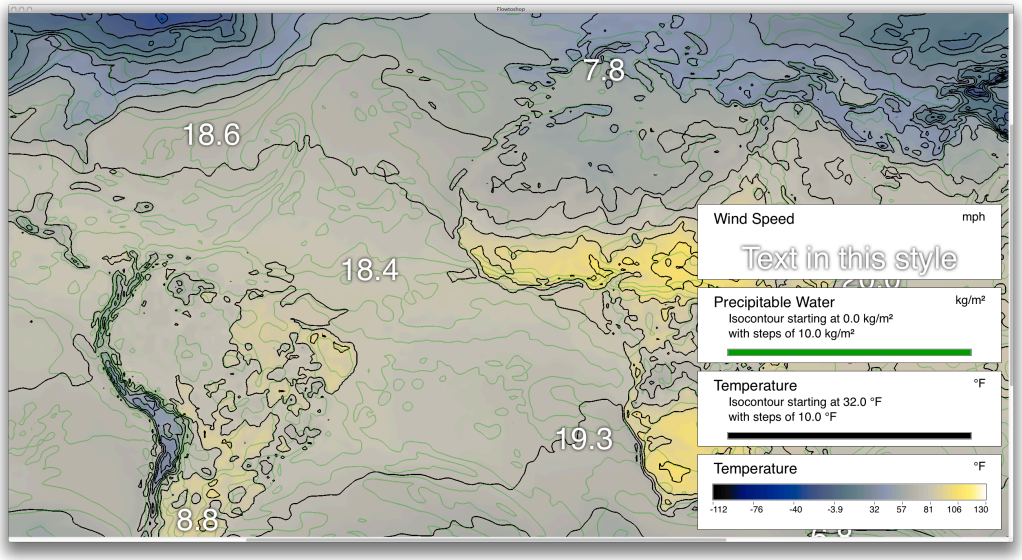

Figure 6.7: Flowtoshop's legend layer shows the mappings used in the other visible layers. Here, a colormap is used to show temperature, two isocontour layers are used to show temperature (in black) and precipitable water (in green), and a measurement layer shows the wind speed. Users are able to adjust the colors, scale, dimensions, and spacing of the legend elements.

Earth's landmasses, optionally coloring the land area, the outlined area, the sea area, or any combination of these. This allows otherwise difficult-to-place visualizations to have a geographic context, such as that shown in Figure 6.6. The grid layer provides both a geographic grid showing lines of longitude and latitude, as well as a data grid, showing the size of the individual data cells. The geographic grid can optionally show textual labels of the number of degrees for each line of latitude or longitude. Users are able to select how densely rendered the gridlines will be, and as the user zooms in and out, Flowtoshop will adjust the spacing between gridlines and smoothly fade between levels of detail. The legend layer draws a legend showing the mappings used by various layers, and is shown in Figure 6.7. Users are able to choose the dimensions, scale, colors, and spacing of the legend elements. Additionally, for the colormap layers, users can specify how many locations on the gradient will be labeled. The endpoints are always labeled, as are any "special" values for each variable, such as mean surface pressure for atmospheric pressure, the freezing point of water for temperature, and zero meters for geopotential heights.

Illustration Layers

Illustration layers are used to add elements to the visualization that aren't tied directly to the underlying data. These layers include an image layer and an annotation layer. Image layers can be used to add elements that provide context for what is currently being shown, such as photographs or research papers, or can be used for an artistic effect, such as embedding the visualization in a screen, as shown in Figure 6.8. Images can be positioned relative to the data, or can be positioned in screen space. The annotation layer is used to draw colored lines on top of the visualization, and can be used to draw text, arrow, lines, or any other figures the user wants. Together, these two layers allow users to add information to the visualization outside of that contained inside the data.

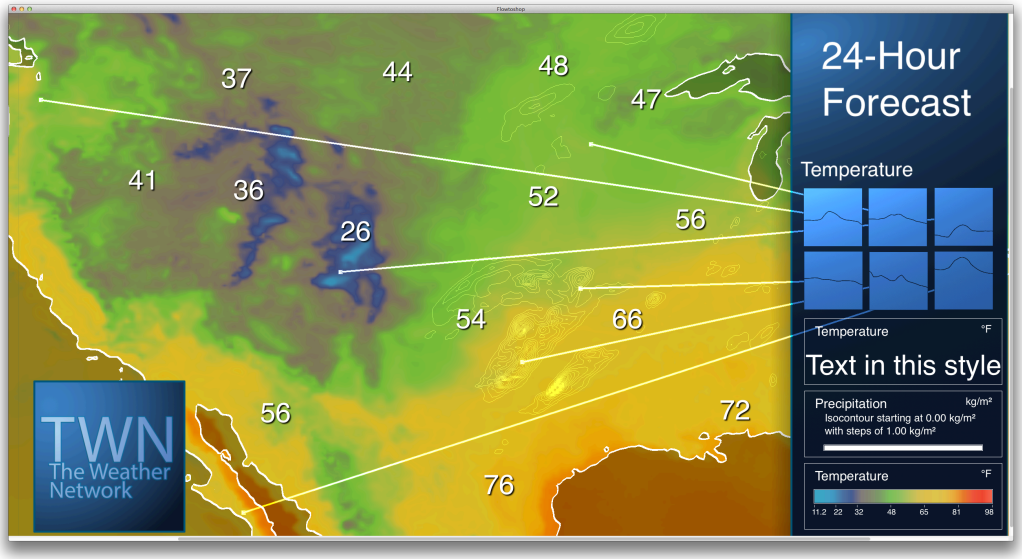

Figure 6.8: Using the image layer and the annotation layer, users can add elements to the visualization that are not tied to the underlying data. Here, a screen-anchored image provides a mockup of a potential on-air image for a hypothetical weather network.

6.2.4 Script Editor

Flowtoshop supports the creation of "scripts" which tell a story about a scientific dataset. Scripts are made up of scenes, each of which has a set of layers associated with it, a geographical position, a duration of the dataset that it shows, and various transition parameters.

Users can choose to enable a zoom-out zoom-in transition between two different scenes to maintain a sense of the global positional difference between the two scenes. This transition is performed by first determining the furthest zoomed zoom level that allows the centers of the two views to be on screen at once. If this zoom level is more zoomed than one of the endpoints, no intermediate zoom-out is required to relate the two scenes, and a simple pan and zoom between the two scenes is performed.

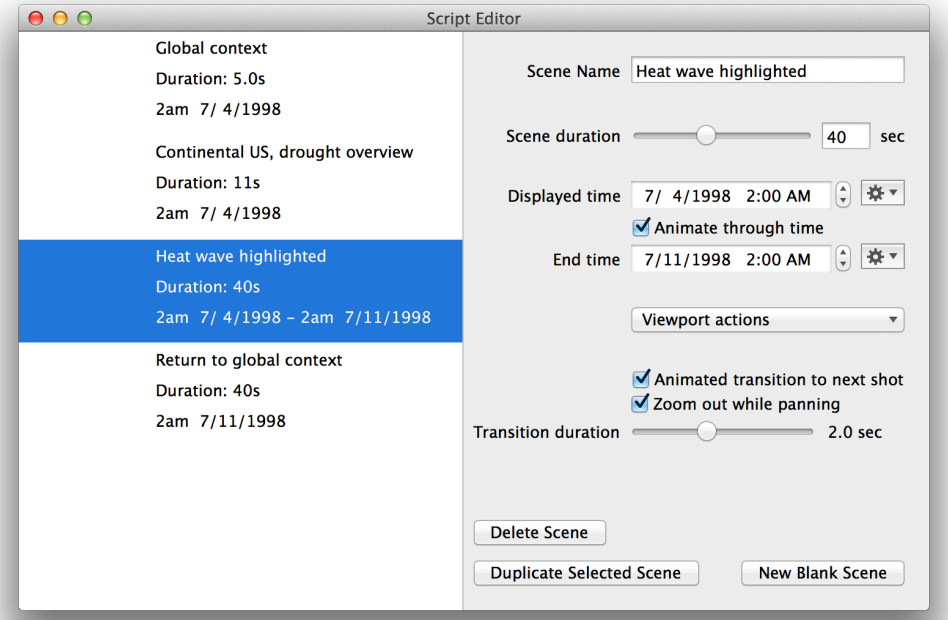

Figure 6.9: The Script Editor allows the user to define a series of scenes that can be played back to tell a story using the climate dataset. Here, a set of scenes examine the 1998 drought that affected the southern United States and show its impact.

