Data-Driven Exploratory Interfaces for Contextualizing Parameter Spaces: Adding Intuition to Big Data

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

Daniel Orban

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Daniel F. Keefe

Aug, 2021

Acknowledgements

I would like to thank everyone who has experienced this process with me over these years. Graduate school has been both challenging and rewarding, but I could not have done it without the support of so many people.

First and formost, I would like to thank my dissertation comittee Daniel Keefe, Stephen Guy, Victoria Interrante, David Odde, and David Rogers. I appreciate your time and commitment to evaluate my work. Thank you also for your feedback that helped make this thesis successful.

I would like to thank my advisor, Dan Keefe, for giving me the opportunity to purse a doctorate degree. You taught me how to write and reason about truth. Though your example, I can now creatively express my excitement about new ideas. I am continually challened by your pursuite of great visuals and artistic storytelling, areas I aspire to grow. Above all, thank you for mentoring me, especially in teaching me how to teach. I look forward to learning more from you over the years.

To my fellow Ph.D. lab mates Seth, Jung, Volcano, and Bridger. Wow, after so many years it is great to celebrate your success as well. It's good to remember our conversations through our shared struggles. Thanks for all the encouragement through difficult times! To those who came before, Dane Coffey, Bret Jackson, and Kate Schroeder. You inspired my research direction and gave me strategic starting hints. I still have so much to learn from you all. I should also mention others in the IVLab who I have looked up to: Liam, Nikki, Liam, Daniel, Morgan, Sean, Kiet, Devin, and others who have been part of the interactive visualization lab.

Special thanks to those who have mentored me in teaching including Timothy Wrenn, Phil Barry, Dan Challou, Amy Larson, Kevin Wendt, Dan Keefe, Victoria Interrante, and Stephen Guy. I have learned so much over the years through each of you, and have developed some great friendships in the process. To my fellow CSCI 3081 TAs, you will always go down in history as legends in my book. I have so many memories tackling some of the hardest student problems with each of you including Libby Ferland, Shruti Verma, Nikki Kyllonen, Hrushikesh Nimkar, Daniel Olson, John Harwell, Suhail Alnahari, Carter Blum, Michael Ung, Aaron Koenigsberg, Frank Bender II, Clair Li, and so many others. You all will always be heros to me because we survived together.

I want to honor my collaborators, without whom this dissertation would not exist. In particular, I would like to thank Arthur Erdman, Bethany Juhnke, Hakizumwami Birali Runesha, Gregory Peterson, David Rogers, Jim Ahrens, Christine Sweeney, Richard Sandberg, Cindy Bolme, Ayan Biswas, Francesca Samsel, Greg Abrams, Divya Banesh, Jay Hou, Sarah Anderson, and David Odde. I will forever be in debt for teaching me about new worlds of knowledge.

I cannot forget my entire family for supporting me through the Ph.D. process. Mom thank you for you encouragement and excitement. Dad thank you for talking through ideas and challenging me to stay the course. Poppy and Arlo, you two are the best children and I love you both so much. Thank you for dealing with Daddy's long days and nights while still making time to play with me. Madaline Clive, the best mother in law ever, I thank you for helping us out with the kids and for taking us on fun road trips. Grandma and Grandpa Orban, this is for you! I'm so happy to have you in my life right now. To the rest of my extended family, I'm excited that you are excited for this being done.

Finally, and most importantly, I owe everything to my beautiful and wonderful wife, Sonja, who stuck with me through thick and thin. You are the reason for all my success. Thank you for sharing this experience, however humbling, with me. You encouraged me to finish well. Thank you for teaching me to pursue truth no matter what the cost and for it's own beauty. I love you forever and ever!

Dedication

For my friend Archie White.
Your life, faith, conviction, integrity, and humility continue to inspire me.
Thank you for speaking truth into my life.

Abstract

This dissertation investigates how to understand the complexities of large parameter spaces through user interaction. Both researchers and practitioners have embraced the hope of using Big Data (extremely large, heterogeneous, and unstructured datasets) to solve complex problems using statistical methods like artificial intelligence or machine learning. Unfortunately, for many disciplines (e.g. the scientific fields), these computational algorithms are black boxes that provide answers without the important explanations required for developing hypotheses. Interactive visualization offers the promise of adding intuition to Big Data, however, current approaches do not scale to the large data sizes, high-dimensionality, and ill-defined complexity. In order to address these challenges, we introduce Data-Driven Exploratory Interfaces (DDEIs), interfaces that are scalable, enable contextual navigation, and use meaningful feature interaction for intuitive exploration.

Using DDEIs, we analyze Big Data in the scientific context using three separate applications, each focusing on a different "Big" aspect of Big Data. These include medical device design (memory intensive, spatially complex), shock physics (high-dimensional, many instances), and cell migration (memory intensive, spatially complex, high-dimensional, and many instances). In each case, we first look at traditional methods for visualizing and understanding these large datasets, then we overcome the limitations with key concepts introduced by DDEIs (specifically enabling users to define the contexts with respect to their questions, and then explore the relevant trends in the parameter space).

In addition to studying the "Big" aspects of Big Data (Challenge 1), in parallel, we study two other challenges (Challenge 2 and Challenge 3) relevant to understanding Big Data. The second challenge involves investigating approaches to solving the high-dimensional sparsity problems (Challenge 2). Here we use prediction and sampling strategies to analyze the gaps in knowledge. In the third challenge, we consider how to use interaction to navigate and understand ill-defined features in the data (Challenge 3). Using DDEIs, we combine natural exploration techniques with data-driven algorithms to understand application specific features across parameter space sampling properties (input, output, local, global). In this dissertation we show that DDEIs can overcome the limitations of traditional visualization approaches while creating new intuitive views into otherwise ill-defined and complex phenomena.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

People around the world are excited about the potential applications enabled with artificial intelligence (AI) and machine learning (ML). Our daily news contains headlines describing how machines can successfully predict patterns and answer difficult questions faster than humans. At the same time, recent technological advances in data collection, communication and storage have led to a data explosion. In fact, our digital universe continues to increase exponentially, and new information is becoming easier to generate [1]. Referred to as Big Data, both researchers and practitioners see opportunities to utilize these large, heterogeneous, and unstructured data sets to solve complex problems. It is no surprise, therefore, that the latest research aims to combine Big Data analysis with data-driven methods like AI and ML to automatically analyze patterns, trends, and associations in the data. This unique marriage has led to many practical successes, ushering in a new data-driven paradigm for exploring ideas [2].

Due to the Big Data's size, however, in order to feasibly take advantage of this paradigm, successful approaches use automation. It is simply not possible for humans to sift through large data sizes (i.e. gigabytes, terabytes, petabytes, etc...) in a reasonable amount of time, let alone find patterns in the data. One of the common criticisms for using these automated methods (e.g. machine learning, interpolation, and regression), is that they act like black boxes. Answers appear to be magical and are given without explanation. How can we trust a machine's answer when we do not know the rationality behind it? This has become increasingly important as algorithms become the backbone behind new technologies like self driving cars, hiring systems, and social media. Does a machine have a person's best interest in mind? For example, huskies have been classified as wolves for being in the snow [3], people have been labeled as animals based on their race [4], and robots have learned to swear and make inflammatory remarks on Twitter [5]. How do we begin to address algorithmic discrimination and bias as machines begin to play increasing roles in hiring, housing, health care, politics, finance, and education [6, 7]? More than ever, we need to go beyond merely using data, and focus on understanding it.

This problem is of importance for the scientific and engineering disciplines where correctness, precision, and understanding are key. Practical solutions or reasonable answers may be fine for some fields (e.g. animation, media, and marketing), but for science and engineering, intuition is vital. Scientists want to know why something is true, so they need to build hypotheses, not simply receive answers. Why should a scientist trust a machine that may have bias or treat noise as a key component in analysis? The sacrifice of understanding is simply not worth the benefits of using data-driven methods. These disciplines, however, also rely heavily on large amounts of simulation and experimental data to inform theory. Since there is much to be gained from using Big Data, solving the intuition problem is a top priority of our time.

Fortunately, interactive visualization is one way to bridge the gap between computation and human intelligence by providing visual metaphors, allowing humans to gain insights about complex phenomena [8]. Utilizing images built to convey information to the human visual system, we can create a communication channel between machines (used for automation) and the human brain (used for intuition). This marriage between the two systems can potentially provide the missing intuition for Big Data problems. One of the most intuitive ways to understand complex scientific phenomena is to view a spatial-temporal representation. For example, consider studying blood flow through the heart by simply viewing the flow dynamics of a computational simulation using 3D streamlines [9]. Visual analytics aims at presenting aggregated statistical data and variable relationships. These approaches often use linked views of scatter plots, parallel coordinate plots, glyphs, and graphs to view data. Interaction allows subject matter experts to apply their expertise to a problem, often influencing the computation done on the machine. With the emergence of Virtual Reality (VR) and Augmented Reality (AR), perhaps soon it will be common practice for scientists and engineers to immerse themselves in their data, directly manipulating variables of interest. Imagine painting stress on a medical device, twisting a pathline to increase vorticity in a fluid, and stretching a solid object to change displacement. These intuitive, user-defined actions allow subject matter experts to ask questions of ill-defined systems in the same way we naturally experience reality.

Surprisingly, the best way to visually navigate, identify, and understand relationships between complex features in large scientific data sets remains an open question. In other words, we are not yet able to interact with data naturally and intuitively. In order to proceed, we must investigate solutions to three major challenges. To make concepts concrete, these challenges are stated in the context of scientific exploration:

1.1 Challenge 1 - Big Data is "Big" in several ways.

Simulation has become a vital part of the scientific exploration process, often replacing expensive and sometimes infeasible experiments. Models represent physical phenomena and can produce large sets of results, providing more data than ever before. These scientific data sets, called ensembles, are collections composed of many parameterized simulation instances [10]. Ensembles are sampled from multi-dimensional parameter spaces that include any number of model variables (temperature, quantity, force, pressure, density, etc...). They are widely used for visual analysis tasks including evaluating model uncertainty [11, 12, 13, 14], discovering trends [15], and making predictions [16, 17, 18].

Ensemble visualization is an active area of research with many open questions, but here we focus on the large data visualization challenge. Scientific ensembles are large both in memory footprint and in the number of instances. The data usually exists on a shared storage system where the simulations are run (i.e. a supercomputer), and due to bandwidth limitations and extreme data sizes, it is necessary to process the data in local proximity. In other words, it is impractical to move the data because the time or storage cost would be too large. Unfortunately, this eliminates simple solutions like studying data with a laptop. It also requires expertise in interacting with supercomputer environments.

Scientific visualization tools are rapidly embracing the necessary challenge of simultaneously visualizing multiple parameterized simulation data sets [19]. In practice, however, this can be a difficult task due to the spatial and high-dimensional nature of scientific data, meaning that we need to apply both sides of this **large** data spectrum (large in size vs. large in number of instances). Accurate simulations are often **large** in size due to dynamic unstructured mesh resolutions and small time steps, making it difficult to simply render even one data set. Scientific data sets are also multi-dimensional, multi-valued, and multi-variate, which are hard to visualize by themselves [20]. Since interactive visualization technically requires high frame rates and optimized use of precious GPU memory, ensemble visualization often requires sophisticated data sampling and rendering techniques [9, 21, 19]. Visualizing instances together multiplies the problem since there is a limited number of perceptual visual encodings and instance occlusion is probable [11]. On the other side of the **large** data spectrum, we have a **large** number of instances. Because the data is large in size, we can only look at a few instances at a time, but we need to understand the relationships over many instances. This not only requires sophisticated methods for searching and interacting with an ensemble [22], but an understanding of navigating sparse parameter spaces [23]. It is important to efficiently decide which instances are important for analysis and, therefore, can be loaded for processing.

Finally, remote visualization techniques are necessary for visualizing large ensemble data that is stored on a server. There are many complications with trying to stream subsets of data effectively and efficiently to remote clients. Orchestrating the amount of communication is important for usability and bandwidth optimization.

1.2 Challenge 2 - Big Data is sparse and hard to understand.

Ensembles live in multi-dimensional parameter spaces, where each simulation instance is characterized with a unique set of parameters. In addition, the output data often lives in a high-dimensional space due to the complexity of time varying features (e.g. time-series, unstructured meshes, volumes, etc...). Unfortunately, due to the physical limitations of displays and our own visual system, it can be challenging to effectively visualize data that has greater than two or three dimensions [24]. Therefore, the problem of visualizing high-dimensional spaces with the human visual system is a fundamental problem for understanding scientific data sets. One way to visualize these complex multi-dimensional spaces is to use dimensionality reduction techniques, which project multi-dimensional data into fewer dimensions that more adequately represent the intrinsic structure of the data. Unfortunately, there is a known "trade-off between the interpretability of the axis and intrinsic structure captured by dimensionality reduction methods." [24] In other words, the visual representation of these dimensionally reduced spaces are usually hard to understand.

In practice, scientists or engineers prefer not to employ these types of approaches since they are less intuitive than viewing the two or three of the original dimensions at a time. Although intuitive to a user, this more traditional approach does not scale in a high-dimensional space. These spaces are often sparse, non-linear, and increase exponentially as the number of dimensions increase. Consider a 10 dimensional system (including substrate stiffness and motility coefficient) that models the dynamics of cell movement [25]. Each data point in the high-dimensional space represents a numerical set of model parameters or outputs (e.g. stiffness, motility, motor number, clutch number, etc...). It is possible to fix 8 of the dimensions and look at the relationships between the two dimensions of interest (e.g. stiffness vs. motility) on a 2D scatter plot. Unfortunately, this view would only show a local view of the stiffness vs. motility at a local 8 dimensional point. To see a more accurate picture of the full space for just these two dimensions, it would be necessary to view n^8 more scatter plots, where n is the number of samples in each dimension. So if we had a Cartesian sampling of $n = 5$ samples for each dimension, which is fairly small, we would need $5^8 = 390,625$ scatter plots to partially understand two dimensions with the traditional scientific approach. Therefore, we need to create new intuitive ways to incorporate the more scalable approaches that visualize high-dimensional data. The visualization solutions need to either complement or outperform the traditional approaches.

In the context of high-dimensional parameter spaces, scientists would also like to understand the gaps in knowledge without the expensive computation time required by simulation or experiment. Fortunately, prediction methods are able to estimate results using the statistical properties of the data set along with advanced statistical methods like regression and machine learning. However, high-dimensional spaces are sparse, implying that the sampling density is practically small. This suggests that predictions, which are calculated from the sparse sampling, need to be evaluated in the context of uncertainty [18]. There are many prediction methods, and they often give different answers, so understanding a prediction's correctness is important. This is especially true for inverse problems, which try to determine the input from the output. For these problems prediction actually represents a probability distribution and there is no "best" estimate, but rather multiple possible estimates [26]. Therefore, the uncertainty distribution, which informs and validates prediction, is perhaps just as important as the prediction itself. Currently, however, it is often easier or more common to visualize a local sample or prediction than compare in the context of an uncertainty distribution [27, 28, 9]. It is no surprise that uncertainty visualization is a top visualization challenge [29], especially as it relates to high-dimensional spaces [18, 23].

To make the situation more complicated, higher dimensional spaces suffer from a sampling problem called the "curse of dimensionality" [30], where the sampling density is extremely sparse compared to the continuous space. This implies that it often impossible to collect enough samples to make any statistical significant claims using the sample set. In other words, even when using a common statistical sampling method like Latin hypercube sampling [31], the number of samples needed to achieve the necessary sampling density increases exponentially as the number of dimensions increase. In these cases, simply collecting more samples will never be enough to analyze the problem globally or locally. Therefore, it is necessary to develop new sample steering methods that both explore the space while realistically answering relevant questions.

1.3 Challenge 3 - Big Data is ill-defined, unstructured, and *complex.*

One of the most difficult problems with analyzing scientific data is that output data is often represented by what Sedlmair et. al. and Munzner define as *complex objects* [32, 8]. These objects are known at the semantic level, not at the same mathematical level as multi-variate/multi-dimensional objects that can be represented as numerical arrays. Subject matter experts are able to identify and understand these ill-defined and unstructured features, but from a computational approach these features can be difficult to quantify.

Consider the application involving a cardiac lead placed in the right atrium of the heart [33]. An engineer must discover what length and stiffness of the lead adequately attaches to the heart wall while minimizing the effect on the surrounding blood flow. At the same time the engineer might also want to reduce the amount of fibrosis build-up where the heart wall and the lead meet for a specific lead stiffness range. These problems are non-linear, spatially complex, and may have more than one viable solution. The lack of a clear objective function and the existence of vague user-defined features makes it hard for computers to automatically find optimal solutions. This problem presents both a challenge and an opportunity for visualization. Since visual interaction has the potential to augment human intelligence with computational models [8, 34], we can investigate how interaction can apply meaning to the scientific data [35]. In the visualization literature, utilizing meaningful interaction (selecting, highlighting, annotating, etc...) to steer computational and visual models is called semantic interaction [36]. Another related method utilizes the direct manipulation of the visual representation to understand and navigate a parameter space [22, 23]. These semantic approaches have the potential to learn intuition and knowledge about the scientific domain directly from an expert as they use the system.

Another complication when studying trends in simulation ensembles, is that Scientists and engineers would like to ask meaningful questions like “How does parameter x affect parameters y and z?”, “What does the input look like for this output?”, or “What assumptions should be made to reduce the uncertainty in this area of our simulation input space?” There are two levels that these questions can be asked: locally or globally. Just like the uncertainty principle in quantum physics, Fourier analysis, and statistics, there is a natural trade-off in the sample granularity for ensembles. The more focused the question (local), the less that is known about the system as a whole (global). Inversely, the more samples that are taken into account in analysis (global), the less that the trend applies to a single sample (local). Traditionally, interactive visualization allows users to explore a data set using either local-to-global [22] or global-to-local [37] navigation strategies [32]. Local-to-global approaches start with an instance and navigates within the overall context, while global-to-local approaches follow Shneiderman’s mantra “Overview first, zoom and filter, then details on demand” [38].

As described in Section 1.2, local approaches do not scale, but they are often easier to understand. Asking a question at the local level makes sense intuitively, but what does it look like to ask ill-defined questions at the global level? For example, in the case of studying blood flow through the heart, how do we compare flow patterns with fibrosis build-up across an entire ensemble? Analyzing global trends in data is an active research area [39, 15, 12, 13], but more work needs to be done to understand global system dynamics in the scientific domain. Are we able to use concepts like semantic interaction [40, 41] or meaningful direct manipulation [22] to add understanding to complex high-dimensional spaces?

One final problem with asking these meaningful questions through interactive techniques, is that results can be hard to reproduce. Human intuition generally does not give absolute numerical values, but rather approximate answers based on expert knowledge. Users may ask the same question in a slightly different way or order and produce entirely different results. Although this may help discover trends in the data or develop a rough hypothesis [23], it may not give consistent answers necessary for reproducing a scientific discovery. It is important to discover methods to make this type of interaction both meaningful to a scientist and reproducible.

1.4 Our Approach: Data-Driven Exploratory Interfaces

To overcome these obstacles we introduce Data-Driven Exploratory Interfaces (DDEIs), user interfaces that analyze data in the context of exploration to drive understanding. As their name suggests, they combine the properties of data-driven analysis and user exploration of parameter spaces. To be more specific, DDEIs are defined by three characteristics related to the three challenges above:

- **Scalability** - A DDEI must be able to scale to meet the needs of the application data. This is the first way that data drives exploration. In order to explore the data interactively within the dataset, the interface needs to be able to visualize multiple data instances simultaneously. This enables users to view variablity within the parameter space through local distributions. In addition, the interface must intuitively scale to meet the demands of the application's dimensionality. In other words, DDEIs need to work with large datasets.

- **Contextual Navigation** - In order to explore relationships within a multi-dimensional parameter space, it is important to view data instances in the context of an ensemble or subspaces of an ensemble. The context is derived directly from the data and users visually navigate within, viewing global relationships throughout the space. When there are gaps in the data, it is important to use the data to predict gaps or drive the sampling process within the continous space. Questions should be asked within local or global uncertainty contexts to see variations within the data.

- **Meaninful Feature Interaction** - DDEIs must enable users to ask multiple ill-defined or complex questions about the features defined by the application. For example, this may mean defining multiple regions of interest within the output space or directly manipulating input and output curves. In order to be data-driven, these questions need to be constrained to the dataset. In other words, DDEIs force questions to be relevant to both the user and the sampled data. It does not make sense to ask questions about useless or invalid parts of a parameter space.

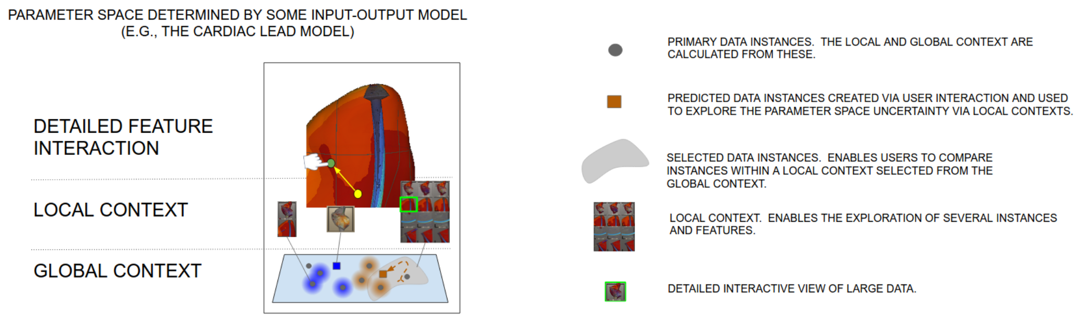

PARAMETER SPACE DETERMINED BY SOME INPUT-OUTPUT MODEL (E.G., THE CARDIAC LEAD MODEL)
DETAILED FEATURE INTERACTION
LOCAL CONTEXT
GLOBAL CONTEXT
PRIMARY DATA INSTANCES. THE LOCAL AND GLOBAL CONTEXT ARE CALCULATED FROM THESE.
PREDICTED DATA INSTANCES CREATED VIA USER INTERACTION AND USED TO EXPLORE THE PARAMETER SPACE UNCERTAINTY VIA LOCAL CONTEXTS.
SELECTED DATA INSTANCES. ENABLES USERS TO COMPARE INSTANCES WITHIN A LOCAL CONTEXT SELECTED FROM THE GLOBAL CONTEXT.
LOCAL CONTEXT. ENABLES THE EXPLORATION OF SEVERAL INSTANCES AND FEATURES.
DETAILED INTERACTIVE VIEW OF LARGE DATA.

Figure 1.1: An overview of Data-Driven Exploratory Interfaces illustrated using a large cardiac lead simulation model. The interface illustrates the three criteria of a DDEI. It is scalable because the interface visualizes many instances of the data set for comparison. It is possible to navigate in both local and global contexts and predict the gaps. Finally, meaningful interaction is illustrated with defining regions of interest and direct manipulation.

Figure 1.1 illustrates an example DDEI concretely using a cardiac lead application. The interface is **scalable** since it can show multiple hearts at the same time to compare the variation. It also enables **contextual navigation** at both the global and local contexts, while predicting the gaps validated by an uncertainty distribution. Finally, users can employ **meaningful feature interaction** by defining regions of interest and directly manipulating stress on the heart wall. These three criteria enable users to ask multiple custom "What if?" questions, contextualizing the space based on expert intuition. In this dissertation, **"What if?" questions** are questions that investigate how data instances change when parameters or features change.

In the next two subsections we explain the theory behind data-driven exploratory interfaces by describing how contextualization capabilities lead to understanding and how interaction adds intuition to Big Data.

1.4.1 The Importance of Contextualization for Understanding

In linguistics, there are three major ways to study a statement:

- **Syntax** refers to structure. To be syntatically correct, the statement needs to follow the structural rules of the language, but does not need to make sense.
- **Semantics** refers to a statement's meaning. Does the statement make sense?
- **Pragmatics** studies how the context contributes to meaning.