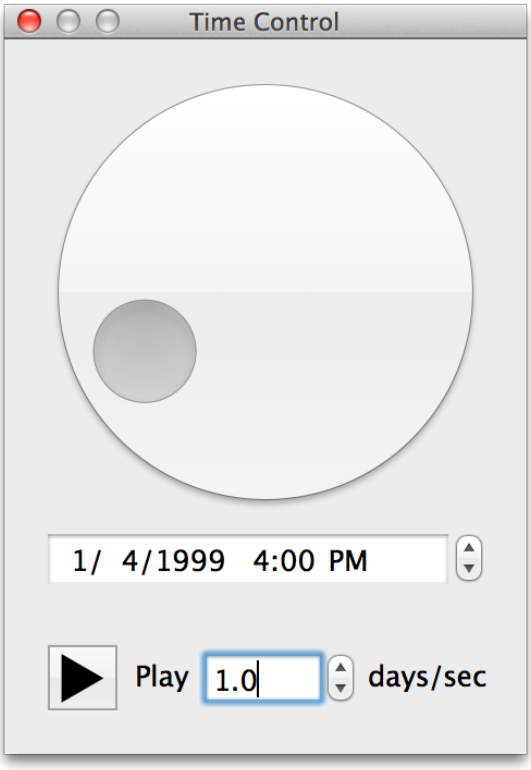

Figure 6.10: Controls for navigating through time had to be specially designed to enable infinite, directly controllable scrolling with stylus-based input. The circular control on top can be rotated clockwise to move forward through time, and counter-clockwise to move backwards through time. One complete revolution advances time by one day. Additional controls at the bottom allow for navigation to a specific date and time, and for automatic advancement through time.

6.2.5 Other Interface Widgets

An effort was made to choose interface elements that are familiar to a wide range of users, to ensure that users are able to transfer their existing knowledge. We implemented custom controls to mimic typical interactions from Photoshop, such as dragging the cursor left and right on a text label to adjust the associated scalar quantity (such as brush size or opacity).

Flowtoshop's interface needed to be usable on graphics tablets, such as a Wacom

Cintiq display. This required a few custom interface elements. Traditional interfaces for navigating a time-varying dataset, such as a horizontal bar, are not suitable for this case. These interfaces either assume that the dataset is small enough to have the entire duration mapped to the length of the bar, have an automatic "panning" action occur near the horizontal extents of the bar, or assume that they are able to "warp" the mouse cursor from the right-most extent to the left-most, to enable infinite dragging. With a graphics tablet, the cursor is constrained to be directly underneath the stylus, preventing warping, and so a circular dial control is used to allow the user to scroll infinitely in a finite space while always having direct control over the movement speed, as shown in Figure 6.10.

In order to be usable by a wide range of users, Flowtoshop also includes a flexible unit conversion framework. Incoming data is typically set with units favored by scientists, such as Kelvins, meters, and Pascals. End users typically want other units, either in SI-compatible units (such as degrees Celsius, kilometers, or millibars) or United States customary units (such as degrees Fahrenheit, miles, or atmospheres). While using Flowtoshop, users are able to bring up a unit manager window (shown in Figure 6.11) where they can choose their preferred units for each variable.

6.2.6 Blend Modes

Flowtoshop supports a wide range of methods to composite one layer on top of the layers beneath it. Digital painting programs such as Corel Painter and Adobe Photoshop allow users to specify the "blend mode" and "opacity" or "fill" of each layer. Blend modes define functions that determine how composition is performed between layers based on the color and transparency of the current layer and of the result of compositing all layers beneath the current layer. Opacity and fill modulate the strength of the composition in subtly different ways. Opacity interpolates between the resulting composite and the

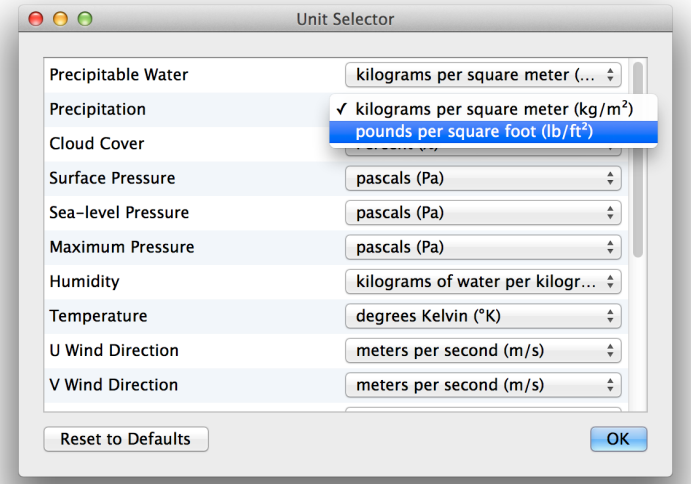

Figure 6.11: Flowtoshop allows users to choose in what units they want to examine their data. Units can be chosen on a per-variable basis, allowing users to examine multiple dimensionally equivalent variables in different units.

composite of previous layers, while fill adjusts the current layer's color values based on blend mode to achieve a similar result. Opacity and flow have identical effects on some blend modes, such as "normal," while other modes have severe differences (such as "divide," especially when clipping occurs in a color channel). In order to simplify the interface, Flowtoshop offers only a "fill" control.

Flowtoshop supports blend modes operating in both the sRGB and CIELAB color spaces. This variability is due to some blend modes (such as 'screen') being defined based on the RGB values of each component, and other blend modes (such as lightness) having a direct implementation in the CIELAB color space. A full list of supported blend modes along with the formulae used is given in Table 6.1.

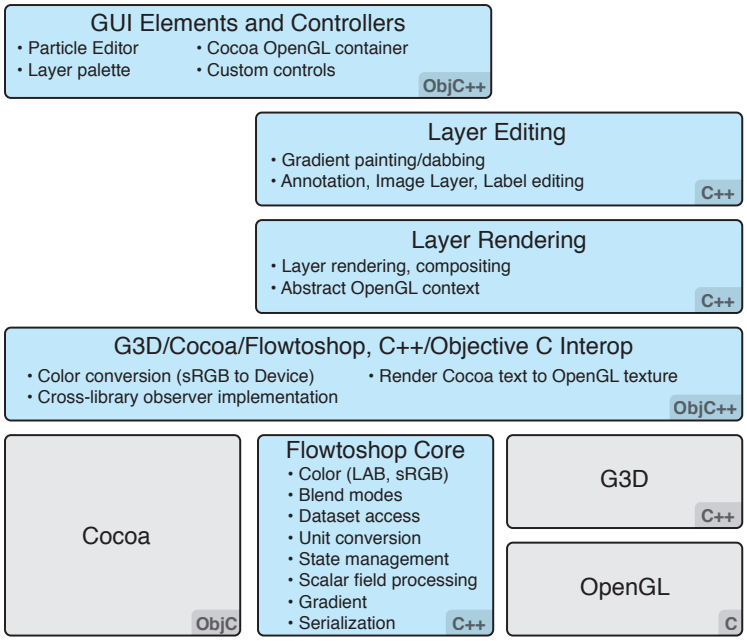

Figure 6.12: Flowtoshop's code structure. Flowtoshop is built on top of G3D (itself built on top of OpenGL) and Apple's Cocoa. Different layers in Flowtoshop are responsible for providing core features, interoperability between different libraries and languages, layer rendering, layer editing, and the user interface. By limiting coupling between the GUI toolkit and the rendering toolkit, an entirely new user interface could be provided without needing to change the layer rendering or editing code.

	Colorspace	Equation
Normal	CIELAB	$c_x = (1-\alpha)a_x + \alpha b_x$
Screen	RGB	$c_x = 1 - (1-a_x)(1-\alpha b_x)$
Multiply	RGB	$c_x = a_x(\alpha b_x + (1-\alpha))$
Divide	RGB	$c_x = \frac{a_x}{\alpha b_x + (1-\alpha)}$
Lighten	RGB	$c_x = (1-\alpha)a_x + \alpha \max(a_x, b_x)$
Darken	RGB	$c_x = (1-\alpha)a_x + \alpha \min(a_x, b_x)$
Lightness	CIELAB	$c_L = (1-\alpha)a_L + \alpha b_L$
		$c_A = a_A$
		$c_B = a_B$
Color	CIELAB	$c_L = a_L$
		$c_A = (1-\alpha)a_A + \alpha b_A$
		$c_B = (1-\alpha)a_B + \alpha b_B$

Table 6.1: Blend modes supported in Flowtoshop. In the formulae, the color b is composited on top of a with opacity/fill α to produce color c. Subscripts represent components in RGB or LAB space, or are 'x' to represent any component.

6.2.7 Software Design

In order to provide this direct interaction, Flowtoshop is organized as shown in Figure 6.12. Flowtoshop uses Cocoa for the GUI implementation and G3D for rendering. All Flowtoshop code depends on Flowtoshop Core which provides foundational classes (such as accessing data, handling color, and mapping between screen and data coordinates). On top of this, an interoperability layer allows higher layers to couple only to this interoperability layer and the required lower layers, enabling, for example, the future possibility of replacing the GUI toolkit without needing to adjust rendering code. On top of this layer sit the rendering and GUI layers. The GUI layer relies on the rendering layer, but not upon the details of how rendering is actually implemented.

6.3 Applications and Design Studies

In this section, we present two different applications of Flowtoshop. The first is applying Flowtoshop to two different climate datasets made available by the National Oceanic and

Atmospheric Administration (NOAA). The second application is the reimplementation of past visualization styles in Flowtoshop, along with an analysis of data collected during this design process.

6.3.1 NOAA Climate Data

Flowtoshop was designed to support the analysis of 2D time-varying datasets with multiple scalar and vector fields. The two main datasets currently supported by Flowtoshop are climate datasets provided by NOAA. The first dataset is the Climate Forecast System Reanalysis (CFSR), which provides hourly data over the entire Earth from 1979 until 2009 [75]. Figures using the CFSR dataset include Figures 6.6 and 6.7. The second dataset is the North American Mesoscale Forecast[2], which is updated every six hours and provides forecasts over the continental United State for the upcoming 84 hour period. The first 36 hours of this dataset are available at one-hour intervals; after 36 hours the data is sampled at three-hour intervals. For use in Flowtoshop, we have limited the dataset to the first 36-hours so that higher temporal resolution can be achieved. The NAM dataset is used in Figures 6.13 and 6.1 (as well as others).

The data files are provided in the grib2 format and are processed prior to being loaded in Flowtoshop. The files are processed in order to remove unnecessary variables and to convert the files into a regularly gridded file of IEEE floats that is conducive to direct memory mapping. The datasets after processing are described in Table 6.2.

By directly memory mapping the dataset, Flowtoshop is able to take advantage of the operating system's virtual memory subsystem to randomly access any variable at any point in time with minimal latency. On a system running OS X 10.9.2, Flowtoshop is able to memory map over a terabyte of data, allowing direct access to eleven years of

[2]http://www.ncdc.noaa.gov/data-access/model-data/model-datasets/north-american-mesoscale-forecast-system-nam

	CFSR	**NAM**
Extents	180°W – 180°E	135°W – 65°W
	90°N – 90°S	55°N – 25°N
Resolution	$0.5° \times 0.5°$	$0.125° \times 0.125°$
Scalar Fields	—	Cloud Cover
	Geopotential height	Geopotential height
	Humidity	Humidity
	Precipitable Water	Precipitable Water
	Precipitation	Precipitation
	—	Pressure (jet stream)
	Sea-level Pressure	Sea-level Pressure
	Surface Pressure	Surface Pressure
	Temperature	Temperature
	Wind Speed (10m)	Wind Speed (10m)
	—	Wind Speed (jet stream)
Vector Fields	Wind Velocity (10m)	Wind Velocity (10m)
	—	Wind Velocity (jet stream)
Start Date/Time	January 1, 1999 0:00 UTC	Today, 0:00 UTC
End Date/Time	December 31, 2009 23:00 UTC	Tomorrow, 12:00 UTC
Sample Rate	Every hour	Every hour
Total Size	1.2 TB	360 MB
Example Figures	6.6, 6.7	6.13, 6.1

Table 6.2: Parameters of the two NOAA climate datasets, after processing.

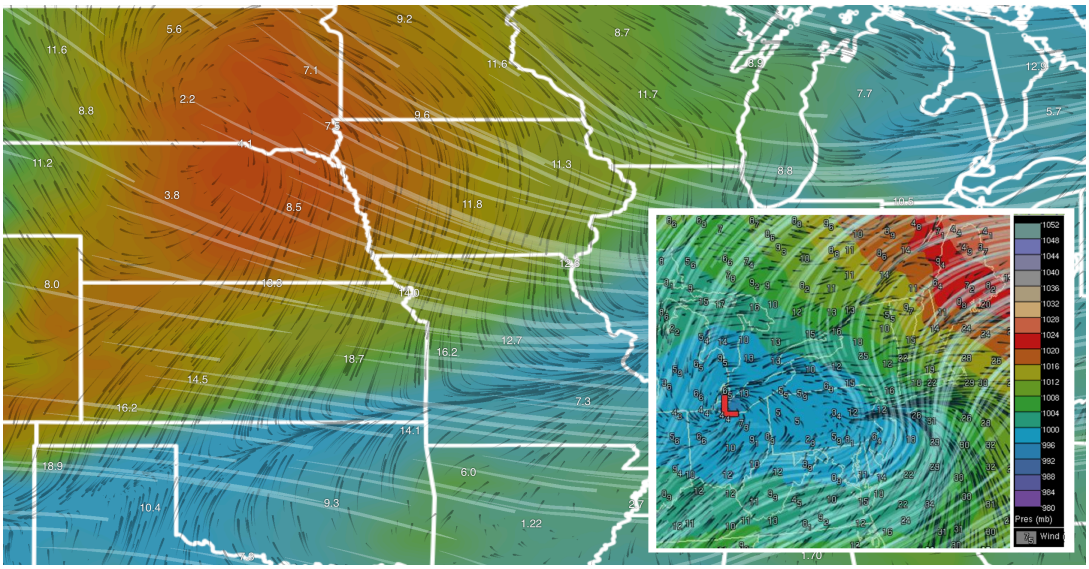

Figure 6.13: Flowtoshop is flexible enough to recreate styles from a number of different visualization publications. Here, a visualization designed by Ware et al.[1] is recreated in Flowtoshop (The original visualization is shown in the inset). The background colormap maps surface barometric pressure to color using a similar color scale to Ware et al. On top of that, black particles trace the direction of surface winds, white particles trace the direction of jet stream winds, and text labels show the wind speed in miles per hour.

global climate data.

6.3.2 Recreating Past Visualizations

Flowtoshop was designed to be a visualization design tool that allowed users a wide range of visual design freedom. In order to assess to what extent we achieved this goal, we attempted to recreate a range of past visualization results in Flowtoshop.

Figure 6.13 shows a re-creation of a weather visualization published by Ware et al. [1]. This visualization is applied to a similar dataset as that used in the original publication, so the same variable mappings were used. This visualization shows wind velocity at the surface and jetstream layer (as black and white particles, respectively), as well as surface pressure. Surface wind speed is additionally depicted as numbers overlaid

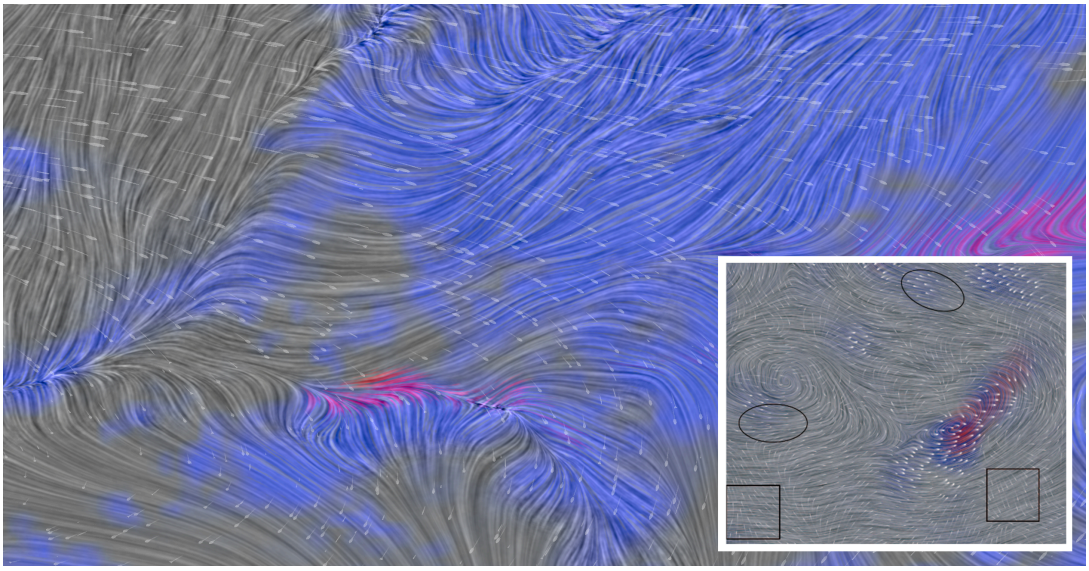

Figure 6.14: Flowtoshop's layer-based approach is sufficiently flexible to produce visualizations similar to those published as the result of a novel algorithm. Here, the color weaving algorithm for LIC visualizations presented by Urness et al.[2] is approximated by Flowtoshop. The original visualization is shown in the inset.

on the visualization space. Re-creating this visualization took around ten minutes, which was mainly spent on trying to match the pressure gradient. The most significant difference between Ware's visualization and our re-creation is our lack of a banded colormap.