Statements can have a valid structure (syntax), but have no logical meaning (semantics). For example, "This sentence is false" follows the structural rules of English, but contradicts itself. Similarily, meaningful sentences (semantics) may have seperate interpretations in different contexts (pragmatics). Does $5 + 4 = 3$? The answer is yes, assuming we are using modular arithmatic with a base of 6.

The current visualization literarture makes a distinction between interaction done at a syntatic level and interaction done at the semantic level [40]. Syntatic interactions are those that allow users to directly modify the parameters of complex mathematical models. For example, imagine a control panel with sliders that allow the users to fine tune individual eigenvalues and eigenvectors for a principal component analysis visualization [42]. Although useful, this technique requires expert theoretical knowledge about dimensionallity reduction, something that users may be familiar with, but unfortunately adds cognitive overhead irrelevant to the problem. Semantic interaction, on the other hand, allows experts to apply meaningful interactions to modify the underlying computational models [36]. For example, a scatter plot application allows users to move points closer to each other since they intuitively should be similar. Based on the interaction, the system can discover a distance function and project the points in a new scatter plot that captures the user's expertise or intuition [41, 43]. The user does not need to understand the underlying dimensionality reduction algorithm since the interaction technique translates meaning into the computational parameterization needed. In this dissertation we define semantic interaction as user interaction that is meaningful in the application domain, and has the potential to steer the complicated underlying computational and visual algorithms by inferring analytical reasoning from the user [36, 44].

Unfortunately, semantic interaction itself acts like a black box. By definition it shields the users from the underlying complexity and mathematical parameters of a computational model [36]. The visual feedback may be interesting, but a scientist who wants to validate a hypothesis needs to ask "Does the result of the semantic interaction make sense?" or "How confident can I be in the result?" Consider the above example of moving points closer to each other. Assume each point was a multi-dimensional vector representing an animal and its properties. In one case, it makes sense to move a dolphin close to a fish because they are both water animals, however, in another case it may make sense to move the dolphin closer to a bear since they are both mammals. In these local contexts, the generated distance function may be completely different, and therefore, separate projections are produced. The results ask two different questions of the data. Pragmatics asks the question, "What can we learn from these two separate contexts about animals and their properties?"

Pragmatics is interested in studying how contextual clues provided by user interaction can provide meaningful information to both a user and the underlying model. A pragmatic approach looks at data from from a local perspective and a global perspective. We can ask, "Are we able to infer meaning from the interaction done within different contexts?" (local-to-global). Alternatively we can ask, "How does the meaning of an interaction change with respect to other contexts?" (global-to-local). The local-to-global approach consists of asking many different questions and discovering trends in the overall space. Inversely, the global-to-local approach involves asking the same question under different contexts to also discover trends in the overall space. From the local perspective, we can ask relevant questions, and from the global perspective, we can understand which questions are worth asking.

1.4.2 Adding Intution to Big Data

Our interaction work is motivated by the modern approach for discovering information epitomized by the popular phrase, "Google it." Users simply type their question into a search engine as a text query and results ordered by relevancy are returned. The experience is simple, organic, and does not require an in-depth knowledge of the intrinsic structure required by most database engines. Imagine a visual search engine that allows users to find data information by simply showing the search engine what they are looking for. In the scientific domain, these exploratory interfaces may involve adjusting parameter sliders [45, 46], editing a CAD model [47, 22], or dragging stress on an object [22]. As the user interacts, the underlying search algorithms return results relevant results to the user. This is the traditional local approach that most exploration tools employ. The interfaces allow for one query and return multiple results (visualization tools, however, often return only one result).

Data-driven exploratory interfaces go beyond the traditional approaches by enabling multiple meaningful user queries in context. These provide a data-driven view of information tied to user intent. In the search engine analogy, this would be like a user asking multiple semi-related questions at the same time. The search algorithms would aggregate the results based on multiple contexts to discover trends, matching the user's intent based on the data. Because navigation is done in the context of the data, users would also know how the questions fit together into the sampled space. Here we can look for alternative solutions. For example, searches engines commonly employ is the fact that queries can be inaccurate, correcting spelling errors or suggesting similar queries (i.e. "Did you mean to search for ...?"). In summary DDEIs enable users to ask multiple custom "What if?" questions to contextually analyze a complex phenomena in an application domain.

1.5 Dissertation

This dissertation posits the following **thesis statement**:

Data-driven exploratory interfaces improve the understanding of input $\longleftrightarrow$ *output relationships, including spatial relationships, across large parameter spaces by providing scalability, contextual navigation, and meaningful feature interaction.*

1.5.1 Research Questions

Realizing the potential of DDEIs requires studying three related visualization research questions:

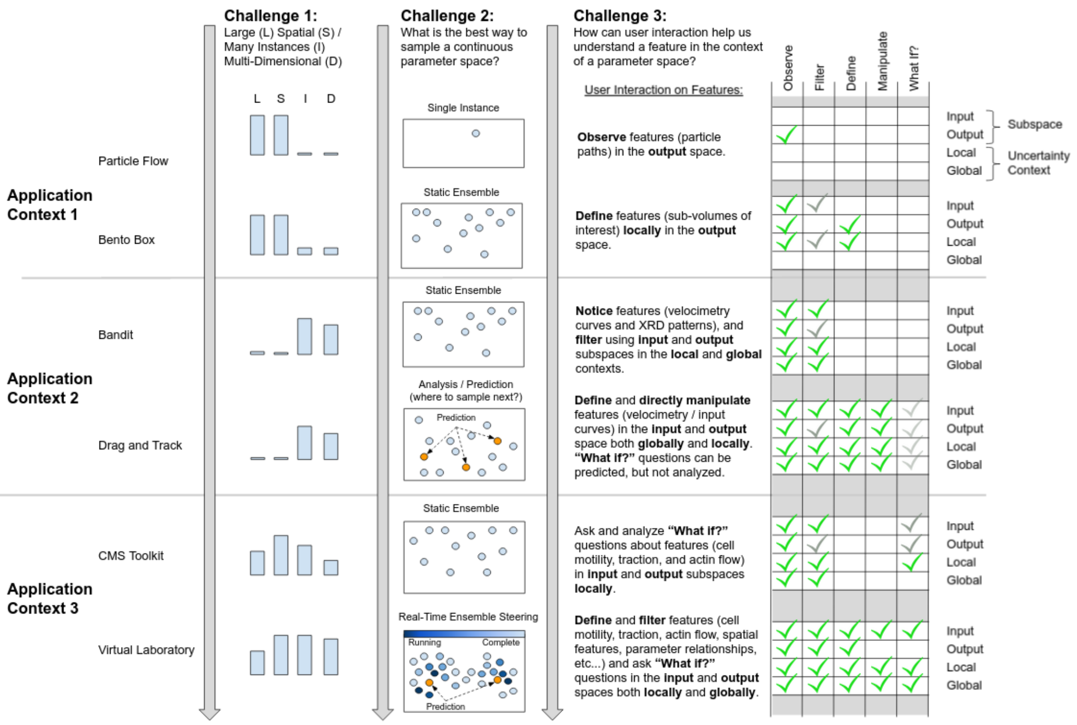
- **Q1 - How can we efficiently visualize large ensembles?** (Challenge 1)
- **Q2 - How can we effectively visualize and understand high-dimensional spaces?** (Challenge 2)
- **Q3 - How can we use expert interaction to explore parameter and feature relationships in an ensemble?** (Challenge 3)

All of these questions need to be asked in the context of several relevant criteria that describe the type of ensemble or visualization. These criteria are ranged variables that represent the difficulty of visualizing or understanding an ensemble. For example, a two dimensional parameter space is trivial to understand compared to a ten dimensional parameter space. We ask the research questions in the context of the following criteria:

- **C1 - Memory Footprint** - Small vs. Large
- **C2 - Output Type** - Multi-variate/Multi-dimensional vs. Complex Object (e.g. Time Series vs. Spatial-Temporal)
- **C3 - Number of Instances** - Low vs. High
- **C4 - Dimensionality** - Low vs. High
- **C5 - Analysis Level** - Local vs. Global

Asking the research questions in context requires sampling of the criteria space, however, the closer we can sample towards the difficult side of the variable ranges, the more confident we can be in the general applicability of our conclusions.

1.5.2 Parallel Challenges



Large (L) Spatial (S) / Many Instances (I) Multi-Dimensional (D)

What is the best way to sample a continuous parameter space?

How can user interaction help us understand a feature in the context of a parameter space?

Observe features (particle paths) in the **output** space.

Define features (sub-volumes of interest) **locally** in the **output** space.

Notice features (velocimetry curves and XRD patterns), and **filter** using **input** and **output** subspaces in the **local** and **global** contexts.

Define and **directly manipulate** features (velocimetry / input curves) in the **input** and **output** space both **globally** and **locally**. **"What if?"** questions can be predicted, but not analyzed.

Ask and analyze **"What if?"** questions about features (cell motility, traction, and actin flow) in **input** and **output** subspaces **locally**.

Define and **filter** features (cell motility, traction, actin flow, spatial features, parameter relationships, etc...) and ask **"What if?"** questions in the **input** and **output** spaces both **locally** and **globally**.

Figure 1.2: Three parallel challenges outlining this dissertation. Addressing three challenges are our approach for investigating our three research questions.

The three research questions Q1, Q2, and Q3, motivated by the three visualization challenges, can be studied in the light of three parallel challenges pictured in Figure 1.2. **Challenge 1 (Q1)** investigates how to efficiently visualize ensembles that have large memory footprints (C1), spatial outputs (C2), a large number of instances (C3), and a large number of dimensions (C4). Challenge 1 acts as the domain for our thesis with three application contexts described in Section 1.5.4. Parallel to Challenge 1 is **Challenge 2 (Q2)**, which studies how sampling the space can lead to effective understanding of higher dimensional spaces. We start with a small number of static instances, enable prediction within a static ensemble, and finally use prediction to steer the sampling of a dynamic ensemble. **Challenge 3 (Q3)**, pictured in the third column of Figure 1.2, simultaneously investigates how user interaction can add intuition to a parameter space in local and global uncertainty contexts (C5). We start simple by observing features (defined in the application domain), add user interaction techniques (e.g. filtering, defining features, directly manipulating features, and asking "What if?" questions), and discover how the interaction techniques contribute to understanding in the local and global domains.

1.5.3 Contributions

The key contributions of this dissertation are:

- Data sampling and rendering techniques for interacting with and comparing the user-defined features from multiple large simulations.
- Visual search approaches that use the direct manipulation of user-defined features to explore a parameter space in the context of uncertainty.
- Mechanisms for specifying user-defined annotations of parameter spaces using the results from multiple searches and continuous prediction across multiple linked views.
- Methods for systematically defining and applying multiple user-defined interactions that generate and reproduce an understanding of the global and local dynamics in an ensemble data set.
- An evaluation of these approaches using three separate real-world scientific applications in collaboration with respective expert mechanical engineers, material scientists, and biomedical engineers.

1.5.4 Structure of this Dissertation

This dissertation first describes the related work and background (Chapter 2), followed by several chapters describing three separate Application Contexts. Each chapter describes motivation and our techniques used to address the research questions. At the end of each chapter we discuss results and how they relate to our thesis. We report on each application context using two chapters. The first chapter in each application context studies our thesis using more traditional visualization approaches. Each application contexts's second chapter, on the other hand, reports on new visualization approaches that are needed to adequately investigate the thesis. Specifically these second chapters describe data-driven exploratory interfaces and their contributions.

Application Context 1 - Investigating ensembles whose instances are both memory intensive and spatial.

In Application Context 1, we focus on how to efficiently (Q1) visualize multiple memory intensive simulations (C1) in order to analyze local spatial (C2) dynamics based on user-defined feature locations (Q3). It is quite common for scientific simulations to be large and unstructured, so it is necessary to enable the ability to interact with these types of data sets. We start in Chapter 3 describing some of the complexities involved when optimizing memory and rendering on the CPU and GPU. Here we share our application, **Particle Flow**, with results and lessons learned from visualizing fluid flow simulations for a hurricane and blood flow in the right atrium of a heart. In Chapter 4 we scale up, simultaneously visualizing multiple fluid structure interaction simulations that work towards understanding a cardiac lead design application [33]. We describe our approach, **Bento Box** [9], an application that uses data sampling and rendering techniques [28] optimized for real-time user-defined spatial-temporal comparison across simulations.

Application Context 2 - Investigating high-dimensional ensembles that have many instances.

In Application Context 2, we investigate the thesis by focusing on the high-dimensional problem (Q2, C4) using ensembles that have many instances (Q1, C3). We also explore how to apply expert knowledge to high-dimensional parameter spaces using user interaction (Q3) both on a local and semi-global analysis level (C5). In Chapter 5, we motivate the high-dimensional problem using a shock physics application named **Cinema:Bandit**, which analyzes experimental results at the beamline in light of a high-dimensional parameter space. We conclude our investigation into context 2 with

Chapter 6 by discussing the **Drag and Track** [23], an application that allows users to intuitively contextualize instances in the context of the ensemble and uncertainty.

Application Context 3 - Investigating the dynamics of high-dimensional ensembles that have many spatial instances.
In Application Context 3, we consider the unique problems that arise from ensembles that are both spatial (C2) and high-dimensional (C4). This phase investigates the thesis (Q1, Q2, Q3) by adding the missing pieces necessary to study an ensemble that has many simulations (C3) where each simulation has a moderately sized memory footprint (C1). Finally, we investigate how to steer a simulation ensemble based on reproducible user-defined interaction (Q3), allowing a user to understand the user-defined local and global dynamics of the data (C5). Chapter 7 describes our foundational toolkit for studying cancer cell migration [48], laying a groundwork for our final investigation in Chapter 8.

Chapter 2

Background

Before discussing related work, it is important to build a firm technical foundation for parameter space analysis by defining relevant terms. From this point forward we will be discussing Big Data in the context of scientific data sets that are composed of many simulation or experimental results. These are called ensembles. An **ensemble** is a set of parameterized data instances, where each **data instance** is a simulation result or experiment characterized by a unique set of parameters [10]. **Parameters** are often inputs into a simulation, but can be any categorical or quantitative value that describe a data instance. For example, the parameters of a cell migration simulator [48] would include substrate stiffness, maximum number of motors, maximum number of clutches, and maximum polymerization. Similarly parameters for a cardiac lead simulator [49] might include a set of cardiac lead thickness and stiffness parameters along with a tissue attachment position.

It is important to understand how an ensemble relates to a **scientific model**, which is a physical, mathematical, and/or conceptual representation of a system of ideas. Scientific models seek to understand physical phenomena by drawing on existing knowledge to predict patterns. Examples include the theory of relativity, quantum physics, and equations for material elasticity. An ensemble samples an **input-output model**, a model that starts with a set of parameters as inputs and arrives at an output or result [32]. This type of model acts like a function in mathematics. The data-driven methods described in Chapter 1, often use another type of model called a statistical model. **Statistical models** (e.g. regression, interpolation, or machine learning) use the statistical properties of sampled data (i.e. an ensemble or an ensemble subset) to predict models. When a statistical model is used to predict the output of an input-output model, it is called a **surrogate model** [32] because it is often used as a quick and efficient approximation to the original model, which can often take days to compute. By this definition, machine learning can be thought of as a surrogate model for an unknown model that can be approximated from the data.

The continuous multi-dimensional set of parameter possibilities for an input-output model is called a **parameter space**. Since the space of possibilities is extremely large, an ensemble can only sample a parameter space. Understanding the dynamics of a parameter space, can provide insight into important relationships in physical systems. For example, if we knew that changing one or two parameters in a cancer cell migration model would slow the speed of cancer, we could investigate drugs that could provide similar functionality and potentially save lives. The study of the relationships between parameter settings and corresponding outputs is called **Parameter space analysis** [32].

2.1 Related Work

Our work combines techniques from two major visualization research areas: scientific visualization and visual analytics. Scientific visualization uses computer graphics to provide key insights into volumetric data sets to understand spatial temporal relationships (e.g. MRI, weather, blood flow, etc...). Visual analytics uses visual interfaces and user interaction to combine computational methods and human reasoning in order to gain analytical insights about the data. Specifically, our work is derived from the sub-areas of ensemble visualization, visual parameter space analysis, and direct manipulation interfaces.

2.1.1 Ensemble Visualization

Ensemble visualization is a broad class of research that focuses on understanding variation across multiple related datasets (i.e. an ensemble), which are usually characterized as being large, temporal, multidimensional and multivariate. Because of the statistical properties of ensembles, visualization approaches often work towards mitigating uncertainty and error in simulation results that arise from misunderstanding real-world phenomena [11, 16, 18, 50]. Other approaches use the ensemble to derive and visualize uncertainty distributions [12, 51]. Dimensionality reduction (DR) techniques or interactive widgets (i.e. graphs or parallel coordinate plots) enable users to view the structure of high-dimensional spaces, allowing feature based views of their data [24, 42, 52, 53]. Cluster analysis partitions the ensembles into subsets with localized statistical properties, providing the ability to compare trends and understand local variances [12, 13, 54, 37].

Scientific Comparative Visualization

Ensemble visualization is considered a top problem in scientific visualization [29]. Visualizing multiple large spatial-temporal simulations can be difficult due the size of the data and the visual complexity as described in Chapter 1. In this context, ensemble visualization often morphs into comparative visualization, analyzing the differences between multiple simulations across space and time. Kim et al [27] provide a taxonomy and a survey of comparative visualization techniques in spatial 3D and 4D data, documenting four fundamental approaches for comparing multiple data sets: juxtaposition, superposition, interchangeable, and explicit encoding. Coffey et al [55], studies the use of interaction, animation, and static views across space and time, and it provides a taxonomy for evaluation. Vohl et al [19] have effectively used juxtaposition for small miniatures across multiple collaborative environments including desktop, mobile, and VR displays for large scientific data sets. Using similarity metrics applied to an ensemble, Schroeder et al [39] analyze the trends in scientific motion along with superimposed volumetric and 2D visualizations that show local uncertainty distributions. Instead of comparing multiple ensemble data instances, this approach computes trends in the data set, and compares how data instances leave and join trends.

Our approaches use juxtaposition and superposition to understand scientific data sets that are spatial (e.g. blood flow) and/or mathematical (e.g. time series data). In the context of an ensemble, we find local uncertainty distributions based on similarity that can be compared with these techniques. The locality of the uncertainty distribution allows us to efficiently and effectively render only relevant data sets, therefore optimizing GPU memory and bandwidth. With this assumption we can assume that we only need to visualize a subset of the ensemble data at the local level.

Dimensionality Reduction in Visual Analytics

Dimensionality reduction plays a significant role when it comes to visualizing an ensemble's context. Because it is hard for the human visual system to comprehend visuals beyond two or three dimensions, DR allows humans to visualize intrinsic structure of an ensemble by transforming a high number of dimensions into a low number of dimensions [24]. Ensembles can then be represented by two or three dimensions (e.g. a scatter plot) in a spatialization, where relative proximity (i.e. nearness) implies that the data instances are similar [40]. In addition, clustering in these lower dimensional spaces helps provide insight into ensemble uncertainty [12, 13]. For example, Steiger et al [56] visualize clusters of time-series data on a DR view and display the mean at each cluster via a visual callout. Ferstl et al [12] use dimensionality reduction to characterize vector field uncertainty using streamline variability plots that show the global dynamics of an ensemble. Similarily, Hummel et al [13] reduce high dimensional data into a classification space in order to characterize uncertainty in flow fields using pathline integration.

For a detailed survey on dimensionality reduction methods as it applies to high-dimensional data, we refer the reader to Van der Maaten et al [57] and Liu et al [24]. Examples of DR include principal component analysis (PCA), which finds projections whose components are ordered to maximize the variation, and multidimensional scaling (MDS), which uses the distances of individual data instances to show the level of similarity. In our work, we use DR projections to visualize ensembles in linked reduced similarity spaces, and then we combine them with visual interactive querying techniques (e.g. interactive scatter plots, spatial representations, and parallel coordinate plots) to search for localized uncertainty distributions. This enables us to evaluate predictive models in the context of the ensemble and uncertainty. Globally, we take a similar approach to streamline variability plots from Ferstl et al [12], only at a more abstract level to include non-spatial data sets. We cluster the dynamics of an an ensemble to understand the intrinsic structure of a model's variability, similar to finding global trends [39, 13, 15].

2.1.2 Visual Parameter Space Analysis

Visual parameter space analysis (VPSA) involves visually analyzing, optimizing, and estimating parameter relationships in an ensemble data set. Sedlmair et al [32] provide an in depth literature review of visual parameter space analysis tools with a conceptual framework that can be used to evaluate new and existing approaches. The goal of VPSA is to visually understand the relationships between the parameters of input-output models and the resulting output. This usually involves sampling the parameter space to create a parameterized ensemble, which can then be used to approximate the scientific model using fast surrogate models. This allows users to analyze continuous relationships that may help in understanding complex models, especially when sampling the parameter space may be costly.

The conceptual VPSA framework describes the navigational strategies *local-to-global* and *global-to-local*. Local-to-global allows users to search a space one instance at a time [22, 46], while global-to-local starts with an overview of the instances and allows users to drill down to instances of interest [37]. In our approach we use both strategies simultaneously. Multiple uncertainty distributions with prediction (local-to-global) are viewed in the context of the ensemble (global-to-local).

In the area of VPSA, our work most closely resembles Design by Dragging [22], a direct manipulation interface that allows users to interactively change both input parameters and output data in order to navigate a set of CAD designs. Although we employ a similar mechanism for exploration, Design by Dragging only allows for navigating between two similar instances, only showing one data instance at a time. We visualize multiple data instances and allow for prediction models that estimate the gaps between multiple instances.

As part of Sedlmair et al's framework, fast *surrogate models* are often used to predict results based on interpolation, regression, or machine learning. This type of estimation enables uncertainty analysis, optimization, and parameter fitting [32]. Our work is similar to Berger et al's [18], which uses regression and nearest neighbor prediction approaches to use a sampled parameter space as a continuous multidimensional function. Our approach, however, also allows for multiple estimations in the context of the ensemble for visual comparison.

Special care needs to be taken when using estimation methods since surrogate models may be less accurate for sparse parameter spaces [18]. Tuner [17] optimizes image segmentation algorithms by comparing Gaussian process estimates with ground-truth images. A Gaussian process provides uncertainty information with estimates, enabling the ability to determine areas of interest. For example, it may be advantageous to add more samples in an area which has a high uncertainty and desired estimates to optimize the expected gain [17]. HyperMoVal [16] also uses estimation with support vector regression. It compares the surrogate models themselves to find regions of best fit and validate their physical plausibility. Our estimates are qualified in the context of local distributions providing parameter uncertainty information based on similarity.

We also want to explore a space based on both input and output parameters. One of the most common approaches is to project the output space into 2D images [46, 58, 37, 59]. These images can be used for feature extraction [60], clustering [37], optimization [59, 22], evaluation [17], and navigation [46, 22, 61]. Other output representations focus on computing correspondences between instances in an ensemble. Yumer et al [62] present an approach that ties computed deformation handles across similar shapes to crowd sourced parameters, making it possible to modify geometry based on interpolated semantic properties. We provide 2D views of the input and output parameters, while maintaining a tightly coupled relationship to the underlying data for inverse queries.

Two additional VSPA research areas relevant to our study is Simulation Steering and Parameter Space Dynamics. These areas are important because they allow us to ask relevant and statistically significant questions about the global parameter space.

Simulation Steering

Sedlmair et al [32] define *simulation steering* as a method for adding instances to an ensemble in real-time by running more simulations based on user interaction. For example, World Lines [63] allows users to ask "What if?" questions by allowing them to change the parameters of a data instance, creating an alternative branch that can be compared. In these new branches, a new simulation is run in real-time and reported to the user when complete for analysis. In this system, users can create flood barriers in a city as the water is rising, and determine optimal locations. Sedmair et al also make a distinction between *simulation steering* and *computational steering*, which focuses on the methods by changing computational parameters like grid resolution or time step size. To complete our work, we focus on *simulation steering* using an ensemble of data instances. In other words, compared to World Lines, we are interested in creating many alternative branches based on user interaction.