Figure 6.14 shows a re-creation of a visualization technique published by Urness et al. [2]. The published visualization technique aims to encode multiple scalar variables on top of a LIC background using color weaving. The Flowtoshop implementation does *not* implement color weaving; instead, the implemented blend modes (such as the Lightness and Lighten blend modes) are used to cause the red regions (representing areas of precipitation) and blue regions (representing cloud cover) to take the form of the LIC background. This structure is easier to see when only one variable is shown in the color map, as in Figure 6.15. The images produced by Urness et al. are clearly

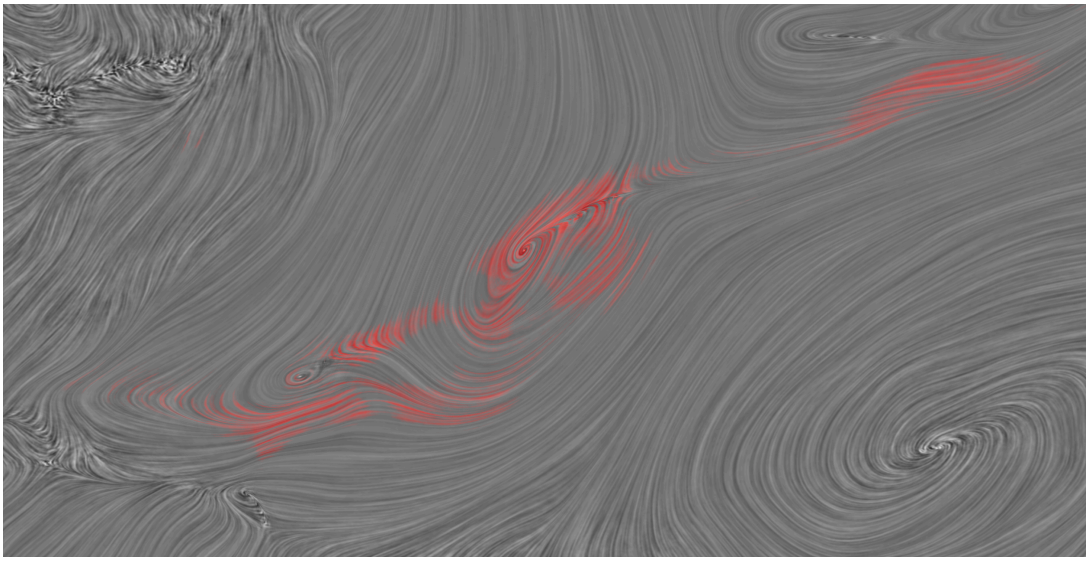

Figure 6.15: By displaying only one variable in a colormap, the similarities between Flowtoshop's output and the algorithm presented by Urness et al.[2] can be seen. Flowtoshop is able to approximate this intricate effect through a combination of its layer-based design and creative use of blend modes.

of a higher quality than that produced by Flowtoshop; nonetheless, the ability for a general tool such as Flowtoshop to recreate this advanced and specialized visualization technique without requiring any programming is a significant accomplishment.

Figures 6.16, 6.17, and 6.18 show additional visualization styles recreated in Flowtoshop. All of these style recreations were produced in around 30 minutes and required no programming experience. Flowtoshop is able to successfully recreate a wide range of visualization styles, showing that the layer and blend mode-based interface is flexible and powerful.

Collected Data

During the recreation of these previous visualizations, Flowtoshop logged information, including the beginning and end times of drawing operations, the computed "weight

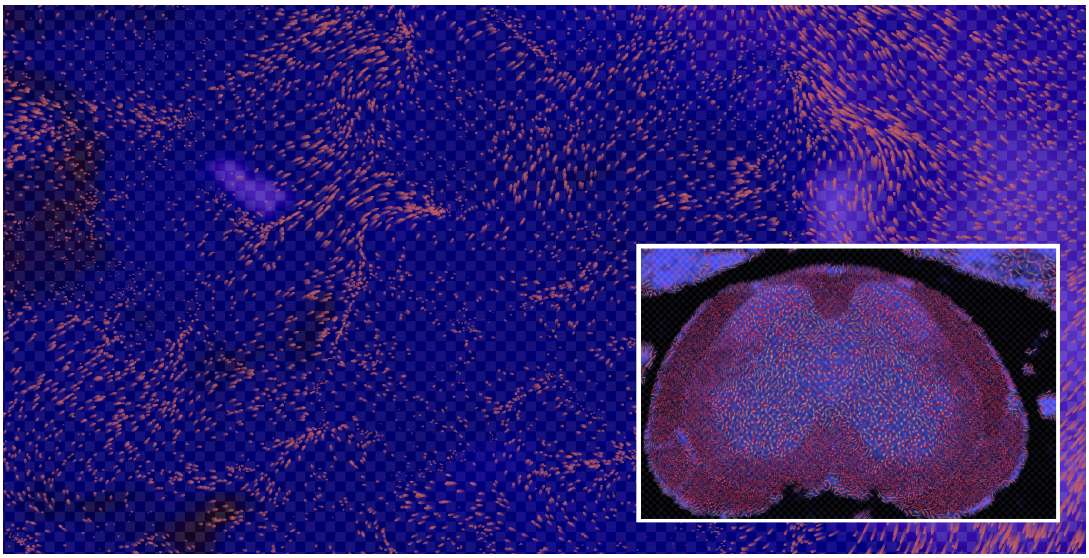

Figure 6.16: Flowtoshop is able to approximate styles of visualizations that represent completely different data. Here, Laidlaw et al.'s diffusion tensor imaging of the mouse spinal cord[3] is approximated by a visualization of weather phenomena. Laidlaw et al.'s visualization is shown in the inset.

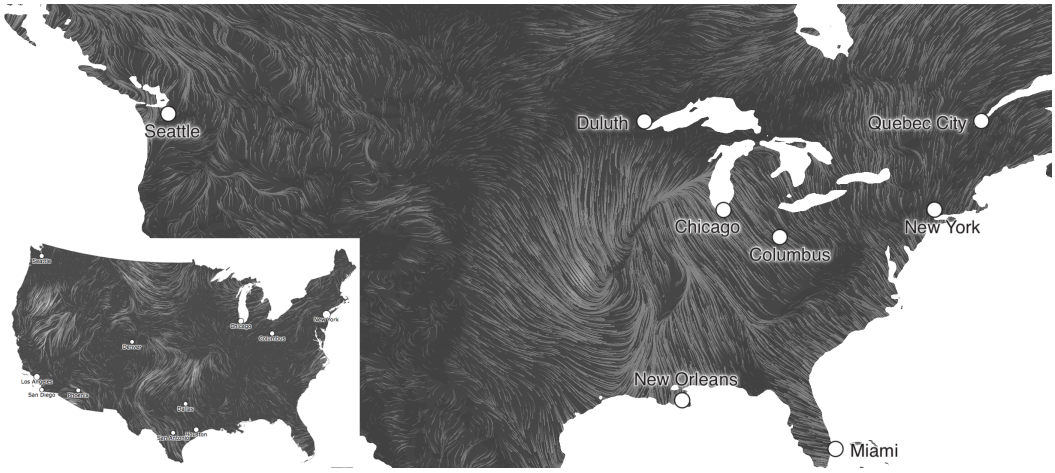

Figure 6.17: Flowtoshop is able to reproduce visualizations that have reached a wide appeal. Here, the "hint.fm wind map" (Viewable at `hint.fm/wind`, and reproduced in the inset) is reproduced in Flowtoshop.

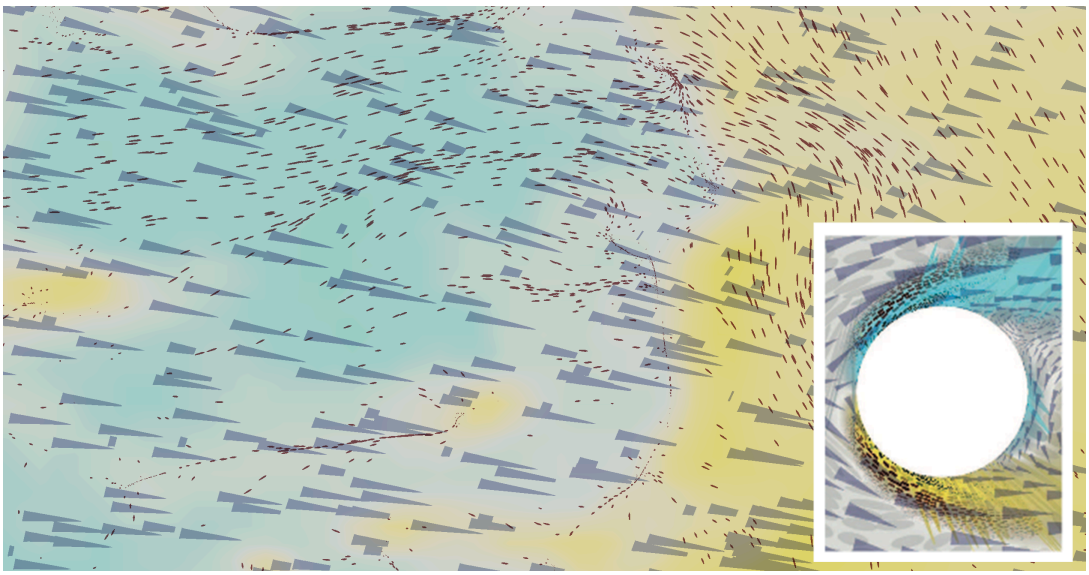

Figure 6.18: Kirby et al. presented a visualization design using concepts from painting to depict many different values in a 2D fluid flow (shown in the inset).[4] Flowtoshop does not support all of the visualization elements used, but is still able to approximate the style.