Integrated sampling [63] allows users to sample the space within a visualization tool. An example of such a system is Design Galleries [59], a tool that allows users to see the output from a number of possible parameter configurations. The user can then refine, or drill down, to optimize the desired result by choosing the image that best fits the user's criteria. As the user refines their search, Design Galleries continues to show multiple alternatives using re-sampling the parameter space. We also use *integrated sampling*, running many simulation alternatives when a user changes the desired criteria. This allows us to present several possibilities that are valid within an experiment of interest, then like Design Galleries, we allow users to create new experiments in order to investigate or "drill down" into the alternatives.

Parameter Space Dynamics

We define *Parameter Space Dynamics* as observing how parameters and output data change throughout the parameter space with respect to a question of interest, also defined as a domain of interest. In many scientific disciplines, scientists are interested in the dynamics of a system, informally referred to as "What if?" questions. For example, in the context of cellular migration, we might ask, "What happens to the velocity of a cancer cell and it's force as we increase parameters A and B?" In this case, knowing how the cell changes can help doctors know which drugs to apply depending on the state of the patient's cell type to slow the spread of cancer. This is analogous to investigating the changes (i.e. derivative) of a function over time, but in our case the independent variable is a set of parameters or output features defined by the user.

In terms of visual parameter analysis and statistics, understanding parameter space dynamics is most related to sensitivity analysis [64], which involves understanding how the output of a simulation is affected by changes in the parameter space. Local sensitivity involves making small changes at one location in the parameter space and observing the uncertainty in the output. Berger et al [18] show the movement in the output for the most important parameters (this is called star sampling since it resembles a star pattern) by focusing at one point a time and systematically changing one parameter at a time. Global sensitivity on the other hand, focuses on any source in the model input that causes uncertainty in the output [65]. For example, perhaps increasing parameter A in general increases a cell's ability to move regardless of where we are sampling in the space. The distinction of global sensitivity exists, since most analysis methods in practice are either local or one parameter at a time [65]. In fact, of the papers surveyed in Sedmair et al [32], the visual parameter space analysis tools only investigated local sensitivity, and none of them considered global sensitivity. Our approach uses global methods, meaning it looks at changes throughout the entire space of interest, however, we make a distinction between dynamics and sensitivity. Dynamics compared to sensitivity looks at changes in both the input and output domain. It also adds the context of a user defined question of interest. In other words, parameter space dynamics is global in the analysis of a local question.

We use a Lagrangian approach for understanding the dynamics of a system. Eularian methods focus on understanding changes at each point locally in a space, while Lagrangian methods follow the changes throughout the space. An example of using a Lagrangian method would be to follow the path of a particle as it moves through a fluid field, monitoring changes in the attributes of the particle (e.g. velocity and density). In the visualization community these approaches are used often for characterizing 2D or 3D curves. Curve Boxplots [14] extends the traditional idea of box plots to curves, showing the uncertainty of particle path lines. Hummel et al [13] combines a Principal Component Analysis (PCA) of pathlines for a fluid flow simulation to create a lower dimensional classification space. Brushing in the classification space allows users to color and characterize similar pathlines in a fluid flow visualization. Simalarly Ferstl et al [12] combine PCA and clustering (e.g. k-means) in the PCA space to characterize several variability plots, showing the uncertainty in pathline similarity.

Like Hummel et al and Ferstl et al, we investigate using dimensionality reduction and clustering to characterize the higher dimensional equivalent of pathlines. These pathlines move through a user-defined time domain that is based on changes in parameters. For example, instead of time, an independent variable might involve increasing the number of motors while increasing the number of clutches. Data instances sampled throughout the parameter space, much like seeding particles in 2D or 3D, will travel along the parameterized path defined by the user. Patterns in these higher-dimensional pathlines are clustered by similarity and then can show the global dynamics within the parameter space.

2.1.3 Direct Manipulation Interfaces

This dissertation is inspired by direct manipulation interfaces, which allow users to interact with geometry [22] or visual data representations [61, 66] to navigate or affect a system. Accessible Animation [67] enables users to explore and modify animation keyframes by "tugging" directly on the animation. Similarly, through Video Browsing [68] it is possible to inversely change frames on a video by clicking and moving objects around on the screen. Design by Dragging [22] moves these techniques into the scientific domain. Users can either change geometrical dimensions (forward design) or drag stress calculated from a finite element method (FEM) around on a mesh (inverse design). These approaches fit well into our natural experience and also enable exciting ways to understand parameters that are related to interesting phenomena. We apply similar forward and inverse direct manipulation approaches, but extend them to virtual data instance interactions.

The visual analytics community continues to invest in direct manipulation and inverse techniques. Sacha et al [69] provide a detailed survey of interaction techniques applied to DR. The authors list seven guiding scenarios for DR interaction. Of these, data manipulation, which allows users to directly manipulate data points similar to interaction in our system, was rarely used. A recent paper by Cavallo and Demiralp [66], however, shows the utility of such interaction, especially as how it relates to understanding DR. They provide an impressive framework that explores interacting with forward and backward projection mappings for both PCA and audoencoders. Users can move data points around to see how parameters change and also directly manipulate output images to see how the point moves in the DR space. With their proline extension, users can see how parameters are related to the underlying projection. Our forward and inverse interaction techniques are similar when exploring a DR space, as both approaches use out-of-sample extensions [57], which allow new data points to be projected into an existing DR. However, we are also interested in understanding the relationship between multiple spaces, an important feature when understanding model sensitivity. We add the extension of being able to manipulate multiple data points that are not part of the original ensemble, allowing the user to explore multiple parameter relationships at the same time. This enables user-defined comparisons between areas of interest across DR views, and the development of custom illustrations that represent the parameter space as a whole.

Our approach is similar to what the visual analytics community calls semantic interaction [44, 40, 70], which focuses on using human cognition to steer the underlying computations through the direct manipulation of spatializations. Many visual analytic tools take advantage of spatial relationships or relative proximity to represent similarity. For example, Dust and Magnets [71] adds the metaphor of using anchor points as magnets that attract data points. Similarly, ForeSPRIE [36, 70] allows users to drag documents around a force directed screen to enable user intuition based clustering. Visualization by Demonstration [72] suggests specific visualizations based on a user's interaction like positioning data points or changing their size or color. Contrasted with syntactic interaction, which requires that the user know about the complex algorithms [40], semantic interaction tries to change the distance function weighting [41] or DR method [43] to optimize meaning. Most semantic interaction tools change the underlying dimensionality reduction mechanisms to improve understanding. We focus specifically on intuitively understanding the DR space and how DR spaces are related to each other without modifying the spaces. Similar to semantic interaction, however, we aim to understand the parameter space from the direct manipulation of high-dimensional data in a natural and intuitive way.

Chapter 3

Application Context 1:

Particle Flow - Interacting with Fluid Flow using Particle Glyphs

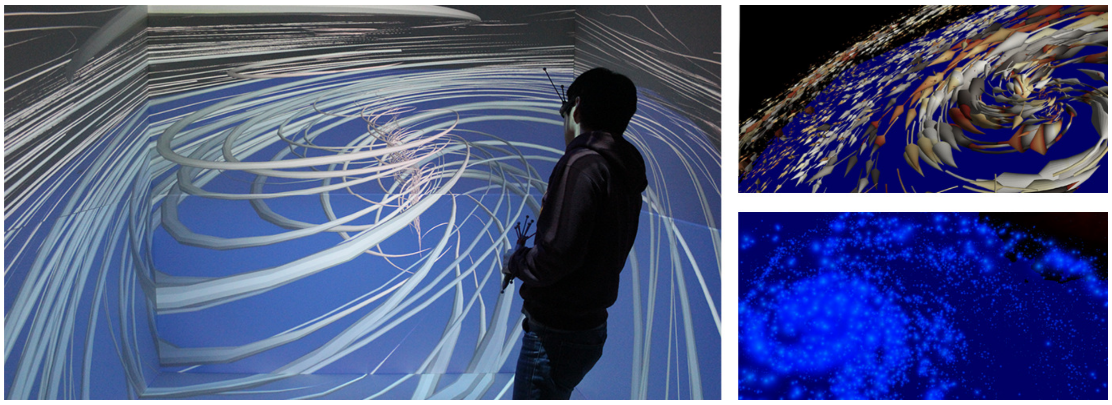


Figure 3.1: Left: Streamlines computed in real-time are displayed in a Cave VR environment. Right: Two closeup views. Particles that advect through time-varying vector fields can also be calculated and displayed in real-time using GPU hardware.

At the heart of our study, we would like to understand scientific data through user interaction. In this chapter we start with a simple, but difficult problem to solve: How do we efficiently visualize large scientific data sets (Q1) while enabling user interaction

(Q3)? Scientific visualization often focuses on real-time rendering and memory management to view data spatially [21, 19]. There are also several examples of systems built specifically for interacting with data [22, 63, 73]. In this dissertation, we start with the assumption that intuition can be gained from natural manipulation of stereoscopic views of 4D spatial-temporal data [27]. Therefore, in order to enable interaction for scientific data sets, we must we consider allowing immersive exploration of these data sets using virtual reality (VR) or augmented reality (AR). VR and AR systems, however, often use sophisticated hardware configurations and multiple GPU / CPU configurations with limited processing and memory, which complicates an already difficult problem. We start our study with an investigation into how to best visualize interactive fluid-flow in virtual reality environments.

We are interested in asking two major questions. First, can we take advantage of the GPU hardware to improve performance? Second, can we achieve real-time fluid-flow interaction at the rates necessary for immersive visualization?

3.1 Motivation and Application Background

Visualizing complex fluid flow in virtual reality (VR) [74] is a necessary part of state-of-the-art climate study, physical system design, and bio-mechanical engineering. However, the most effective way to support real-time advection and rendering of hundreds of thousands of multi-dimensional particles on multiple high-framerate stereo displays remains an open challenge [73] [75] [76]. We present a particle advection and rendering system, Particle Flow, for addressing this challenge in the context of complex graphics hardware configurations. Specifically, we take advantage of the ability to drive VR projection environments from a single computer with multiple graphics cards. We explore the potential of using hardware configuration that can include both traditional graphics processing units (GPUs) and general-purpose GPUs, for scientific visualization. We report upon the feasibility of this approach for use in exploring large-scale computational simulations of both a hurricane and blood flow through the heart.

3.2 Visualization Techniques and Implementation

Particle Flow uses a GPU-based implementation of 4th order Runga-Kutta integration to update the space-time state of each particle as it moves through vector fields of interest to scientists, typically pre-computed with offline high-resolution simulations. Hundreds of thousands of animated particles are then displayed using GPU shader programs that support interactive head-tracked stereoscopic rendering. This section describes our techniques used to enable the immersive experience.

3.2.1 Virtual Reality Considerations

In order to enable multiple GPU configurations we utilize two VR environments, each with different hardware constraints. One is a four wall cave and the other is a projected table top environment. We use an open source virtual reality software toolkit, MinVR [77], allowing us to configure an application to run across multiple threads and several OpenGL contexts, without changes to the underlying application. This allows us to completely control the context memory sent to specific GPUs on the multi-GPU system which renders to the stereoscopic displays. In addition, we consider the use of general purpose GPUs for optimizing computational performance.

3.2.2 Large Dataset Support

The data imported into our system includes the vector field data necessary for particle advection as well as relevant multidimensional spatial variables like temperature, vorticity, and pressure. In order to enable large dataset support, we employ two capabilities:

1. The data can be spatially sub-sampled for lower resolution results.
2. Time-varying datasets can be streamed into memory as the visualization progresses over time.

Both of these approaches can be combined to enable real-time approximation followed by a higher resolution view when ready. This approach is often used in volume rendering [78] and visual mapping systems [79], which display a view into the results before they are precise.

The time-varying streaming works by buffering three time-steps in CPU memory and on the compute graphics card. At any time, two time-steps are being viewed, while the third time-step is being loaded into memory. This assumes that the loading time for a time-step is less than the viewing speed of one time-step. As soon as the viewing window moves to the second and third time-step, the first time-step becomes the new loading time-step. This continues until the end of the dataset and repeats if necessary. If the data viewing speed is greater than one time-step, it is possible to increase the buffer size, or temporally sample the dataset. If a user is viewing the data in reverse, the order of buffering is also reversed.

3.2.3 GPU Particle Advection and Interaction

For our GPU-based particle advection implementation, we use 4th order Runga-Kutta integration to update the space-time state of each particle as it moves through vector fields of interest. Initially particles are loaded into GPU memory and initialized on the GPU with multiple emitters, which can be placed anywhere in the scene. Every frame, these particles are iteratively advected through the vector field. After advecting the particle, the corresponding data values (e.g. velocity, vorticity, pressure) are interpolated based on the particle position.

For stable fluid flow simulation results, the GPU will update the position of the particle based on interpolated vector field velocity at the position. We use trilinear interpolation on the GPU, which requires that the data be represented as a structured grid. Therefore, for unstructured fluid flow data, we sample the flow fields into structured grids using interpolation to determine each grid point (using methods like weighted distance interpolation or cell based interpolation). For unsteady flows trilinear interpolation is used on GPU by including time as another dimension.

At any time, users are able to create multiple emitters, which each take a subset of the total number of particles and control where and how particles start. Each emitter also includes visualization properties like shape and color encodings. This is key for interacting with a large fluid-flow simulation since users can fine tune the visual based on their analysis. For example, in VR a user can zoom into a spatial region of interest and release more particles by clicking a button on a 3D stylus, therefore, studying the flow patterns at many levels of detail. Other interaction techniques include zooming, panning, and rotating.

3.2.4 Rendering Techniques

In order to efficiently render and compute particles from the vector fields, we take advantage of multiple graphics cards. While graphics are being rendered on a set of GPUs, we simultaneously compute on the compute GPUs. This, however, requires synchronization of data between GPUs.

In order to copy data between the compute GPUs and render GPUs, all GPUs are synchronized between the render/compute phase and the update phase. Specific and relevant time-step data is copied from GPU to GPU at this point. We enable OpenGL/Compute interoperability so that compute and rendering can be done without copying if the compute and rendering is done on the same GPU. This, however, requires that the two steps be done synchronously, meaning that there is a design trade off between compute time vs. copy time.

We follow two approaches for rendering the particles. The first uses a geometry shader to draw camera oriented billboards where the particles are rendered as spheres. If transparency is necessary, we sort the particle vertices and send the indices in the correct order. The second approach takes advantage of geometry instancing. This allows a user to define any Vertex Buffer Object (VBO) based mesh which will be repeated for each instance. A VBO is a memory locattion on the graphics card that stores geometry attribute information. In the vertex shader, we move and manipulate the mesh to match the particle pathline. This allows us to create new trajectory related shapes like streamlines, arrows, comets based on the particle history. An added advantage is that we can change the shape of the particles in real-time to encode other relevant data values like velocity, vorticity, or pressure.

3.3 Results

The results show how we can take advantage of the GPU hardware to improve performance. They also provide insights into how to achieve real-time fluid-flow interaction at the rates necessary for immersive visualization.

3.3.1 GPU Hardware Optimization

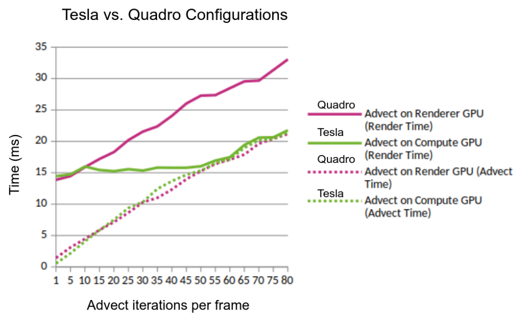


Figure 3.2: A comparison of the rendering and compute time between the Tesla congifuration (computing on the Tesla GPU) and the Quadro configuration (computing on the Quadro GPUs).

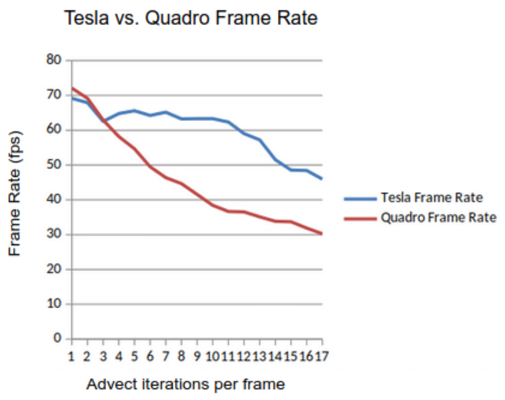


Figure 3.3: A comparison of framerates between the Tesla congifuration and the Quadro configuration.

We studied GPU hardware optimization using a 4-wall Cave VR environment on a single machine with three NVidia Quadro K5000 cards for rendering, and one NVidia Tesla K20c for compute. One Quadro card was used for the left and right wall, while the other two rendered to the front and floor respectively, for a total of 8 projectors. Both the Tesla and Quadro cards support CUDA for general-purpose parallel programming.

We compared two different configurations, (1) using only the Quadros for both compute and rendering with OpenGL/CUDA interop and (2) using the Tesla for compute and the Quadros for rendering. To compare the two configurations, we fixed the number of particles so that the rendering time remains constant while changing the number of advect iterations (smaller timestep) for each frame. As we increase the number of advect iterations, it means that we need more compute processing. The total frame time is the render time plus the compute time. Figure 3.2 shows that using the Tesla for compute while the Quadros for rendering achieves the best scalability as we increase the compute power need (i.e. number of advection iterations). This implies that simultaneously computing and rendering on separate GPUs can be more efficient than using the same GPU to do both. Copying from one GPU to another does not appear to be a bottleneck. The per frame time for advecting on the Tesla and rendering on the Quadros remains constant until the compute time (advect time) becomes more expensive than the render time (between 15 and 16 milliseconds).

Figure 3.3 shows this same relationship in terms of the frame rate. Using the Tesla for compute produces a shift in performance where the frame rate stays high (greater than 60 fps) until the compute catches up. This is nearly double the frame rate for the Quadro case (36 fps), which means that with the Tesla configuration, more computational processing can be done without sacrificing the increased performance needed for virtual reality.

3.3.2 Real-Time Interactive Visualization

For a second application, we visualized fluid flow in the right atrium of the heart on a Quadro K4000 graphics card with a ceiling mounted projector shining down on a table top. The projection was rendered in head-tracked quad-buffered stereo allowing a collaborative virtual reality experience. We 3D printed a model of the right atrium and added tracking so users could pick it up physically and virtually translate or rotate the object.

Figures 3.4 and 3.5 show the various visualization techniques that our Particle Flow framework allows including different pathline shapes and billboards. Figure 3.4 also shows that the same fluid-flow for the heart also works in other environments including our 4 wall cave described above. Framerates for these environments ranged from 30-60 fps depending on the visualization technique and number of particles. Compute and rendering were both done on the Quadro GPU.

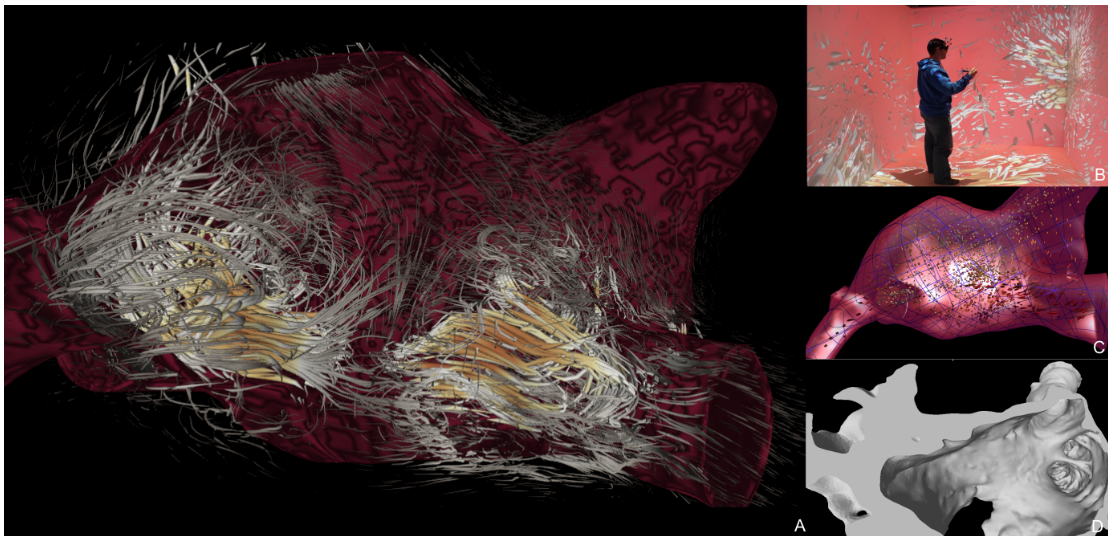


Figure 3.4: Left: (A) Real-time computation and rendering of animated streamlines as particles advecting through blood flow in the right atrium of the heart. Right: (B) Interacting with blood flow in a Cave environment. (C) Alternative rendering of the heart using smaller particles. (D) A 3D model of the right atrium that was also printed for interactive navigation.

Dependence on trilinear interpolation ended up being a major problem with the heart model since the mesh itself was an unstructured grid. In order to efficiently fit the data onto the GPU in a structured grid, we had to use a fine grid resolution in certain regions of the mesh. This caused two major problems: (1) First, we were unable to hold many timesteps in memory in order to finely sample the mesh for accuracy, implying that we could not easily show time-varying flow. (2) Second, since we could not sample the mesh into a small enough grid that would work in all parts of the heart, this caused strange anomalies in the blood flow where particles would go through the heart wall (see Figure 3.4), or they would circle around and fall into apparent singularities (see Figure 3.6).

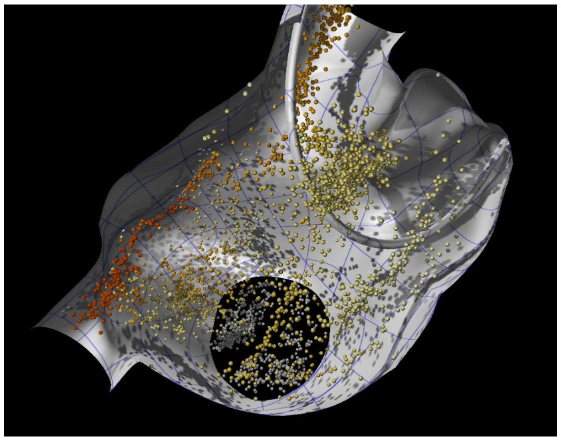


Figure 3.5: A visualization of blood flow through the right atrium of the heart using billboards as particles and shadow effects.

3.4 Discussion

Particle Flow approaches our thesis by looking primarily at the research question “How can we visualize large ensembles?” (Q1) as it relates to visualizing large memory intensive datasets (C1). We start simple by investigating how to render and interact with one local instance. Here we investigate how to enable user-defined interaction (Q3) for complex objects (C2) (e.g. fluid flow).

One major take away from our Particle Flow study is that interacting with large data sets is costly. First, it requires data sampling and interpolation, which may not provide the most accurate view of the data. Second, sophisticated hardware configurations, synchronization, and GPU communication is necessary to optimize performance. These are the types of problems fundamental to scientific visualization.

Even though there are performance and accuracy trade-offs, Particle Flow shows that it is possible to optimize a system to address these types of problems. For example, based on our investigation into GPU configurations, the Tesla configuration provides insight into fine tuning performance and accuracy. If a user has a desired frame rate, we can fix the render time and force an upper limit on the accuracy (i.e. advect iterations per frame) so that the render time matches the advect time. Alternatively, based on a desired accuracy per frame, we can determine a compute time that will imply the type of complexity of the rendering (i.e. render time) that can be done. In this case,

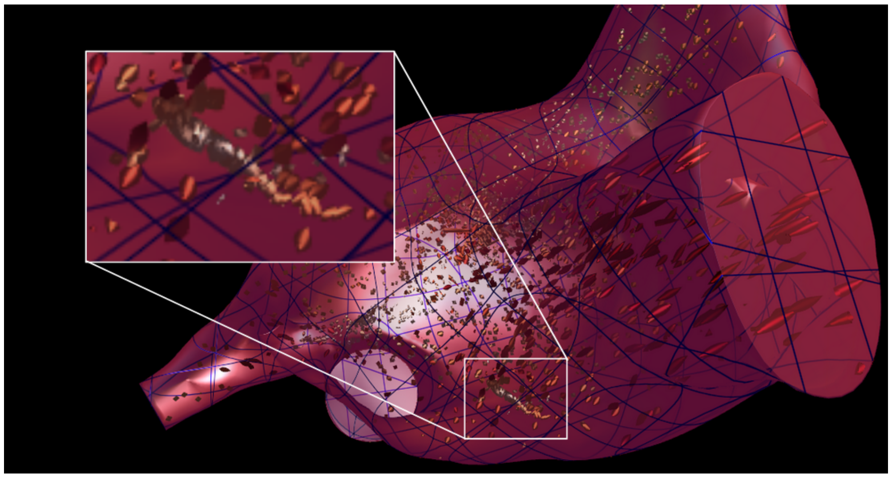


Figure 3.6: An anomaly in the particle advection due to sampling from an unstructured grid into a structured grid.

for example, fitting the rendering performance to the compute time may mean using billboards instead of geometry or decreasing the number of particles rendered, or vice versa. Optimizing these types of trade-offs and design decisions becomes even more important as we increase the number of instances. We will discuss this in the next chapter (Chapter 4).

Interestingly, optimizing Particle Flow is itself a high-dimensional problem (Q2, C4). As we have seen, fine tuning both visualization and compute parameters have a profound affect on understanding a data set. For example, the choice of trilinear interpolation appears to work well for the hurricane, but not as well for blood flow, due to the higher resolution grid needed for the unstructured heart mesh. Similarly, there are many other parameters that affect our fluid flow visualization including the number of particles, particle geometry complexity, grid resolution, temporal sampling, integration method, advection iterations, pathline steps, and render method. A parameter space analysis could be done by sampling the visualization parameters to create the application specific ensemble needed to understand the space of visualization possibilities. In other words, this chapter motivates our high-dimensional work by showing that our visualization results are merely a local view of a larger visualization space. Particle Flow works towards understanding this space, but more work needs to be done to investigate the best way to visualize fluid flow for an application.

3.5 Relevant Publications

- Bethany Tourek, Daniel Orban, Bogden Tanasoiu, Hakizumwami Birali Runesha, Daniel F. Keefe, and Arthur G. Erdman. Poster: Inverse design process: New methodology to design medical devices with big data. Minnesota Supercomputing Institute Research Exhibition, April 2016. [49]

3.6 Acknowledgements

This research was supported in part by grants from the NSF (IIS-1251069) and NIH (1R01EB018205-01). Hurricane Isabel data produced by the Weather Research and Forecast (WRF) model, courtesy of NCAR, and the U.S. National Science Foundation(NSF).

Chapter 4

Application Context 1:

Bento Box - Immersively Exploring the Cardiac Lead Design Space by Defining Sub-Volumes of Interest

Scientific visualization tools are rapidly embracing the necessary challenge of simultaneously visualizing multiple parameterized simulation data sets [19]. In the new paradigm, scientists hope to understand parameter relationships and stochastic trends that exist in a parameter space [32, 39]. At the same time, virtual reality (VR) environments have enabled exciting possible opportunities [80, 81, 82, 83, 84] including the exploration and comparison of time varying spatial data sets [27]. To name a few, engineers can now use augmented reality to view stress and deformation on a step ladder as they climb it [80], firefighters are able to train by experiencing complex fire scenarios in virtual reality [84], and neurosurgeons can visualize tissue changes during an operation [83]. Although VR offers a unique perspective to view 3D and 4D data, it requires high framerates for interactivity and optimized use of precious GPU memory. Accurate simulations, on the other hand, are often very large due to dynamic unstructured mesh resolutions and small timesteps, making it difficult to simply render even one data set. To solve this, large data visualization frameworks often use data sampling and efficient rendering techniques to engage the GPU [21, 19]. Even then, VR is mostly used to add a stereoscopic view, and is rarely an integral part of interactive data instance comparison [27].

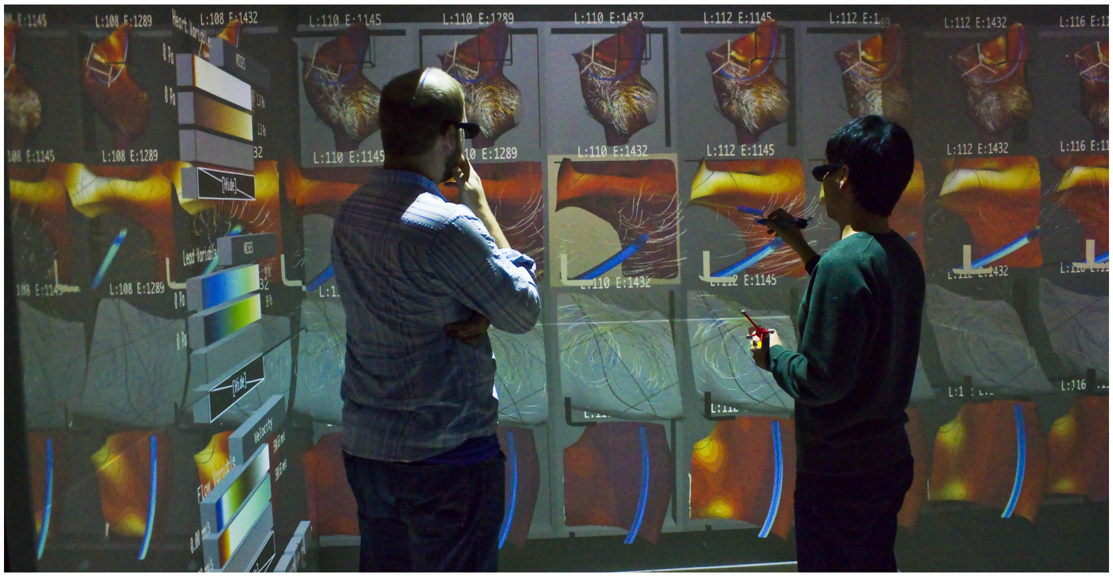


Figure 4.1: Bento Box is a virtual reality visualization and 3D user interface technique for comparative analysis of data ensembles, such as this set of 10 time-varying simulations of blood flow around a cardiac lead in the right atrium of the heart. Each *column* shows a simulation with different parameters, here varying the length and stiffness of the lead. Each *row* shows a different view of the data. The top row is a zoomed-out overview. Users add additional rows of complementary, zoomed-in views during analysis.

This chapter investigates how to efficiently render **multiple** large data sets for comparison (Q1, C1) while adding expert intuition through interaction (Q3, C2). In contrast to the previous chapter, we are now interested in immersively rendering more than one instance (C3) in an ensemble, a rare feat in the visualization literature [27]. Here we describe our first data-driven exploratory interface. We discuss the sophisticated data sampling and rendering algorithms needed to make the visualization scalable, while considering the comparison of *complex*, ill-defined spatial features that are of interest to subject matter experts.

The project that is described in this chapter is part of a larger collaborative effort with other computer science researchers studying human computer interaction. The interface and interaction techniques are described in detail in Palpable Visualizations [85]. The specific contributions in this dissertation include the data processing, sampling, and rendering methods that enable the comparision of multiple fluid structure interaction in virtual reality. In short, we add the scalable component to make the existing application into a data-driven exploratory interface.

4.1 Motivation and Application Background

Fluid-Structure Interactions (FSI), the coupling of fluid and solid domains, are critical to the success of many engineering applications including developing medical devices, bridges, airplanes, and engines. The oscillatory patterns produced by the solid materials and the surrounding fluids must be considered in order to accurately simulate the complex system and avoid catastrophic consequences [86]. In order to design stable, but effective solutions, engineers often employ Computer-Aided Engineering (CAE) methods to build virtual prototypes through simulation and computational analysis techniques.

There are usually several competing design decisions worth investigating. In our case, we focus on a medical device engineering application involving a cardiac lead placed in the right atrium of the heart [33]. An engineer must discover what length and stiffness of the lead adequately attaches to the heart wall while minimizing the effect on the surrounding blood flow. At the same time the engineer might also want to reduce the amount of fibrosis build-up where the heart wall and the lead meet for a specific lead stiffness range.

These problems are non-linear, spatially complex, and may have more than one viable solution. The lack of a clear objective function and the existence of vague user-defined features makes it hard for computers to automatically find optimal solutions. Interactive visualization helps bridge the gap by using the human visual system to augment human intelligence with powerful memory intensive computational models [8, 34].

For FSI design space exploration, there are two major problems we work towards solving:

1. It is challenging to know how to visually compare simulation instances. This is because FSI simulations are naturally time-varying, have elaborate feature relationships, and often contain unnecessary detail (visual clutter).

2. Accurate simulations tend to be very large in size due to dynamic unstructured

mesh resolutions. This can make it impossible to load multiple simulation instances into GPU memory.

4.2 Visualization Techniques and Implementation

We present a framework for interactively comparing multiple large fluid structure interaction (FSI) simulations by extending a VR application called Bento Box. Our approach uses rendering, sampling, and streaming strategies to optimize memory on the GPU while achieving interactive framerates. Our contribution provides a case study for building a multi-instance VR comparison tool for FSI data sets that are too large to fit onto a GPU.

4.2.1 Visual Instance Comparison

In order to understand the differences between multiple FSI simulations, like the cardiac lead model, we extend a general VR comparison tool called Bento Box. Bento Box allows users to interactively select and drill down to linked regions of interest across several instances. 4.1 shows the bimanual 3D interface specifically built for immersive VR environments where users can view and build a comparison grid. The columns in the grid represent instances and the rows linked sub-volumes of interest. The first row shows an overview of each simulation. From any cell, engineers can interactively create linked regions of interest (new rows) by clicking and dragging a bounding box inside the zoomed-in volume. Therefore, users can compare many spatially relevant views side-by-side based on user-defined volumes of interest (i.e. stress where the lead touches the heart and blood flow patterns elsewhere). Time steps can be compared by adding new columns for each visible instance. Users interact by selecting and zooming into a set of cells using a laser pointer. They can then immersively reposition, resize, change color maps, change visible variables, scroll through time, and create new sub-volumes of interest to customize each linked row.

The power of using Bento Box lies in the ability to visually analyze the relationship between input parameters and multiple output features. For example, users can compare the sensitivity of the stiffness in a lead with respect to both the stress on the heart wall and variations in fluid flow. Users can change color maps on a row by row basis in order to highlight specific trends in the FSI variables. Due to large data instance sizes, we can not simply load all the data onto the GPU. In order to make Bento Box possible for multiple large FSI simulations, we implement different rendering, sampling, and streaming strategies for the solid and fluid domains.

4.2.2 Data Sampling and Rendering

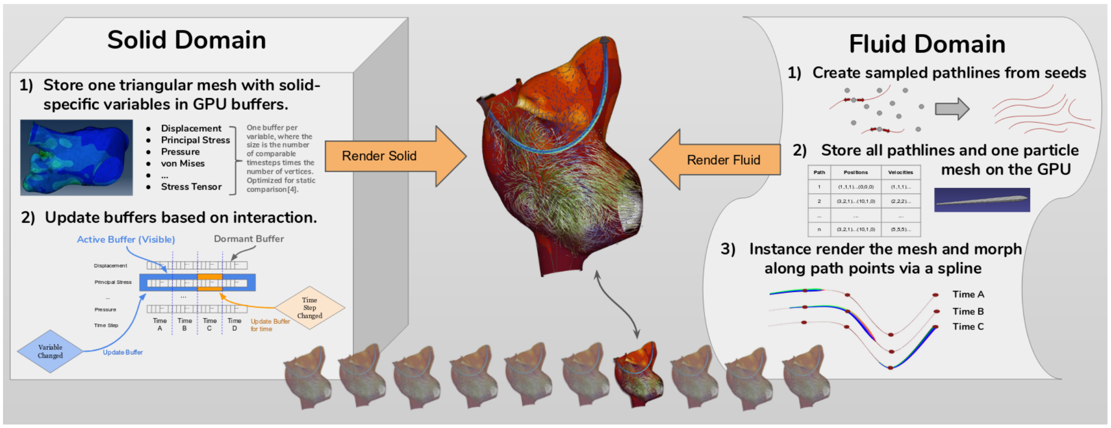

Solid Domain
1) Store one triangular mesh with solid-specific variables in GPU buffers.
Displacement
Principal Stress
Pressure
von Mises
Stress Tensor
One buffer per variable, where the size is the number of comparable timesteps times the number of vertices. Optimized for static comparison[4].
2) Update buffers based on interaction.
Active Buffer (Visible)
Dormant Buffer
Time Step Changed
Variable Changed
Update Buffer
Update Buffer for time
Render Solid
Render Fluid
Fluid Domain
1) Create sampled pathlines from seeds
2) Store all pathlines and one particle mesh on the GPU
3) Instance render the mesh and morph along path points via a spline
Time A
Time B
Time C

Figure 4.2: Data sampling and rendering techniques for multiple FSI simulations. The visualization algorithms for rendering the solid and fluid domains are handled as separate processes to optimize memory.

For the purposes of this chapter's position on comparing multiple large complex FSI simulations we will focus on the data sampling and rendering techniques needed to make Bento Box possible. For each cell in the Bento Box grid we need to render an FSI instance at a specific time and specific clipped view. Whenever a sub-volume of interest is updated via 3D interactions, the view settings are calculated and reported to the renderer. Our approach uses different strategies for solid domain data (Abaqus Implicit Solver) and fluid domain data (Abaqus CFD Solver). Figure 4.2 describes how we integrate our solid and fluid methods.

Visualizing the Solid-Domain The solid domain describes properties of deformable meshes like displacement and stress and defines the physical structure of the volumetric solid with a mesh. The element and node properties of the simulation are used to generate a triangulated mesh that can be rendered. Triangles are constructed from the supplied primitives and passed to the GPU. Each instance uses its own index and vertex array since the meshes deform as a result of the simulation.

The solid domain data are loaded into GPU memory on an as-needed basis, only for the critical times currently displayed. Each solid domain variable (e.g., stress) is assigned its own GPU buffer with size equal to the number of critical times multiplied by the number of mesh nodes and then by the number of components (1 float for scalar values, 3 for vector values). Although for perceptual reasons, our default is to view multiple critical times in static juxtaposed views, the visualizations can be animated by simply swapping data every frame, and this can be done automatically or interactively using an animated timeline widget.

Visualizing the Fluid Domain Since the fluid domain data are rendered as flowing particles, pre-computing the set of pathlines that the particles follow significantly reduces the size of the data that must be stored in order to render the visualization. Our implementation uses a cell location technique, similar to a CellTree [87] data structure, to calculate path lines directly from the unstructured grid data using a Fourth Order Runga-Kutta integration method and random seeding. Since path lines inherently encode time; the display for a given critical time can be determined from a single set of path lines by simply determining which portion of each line to display.

The specific path lines to draw within each cubby are determined based upon the view. When the view settings produce an external view of the dataset, as in the top row of the widget (scale = 1.0), flow data are displayed using a random selection of approximately N paths from the pre-calculated pathset. We adjust the number of particles to render for each row of cells as a function of the scale of the data (i.e., the zoom factor), drawing a larger number of particles at random from the pre-computed pathset when our view is zoomed in to a smaller sub-volume of the data.

This rendering method makes it possible to visualize the flow at any scale and any critical time from the same pre-calculated data. Thus, changing the critical time does not use any additional GPU memory or require additional CPU-GPU memory updates. One path buffer array is stored on the GPU for each data instance in the display. The buffer is arranged according to a path index and each path has the same path length. The GPU also stores path value buffers for data variables such as velocity and pressure that may be used to color the particles.

The comet geometry is defined as an axis-aligned 3D mesh. Since each visualization will include thousands of these meshes, the mesh is rendered using instancing and deformed in a shader to fit within an appropriate start and end position along the path line. Our implementation uses a mesh with 72 triangles.

4.3 Results

The Bento Box approach was successfully applied to the blood flow simulation ensemble described in Section 4.1. We describe both visual comparison and performance results below.

4.3.1 Visual Comparison Results

Our goal was to explore the cardiac lead data sets in order to find parameter relationships determined by the ensemble. Our interdisciplinary collaborators iteratively worked with us to validate and discover new insights. As subject matter experts, they noticed that the lead stress patterns at cross sections were appropriate by creating sub-volumes at key locations along the lead. They also verified the observed differences in blood flow based on lead stress due to changes in the heart wall deformation.

Figure 4.3 shows a comparison of lead stiffness in columns increasing from left to right. The first column shows an overview of the simulation instance along with the various locations of the spatially aligned cubbies below. This shows major spatial trends, but also displays the visual clutter that is fixed by the feature rows below. The second and third row show the expected stress pattern on the lead as the variable increases. The fourth column gives insight into how the stress in the lead affects the stress on the heart wall at the attachment point, which also relates to displacement.

We also wanted to study how the output changes in relation to the lead length input parameter. Figure 4.4 shows another view of the heart dataset by comparing an increasing lead length from left to right. The engineers observed a difference in fluid flow patterns. For the longest lead length, the velocity appears to be slower (darker red) and slightly out of phase with the other simulations. The scientists hypothesized that the larger lead length stretches the heart wall, making the atrium bigger, and the increased volume slows the flow pattern.

4.3.2 Evaluation of Rendering Performance

For the purposes of this chapter we also wanted to show that multiple large data data simulations could be loaded and interactively compared. Characteristics for the ten data instances visualized are reported in Table 4.5. After processing the 39 GB of raw data, the amount of memory needed to accurately visualize the solid and fluid attributes is over 8 GB, exceeding a 4 GB GPU hardware limit on our 4-wall cave environment, a 2 processor Intel(R) Xeon(R) CPU E5-2640 @2.50GHz machine with two NVIDIA Quadro K5000 cards and 192 GB of RAM. Since this machine has more than 8 GB of RAM, it is possible to stream solid attribute data from memory into the GPU when needed. Streaming combined with the pathline sampling of the fluid, provides an extremely low memory footprint on the GPU, allowing us to visualize many instances and variables.

Using this application as a testbed, we also report some rendering performance measures, summarized in Figure 4.6. These timings were recorded on a 4 core processor Intel(R) CORE(TM) i7-7700HQ CPU @2.80GHz machine with 16 GB of RAM and a NVIDIA GeForce GTX 1070 graphics card, which was configured to drive an HTC Vive with a resolution of 2160x1200 pixels. The data sets are streamed into memory from a 128 GB M.2 PCIe SSD. The scatter plot in Figure 4.6 shows a systematic sampling of Bento Box resolutions with varying sub-volume scales. As expected, the logarithmic trendline shows that frame rate decreases as the number of Bento Box cubbies are increased. The scatter plot also shows that there is a relationship between frame rate and the number of particles drawn. This is because it is expensive to render an increased number of particles for smaller sub-volumes in order to display a constant particle density as explained in Section 4.2.2.

Although we collect data for large Bento Box resolutions, in practice, we find that having more than 20 or so cubbies is not useful because it is too visually complex. In this chapter we show various useful arrangements (e.g., 10x2, 4x5, 3x6) that run at interactive rates consistent with the above evaluation.

4.4 Discussion

One major limitation with our Particle Flow approach in Chapter 3 was that it focused on only one simulation, but for our larger study we want to understand ensembles. Our FSI implementation using Bento Box has shown that it is indeed possible to efficiently (Q1) and accurately interact with multiple (C3) large simulation data sets (C1). In addition we investigated how to apply user-defined interaction by spatially (C2) defining (Q3) sub-volumes of interest in a Bento Box. The user-defined visual layout has shown that we can use a local view of an ensemble to compare parameter relationships for complex spatial features relating to both the fluid and structure domains. In fact, our results in Figure 4.5 show that there is a potential to scale to higher numbers of available instances (only 1.22 GB out of 4 GB are used on the GPU for 10 instances). Figure 4.6 also suggests that performance is dependent upon the number of instances shown, not necessarily the number of instances available.

In terms of our thesis, we have a result that suggests we can efficiently and intuitively interact (Q1, Q3) with a local uncertainty distribution (C5) for several memory intensive (C1), spatial (C2) data sets. However, for this case the parameter space is small with only two parameters (C4), and the the number of instances in the ensemble is also small (C3). It is therefore not possible to adequately explore research question Q2, which asks how to visually understand large high-dimensional spaces. On the other hand, it does show that perhaps we could use results from our investigation into Q2. If we had a way to visually explore a high-dimensional parameter space, we could perhaps determine the important subset of instances (local uncertainty distribution) that would allow us to intuitively answer what-if questions. For example, even using the Cardiac Lead application, we can study two subsets: stiffness or stress. Imagine using dimensionality reduction, to determine which set of instances are similar, giving us insight that would be more closely related to the underlying local structure of a larger parameter space.

Finally, our FSI investigation allows us to consider how we might use user-defined interaction to drive the underlying visualization algorithms, adding expert intuition (Q3) to complex data (C2). Automated methods (e.g. machine learning) that analyze Big Data by themselves do not consider what is important to the user, therefore, the algorithms could see the spatial clutter in an FSI simulation as a vital pattern. However, using Bento Box, we have shown that users can define sub-volumes of interest or importance to the application. This means we can perhaps infer information based on regions defined by user interaction. For example, Figure 4.3 shows that the inner stresses in the lead combined with both the stress and fluid at the lead's attachment point are important. This means that it is perhaps possible to use this information to generate distance functions or fine tune underlying high-dimensional models. We will investigate this idea more closely in Chapter 6.

4.5 Relevant Publications

- Seth A Johnson, Daniel Orban, Hakizumwami Birali Runesha, Lingyu Meng, Bethany Juhnke, Arthur Erdman, Francesca Samsel, and Daniel F Keefe. Bento box: An interactive and zoomable small multiples technique for visualizing 4d simulation ensembles in virtual reality. *Frontiers in Robotics and AI*, 6:61, 2019. [9]

- Daniel Orban, Seth Johnson, Hakizumwami Birali Runesha, Lingyu Meng, Bethany Juhnke, Arthur Erdman, Francesca Samsel, and Daniel F Keefe. Com- parison of multiple large fluid-structure interaction simulations in virtual reality. In *2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)*, pages 110–111. IEEE, 2018. [28]

4.6 Acknowledgments

This work was supported in part by grants from the National Science Foundation (IIS-1251069,IIS-1218058) and the National Institutes of Health (1R01EB018205-01). The authors thank Bogdan Tanasolu, Georgi Subashki, and Shan "Sandy" Wang for assistance with developing simulations and data wrangling. Thanks to Dr. Paul Iaizzo and the University of Minnesota Visible Heart Lab for access to heart geometry data and feedback during iterative development. This work was supported in part with computing resources from the Minnesota Supercomputing Institute and hardware donations from NVidia. Our software utilizes the VRPN library maintained by UNC-Chapel Hill's CISMM project with support from NIH/NCRR and NIH/NIBIB award 2P41EB002025.