	Ware et al.	**Urness et al.**	**Laidlaw et al.**	**hint.fm**	**Kirby et al.**
Figure	6.13	6.14	6.16	6.17	6.18
Time Taken	44:14	37:25	6:09	13:27	15:14
Number of Draw Operations	56	54	2	52	31
Drawing Time					
Minumum	0s	0s	4s	0s	0s
Maximum	7s	10s	7s	6s	4s
Median	2s	0s	5.5s	1s	1.0s
Mean	1.7s	1.4s	5.5s	1.4s	1.7s
Standard Deviation	1.5s	2.4s	1.5s	1.5s	1.4s
Gap Between Draw Operations					
Minimum	1s	0s	47s	0s	0s
Maximum	248s	101s	47s	72s	190s
Median	5.0s	1s	47s	2s	4s
Mean	15.0s	5.7	47s	6.5s	11.8s
Standard Deviation	34.9s	14.5s	0s	13.6s	34.1s

Table 6.3: Statistics on the use of Flowtoshop in the creation of Figures 6.13, 6.14, 6.16, 6.17, and 6.18. During use, timestamps of user interface actions were tracked to an accuracy of one second. Drawing time measures the time elapsed between the mouse down action initiating gradient drawing and the mouse up action ending drawing. Gap between draw operations measures the time elapsed between the mouse up action ending a drawing operation and the mouse down action initiating the next drawing operations. Gaps that spanned multiple executions of Flowtoshop were ignored.

influence" for each draw operation (discussed in Section 6.2.1), and the time that Flowtoshop was started and quit. All of these were logged along with a timestamp with a one-second accuracy. This data is summarized in Table 6.3. Time taken is computed as being the difference between Flowtoshop being launched and the final "save" operation performed. When multiple sessions were used to make a single figure, the timing for the individual sessions were added together. Drawing time is computed as the difference in time between the mouse down even and the gradient being updated. None of these drawing operations required the user to wait for the bandpassed scalar field, avoiding the need to account for this time. Gap between draw operations is computed as the difference in time between the end of one drawing operation and the start of the next operation.

Our goal with the analysis is to perform a first polot study to lay the groundwork for a larger study with practicing artists. One of the authors served as the designer for this pilot study.

First, we want to understand how effective the painting vs. dabbing colormap interface is at distinguishing users' intent. Recall from Section 6.2.1 how "weight influence" is used in the brushing interface to distinguish between painting and dabbing. This is one of the most critical user interface decisions, so understanding the computed weights is critical to evaluating Flowtoshop. Figure 6.19 shows the computed weight influence for all of the brushing operations performed for all of the figures presented in Table 6.3. Clearly many draw operations were performed with a low weight influence. Over 13% of gradient paint operations had a weight influence of less than 0.01, and over 60% had a weight influence of less than 0.1. This suggests that most gradient draw operations were performed on relatively smooth areas of the dataset. At the same time, there is a loose group of weight influences between 0.4 and 0.6, which could correspond to when the user was making a clear adjustment to a minimum or maximum. Future user studies

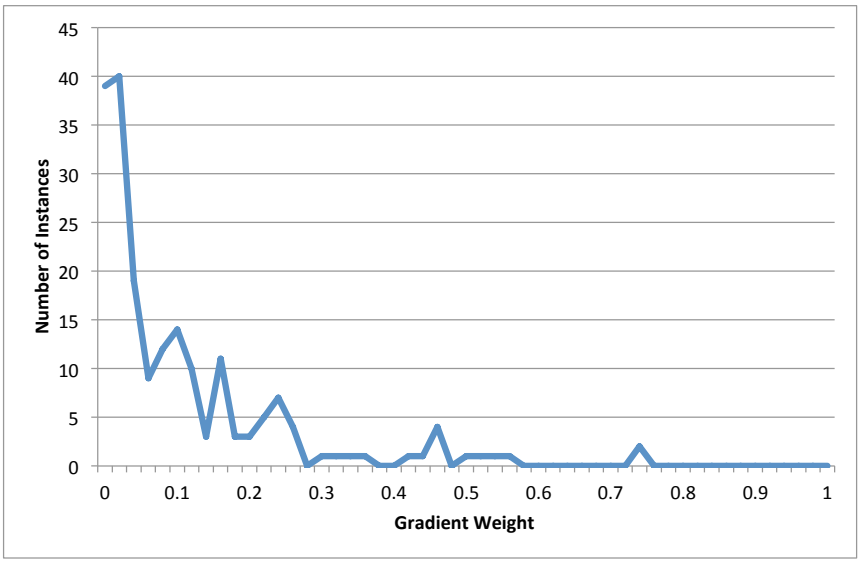

Figure 6.19: A histogram of all computed weight influences during the recreation of previous visualization results. Between the fact that over 60% of all paint operations had a weight influence below 0.1, and the appearance of a slight increase in weight influences in the range of 0.4 – 0.6, it seems that the weight influence may be able to successfully determine the difference between painting and dabbing.

will be needed to determine what relations if any exist between the computed weight influence and how well the user feels their input was interpreted.

Second, we examined the data in Table 6.3, to see what patterns might exist. Looking at the drawing time, we noticed that there seemed to be a non-normal distribution. For example, for the Urness et al. design, the mean time taken to perform a draw operation is 1.4s, yet the standard deviation is 2.4s (with a maximum of 10s). This tells us that the designer generally took a short amount of time for each draw operation, yet had a few that were significantly longer. A similar pattern is visible in the Ware et al. design, with a mean of 1.7s, standard deviation of 1.5s, and maximum of 7s. In the other designs, this patterns is less visible but possibly still present. This is interesting because it suggests that the time taken to perform a draw operation is not simply distributed, indicating that there may be two or more separate conceptual "modes" users are in while using the drawing interface. This leads to one hypothesis for follow on studies: Within a single user's recorded drawing times, there will be a division between short and gestural inputs and longer and more precise inputs. and this division will correspond to whether users are intending to make a rough or precise adjustment to the color map.

Finally, we noticed that users used the drawing tool in stages, spending periods of time spent in the drawing tool, and periods of time outside the drawing tool. Examining the gap between the draw operations, we see evidence for this hypothesis. Gaps between drawing operations range between less than one second[3] to over four minutes. The relatively large discrepancy between median and mean for all cases with a large number of draw operations and the fact that the mean is consistently higher than the median

[3]With the quantization of timesteps to the nearest second, it is possible for a gap of 0.9 seconds between draw operations to be recorded as a gap of zero seconds, as in the case where one draw operation ends at timestep 10.0 and the next one begins at 10.9. Similarly, a gap of 0.1 seconds can be recorded as a gap of one second. With this quantization in mind, minimum and maximum values should be viewed with an accuracy of $\pm 1s$.

suggests that most gaps between draw operations are on the order of a few seconds, while a few gaps are of a much longer length.

6.4 Discussion

6.4.1 Style Transfer

All of the visual elements supported by Flowtoshop (with the exclusion of the image and annotation layers) are defined based only on the variable being visualized and some variable-agnostic parameters. Therefore, transferring a visualization style from one timestep to a different timestep, or from one dataset to a different dataset with the same variables is easily accomplished. In this way, Flowtoshop can be used to define visualization presets that can be immediately applied to applicable datasets. This is particularly interesting for datasets such as the NAM which are updated every six hours, in that visualizations designed one day can be applied to the next day's data.

6.4.2 Future Work

We are already extending on the work presented in this chapter. We have started multiple collaborations with domain scientists and illustrators studying various climate datasets. We are working with atmospheric scientists at the University of Utah on the visualization of radiant energy simulations and on enabling scientists to efficiently compare multiple simulation runs to reference data collected in the field. We are also working with an artist employed by scientists at the Los Alamos National Laboratories studying the change in oceanic properties (such as salinity and temperature) as a result of anthropogenic environmental changes. In both of these collaborations, we seek to evaluate the performance of interfaces such as Flowtoshop for designing visualizations and examining complicated datasets.

Currently, Flowtoshop's layer support is simple compared to that of tools such as Photoshop. Flowtoshop has a stack of layers, while Photoshop supports the ability to arbitrarily nest layers and to extract a set of layers into a "Smart Object" that can then exist at multiple points in the layer hierarchy. Additionally, "Adjustment Layers" allow global adjustments to be applied to all underlying layers, such as adjusting color balance or exposure. Changing Flowtoshop's layer model from its current single list to a more complicated directed acyclic graph would greatly increase its power and ability to recreate a wide range of different visualizations. Adding in adjustment layers would allow users to more easily tweak their visualizations without needing to go through complicated or laborious processes.

Scientists generate many 3D, time-varying datasets. Flowtoshop currently only supports 2D datasets which limits its applicability to these datasets. Flowtoshop can be fairly easily embedded as a cutting plane in a higher-dimensional dataset, but there is also the possibility of either applying Flowtoshop's visualization elements to a potentially complicated 3D surface or extending the interactions from Flowtoshop into a 3D, volumetric dataset. Making this transition would be difficult, but could enable scientists and other users to take the intuitive interaction style from Flowtoshop to much more complicated 3D datasets.