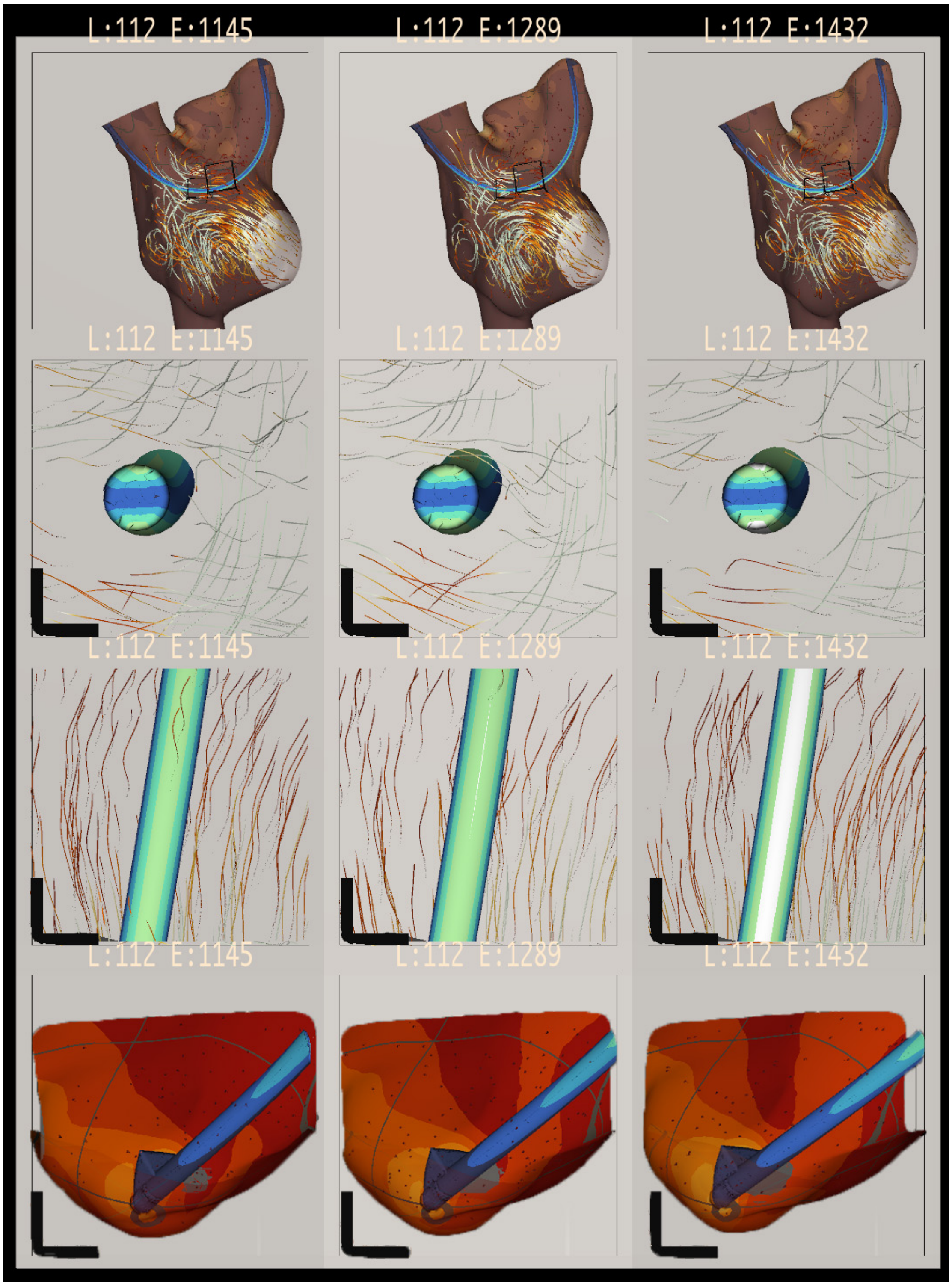


Figure 4.3: Bento Box arranged for a comparison of cardiac leads with three different stiffness parameters, increasing in stiffness from left to right. The top row shows an overview of the dataset. The middle two rows highlight stress on the lead itself, which appears to increase with stiffer leads. The bottom row zooms in on the attachment point of the lead in the atrial appendix. Here, stress on the atrial walls also appears to increase with stiffer leads.

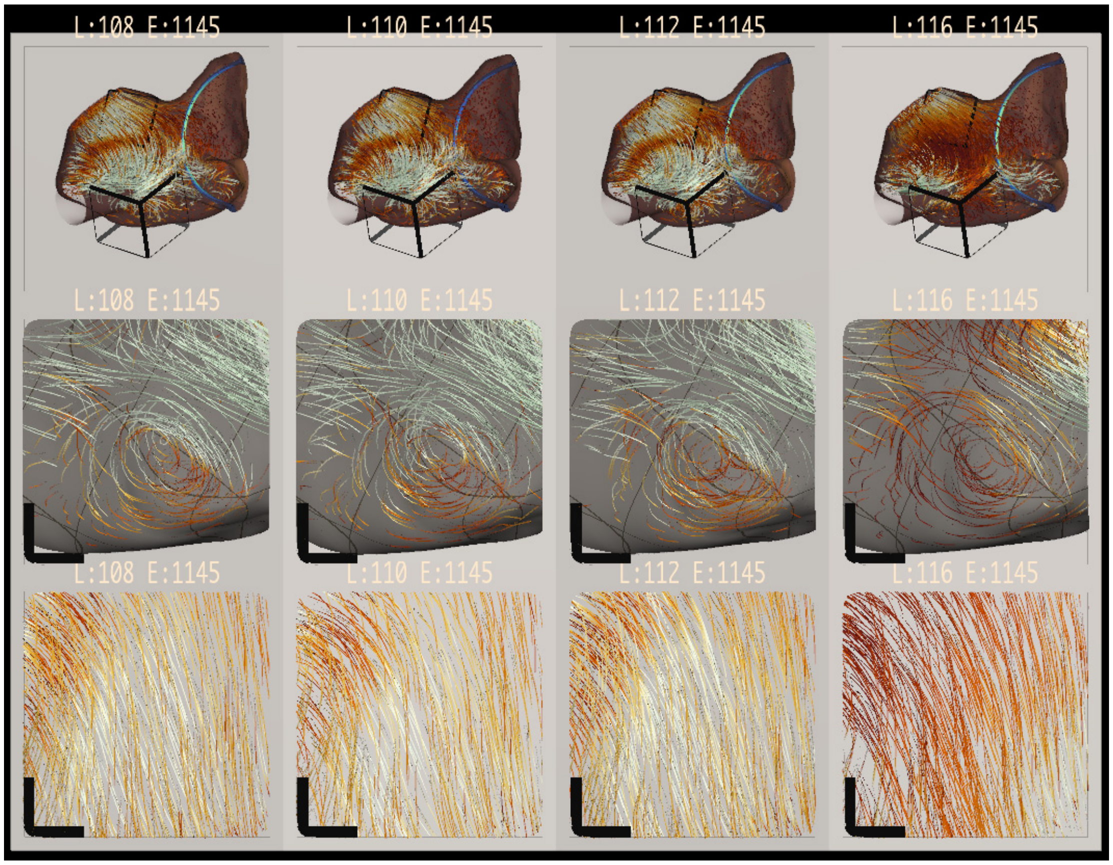


Figure 4.4: Bento Box arranged for a comparison of different length cardiac leads. Lead length increases from left to right. At this timestep in the simulation, the longest lead length creates a slower flow (i.e., darker red path lines).

ID	Raw (MB)	Solid (MB) Processed	GPU	Fluid (MB) Proc. / GPU	Total (MB) Processed	GPU
108-1145	5,252.6	892.6	7.0	115.2	1,007.8	122.2
108-1289	4,724.4	682.2	7.0	115.2	797.4	122.2
108-1432	4,911.5	682.2	7.0	115.2	797.4	122.2
110-1145	2,933.7	660.9	6.7	115.2	776.2	122.0
110-1289	2,933.7	660.9	6.7	115.2	776.2	122.0
110-1432	2,933.7	660.9	6.7	115.2	776.2	122.0
112-1145	4,926.1	687.5	7.0	115.2	802.7	122.3
112-1289	4,934.6	687.5	7.0	115.2	802.7	122.3
112-1432	4,934.6	687.5	7.0	115.2	802.7	122.3
116-1145	1,588.9	832.4	6.9	115.2	947.6	122.1
	40,073.6	7,134.7	69.1	1,152.3	**8,287.0**	**1,221.4**

Figure 4.5: Characteristics and memory usage for the ten data instances.

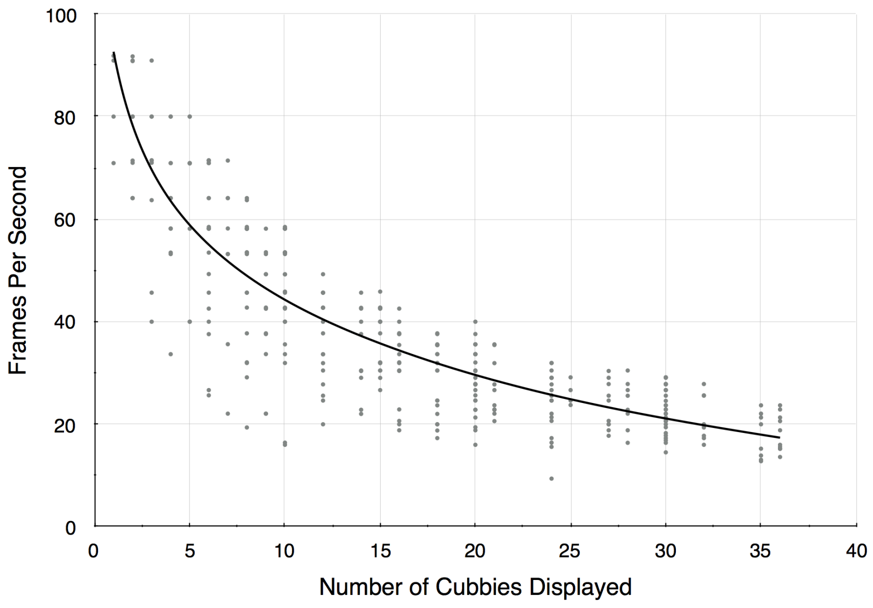


Figure 4.6: Performance evaluation shows that the frame rates decrease as the number of cubbies increase.

Chapter 5

Application Context 2: Cinema:Bandit - Augmenting Shock Physics Experimental Design using Linked Ensemble Visualization Techniques

In this dissertation we have discussed how to interact with scientific ensembles that are both spatial and have a large memory footprint. Ensembles are also characterized as large (Q1) in terms of number of instances (C3) and the number of parameters (C4). This chapter begins our study into intuitively understanding multi-dimensional parameter spaces (Q2), a difficult challenge since the human visual system has trouble visually comprehending anything above two or three dimensions [24]. We start with an application, Cinema:Bandit, that is used to inform real-time decisions at shock physics experiments through linked ensemble visualization techniques like parallel coordinate plots, time series supposition, and side-by-side image analysis. This motivates our work in next chapter (Chapter 6), where we go beyond traditional ensemble visualization approaches by introducing user-defined direct manipulation of virtual data instances in the context of dimensionality reduced views.

When faced with a set of decisions, we must consider when to execute on the information we have and when to gather more information. This is especially difficult when we are working with multi-dimensional parameter spaces where the space is sparse and the number of possible decisions increases exponentially with each dimension. To complicate the situation, there is usually a high cost associated with each sample. In the category of stochastic scheduling, this situation is called the N-armed bandit problem [88]. The problem, inspired by gamblers who want to choose the right slot machine (a.k.a. one-armed bandit), involves making sequential choices to optimize the expected gain based on known information, which may change as more choices are made or as resources are allocated to a choice [89]. The bandit model applies to many practical applications, including clinical trials, adaptive routing, and financial portfolio design. In these problems, there are trade-offs between gaining new knowledge (exploration) and using the existing knowledge to make advantageous decisions (exploitation). Scientific exploration fits well into this model because the experimental space is often unexplored but there is a limited budget of experiments that can be made (often due to cost), so experiments need to be made strategically. Therefore, it is important to use the known information to make experimental decisions while also exploring the space.

We approach this problem through interactive visualization by combining previously explored data with ensemble visualization techniques in order to provide information that helps augment the experimental design (i.e. choice of parameters for the next experimental run). This leads to an informational feedback loop that helps us understand the space better since each sample in the experimental space becomes part of the ensemble. Ensemble visualization also helps users see which parts of the space need more exploration so that advantageous decisions can be made. In this chapter we consider traditional scientific approaches that are used to explore multi-dimensional data.

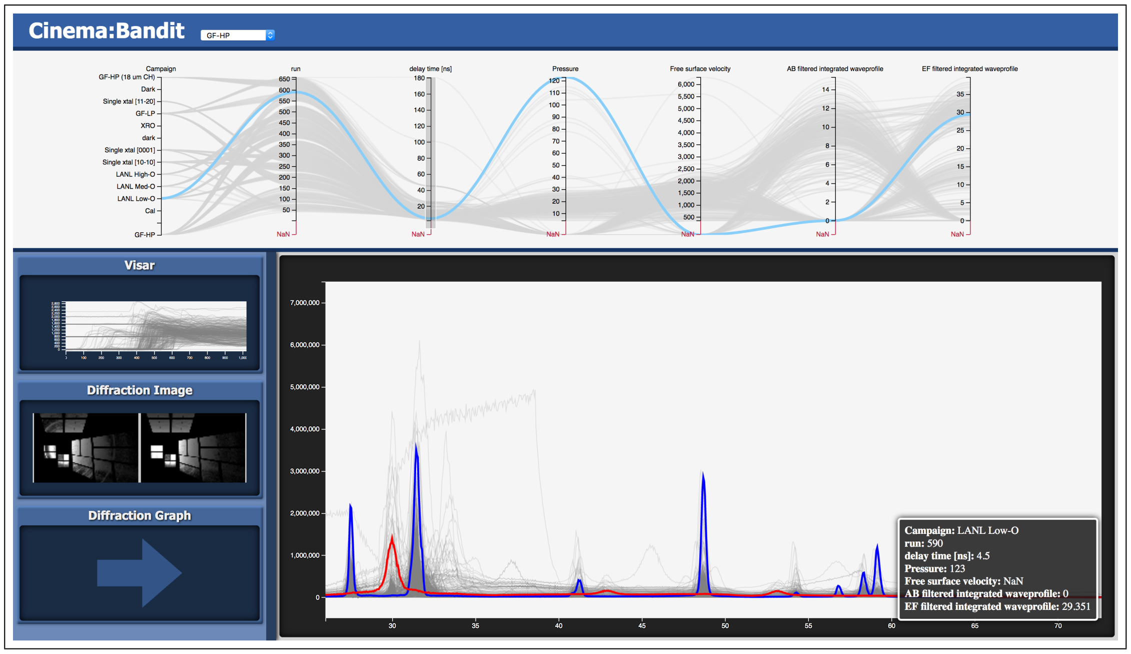
5.1 Motivation and Application Background

Experimental workflows and visualization techniques need to adapt to meet the demands of the next generation of shock physics experimental facilities. Recent technological advances are allowing material scientists to better investigate shock compressed matter under extreme conditions [90, 91]. For example, X-ray free electron lasers now provide unprecedented spatial and temporal resolution to investigate phase transitions, compression, and strength of materials [90, 92]. Parametric studies that combine high-pressure and high-temperature methods [91] with X-ray diffraction [90, 92] and optical diagnostic techniques [93] allow scientists to not only study astrophysics, geophysics, and planetary physics, but also to synthesize new materials with useful strength properties. In the near future, these experiments will be performed faster with orders of magnitude more data, up to 10 Hz at the High Energy Density Science (HED) Instrument at the European X-ray Free Electron Laser [94, 95, 96]. However, at data rates up to 10 Hz, the data feed will be too fast for scientists to interact with each dynamic event or shot, which necessitates the development of automated processing and prompt display tools. These tools will facilitate informed decisions by scientists *during an experiment* to optimize their use of time at the beamline facility.

Traditional shock physics datasets have used diagnostics that measured the bulk volumetric response of the materials. The most common type of measurement made is velocimetry, which uses optical light (usually from a laser) to measure the velocity or displacement of material interfaces. These velocities can be related back to thermodynamic state variables using the fundamental conversation laws of mass, momentum, and energy (*i.e.* the Rankine-Hugoniot relationships). The Velocity Interferometry for Any Reflector (VISAR) is a common technique that was first implemented to make measurements on a single point [97] and is now also used in a line-imaging version, which provides one dimension of spatial resolution [98, 99, 100].

The utility of X-ray diffraction (XRD) for diagnosing material response (*e.g.* phase changes, strain, and texture) to shock compression was recognized many years ago and was originally achieved by Johnson *et al.* [101, 102, 103]. Due to the X-ray source technology at that time, the experiments were extremely difficult to perform, and XRD was not adopted as a common technique in shock compression experimentation. During the past two decades, improvements in X-ray brightness at high-energy laser facilities [104], at X-ray synchrotrons [105], and at X-ray free electron lasers [90, 106, 107, 108] has resulted in the wide-spread use of XRD for *in situ* measurements in dynamically compressed materials.

Real-time analysis and visualization of data is important to maximize the scientific productivity at beamline facilities. Automation within the heterogeneous hardware and software environments [109, 110] at beamline facilities has always been an active area of development. Many facilities have data management systems that enable users to automate analyses, *e.g.* [111], and there are a number of cross-platform tools to



Campaign: LANL Low-O
run: 590
delay time [ns]: 4.5
Pressure: 123
Free surface velocity: NaN
AB filtered integrated waveprofile: 0
EF filtered integrated waveprofile: 29.351

Figure 5.1: An overview of the *Cinema:Bandit* interface. The parallel coordinates plot at the top shows parametric (*i.e.* inputs and outputs) information about each run in an experimental dataset. At the bottom, the user can alternate between the VISAR, Diffraction Image, and Diffraction Graph views. Each view shows information that has been extracted from the experimental runs' data.

process and visualize XRD data, such as Mantid [112], GSAS-II [113], MAUD [114], FIT2D [115], and DIOPTAS [116].

Our goal is to improve visualization of datasets for shock physics experiments at XFELs, particularly visualizing aggregated data in interactive views. In this chapter, we present a visualization tool for experimental data exploration at beamline facilities and describe its inclusion in a continuous workflow. Our visualization tool, called *Cinema:Bandit*, uses multiple linked views [117] and the visual comparison techniques of superposition and juxtaposition [118] to understand the relationship between many samples in an experiment as a whole. It allows disparate data types to be coherently visualized and explored, and it supports irregular sampling of experimental parameters, including experimental inputs, measurements and outputs, that can be displayed and interactively adjusted. To address the automation issues at beamline facilities, we demonstrate the use of *Cinema:Bandit* in a modular workflow with user-defined algorithms and tools that process the raw detector data at the beamline facility. In this chapter, we provide a detailed description of our visualization tool *Cinema:Bandit* and the distribution of the software (Section 5.2.1). Then, we discuss the standardized Cinema database specification (Section 5.2.1) that our tool uses. To describe the visualization process at a beamline facility, we present an integrated experimental workflow implementation that was used for data processing and visualization during experiments at the LCLS facility (Section 5.2.2). We show how *Cinema:Bandit* was used to analyze the results from a titanium experiment (Section 5.3). In particular, we demonstrate its ability to identify outliers or problematic data, direct experimental design, and compare ensembles of samples.

5.2 Visualization Techniques and Implementation

5.2.1 *Cinema:Bandit* Implementation and Description

Cinema:Bandit is a modular visualization tool that enables collation, visual comparison, and filtering for an ensemble of samples. It is a web-based tool that is part of a group of open-source Cinema tools, which is a visualization approach to address extreme-scale data analysis problems in the supercomputing and the visualization community [46]. The tool is a Cinema viewer, hierarchically named *Cinema:Bandit* due to a visual similarity with a slot machine (a slot machine is also known as a One-armed Bandit). In Figure 5.1 we see the set of three data views (to the left of the screen) that respond to the user's interaction with the tool. These constantly updating data views are reminiscent of the three spinning wheels of a slot machine.

Cinema:Bandit falls under the general category of ensemble visualization, a method for using parameterized datasets, or ensembles, to display information to a user [119, 11, 12]. This is illustrated in Figure 5.1 where all runs are shown in the interactive parallel coordinate plot at the top. The interactive parallel coordinate plot is a multi-dimensional visualization that displays each parameter as a parallel vertical axis and each run as a polyline whose vertices are located on the these axes. Additionally, Figure 5.1 shows four interactive linked views: an interactive parallel coordinate plot at the top and three other views on the left-hand side. In Figure 5.1, we display velocimetry profiles, XRD images, and XRD patterns; however, there can be any number of side views characterized by either line graphs or side-by-side images.

Implementation

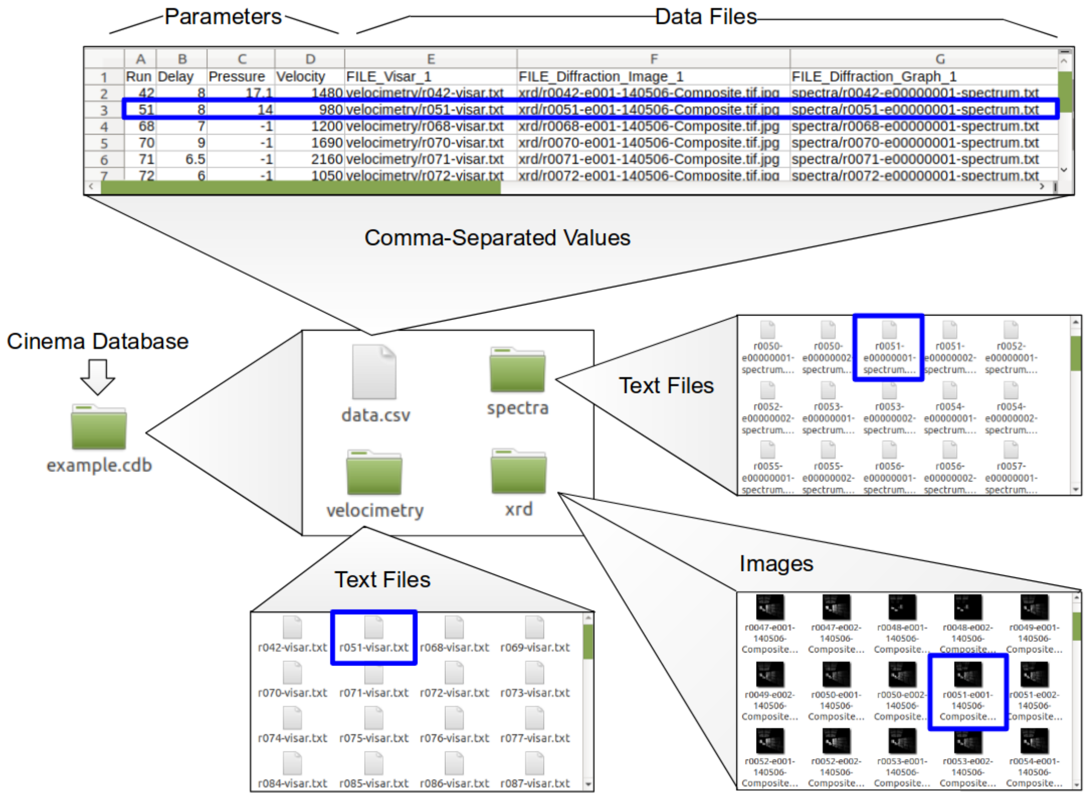
The web-based tool is written in HTML and JavaScript using the Cinema Components Library [120] and D3 [121]. The Cinema Components Library is an open-source library containing components for querying Cinema SpecD databases and building web-based visualization tools (*e.g.* parallel coordinate plot or scatter plot). Since we leverage web-based technologies, remote visualization of data is possible, and users can open up many instances of the application by simply navigating to or double clicking the HTML page. Additionally, there are no necessary external libraries to install, which is ideal for the heterogeneous environments at different beamline facilities.

Due to web application security, Cinema databases need to be stored in a sub-folder under the *Cinema:Bandit* application folder on the filesystem. The source code and documentation are available for download from `https://github.com/cinemascience/cinema_bandit`. New releases of *Cinema:Bandit* will be posted at the same location.

Cinema Database Specification

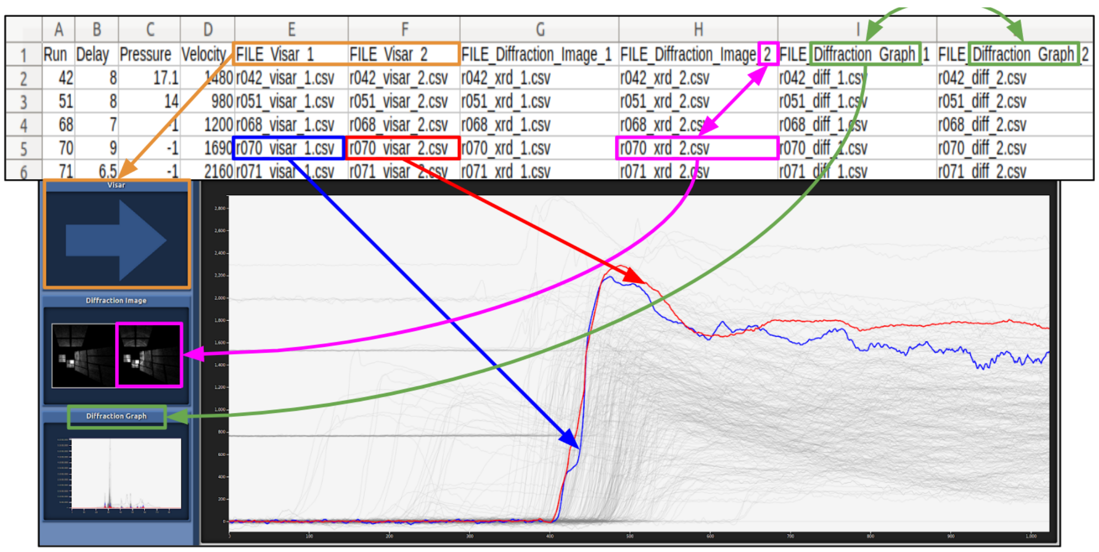
Cinema visualization tools, such as *Cinema:Bandit*, use a standardized database [122], which contains a set of sampled data, often in the form of images, along with simulation/experiment parameter information (*e.g.* pressure, temperature, and time delay). Each item in the database is a parameterized run that has a set of files representing line graphs or images. Cinema databases can be visualized through other open-source tools since they are composed of commonly used files types, *e.g.* CSV files or `png` images.

Figure 5.2 shows an example of a Cinema Specification-D (Spec-D) database [122] for a shock physics experiment. A Spec-D database is a folder that contains a `data.csv` at the top level and any number of data files (*e.g.* images or text files) that could be nested in sub-folders. The `data.csv` file is a CSV file with columns and rows, which can be saved from common spreadsheet applications. Each row represents a separate instance, experiment, sample, or run. Each column represents a parameter of interest, such as pressure, temperature, or velocity. In the `data.csv` file, users can optionally specify multiple associated data files at the end of the parameter list by adding the prefix `FILE` to the title. Each row in the database can then designate relative paths



	A	B	C	D	E	F	G
1	Run	Delay	Pressure	Velocity	FILE_Visar_1	FILE_Diffraction_Image_1	FILE_Diffraction_Graph_1
2	42	8	17.1	1480	velocimetry/r042-visar.txt	xrd/r0042-e001-140506-Composite.tif.jpg	spectra/r0042-e00000001-spectrum.txt
3	51	8	14	980	velocimetry/r051-visar.txt	xrd/r0051-e001-140506-Composite.tif.jpg	spectra/r0051-e00000001-spectrum.txt
4	68	7	-1	1200	velocimetry/r068-visar.txt	xrd/r0068-e001-140506-Composite.tif.jpg	spectra/r0068-e00000001-spectrum.txt
5	70	9	-1	1690	velocimetry/r070-visar.txt	xrd/r0070-e001-140506-Composite.tif.jpg	spectra/r0070-e00000001-spectrum.txt
6	71	6.5	-1	2160	velocimetry/r071-visar.txt	xrd/r0071-e001-140506-Composite.tif.jpg	spectra/r0071-e00000001-spectrum.txt
7	72	6	-1	1050	velocimetry/r072-visar.txt	xrd/r0072-e001-140506-Composite.tif.jpg	spectra/r0072-e00000001-spectrum.txt

Figure 5.2: Example Cinema Spec-D database. Each Cinema database has a `data.csv` file as well as multiple folders that contain types of data (*e.g.* velocimetry profiles, XRD images, and XRD patterns). The `data.csv` contains parametric information about the run as well as paths to corresponding file names.



	A	B	C	D	E	F	G	H	I	J
1	Run	Delay	Pressure	Velocity	FILE_Visar_1	FILE_Visar_2	FILE_Diffraction_Image_1	FILE_Diffraction_Image_2	FILE_Diffraction_Graph_1	FILE_Diffraction_Graph_2
2	42	8	17.1	1480	r042_visar_1.csv	r042_visar_2.csv	r042_xrd_1.csv	r042_xrd_2.csv	r042_diff_1.csv	r042_diff_2.csv
3	51	8	14	980	r051_visar_1.csv	r051_visar_2.csv	r051_xrd_1.csv	r051_xrd_2.csv	r051_diff_1.csv	r051_diff_2.csv
4	68	7	-1	1200	r068_visar_1.csv	r068_visar_2.csv	r068_xrd_1.csv	r068_xrd_2.csv	r068_diff_1.csv	r068_diff_2.csv
5	70	9	-1	1690	r070_visar_1.csv	r070_visar_2.csv	r070_xrd_1.csv	r070_xrd_2.csv	r070_diff_1.csv	r070_diff_2.csv
6	71	6.5	-1	2160	r071_visar_1.csv	r071_visar_2.csv	r071_xrd_1.csv	r071_xrd_2.csv	r071_diff_1.csv	r071_diff_2.csv

Figure 5.3: A breakdown of how columns in the `data.csv` relate to views in *Cinema:Bandit*. Column headers correspond to the type of data to be visualized and entries point to file names containing data values or images.

to the run-specific file for these columns. In Figure 5.2, notice that the parameters are specified first and the data files are listed last. A blue highlight shows how one row in the CSV file points to many files in the Cinema database folder structure.

Two run-specific line graphs or images can be defined in the Cinema database's `data.csv` file via the `FILE` column headers. The format is as follows: `FILE_Title_Index` (*e.g.* `FILE_Diffraction_Graph_1`). Columns with the same title will be grouped in the same view. For example, the Diffraction Graph view in Figure 5.1 uses an index of 1 to represent the static XRD pattern (blue) and 2 to represent the dynamic XRD pattern (red). This is the mechanism to create new views in the left-hand panel of *Cinema:Bandit*.

View Input Format

Graph Data Files. *Cinema:Bandit* will draw superimposed line graphs if the value in a `FILE` column references a CSV file. The application will assume that the data is stored as a table of numerical data with two columns representing the x and y dimensions of a 2D line graph. The CSV files do not contain headers.

Image Files. *Cinema:Bandit* will draw images if the value in a `FILE` column references an image file. The resulting images need to be stored as common Internet image file types, such as `png`, `jpg`, or `gif`.

Visualization Features

Filter and Select. The parallel coordinate plot shows a multi-dimensional view of the parameter relationships between runs, and users can filter results by interactively selecting valid parameter axis regions for all dimensions on this plot. The line graph views show a superposition of curves from the selected regions in the parallel coordinate plot. Users can make an instance selection by either hovering over or clicking a line in the parallel coordinate plot or in a line graph view. Both the parallel coordinate plot and graph view highlight the selected instance (*i.e.* the red and blue curve in the line graph view from Figure 5.1). The image views display two images that represent the selected instance for side-by-side comparison, and an info panel also appears, showing the selection's detailed parameter information. This unique visualization format allows users to interactively explore multiple data types through a single tool, while maintaining the context of their data through the parallel coordinate plot.

Multiple Linked Views. The filtered results and selection are linked across views making it easy to see parameter and disparate data relationships.

Pop-up Display for Parameters. When a run is selected, its data are highlighted in all linked views, including the parallel-coordinates plot, as shown in Figure 5.1. A pop-up window also appears, as shown in the bottom right corner of Figure 5.1, which allows the analyst to view the precise values for each parameter for the highlighted run.

Level of Detail. *Cinema:Bandit* includes toggle zooming by positioning the mouse at the zoom location and moving the mouse scroll button up or down. Once zoomed in on an area, the user can pan by clicking and dragging the view. Figure 5.4 shows three examples of zooming using *Cinema:Bandit.* Notice that for the image view, the images are coordinated such that they are zoomed into the same location for comparison.

Data Refresh. *Cinema:Bandit* refreshes automatically based on changes to the Cinema database. The application will monitor the `data.csv` described in Section 5.2.1 every few seconds for file size changes. The file size is monitored to reduce overhead and increase interactivity. For example, if a new run is added, the application will detect it

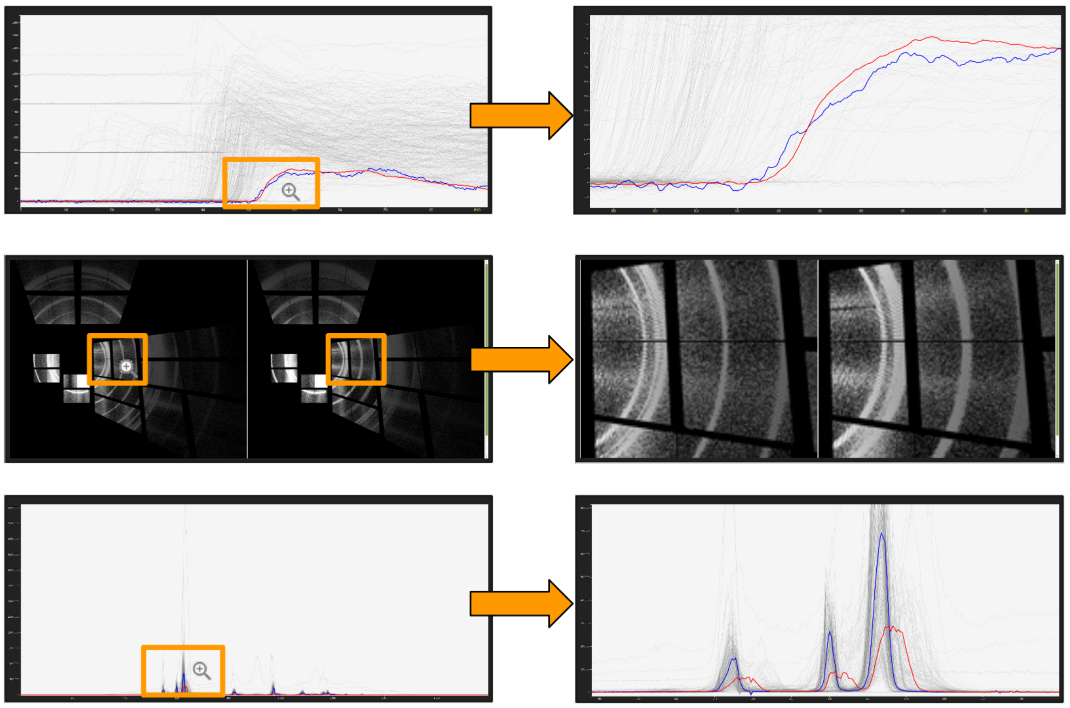


Figure 5.4: *Cinema:Bandit* provides a zooming capability that allows the user the examine finer details of their experimental results. Note that with diffraction images, views are linked so a zoom into one region provides the equivalent zoom in both images.

immediately and add the new run to the visualization. The current filter and selection will remain when a change is detected and added to the visualization, allowing the user to continue to investigate their current query. Occasionally, changes are made to the `data.csv` file that do not affect the file size. Therefore, periodically the application will load in the `data.csv` file to determine if there are any subtle differences. This feature is key to providing automated visualization updates in a workflow.

5.2.2 LCLS Workflow Implementation

Cinema:Bandit was integrated into an automated workflow for a shock physics experiment at the LCLS beamline facility. Figure 5.5 shows a high-level overview of our workflow where a single control script, using LCLS's data analysis software called psana, executes the data-processing workflow that includes three distinct steps. The left side of the figure shows the incoming raw data being distributed to three data analysis modules that perform data-processing and analysis tasks. Our workflow includes modules for extracting and recording the parameters of an experiment, processing velocimetry data, and processing XRD data. These modules are further described in Sections 5.2.2, 5.2.2, and 5.2.2. The output of each module is saved in text files and images, and then appended to a Cinema database, pictured in the center of Figure 5.5. In this case, there are four sets of files: the `data.csv` file which stores parameter values and file locations (described in Section 5.2.1), a set of text files for the velocimetry profile data, a set of images for the XRD image data, and a set of text files for the radially-integrated XRD patterns. Finally, the right side of Figure 5.5 shows visualization of the data using *Cinema:Bandit*.

Parameter Module

For LCLS, we used the psana [123] package to build a Python script that queries parameter values and saves them to the `data.csv` file. Common parameters that we have queried from the LCLS data systems are X-ray photon energy, delay between optical laser and X-ray pulse, and optical laser pulse energy. Similarly, the user can query and append parameters from local or shared spreadsheets (*e.g.* Google Sheets), including text descriptions or notes. The shared spreadsheet is continuously downloaded and

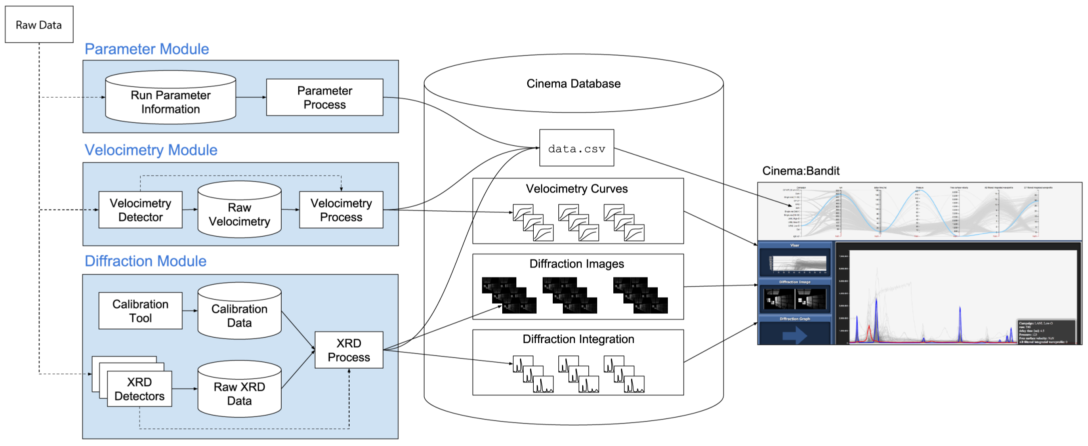
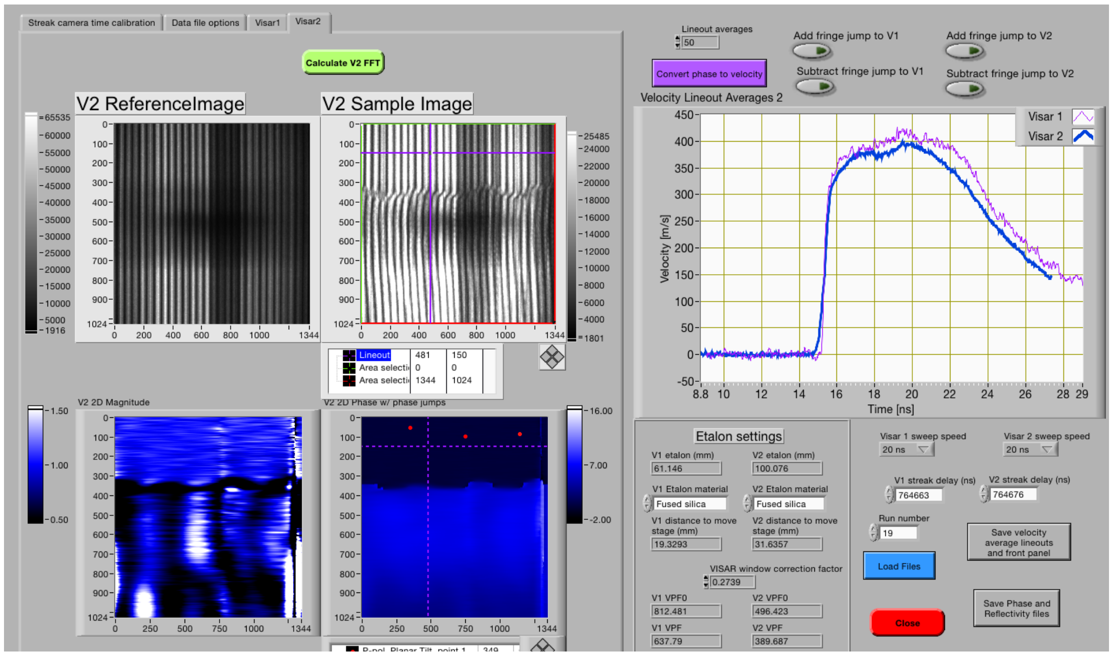


Figure 5.5: Data processing workflow for analysis of shock experiments and subsequent visualization with *Cinema:Bandit*. When a new run executes, raw data is read, processed, and saved into a Cinema database. The processed data are then visualized using *Cinema:Bandit*.

used to update the `data.csv` file, which facilitates a collaborative workflow and allows multiple users to concurrently update run-specific parameters.

Velocimetry Module

The optical VISAR diagnostic [97] measures the interface velocity of a sample. These data allow inference of material changes (*e.g.* elastic compression, plasticity, and phase transitions) by probing the bulk volumetric response during shock loading [93]. In the Velocimetry Module, a LabVIEW executable integrated with Python automates the processing of VISAR data to produce velocimetry profiles. The graphical user interface (GUI) is shown in Figure 5.6. The panels titled Visar1 and Visar2 in Figure 5.6 contain images of the spatial dependence of the material interface motion. For the Visar1 and Visar2 panels, there is one reference image when the shock has not arrived yet (top left) and the shocked sample image (top center). This LabVIEW executable generates the initial profiles, which then require the detection of the shock wave arrival (*i.e.* the abrupt rise in the velocimetry profile) and phase matching for the finalized velocimetry profiles. We employ a Python-based module that determines the discontinuity in the


Streak camera time calibration
Data file options
Visar1
Visar2
Calculate V2 FFT
V2 ReferenceImage
V2 Sample Image
V2 2D Magnitude
V2 2D Phase w/ phase jumps
Convert phase to velocity
Add fringe jump to V1
Subtract fringe jump to V1
Add fringe jump to V2
Subtract fringe jump to V2
Velocity Lineout Averages 2
Velocity [m/s]
Time [ns]
Visar 1
Visar 2
Load Files
Close
Save velocity average lineouts and front panel
Save Phase and Reflectivity files

Lineout averages: 50

Lineout	481	150
Area selecti	0	0
Area selecti	1344	1024

Etalon settings

V1 etalon (mm): 61.146
V2 etalon (mm): 100.076
V1 Etalon material: Fused silica
V2 Etalon material: Fused silica
V1 distance to move stage (mm): 19.3293
V2 distance to move stage (mm): 31.6357
VISAR window correction factor: 0.2739
V1 VPF0: 812.481
V2 VPF0: 496.423
V1 VPF: 637.79
V2 VPF: 389.687

Visar 1 sweep speed: 20 ns
Visar 2 sweep speed: 20 ns
V1 streak delay (ns): 764663
V2 streak delay (ns): 764676
Run number: 19

Figure 5.6: Screenshot of the automated VISAR analysis tool for velocimetry extraction.

profile due to the arrival of the shock wave and then searches for the best-matching phase differences.

X-Ray Diffraction Module

The spatial locations and orientations of the X-ray detectors must be calibrated before they are processed to produce images and radially-integrated diffraction patterns. Experiments at LCLS typically use several X-ray detectors, which are calibrated and assembled into a single image. In order to automate the geometric projection and aggregation of the detectors, our LCLS workflow contains a Diffraction Module as depicted in Figure 5.5. A Python executable within the module transforms image data from a set of area detectors to a plane perpendicular to the beam path, and then writes the image and diffraction pattern to disk. This executable makes extensive use of tools from the open-source PyFAI [124] package. The ensemble of images and diffraction patterns from experiments are displayed in *Cinema:Bandit* as shown in Figure 5.1.

Timing

The LCLS workflow described in this section has successfully been used during several experiments in the MEC hutch at the LCLS for real-time data processing and visualization. In each experiment, there was ∼7 minutes between shots, *i.e.* the entire data-processing workflow for a shot should be completed within 7 minutes. The time until the data files are visible, or can be queried and accessed using the psana interface LCLS, is variable, but typically the data are available within 10-30 seconds of the shot. In the described workflow, the automated parameter, velocimetry, and XRD modules (Section 5.2.2) were serially executed. The automated parameters module completed in $<$ 5 seconds, the velocimetry module completed in $<$ 5 seconds for a time series of 1000 timesteps, and the XRD module completed in $<$ 5 seconds for an array of two CSPad [125] detectors each taking 1940x1942 pixel images. The auto-refresh rate, described in Section 5.2.1, for the *Cinema:Bandit* HTML page was set to 5 seconds. Therefore, the latest shot parameters, velocimetry, XRD images, and the XRD patterns were visible in *Cinema:Bandit* within 50 seconds of the shot (or 20 seconds after the data was available). This timescale was acceptable based on our shot rate, but for faster data acquisition rates, additional optimizations could be made, *e.g.* execute modules in parallel.

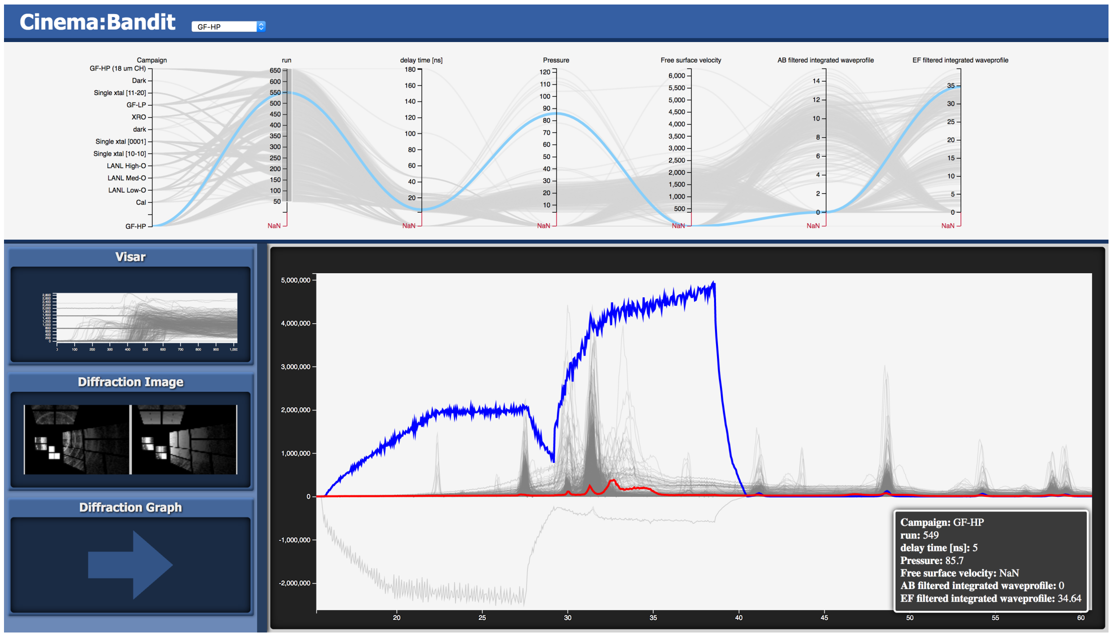
Aside from reducing the time from shot to visualization, the implementation of an automated workflow frees two additional analysts during the experiments. Prior to the implementation of the workflow, the velocimetry and XRD analyses used GUIs for the tools described in Section 5.2.2. A dedicated person on each tool could load, analyze, and visualize the latest shot within 3-5 minutes, leaving approximately 1-3 minutes to interpret the results before the next shot.

5.3 Results

The web-based visualization tool *Cinema:Bandit* (Section 5.2.1) and the workflow tools (Section 5.2.2) have been used to process and visualize data from a titanium experiment at the LCLS. Titanium is a constituent for many alloys used in demanding engineering applications because it is both strong and lightweight. An experiment was run at the LCLS to understand the elastic response, plastic flow, and phase transitions of titanium via the shocked response from pure and alloyed titanium. The purpose was to determine the relative contribution of defects and impurities to the dynamic response at high pressures. Here, the experimental samples consisted of polycrystalline or single crystal titanium foils, and the shock compression was created using an optical laser. The dynamic response was examined while varying the stress (by adjusting the laser energy), varying the delay between the optical laser and the X-ray pulse, and varying features of the initial material. Velocimetry data was collected using a VISAR system and XRD images were collected with an array of CSPad [125] area detectors at the Matter in Extreme Conditions [90] instrument at LCLS. In this section, we use this titanium dataset to demonstrate how the visualization tool *Cinema:Bandit* can be used to identify outliers or problematic data, direct experimental design, and compare ensembles of samples.