We are also keen in the future to evaluate Flowtoshop's performance when used by scientists working with real datasets. Goals from this evualuation include: assessing the utility of Flowtoshop compared to scientists' current tools, evaluating the performance of the colormap painting interface, gathering general feedback about the tool and tools in this direction, and evaluating its ability to create stories about relevant patterns in the data.

User Study

In the future, we plan to build on the pilot study in Section 6.3.2 and to evaluate Flowtoshop with a larger study to determine to what extent our design decisions enable the creation of useful and accurate visualizations.

Since Flowtoshop takes a fundamentally different approach to visualization design than tools like ParaView, a direct quantitative comparison of the tool as a whole is not possible. Nonetheless, subsets of Flowtoshop's features can be compared. In particular, the gradient painting tool for defining color maps is ideal for comparison since an existing interface is a de facto standard (the gradient control point interface seen in tool such as Adobe Photoshop, ParaView, and other tools). Making this comparison trickier, some aspects of the standard interface (e.g., linear vs. spline color interpolation, interpolation color space, color of newly inserted control points, behavior of a gradient segment's "halfway" control) are not consistently chosen, and this inconsistency could very likely change the performance of the de facto standard interface.

The evaluation of the gradient tools should involve both quantitative and qualitative elements. A quantitative comparison could task users with reproducing either exemplar gradients or exemplar colormapped data, with performing adjustments with varying degrees of locality, or with highlighting specific features in the data. Once these tasks are performed, different metrics such as time taken, accuracy, gradient smoothness and perceptual linearity, and number of "undo" operations could be compared to explore the differences between these two interface. The qualitative comparison could compare aspects of the interfaces that are difficult to quantify, such as the perceived amount of control, freedom, and enjoyment, as well as the possibility of serendipitous colormap creation.

For such a study, there are a few hypotheses. First, we hypothesize that matching

exemplar gradients will be faster and more accurate with a traditional gradient interface, but that matching exemplar colormapped images will be faster and more accurate with the gradient painting interface. Second, we hypothesize that gradient painting will allow users to more rapidly create colormaps to emphasize specific subsets of the data. Third, we hypothesize that users will find gradient painting to be more free and enjoyable, and similarly controllable to the conventional gradient interface. Fourth, we hypothesize that users will use the "undo" operation far more often with the gradient painting interface since it introduces a layer of indirection between the user's input and the resulting effect on the gradient. Finally, we hypothesize that users will enjoy using the color painting interface more, and that they will find the interaction more intuitive.

6.5 Conclusion

In this chapter, we presented Flowtoshop, a visualization design tool that allows users to create intricate and complicated visualizations using a novel direct visualization creation interface. This work provides evidence that visualization design tools accessible to artists and scientists can support the creation of difficult-to-design visualizations. For multiple previously published visualization designs, we were able to, in less than an hour, recreate these visualizations. Interfaces in a similar style to Flowtoshop have the potential to more efficiently support visualization development and design.

Chapter 7

Conclusions and Discussion

In this dissertation, we have presented four major research contributions in two different but complementary research thrusts that advance the field of visualization as a whole. In this chapter, we will review the contributions, will discuss potential directions of future work, and end with some concluding remarks.

7.1 Summary

7.1.1 Visualizing Surgical Training Databases

The first two technical chapters in this dissertation cover advanced visualization practice, with an emphasis on the difficult problem of motion visualization. Chapter 3 addresses some problems that arise in exploring large, multidimensional datasets. These datasets include multiple motions, each with multiple complementary forms of data, such as time-indexed scalar variables, time-indexed videos, and whole-motion statistics. The engineers and surgeons who collected this data were interested in finding patterns and trends that might help to distinguish between expert and novice surgeons, and that may suggest methods to improve surgeon training. In order to enable this analysis, we

developed several novel visualization features.

We introduce an augmented video view that maintains the view surgeons typically use to evaluate surgical performance but adds to it additional information on invisible quantities that are nonetheless important to patient outcomes. By showing on either side of the screen the force applied to the tools, surgeons are able to observe more of the task being performed, without needing to abandon the views they are already used to analyzing.

An intelligent selection expansion mechanism allows users to select regions of interest, with the system then intelligently expanding the selection to maintain the context of the selected frames. The entire surgical training dataset consisted of nearly four hours of tasks being performed, and observing this entire process would take a prohibitive amount of time. By adding this selection mechanism, users can select subsets of the data that would be impossible to select with a standard brushing operation.

We also created a 3D visualization where users can rotate around the space and see in 3D a task they are used to evaluating in two dimensions. Despite the surgical task occurring in three dimensions, surgeons typically evaluate performance by watching a two-dimensional video. In our 3D visualization, tool traces are drawn in 3D, either as an annotated ribbon that depicts orientation, speed, grasp force, and grasping events, or as a simple tube that shows which tools are currently grabbing. With these visualizations, depth mis-estimation errors that are very difficult to see were made readily apparent.

Together, these contributions help improve the process of visualizing a complicated motion and of filtering the motion down to a relevant subset for further analysis.

7.1.2 Trend-Centric Motion-Set Visualization

In Chapter 4, we present a system that allows users to consider an entire collection of motions at once. With this system, users can take a collection of hundreds of patients

performing a single task, and can find the patterns that emerge and change over time. Several contributions were required to enable this form of interaction.

First, we define trends and present an algorithm for identifying them in temporally aligned data. Trends represent patterns of multiple motions behaving in a coherent manner for a stretch of time. In contrast with the results of conventional clustering, trends have a defined temporal extent, and a single motion can belong to different trends and multiple points in time. This captures the way that a motion can be similar to one set of motions at one point in time, but a different set of motions at a different point in time.

Second, we present a 2D subway map-style visualization of the trends found in a motion dataset. This visualization provides an overview of the general structure of the motion dataset, and identifies the major trends in a motion collection, as well as the smaller trends that branch between them. In an interactive visualization system, this visualization is used to select a trend (and its child trends) for more detailed visual analysis. In this role, the subway map visualization is used both as a reference for position in a larger dataset, and also as way to navigate around the dataset.

Finally, we present an illustrative visualization that depicts an entire motion at a time. A number of visual elements are used to visually identify the median motion in a trend, to depict the motions that are "plus or minus one standard deviation" from the median motion, and to show the overall range of motions in the dataset. Discs located to the right of the neck bones provide a view into the physiologically important distances between neck vertebrae. This visualization was developed in close collaboration with a trained graphic designer which led to a design that would not have been reached otherwise.

With these three major contributions, users are able to get an overview of, navigate, filter, and understand complicated biomechanics dataset that can include hundreds of

motions and complex and subtle patterns.

7.1.3 Sketch-Based 2D Vector Field Visualization

Based on the results from Chapter 4, we realized the need for visualization design tools, which are examined in the last two technical chapters. In Chapter 5, we present a first effort in this direction: a streamline visualization tool where users are able to draw the streamlines they are interested in examining. In order for this tool to be realized, several problems had to be solved. First, users' input had to be rectified with the underlying data to ensure that users' input is converted into accurate streamlines that still are accurate to the users' intent. Second, gestures needed to be provided to allow the user to modify the current state of the visualization. Third, an algorithm had to be determined to allow the tool to automatically perform repetitive tasks when the users' intent can be determined.

Illustrators and other artists who used this tool were pleased that they finally had a tool that allowed them to take their visual expertise and apply it to a scientific problem. More interestingly, engineers that we showed this tool to were interested in using the tool for their own analysis and dataset exploration even in the absence of an illustrator. This suggests that it is worth continuing to explore tools in this direction.

7.1.4 Artist's Interface for Visualization Creation

In Chapter 6, we extended the ideas from Chapter 5 and applied them to time-varying, multidimensional data. We created a tool that provided a familiar interface to artists and designers and allowed them to use their expertise to design visualizations of complicated multidimensional data. Several research contributions allow this tool to be natural to use and to allow a user to directly manipulate the visualization, instead of manipulating sliders and parameters.

A layer-based design allows users to compose various visual elements, ranging from colormaps and isocontours to external image files used as either reference or design elements. These layers have a linked temporal and physical position, ensuring that the resulting visualization represents a coherent view of the data. Various blend modes allow users to compose the visualization elements in a variety of ways.

A gradient painting algorithm allows users to draw directly on a colormap visualization to update the underlying colormap. This algorithm provides users with a large assortment of the digital painting tools and controls they are familiar with from tools like Adobe Photoshop. A "painting versus dabbing" algorithm determines how to interpret user's input based on the values and structure of the underlying data. As a result, users are able to adjust sections of the colormap, ranging from a very narrow section to a broad range.