5.3.1 Identify Outliers or Problematic Data.

The axes in the parallel coordinate plot of Figure 5.7 display information about the material tested (“Campaign”), run number, delay time, pressure, free surface velocity, and information about the optical laser energy. To demonstrate the capability of *Cinema:Bandit* to identify problematic data and outliers, Figure 5.7 shows the integrated XRD data for many experiments. There are two plots that are visually easy to identify as outliers — one is highlighted in blue and the other one lies below the baseline. These outliers were the result of the baseline detector readout being suddenly influenced by the strong electromagnetic pulse that is generated when the high energy optical laser hits the target near the X-ray detectors. These effects are intermittent, which is why they were only observed on a few of the shots. By using *Cinema:Bandit*, these outliers can be easily identified, and the parameters for these runs evaluated through the interface. These samples can be repeated with the same parameters, which would improve the completeness of the dataset for subsequent analyses. Due to the modular structure of *Cinema:Bandit*, one could envision incorporation of statistically-based error-checking algorithms in future versions.



Campaign: GF-HP
run: 549
delay time [ns]: 5
Pressure: 85.7
Free surface velocity: NaN
AB filtered integrated waveprofile: 0
EF filtered integrated waveprofile: 34.64

Figure 5.7: Outliers where the baseline of the X-ray detector created errors in the data. The blue line in the Diffraction Graph shows where the detector readout after the shot is erroneously being influenced by a strong electromagnetic pulse. The red line in the same graph shows the static integration curve before the shot.

5.3.2 Direct Experimental Design.

The parallel coordinates plots along the top of the *Cinema:Bandit* display window allows filtering and exploration of the scalar or text values associated with an input or output of the experiment. A major utility of the parallel coordinates is that the values displayed can be gathered from a variety of sources, and they are added to the database to facilitate quick display and interaction. Two examples are shown in Figure 5.8 of how the parallel coordinates display can assist in experimental design. Figure 5.8(a) shows plots where the line plots have been filtered to a single type of starting material on the first parallel coordinate axis (Campaign = 'GF-HP'). On the middle axis (Pressure), it is clear that for this starting material, in the pressure region of $40 - 85$ GPa, data is absent. *Cinema:Bandit* makes such observations easy to visualize, allowing the user to quickly determine that data in this range of conditions was missing and allowing the experimental plan to be altered accordingly. Figure 5.8(b) shows a different example in which the filter on delay time was set to values > 11 ns. On the center axis (Pressure), it is clear that there are no data at these late delay times for pressures > 32 GPa. Since the objective of the experiment included examination of the material behavior as a function of delay time and pressure, the experimenter might choose to revise the experimental plan to fill in data in this region.

This illustrates how *Cinema:Bandit* can be used to quickly identify and refine the parameter ranges to ensure adequate sampling to achieve the scientific objectives.

5.3.3 Compare Ensembles of Samples.

For efficiency two separate titanium experiments were being run at the same time (runs labeled 339-360 in Figure 5.9) using a single material type (Campaign = GF-HP). The odd-numbered shots had pressures between 4.2 GPa and 13.7 GPa while the even-numbered shots had pressures between 21.9 GPa and 31.2 GPa. Figure 5.9 shows the visualization tool for all, odd-numbered, and even-numbered runs. It is clear from the velocimetry view in Figure 5.9 that the two sets of shots exhibit different behavior and that it is useful to visualize them separately. *Cinema:Bandit* allows users to open up several applications by opening up multiple tabs in a web browser application. Each tab can have different parallel coordinate plot filters, allowing users to monitor and explore

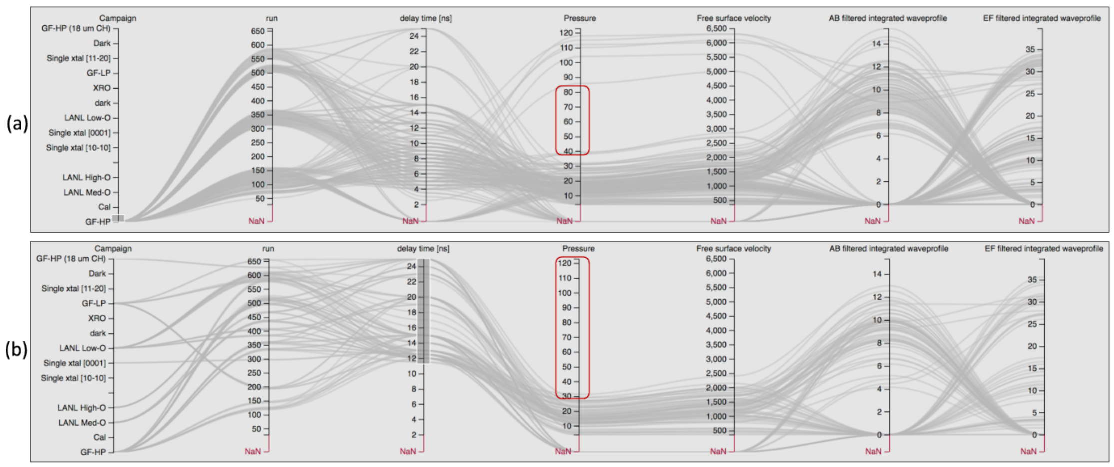


Figure 5.8: Parallel coordinates region of the *Cinema:Bandit* display showing examples of how the parallel coordinates can be used to assist experimental design. (a) Data was filtered to show a single material type (GF-HP). On inspection of the center axis (Pressure), it is apparent there are no data in the pressure range of $40 - 85$ GPa (highlighted in read). (b) Data was filtered by delay time > 11 ns. The center axis shows that there are no data at pressures > 32 GPa (highlighted in red).

multiple experiments at the same time.

5.4 Discussion

Cinema:Bandit helps us understand how scientists can combine ensemble visualization with human intelligence to explore multi-dimensional spaces. With this tool, users can filter and make important experimental decisions based on existing knowledge across multiple criteria and multiple output types. We now have an example that is multi-dimensional (C4), multi-variate (C2), and whose number of instances continues to grow larger (C3).

This chapter has considered how to show these large number of instances together (Q1) using linked views with traditional ensemble visuals (e.g. superposition, juxtaposition, and parallel coordinate plots). *Cinema:Bandit* touches upon understanding high-dimensional data (Q2) by allowing the human visual system to see and compare output curves in light of the input parameters. For example, user can quickly find and filter outliers (Section 5.3.1) while making decisions about the sampling gaps in the experimental

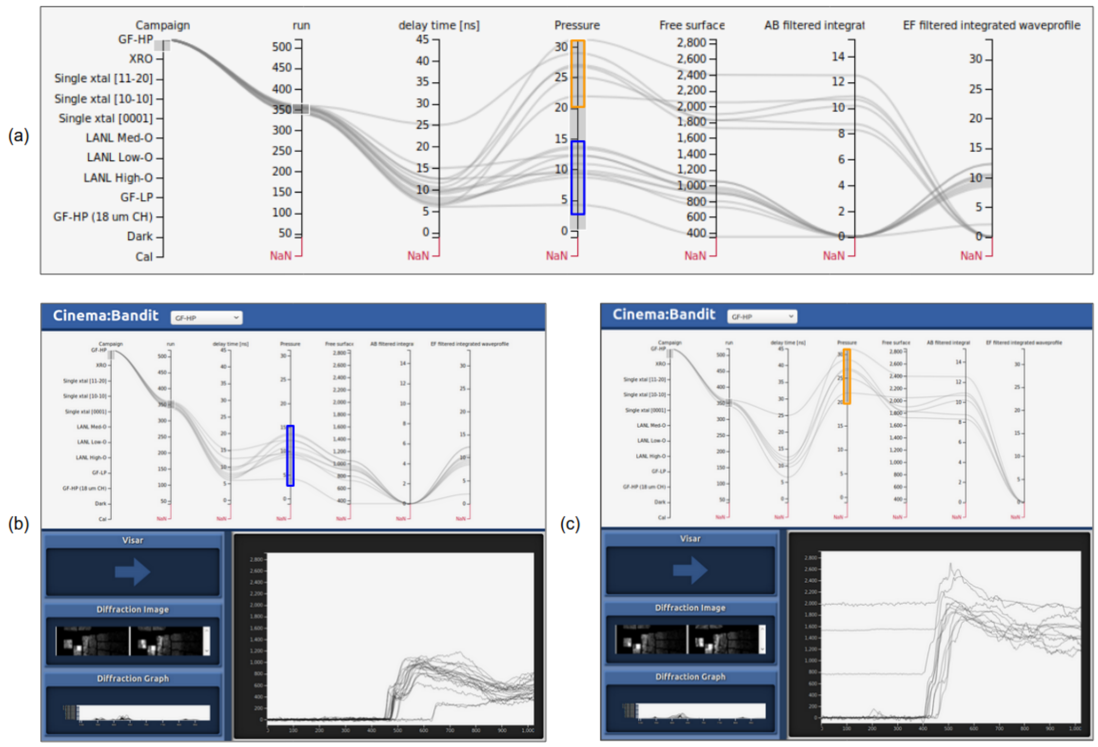


Figure 5.9: Example of separate studies being run at the same time using a single material type (GF-HP, runs 339-360). (a) *Cinema Bandit* shows the studies together with both low (blue) and high (orange) pressures. Alternatively the studies can be viewed separately by filtering out low (b) and high (c) pressures in two different browser windows.

design (Section 5.3.2).

Although useful for steering experiments, this approach relies heavily on human intelligence and the human visual system. The interface itself does not show the user the overall patterns in the parameter space, but rather the user needs to find them. This may become more difficult as the number of samples grow due to visual clutter and the parameter space complexity. In addition, the interface focuses on filtering a data set by input parameters, which may not always be the best approach intuitively. For example, users may want to establish a connection between the input parameters and the output curves (i.e. “Which input parameters cause this feature in the output?”) (Q3). It is not clear how to ask these kinds of questions using *Cinema:Bandit*.

In the next chapter, we look at how to understand relationships between the input parameters and the output data (Q2) based on interactive user-defined queries in both spaces (Q3). We also use prediction in the context of local uncertainty and the ensemble (C5), allowing users to ask qualified custom “What if?” questions. The next approach focuses more on multi-dimensional similarity than filtering individual input parameters.

5.5 Relevant Publications

- D Orban, D Banesh, C Tauxe, CM Biwer, A Biswas, R Saavedra, C Sweeney, RL Sandberg, CA Bolme, J Ahrens, D Rogers. Cinema: Bandit: a visualization application for beamline science demonstrated on xfel shock physics experiments. *Journal of Synchrotron Radiation*, 27(1), 2020. [126]

5.6 Acknowledgements

This work was performed at Los Alamos National Laboratory and was funded by LANL Laboratory Directed Research and Development project 20170029DR. LANL is operated by Triad National Security, LLC, for the National Nuclear Security Administration of the U.S. DOE under Contract No. 89233218CNA000001. We thank the LCLS-LA11 experiment team and the LCLS-Matter in Extreme Conditions instrument for the use of data in developing this workflow and software.

Chapter 6

Application Context 2:

Drag and Track - Contextualizing Shock Physics Simulations through Direct Manipulation and Dimensionality Reduction

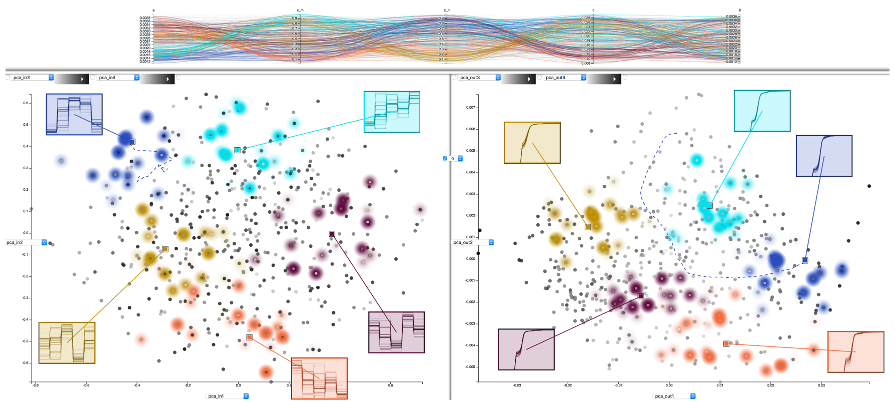


Figure 6.1: Our parameter space exploration tool allows users to query linked dimensionally reduced spaces by interactive clicking and dragging virtual data instances. These are visualized via interactive callouts that can be directly manipulated to search and display the nearest ensemble instances. The linked parallel coordinate plot allows for filtering the ensemble and displays trends of selected parameters based on the user-defined annotation.

In this chapter we investigate how to understand multi-dimensional parameter spaces

(Q2, C4) using custom “What if?” questions (Q3) for simulation ensembles that have many instances (Q1, C3). We are interested in analyzing these questions in the context of local uncertainty and in relation to the ensemble globally (C5). Here we find our second example DDEI, Drag and Track, pictured in Figure 6.1. The interface focuses on scalable approaches for understanding multi-dimensional data by analyzing the gaps in the global context using prediction validated within local uncertainty contexts.

A major missing piece of the DDEI, meaningful feature interaction, has been added from the previous chapter. Although it was possible to filter by input parameters and well defined output data properties, the goal is to understand the features in the high-dimensional output space. For example, a feature in the shock physics application is the height and slope of sections inside the velocimetry and diffraction curves. Here a feature is more than a single number, but rather a set of localized values. Unfortunately, these are not well defined, but experts can make sense of the visual, and they are perhaps more important in understanding the dynamics of the system. In this chapter we enable users to search in both the input and output domains by simply clicking and dragging ill-defined features around.

To accomplish this visually, we take advantage of dimensionality reduction, a mechanism that translates high-dimensional structures into lower dimensional similarity measures [44, 40, 70] (i.e. a 2D or 3D representation that can be visualized spatially). Intuitively, this means that instances that are close to each other (e.g. points on a scatter plot) are similar, and therefore related. We use the direct manipulation of virtual data instances (described in Section 6.2.1) to understand how dimensionally reduced sub-spaces (e.g. input and output spaces) relate to each other. This gives insight into how the parameter space can be annotated and analyzed using expert intuition and knowledge.

6.1 Motivation and Application Background

Simulation has become a vital part of the scientific exploration process, often replacing expensive and sometimes infeasible experiments. Models represent physical phenomena and can produce large sets of results, providing more data than ever before. Ensembles, concrete data sets whose instances are each characterized by a unique set of parameters [10], are widely used for visual analysis tasks including evaluating model uncertainty [11, 12, 13, 14], discovering trends [15], and making predictions [16, 17, 18].

Recent advances in interactive ensemble visualization tools have enabled new exciting modes for subject matter experts to explore their data. Mechanical engineers can directly manipulate 3D models to search a design space [22] and smoothly interpolate [47] between data instances. Flood management experts interactively make decisions by steering fluid flow simulations through multiple branching flood scenarios [63]. Graphics artists can inversely design animations by choosing their desired output from a set of clustered visuals [59, 37]. Scientists use dimensionally reduced (DR) linked views and interactive query widgets to annotate, cluster, extract features, explore, and modify their high-dimensional data [69, 42, 127, 128, 53]. Even in our daily lives, weighted views of ensembles can inform common, but difficult, multi-criteria decision making tasks like "Which car should I buy?" [129]

Although much work has been done, visualizing ensembles is still an open problem, studied recently by many researchers in the visualization community. The call to action has only increased along with the demand to explore parameter spaces representing real-world scientific problems. Consider experiments in shock physics [130] as discussed previously in Chapter 5, where scientists endeavor to understand complex deformation properties of material under extreme thermodynamic conditions. Compressed gas is released from a gas gun to accelerate a projectile down a barrel, which collides with a target composed of a material of interest. The resulting shock wave traveling through the target produces time-series rear-surface velocity curves, also called velocimetry profiles. These profiles provide valuable feature information regarding the elastic and plastic material properties.

Shock physics experimental techniques are about to become orders of magnitude faster, enabling the collection of extremely large amounts of parameterized data [94, 95, 96]. Currently scientists determine which tests to run based on experience and knowledge, but this will be inadequate as the next generation of experimental facilities come online. Instead of planning the next experiment, the scientist will be asked to plan the next hundred experiments. This is especially hard when scientists are unsure which questions are worth exploring in order to maximize the knowledge of a system while minimizing expensive experimental overhead. Fortunately, simulation models can help sample the space of possibilities and provide insight into the unknown by predicting velocimetry profiles. The hope is that material scientists can use a simulation ensemble to ask questions like, “What happens to this elastic feature if this subset of input parameters is explored?”, “Which input parameters reduce the uncertainty in the plasticity?”, or “What experiments should I run to reproduce this velocimetry curve?”

Our work is motivated by the modern approach for discovering information epitomized by the popular phrase, “Google it.” Users simply type their question into a search engine as a text query and results ordered by relevancy are returned. The experience is simple, organic, and does not require an in-depth knowledge of the intrinsic structure required by most database engines. Another important feature that relevancy searches commonly employ is the fact that queries can be inaccurate. These systems often correct spelling errors or suggest similar queries (i.e. “Did you mean to search for ...?”).

Imagine a visual search engine that allows users to find data instances in an ensemble by simply showing the search engine what they are looking for. In the shock physics application described above, a scientist may have a hypothesis that a certain velocimetry profile is created when parameter x is high and parameter y is low. To find out whether this is a reasonable assumption, the scientist can use the visual search engine to retrieve similar data instances by suggesting an approximate velocimetry curve and a high x parameter. The results can be analyzed to see if, in fact, y is statistically low. In order to change the query, the user simply changes the curve. One way this is done for interactive visualizations is to use direct manipulation techniques, which allow users to directly modify data through representative visuals [67, 68, 22]. These organic and qualitative scientific inquiries motivate interactive solutions to the following three general ensemble visualization problems:

Problem 1: Can forward and inverse prediction approaches be combined into one approach that uses both input and output parameters? Ensembles are calculated from an input-output model, a function that maps input parameters to output parameters [32], which is typically a simulation model. In relation to inputs and outputs, research focuses on either 1) forward approaches [47, 63], which use the input space to predict results and determine sensitivity, or 2) inverse approaches [59, 37], which use output parameters to fit input parameters. Similar to just a few systems that have been presented thus far in the literature [22, 18, 66], we ask whether forward and inverse approaches can be combined to creatively search parameter spaces.

Problem 2: High-dimensional projections are hard to interpret. Simulation data sets usually contain many input and output parameters. Due to the physical limitations of displays and our own visual system, it can be challenging to effectively visualize data that has greater than two or three dimensions [24]. One way to visualize these complex multi-dimensional spaces is to use dimensionality reduction techniques, which project high-dimensional data into fewer dimensions that more adequately represent the intrinsic structure of the data. Unfortunately, there is a known "trade-off between the interpretability of the axis and intrinsic structure captured by dimensionality reduction methods." [24] In other words, the visual representation of these dimensionally reduced spaces are usually hard to understand.

Problem 3: Estimation methods produce uncertain results. Scientists would like to interactively view and visually understand the gaps in continuous spaces without the expensive computation time required by simulation. Fortunately, estimation methods are able to predict results using the statistical properties of ensembles. However, ensemble parameter spaces are often sparse and statistical phenomena like the sampling density can cause uncertainty [18]. Therefore, estimates need to be displayed in the context of the ensemble for validation. This is especially true for inverse problems, which try to determine the input from the output. For these problems prediction actually represents a probability distribution and there is no "best" estimate, but rather multiple possible estimates [26]. How should we evaluate and adequately use estimates as they relate to navigating sparse parameter spaces?

Although simple and intuitive, there is one fundamental flaw with the search engine approach described above: users need to know what they are looking for. This assumption is not always true if the user is in exploration mode. What does it mean to search for "something interesting"? Often, models like those in the shock physics application, are non-linear and simple parameter relationships can be difficult to discover. Visual analytics approaches like dimensionality reduction interaction techniques help visualize the ensemble as a whole and understand trends in the data [69]. In the literature this navigation strategy difference is described as *local-to-global* vs. *global-to-local* [32]. We can either start from a data instance to find others (local-to-global) or start with an overview and filter down to relevant data instances (global-to-local). We work towards addressing visualization problems 1 and 2 above by tightly integrating these two navigational strategies.

As problem 3 states, estimates may not be completely accurate, however, their utility in navigating a continuous space is still valid when searching for similar instances. Queries for similarity searches can be approximate. In our approach, we use estimates as initial guesses and allow the user to refine them through direct manipulation. The similarity search provides the actual valid data instances that the user is looking for, which are accurate. Naturally, the returned data instances represent an uncertainty distribution that can be used to evaluate the prediction.

In this chapter we present Drag and Track, a direct manipulation interface for contextualizing ensemble data instances within a continuous parameter space. We combine direct manipulation search techniques (drag) with ensemble dimensionality reduction methods to annotate (track) parameter changes as an ensemble is being explored. Our approach uses multiple virtual data instances (VDI) to ask "what if" questions. These user-defined virtual data instances have the same parameters as regular data instances, but also can be edited. Ensemble data instances along with VDIs are spatially positioned in dimensionally reduced (DR) linked views. VDIs are visualized as interactive callouts, which visually represent the data instance parameters. When the parameters in these callouts are directly manipulated, the underlying VDI is updated and acts as input to a similarity search that returns the nearest ensemble neighbors. The relevant data instances are highlighted, allowing users to annotate the low dimensional space and therefore contextualize data instances based on user-defined goals. As interactive callouts are manipulated, the VDI's movement is visualized as separate paths on the dimensionally reduced spaces, therefore tracking parameter sensitivity across the spaces. We apply our approach to a real-world shock physics application, and evaluate it based on expert material scientist feedback.

6.2 Visualization Techniques and Implementation

6.2.1 Overview, Example Problem, and Definitions

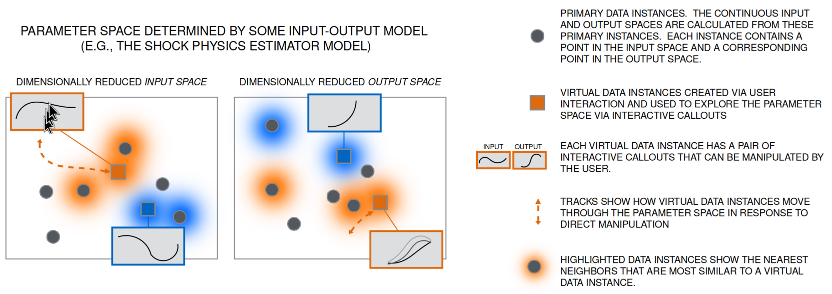

PARAMETER SPACE DETERMINED BY SOME INPUT-OUTPUT MODEL
(E.G., THE SHOCK PHYSICS ESTIMATOR MODEL)
DIMENSIONALLY REDUCED *INPUT SPACE*
DIMENSIONALLY REDUCED *OUTPUT SPACE*
PRIMARY DATA INSTANCES. THE CONTINUOUS INPUT AND OUTPUT SPACES ARE CALCULATED FROM THESE PRIMARY INSTANCES. EACH INSTANCE CONTAINS A POINT IN THE INPUT SPACE AND A CORRESPONDING POINT IN THE OUTPUT SPACE.
VIRTUAL DATA INSTANCES CREATED VIA USER INTERACTION AND USED TO EXPLORE THE PARAMETER SPACE VIA INTERACTIVE CALLOUTS
INPUT
OUTPUT
EACH VIRTUAL DATA INSTANCE HAS A PAIR OF INTERACTIVE CALLOUTS THAT CAN BE MANIPULATED BY THE USER.
TRACKS SHOW HOW VIRTUAL DATA INSTANCES MOVE THROUGH THE PARAMETER SPACE IN RESPONSE TO DIRECT MANIPULATION
HIGHLIGHTED DATA INSTANCES SHOW THE NEAREST NEIGHBORS THAT ARE MOST SIMILAR TO A VIRTUAL DATA INSTANCE.