A particle design interface allows users to finely adjust the parameters of particles that advect along the direction of wind flow. The particles' profile is adjusted by drawing a new profile over an exemplar stroke. Other particle parameters are adjusted in a display that shows variations of the current settings. By clicking on a variation, the user can select that variation as the current variation, resulting in a new set of variations.

When combined with the ability to define different scenes with different visualization designs, this allows users to tell stories about the dataset they are working with. One of the datasets used is an 11-year dataset of global climate conditions. With this data, users can tell the stories of how climate affects everyone, ranging from drought to wildfires to hurricanes.

7.2 Future Work

The work presented in this dissertation present many potential directions of future work. Below, we detail three potential future research directions.

7.2.1 Extending Trends

In Chapter 4, we presented trends as a pattern in motion datasets that are useful for directing analysis. However, our trend formulation is not directly applicable to datasets such as scalar or vector fields, either static or time-varying. Quantities such as these can arise in contexts ranging from finite element analysis of a surgical device to power use of a proposed building in a dense urban environment. In both of these cases, a wide range of parameters can vary that change the behavior of the system. We are interested in investigating to what extent a trend-based system could allow users to more easily understand the behavior of the system and to arrive at desired end results.

7.2.2 Designing for Comparisons and Uncertainty

Comparisaon (either between multiple datasets, or within a single dataset) is a fundamental task, and yet designing visualizations to support comparisons is difficult. Similarly, depicting the uncertainty in a single dataset, or depicting the range in a set of datasets is still an open challenge that will likely require novel solutions to solve. If the power of tools such as Flowtoshop were extended to support multiple datasets and the notion of the range of a variable or the uncertainty in a measurement, many additional users ranging from scientists to visual designers could help solve this problem.

Adding this increased power is a difficult challenge. Views such as the particle flow layer will need to be able to incorporate information on uncertainty in a wide range of potential ways (e.g., blurring particles as the vector field becomes more uncertain,

modifying particles' parameters based on their current positional uncertainty, etc.). Users will need to be able to define multiple views, either on one dataset or on multiple datasets, and define how the views are linked (e.g., always showing the same geographic region, having a fixed time offset between multiple views, etc.).

7.2.3 Future Visualization Design Tools

Drawing with the Flow (Chapter 5) and Flowtoshop (Chapter 6) are the first two interfaces created in this style. There are different types of data (e.g., volumetric, categorical, tensor) different data dimensionalities (e.g., 2D, 2D+time, 3D+time), different display and interaction modalities (e.g., 2D WIMP, 3D wand-based, 3D multitouch), and different data visualization purposes (e.g., within-single-dataset comparison, between-dataset comparison, pedagogical) that all affect how a visualization design tool should be designed. While not all combinations of these parameters make for a reasonable visualization design, visualization design tools are a currently underexplored research direction, and we plan to continue our research in this direction.

7.3 Outlook

In this dissertation, we presented work in both advanced visualization practice and in visualization design tools. Research in both of these areas will doubtless lead to advances for the analysis of scientific data and will advance scientific knowledge. Future advances in motion visualization will allow users to more easily see patterns and act upon the data they've collected, and are critical to advancing scientific knowledge.

However, while advanced visualization practice has a large body of work, there has been comparably little research into visualization design tools. With the increasing size and complexity of scientific datasets, and our continually increasing computational

resources, visualization design is becoming a harder problem to solve. It's time to create the tools that will allow illustrators, artists, scientists, and the general public to help solve these problems by increasing the number of people who have the necessary skills and resources to create and design visualizations. Visualization design tools have the potential to *fundamentally change* how visualization research is performed, and to help us as visualization researchers solve the big problems facing our field.

References

[1] Colin Ware and Matthew Plumlee. Designing a better weather display. In *IS&T/SPIE Electronic Imaging*, pages 829409–829409. International Society for Optics and Photonics, 2012.

[2] Timothy Urness, Victoria Interrante, Ivan Marusic, Ellen Longmire, and Bharathram Ganapathisubramani. Effectively visualizing multi-valued flow data using color and texture. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 16–, Washington, DC, USA, 2003. IEEE Computer Society.

[3] David H. Laidlaw, Eric T. Ahrens, davidkremers, Matthew J. Avalos, Carol Readhead, and Russell E. Jacobs. Visualizing diffusion tensor images of the mouse spinal cord. In *Proceedings of IEEE Visualization 1998*, pages 127–134, 1998.

[4] Michael Kirby, H. Marmanis, and David H. Laidlaw. Visualizing multivalued data from 2D incompressible flows using concepts from painting. In *Proceedings of IEEE Visualization 1999*, pages 333–340, 1999.

[5] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages*, page 336, Washington, DC, USA, 1996. IEEE Computer Society.

[6] D. Keefe, M. Ewert, W. Ribarsky, and R. Chang. Interactive coordinated multiple-view visualization of biomechanical motion data. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1383–1390, 2009.

[7] J. Ma, I. Liao, Kwan-Liu . L. Ma, and J. Frazier. Living liquid: Design and evaluation of an exploratory visualization tool for museum visitors. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2799–2808, 12 2012.

[8] G Elisabeta Marai, Cagatay Demiralp, Stuart Andrews, and David H Laidlaw. JointViewer – an interactive system for exploring orthopedic data. In *Proceedings of the conference on Visualization'04*, pages 598–35. IEEE Computer Society, 2004.

[9] Gopal Pingali, Agata Opalach, Yves Jean, and Ingrid Carlbom. Visualization of sports using motion trajectories: Providing insights into performance, style, and strategy. In *Proceedings of the Conference on Visualization '01*, VIS '01, pages 75–82, Washington, DC, USA, 2001. IEEE Computer Society.

[10] Jurgen Bernard, Nils Wilhelm, Bjorn Kruger, Thorsten May, Tobias Schreck, and Jorn Kohlhammer. MotionExplorer: Exploratory search in human motion capture data based on hierarchical aggregation. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2257–2266, 2013.

[11] T. Crnovrsanin, C. Muelder, C. Correa, and K.L. Ma. Proximity-based visualization of movement trace data. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, pages 11–18. IEEE, 2009.

[12] Gennady Andrienko, Natalia Andrienko, Christophe Hurter, Salvatore Rinzivillo, and Stefan Wrobel. From movement tracks through events to places: Extracting and characterizing significant places from mobility data. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 161–170. IEEE, 2011.

[13] 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, March/April 2004.

[14] Colin Ware, Roland Arsenault, Matthew Plumlee, and David Wiley. Visualizing the underwater behavior of humpback whales. *IEEE Computer Graphics and Applications*, 26(4):14–18, 2006.

[15] S. Bouvier-Zappa, V. Ostromoukhov, and P. Poulin. Motion cues for illustration of skeletal motion capture data. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 133–140. ACM, 2007.

[16] D. Coffey, F. Korsakov, M. Ewert, H. Hagh-Shenas, L. Thorson, A. Ellingson, D. Nuckley, and D.F Keefe. Visualizing motion data in virtual reality: Understanding the roles of animation, interaction, and static presentation. *Computer Graphics Forum*, 31(3pt3):1215–1224, 2012.

[17] D. Acevedo, C.D. Jackson, F. Drury, and D.H. Laidlaw. Using visual design experts in critique-based evaluation of 2D vector visualization methods. *Visualization and Computer Graphics, IEEE Transactions on*, 14(4):877–884, 2008.

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

[19] S. Bruckner and M. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 2 2007.

[20] Michael M. Kirby, Daniel F. Keefe, and David H. Laidlaw. *The Visualization Handbook. Painting and Visualization*, pages 873–891. Elsevier Inc., 2005.

[21] P. Mitchell, C. Ware, and J. Kelley. Investigating flow visualizations using interactive design space hill climbing. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 355–361, Oct 2009.

[22] Daniel F. Keefe, David B. Karelitz, Eileen L. Vote, and David H. Laidlaw. Artistic collaboration in designing VR visualizations. *IEEE Computer Graphics and Applications*, 25(2):18–23, 2005.

[23] David Schroeder, Timothy Kowalewski, Lee White, John Carlis, Erlan Santos, Robert Sweet, T. S. Lendvay, Troy Reihsen, and D.F. Keefe. Exploratory Visualization of Surgical Training Databases for Improving Skill Acquisition. *IEEE Computer Graphics and Applications*, 32(6):71–81, November 2012.

[24] Jeffrey H Peters, Gerald M Fried, Lee L Swanstrom, Nathaniel J Soper, Lelan F Sillin, Bruce Schirmer, Kaaren Hoffman, Sages FLS Committee, et al. Development and validation of a comprehensive program of education and assessment of the basic fundamentals of laparoscopic surgery. *Surgery*, 135(1):21–27, 2004.

[25] J. Rosen, J.D. Brown, L. Chang, M.N. Sinanan, and B. Hannaford. Generalized approach for modeling minimally invasive surgery as a stochastic process using a discrete markov model. *Biomedical Engineering, IEEE Transactions on*, 53(3):399–413, March 2006.

[26] N. Zhang, Xiangmin Zhou, Yunhe Shen, and R. Sweet. Volumetric modeling in laser BPH therapy simulation. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1405–1412, Nov 2010.

[27] Stefan Diepenbrock, Jörg-Stefan Praßni, Florian Lindemann, Hans-Werner Bothe, and Timo Ropinski. Pre-operative planning of brain tumor resections. *IEEE Visualization Contest*, 2010.

[28] Henry Fuchs, Mark A. Livingston, Ramesh Raskar, D'nardo Colucci, Kurtis Keller, Andrei State, Jessica R. Crawford, Paul Rademacher, Samuel H. Drake, and Anthony A. Meyer. Augmented reality visualization for laparoscopic surgery. In William M. Wells, Alan Colchester, and Scott Delp, editors, *Medical Image Computing and Computer-Assisted Interventation – MICCAI'98*, volume 1496 of *Lecture Notes in Computer Science*, pages 934–943. Springer Berlin Heidelberg, 1998.

[29] 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.

[30] David Schroeder, Fedor Korsakov, Carissa Mai-Ping Knipe, Lauren Thorson, Arin M. Ellingson, David Nuckley, John Carlis, , and Daniel F Keefe. Trend-centric motion visualization: Designing and applying a new strategy for analyzing scientific motion collections. In *IEEE Visualization 2014 (to appear)*, 2014.

[31] Daniel F. Keefe, Trevor 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. 27(3):863–870, 2008.

[32] Nathaniel E Helwig, Sungjin Hong, Elizabeth T Hsiao-Wecksler, and John D Polk. Methods to temporally align gait cycle data. *Journal of biomechanics*, 44(3):561–566, 2011.

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

[34] Wei-Hsien Hsu, Jianqiang Mei, Carlos D Correa, and Kwan-Liu Ma. Depicting time evolving flow with illustrative visualization techniques, 2009.

[35] Alark Joshi and Penny Rheingans. Evaluation of illustration-inspired techniques for time-varying data visualization. *Computer Graphics Forum*, 27(3):999–1006, 2008.

[36] Aidong Lu and Han-Wei Shen. Interactive storyboard for overall time-varying data visualization. *Pacific Visualization Symposium, 2008 IEEE*, 2008.

[37] Niloy J Mitra, Yong-Liang Yang, Dong-Ming Yan, Wilmot Li, and Maneesh Agrawala. Illustrating how mechanical assemblies work. *ACM Transactions on Graphics*, 29(4):1, July 2010.

[38] Maneesh Agrawala and Chris Stolte. Rendering effective route maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, pages 241–249, New York, New York, USA, 2001. ACM Press.

[39] Hongwei Li, Chi-Wing Fu, Yinggang Li, and Andrew Hanson. Visualizing large-scale uncertainty in astrophysical data. *IEEE transactions on visualization and computer graphics*, 13(6):1640–7, 2007.

[40] Claes Lundstrom, Patric Ljung, Anders Persson, and Anders Ynnerman. Uncertainty visualization in medical volume rendering using probabilistic animation. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1648–1655, 2007.

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

[42] 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.

[43] Arin M Ellingson, Vishal Yelisetti, Craig A Schulz, Gert Bronfort, Joseph Downing, Daniel F Keefe, and David J Nuckley. Instantaneous helical axis methodology to identify aberrant neck motion. *Clinical Biomechanics*, 28(7):731–735, 2013.

[44] Elizabeth L. Brainerd, David B. Baier, Stephen M. Gatesy, Tyson L. Hedrick, Keith A. Metzger, Susannah L. Gilbert, and Joseph J. Crisco. X-ray 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, 2010.

[45] Anita N Vasavada, Siping Li, and Scott L Delp. Influence of muscle morphometry and moment arms on the moment-generating capacity of human neck muscles. *Spine*, 23(4):412–422, 1998.

[46] R. Kosara. Visualization criticism – the missing link between information visualization and art. In *Information Visualization, 2007. IV '07. 11th International Conference*, pages 631–636, 2007.

[47] Bret Jackson, Dane Coffey, Lauren Thorson, David Schroeder, Arin M. Ellingson, David J. Nuckley, and Daniel F. Keefe. Toward mixed method evaluations of scientific visualizations and design process as an evaluation tool. In *Proceedings of the 2012 BELIV Workshop: Beyond Time and Errors - Novel Evaluation Methods for Visualization*, BELIV '12, pages 4:1–4:6, New York, NY, USA, 2012. ACM.

[48] Maxwell J Roberts. *Underground maps unravelled: Explorations in information design*. 2012.

[49] 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.

[50] Christopher G. Healey and James T. Enns. Perception and painting: A search for effective, engaging visualizations. *IEEE Computer Graphics and Applications*, 22(2):10–15, 2002.

[51] L.G. Tateosian, C.G. Healey, and J.T. Enns. Engaging viewers through nonphotorealistic visualizations. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, page 102. ACM, 2007.

[52] Penny Rheingans and David Ebert. Volume illustration: Non-photorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):253–264, 2001.

[53] NA Svakhine, Y. Jang, D. Ebert, and K. Gaither. Illustration and photography inspired visualization of flows and volumes. *IEEE Visualization, 2005. VIS 05*, pages 687–694, 2005.

[54] Donna Cox. Using the supercomputer to visualize higher dimensions: An artist's contribution to scientific visualization. *Leonardo*, 21(3):233–242, 1988.

[55] Cullen Jackson, Daniel Acevedo, David H. Laidlaw, Fritz Drury, Eileen Vote, and Daniel Keefe. Designer-critiqued comparison of 2D vector visualization methods: A pilot study. ACM SIGGRAPH 2003 Sketches and Applications, July 2003.

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

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

[58] G. Turk and D. Banks. Image-guided streamline placement. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, page 460. ACM, 1996.

[59] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. *Visualization in Scientific Computing*, 97:43–56, 1997.

[60] V. Verma, D. Kao, and A. Pang. A flow-guided streamline seeding strategy. In *Proceedings of the conference on Visualization'00*, page 170. IEEE Computer Society Press, 2000.

[61] L. Li, H.H. Hsieh, and H.W. Shen. Illustrative streamline placement and visualization. In *IEEE Pacific Visualization Symposium*, pages 79–86, 2008.

[62] S.H. Bae, R. Balakrishnan, and K. Singh. ILoveSketch: as-natural-as-possible sketching system for creating 3D curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 151–160. ACM, 2008.

[63] Robert C. Zeleznik, Andrew Bragdon, Chu-Chi Liu, and Andrew Forsberg. Lineogrammer: creating diagrams by drawing. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 161–170, New York, NY, USA, 2008. ACM.

[64] David Akers. CINCH: A cooperatively designed marking interface for 3D pathway selection. In *Symposium on User Interface Software and Technology*, pages 33–42, 2006.

[65] R. Sowell, L. Liu, T. Ju, C. Grimm, C. Abraham, G. Gokhroo, and D. Low. Volume viewer: an interactive tool for fitting surfaces to volume data. In *SBIM '09: Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 141–148, New York, NY, USA, 2009. ACM.

[66] T. Isenberg, M.H. Everts, J. Grubert, and S. Carpendale. Interactive exploratory visualization of 2d vector fields. In *Computer Graphics Forum*, volume 27, pages 983–990. Blackwell Publishing, 2008.

[67] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann, 2007.

[68] Brian Cabral and Leith Casey Leedom. Imaging vector fields using line integral convolution. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 263–270, New York, NY, USA, 1993. ACM.

[69] David H. Laidlaw, Michael Kirby, Cullen Jackson, J. Scott Davidson, Timothy Miller, Marco DaSilva, William Warren, and Michael Tarr. Comparing 2d vector field visualization methods: A user study. *IEEE Transactions on Visualization and Computer Graphics*, 11(1):59–70, 2005.

[70] Robert Kosara and Jock Mackinlay. Storytelling: The next step for visualization. *Computer*, 46(5):44–50, 2013.

[71] Kwan-Liu . L. Ma, Isaac Liao, Jennifer Frazier, Helwig Hauser, and H-N . N. Kostis. Scientific storytelling using visualization. *Computer Graphics and Applications, IEEE*, 32(1):12–19, 2012.

[72] H. Akiba, Chaoli Wang, and Kwan-Liu . L. Ma. AniViz: A template-based animation tool for volume visualization. *Computer Graphics and Applications, IEEE*, 30(5):61–71, 9 2010.

[73] Colin Ware. Color sequences for univariate maps: Theory, experiments and principles. *Computer Graphics and Applications, IEEE*, 8(5):41–49, 1988.

[74] David Fowler and Colin Ware. Strokes for representing univariate vector field maps. *Graphics Interface89*, pages 249–253.

[75] Suranjana Saha, Shrinivas Moorthi, Hua-Lu. L. Pan, Xingren Wu, Jiande Wang, Sudhir Nadiga, Patrick Tripp, Robert Kistler, John Woollen, and David Behringer. The NCEP climate forecast system reanalysis. *Bulletin of the American Meteorological Society*, 91(8):1015–1057, 2010.