Figure 6.2: Diagram explaining how direct manipulation is tracked on dimensionally reduced views. Virtual data instances are displayed as linked interactive callouts that are positioned in both the dimensionally reduced input and output spaces. After dragging to manipulate a parameter via the interactive callout, the corresponding VDI points move in both spaces and a track line shows how they travel through the space.

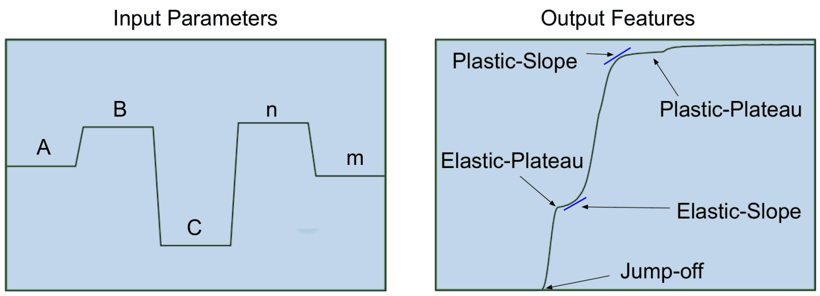


Figure 6.3: A visual representation of the input and output of a shock physics simulation. The input parameters (left) are the A,B,C,n, and m parameters in Equation 6.1, and the output parameters (right) is the velocimitry profile.

The Drag and Track framework allows users to creatively annotate a set of dimensionally reduced spaces through user-defined virtual data instances. Our approach requires four pre-conditions:

1. An input-output model, typically a simulation model, that defines a set of parameterized simulation runs we call data instances. The ensemble being studied is a set of data instances.

2. A set of dimensionality reduction (DR) methods that describe the spaces to explore. For example, in our analysis we use principal component analysis (PCA) and multidimensional scaling (MDS) as DR methods.

3. A distance metric between data instances and a similarity search method, both used for finding nearest neighbors in an ensemble.

4. An estimation method, which takes a subset of input and output parameters and returns a predicted data instance.

Example Input-Output Model. Figure 6.3 shows an example parameterization of a shock physics simulation using the Johnson-Cook strength model [131] given as

$$Y = \left[A + B\epsilon_p^n\right] \left[1 + C \ln\left(\frac{\dot{\epsilon}_p}{\dot{\epsilon}_0}\right)\right] \left[1 - \left(\frac{T - T_{\text{ref}}}{T_{\text{melt}} - T_{\text{ref}}}\right)^m\right], \tag{6.1}$$

The figure shows visual representations of a data instance for both input and output parameters. The left shows the input, where A,B,C,n, and m are the input parameters in Equation 6.1. The right shows the the result of the simulation as an output curve, along with the features the scientists are interested in. Users are interested in how the input parameters affect the features in the output curve and the inverse. For example, material scientists might want to know what happens to the elastic plateau as A is increased. We now describe the Drag and Track technique using the example in Figure 6.2.

Dimensionally Reduced (DR) Spaces. In the figure, we start with two dimensionally reduced linked views of the ensemble. The two views can be any dimensionally reduced spaces, however in this chapter we focus on the input and output spaces as example spaces that help describe our approach. These two spaces are conceptually concrete and are directly applicable to analyzing sensitivity in the shock physics application. Although our approach can scale to several DR spaces, we assume two for our examples.

Virtual Data Instances (VDI). The figure shows each ensemble data instance as a point in a scatter plot. Virtual data instances are shown as squares, and they represent user-defined data instances positioned in the DR space using each view's DR method. VDIs have the same parameters as regular ensemble instances, but their parameters are editable. Figure 6.2 shows linked interactive callouts that visually display a subset of

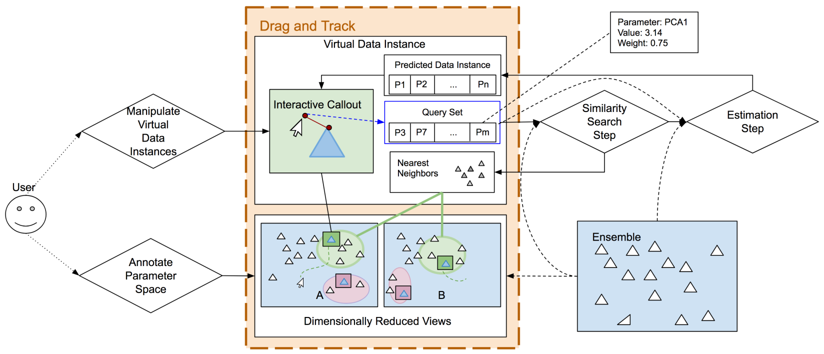


Figure 6.4: Overview of the Drag and Track framework as it relates to user interaction influence over the similarity search and data instance estimation processes.

VDI parameters that are relevant to the DR view. In this case, the callout in the input space shows the input curve, and the output callout shows the output curve.

Linked Interactive Callouts. Linked interactive callouts can be directly manipulated to change the parameters in the VDI they represent. For example, in Figure 6.2 the user drags the center of the input curve up, changing the VDI and the its location in both DR spaces. Finally, Figure 6.4 shows how multiple VDI's annotate the parameter space by highlighting similar instances in the DR spaces.

System Overview. The Drag and Track work flow tightly integrates user interaction techniques like direct manipulation with similarity search and data instance estimation techniques. Figure 6.4 shows the technical framework describing the visualization interface. Virtual data instances act as the glue between user interaction and back-end processes. Users can annotate DR spaces by creating VDIs, which in turn can be directly manipulated. VDIs hold query information related to parameters of interest, and when interactive callouts are moved or changed, the query is updated and a similarity search is invoked, which provides a set of nearest neighbors. Finally, as in Figure 6.4, an estimation step updates the VDI's predicted data instance, which affects the visual callout and the positions of the VDI in the DR spaces.

In the following two sections, we describe our technical framework, first from the user interaction perspective (Drag), and then from the visual analytics perspective (Track). Drag includes tasks like direct manipulation and positioning VDIs inside the DR views. Track involves calculating and displaying similarity search results with predictions to ultimately annotate the parameter space. After describing the general framework in Sections 6.2.2 and 6.2.3, we provide a concrete implementation in Section 6.2.4.

6.2.2 Drag: Direct Manipulation of Virtual Data Instances

The primary method for user interaction in the Drag and Track framework is to update virtual data instances. There are two modification modes:

1. Annotate parameter spaces by creating or dragging the VDI points on the dimensionally reduced space.

2. Directly manipulate callouts to change parameters in the VDI.

Although a VDI looks like a regular data instance, the data structure itself is a bit more complicated as detailed in Figure 6.4. A VDI contains a query set, a predicted data instance, and a set of nearest ensemble neighbors. The query set is a subset of input and output parameters that contain parameter values and weights. This data structure is used for the similarity search step and estimation step shown in the figure. The nearest neighbors are the output from the similarity search, and the predicted data instance is calculated from the estimation step. The predicted data instance has the same parameters as regular ensemble data instances. Interactive callouts display the components of the VDI, and the VDI positions on the DR spaces calculated from the predicted data instance and the user's interaction.

Every data ensemble instance and predicted data instance also contains a set of derived parameters that are calculated from projecting the data instance to the dimensionally reduced space. For example, a data instance projected to both the PCA input and output spaces would have four derived parameters, each representing one axis from the two 2D spaces. These parameters are available in the VDI's predicted data instance and can be used as weighted query parameters.

Annotate Parameter Spaces

Users can interactively annotate the parameter space by creating multiple VDIs. This is done by simply clicking on any DR space at the locations the user would like to annotate. As shown in Figure 6.2, the VDI shows up on all DR spaces at the calculated

VDI position (described below). Users can then move the VDI points around any DR space to see how movement in one space changes the position in another space. Figure 6.2 illustrates the tracked path that is displayed in both spaces when a VDI changes. These paths are calculated from iterative position estimates as users move points from their start to their goal. Annotation comes from the highlights provided by the calculated nearest neighbors that are displayed across all DR views. Users can customize the VDI through direct manipulation to creatively highlight points in the space that match their specific investigation.

When a point is created or moved, the two dimensions on the selected view are added to the VDI's query set. As described in Section 6.2.3, the similarity search and estimate steps update the VDI's nearest neighbors and predicted data instance respectively. The nearest neighbor highlights, VDI position, and visual callouts are then updated. Reprojection of the VDI to the lower dimensional views is handled by the estimation step described in Section 6.2.3. The predicted data instance includes these estimated projected axes values. Here the user's lower dimensional query parameter values (i.e. the location the user clicks) and the predicted values may not match, causing an ambiguity on where to position the VDI on the dimensionally reduced spaces. Our approach is to present both sets of values and allow the specific implementations of the Drag and Track framework to interpolate between them. This design decision is discussed in detail in Section 6.4.

Our framework also integrates a linked parallel coordinate plot as displayed in Figure 6.1. The parallel coordinate plot lines, each representing a data instance, will highlight with the color of the user-defined VDI if it is a nearest neighbor. This enables users to see parameter associations between VDI regions and how they relate to the lower dimensional spaces. The parallel coordinate plot also allows filtering of the data set. The VDI points then describe new highlighted nearest neighbors and the estimate will be based on the filtered set. As described in section 6.3.4, this type of filtering is important in helping scientists constrain the ensemble to investigate sub-areas in the parameter space.

Directly Manipulate Callouts

On each DR space, every VDI point also has a space specific interactive callout as shown in Figure 6.2. The callouts are linked together across spaces through the VDI. These are different view related pictures of the ensemble parameters. Interactive callouts can use any information from the VDI to visualize the data including the query set, the predicted data instance, and the nearest neighbors. For example, referring to Figure 6.3 for the shock physics application, we display estimated inputs as step lines on the input callout and show the velocimetry profile curve estimate for the output callout. Since these are 1D curves, we also superimpose the nearest neighbors giving insight into the uncertainty in the distribution.

Interactive callouts are powerful because they tightly couple direct manipulation with the visualization. Like magic, users can drag a visual representation of a parameter and see the callout and the VDI position immediately change. For example, material scientists can click on the elastic plateau on the velocimetry profile and pull it up and watch other features like the plastic plateau change. In doing this, the user is asking the question, “What if the elastic plateau is higher?”

When a callout is directly manipulated, the parameter values related to the interaction are added to the query set along with parameter weights. Application specific callouts define how the visualization and interaction are related to the query set. In Section 6.2.4 we describe an example visual callout type and weighting scheme. As described in Section 6.2.2, a new similarity search is performed based on the updated query set and a predicted data instance is calculated. These feedback into the visualization loop by changing the VDI’s positions and refreshing its visual representation in the interactive callouts.

6.2.3 Track: Using Virtual Data Instances to Visually Analyze Parameter Spaces

The Track part of Drag and Track involves the back-end algorithms necessary to enable the visual analytics of the parameter space. Figure 6.4 shows the Track actions on the right side of the diagram and they include the Similarity Search Step and the Estimation Step. Both are necessary to enable the visualization piece described in Section 6.2.2.

Similarity Search Step

The nearest neighbors search is a vital part of our framework since the nearest neighbors provide the highlights for the annotation in the visualization. Differences in customized VDIs can be tracked across the DR spaces while providing uncertainty information about the specific query set and predicted data instance. Figure 6.4 shows the search step being invoked when the query set in a VDI is updated. After the nearest neighbors are updated, the estimation step is called.

The similarity search requires a query point, $q \in \mathbb{R}^p$, a set of data points, $S \subseteq \mathbb{R}^p$, and a distance metric, $d : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, to calculate similarity where p is the number of parameters being compared. The query set defined in the VDI has a list of parameters, their values, and their weights as shown in Figure 6.4. The size of the query set determines p, and the query set elements determine the parameters for the data points created from data instances. The query point, q, is created by taking each parameter in the VDI query set and listing its value. Similarly, a point for each data instance, $x_i \in Ensemble$, is created by listing the corresponding parameter values to create a point $s_i \in S$. A weight vector, $w \in \mathbb{R}^p$, is also created by listing the corresponding VDI's query set parameter weights defined by the application.

The distance function $d(A, B)$ can be any measure of similarity between these points. The nearest neighbors are calculated by using the distance function to determine the difference between every $s_i \in S$ and the query point, q, as $y_j = d(s_j, q), j \in \{1...n\}$, where n is the size of the ensemble. The points in S are sorted by the vector y and the top k are used as the new VDI nearest neighbors.

Estimation Step

Finally, the last step in the track process is to provide a predicted data instance based on the query set in the VDI. This is important because we use the predicted data instance to determine the positions of the VDI in the DR spaces. The prediction is visualized in the VDI callouts to provide a starting point for the direct manipulation described in Section 6.2.2. The tracked lines on the dimensionally reduced spaces are also calculated from estimate iterations as parameters are manipulated.

Estimation methods include regression, interpolation, statistical emulators, or machine learning methods. For example, a subset of ensemble visualization tools incorporate fast *surrogate models* [32], which combine the statistical properties of ensembles and estimation techniques to predict results from an input-output model [47, 22, 18, 17, 16]. In this case, a surrogate model is defined as $\hat{f} : A \to B$, where A is the input parameter set and B is the output parameter set, such that $\hat{f}$ approximates a more complicated function f representing the input-output model. We relax the requirement that an estimator must go from input to output by defining it as any function that takes a VDI query set and returns a predicted data instance, which includes both and output parameters.

Finally, the estimation step re-projects the predicted data instance onto the dimensionally reduced spaces using out-of-sample extensions [57, 66] of the existing DR methods. If out-of-sample extensions are not available for the DR methods, we use the estimated values in the predicted data instance. The end result is a predicted data instance that is added to the VDI, which can then be visualized and qualified by a nearest neighbor distribution.

6.2.4 Implementation

We implemented the Drag and Track framework to study the parameters of 1D output curves as in the shock physics input-output model. Two linked scatter plots display the dimensionally reduced views of the input and output spaces as shown in Figure 6.1. As described in Section 6.2.1, implementations need to first provide four components: an input-output model, a set of dimensionality reduction methods, a distance metric with a search method, and an estimation method.

Our implementation allows for any input-output model where the input is a discrete set of quantitative parameters and the output is a 1D curve. Interactive callouts are displayed as 1D curves representing the predicted input or output. Each point on the callout's horizontal axis represents a separate parameter. To facilitate direct manipulation, users can click and drag points on a curve up and down. When the user clicks on a point, the horizontal axis parameter is added to the VDI's query set and updated with the curve's value at that point. Dragging the point up or down also updates the value for the parameter in the query set. Since the output is a 1D curve, we superimpose the curves of the nearest neighbors to show an uncertainty distribution along with the more prominent prediction curve.

The implementation also provides both PCA and MDS dimensionality reduction methods for the estimation step to use. When a user clicks on a scatter plot, a VDI is created and the x and y values are added to the VDI's query set for the specific derived dimensionally reduced parameters described in Section 6.2.2. The similarity search step and estimation steps are invoked and the result is displayed as VDI points and visual callouts in both dimensionally reduced spaces. When a user drags a VDI point, the same DR parameters are updated in the query set, and the algorithm is repeated.

In our implementation, multiple parameters can be added to a VDI query set and their weights are systematically updated after each interaction. We consider the most recent interaction to have the most weight. After each manipulation, the weights of previously added parameters are reduced, allowing the VDI to be naturally constrained based on several user-interactions without over-constraining the parameters. Dragging a VDI point in a DR space is considered a more drastic change than manipulating one variable. In this case, we increase the dimensional axis parameter weights and substantially lower the existing weights on the other query parameters.

For our similarity search, we look for the top-25 neighbors by comparing each data instance point with a query point calculated from the VDI query set. The distance metric we used is the weighted Euclidean distance. For the estimation step, we use the multidimensional form of inverse weighted distance interpolation [132] using 5 neighbors. Since the similarity search step is already performed, we can use the inverse distances of the nearest neighbors that were calculated using the VDI query set. The result is a predicted data instance that uses a percent of each nearest ensemble data instance based on how close the query point is to the data point.

Finally, VDI positions are calculated based on the interaction type. This is to address the ambiguity of a VDI's position mentioned in Section 6.2.2. To avoid confusion, when a VDI position is directly manipulated, the VDI's position is set to the query set value, which represents the user's desired location. For all other interactions we use the VDI's predicted data instance value as its position. Our solution to this problem is discussed in Section 6.4.

To analyze the Drag and Track framework, we use our implementation to investigate two separate application domains. First we apply it to a real-world high-dimensional shock physics problem (Section 6.3). Here we provide a set of interactive tasks by giving an example work-flow. We then evaluate usefulness Drag and Track via known information and expert user feedback. For the second application, we analyze the effectiveness of using Drag and Track across several synthetic models (Section 6.3.5).

6.3 Results

Our Drag and Track solution was applied to a set of simulation reproductions of a shock physics experiment involving aluminum alloy Al-5083, specifically Shot 104S from Botler and Dandekar [130]. Aluminum alloys are used in many applications including baseball bats, transportation, tent poles, armored vehicles, and aircraft. Different elemental compositions of aluminum with other metals provide a variation of useful material properties. Due to its high strength and resistance to corrosion, Al-5083 is a preferred material used in many military and commercial applications including boat hulls, missile containers, military vehicles, and storage tanks for jet fuel [130].

In some thermodynamic conditions due to pressure, temperature or volume, Al-5083 may deform irreversibly, causing a plastic deformation. This makes the aluminum alloy not suitable for some applications, and therefore it is important to understand and predict the alloy's limitations. Shock physics experiments test these extreme conditions by using the pressure in gas guns to accelerate a projectile at the material of study. These experimental configurations are called shots, and their results can be analyzed to investigate the material. Simulations are used to try to approximate the experiment and understand which inputs might cause these plastic deformations.

Figure 6.3 shows the output for these experiments, called velocimetry profiles. The various features on the curve like the jump-off, elastic plateau, or plastic slope describe desirable and undesirable deformation properties. The input for the simulations represent variables in the Johnson-Cook strength model given as Equation 6.1. These variables can change the model to represent different thermodynamic conditions. The goal is to qualitatively understand the material properties of Al-5083 and the deformation causes based on the simulation model.

6.3.1 Example

This section describes an example usage scenario of our system and the tasks that would be used to evaluate its effectiveness. Sonja is a material scientist studying the material properties of aluminum with a new set of 500 simulations based on the Johnson-Cook model. She opens up our implementation of Drag and Track and loads her ensemble hoping to quickly understand some of the major parameter trends worth investigating (Work-flow shown in Figure 6.5). The first thing she does is try to explore the parameter space as a whole by clicking on seven different locations in the output space on the right (see Figure 6.5-(a)). Sonja observes that there are two major features that describe the output space, the elastic-plateau and the plastic-plateau. She also sees that, although there is a trend in the input space, the nearest neighbor highlights representing each VDI are very spread out, meaning that for each output curve there are multiple possible inputs. However, on the interactive parallel coordinates plot she notices that based on her initial annotation the A and m variable are both low for the output curve that is furthest right in the output space, implying there is an input parameter relationship for that curve.

Sonja finds one of the curves that look interesting and wonders how that curve's features are distributed throughout the spaces. She clears the screen and clicks the output space locating the curve of interest (see Figure 6.5-(b)-1). In order to understand how the elastic-plateau changes in both spaces, she clicks on the elastic-plateau on the visual callout and drags it up (see Figure 6.5-(b)-2). A path is interactively drawn across both the output and input spaces, informing her that the output DR space's left and right align with the height of the elastic-plateau. Similarly, she can observe that the input space is also correlated, but on a diagonal. Finally, she directly grabs the plastic-plateau and observes that a spatial relationship also exists in both spaces (see Figure 6.5-(b)-3). The highlighted data points are displayed in the input callout and the parallel coordinate plots, which show that the manipulated velocimitry curve is associated with a high C value and a low n value. This interaction can help her in the future when she is trying to fit an output curve generated from an experiment to the simulation output.

Sonja then wants to understand more deeply the sensitivity of her model for her original curve of interest. She clicks on the curve's position in the output space five

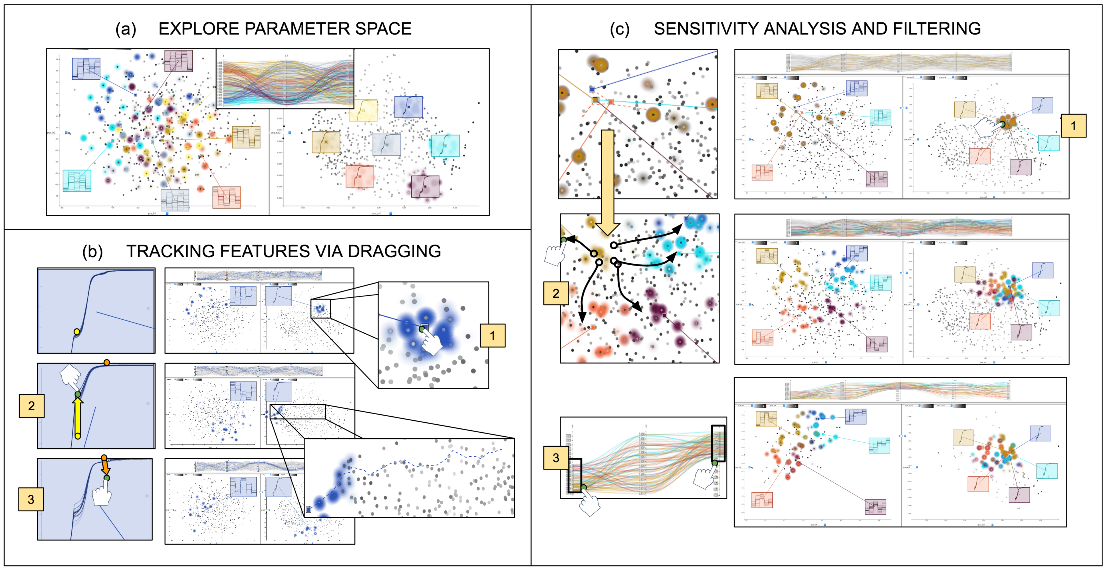

(a) EXPLORE PARAMETER SPACE
(b) TRACKING FEATURES VIA DRAGGING
(c) SENSITIVITY ANALYSIS AND FILTERING
1
2
3

Figure 6.5: Users interact with the system by (a) exploring the parameter space, (b) understanding features through direct manipulation, and (c) performing sensitivity analysis with filtering.

times and notices that the data instance highlights and the VDI positions are all the same in both spaces (see Figure 6.5-(c)-1). In order to sample the input space more completely, she moves the VDI points to several different locations in the input space (see Figure 6.5-(c)-2). This will give her an idea of the various categories of inputs that cause the specific output curve. From the visual callouts in the input space and the parallel coordinate plot, she notices two relationships immediately. First, all of the inputs show a fairly low A value, meaning that a low A is associated with the specific curve. Second, the bottom right callout in the input space, which has the highest set of A values, requires a low C value. Sonja decides that she would like to continue her investigation by using the parallel coordinate plot to filter out high A values and low C values (see Figure 6.5-(c)-3). Fortunately, her preferences were saved in the various VDI's, so the positions and highlighted instances are updated to fit her user-defined illustration. We now describe our results using data from a shock physics simulation.

6.3.2 Data

A shock physics simulation ensemble was generated using the hydrodynamics code FLAG developed at Los Alamos National Laboratory [133, 134, 135]. The data was computed using the Los Alamos National Lab's Moonlight supercomputing platform, which consists of 308 Intel Xeon E5-267 nodes: 4928 CPU cores. Besides the parameters listed in Figure 6.3 for the Johnson-Cook model, the FLAG simulation model also need a shear modulus and an impact velocity. We used the properties of Shot 104S [130] to provide these two values. Out of 1000 simulation runs generated, our analysis is based on a randomly sampled ensemble of 500. The ensemble represents 500 data instances with the Johnson-Cook model as input parameters and the velocimetry profiles as output parameters. In Section 6.3.3 we evaluate our approach based on known information about the ensemble, and in Section 6.3.4 we describe expert user feedback.

6.3.3 Parameter Sensitivity Analysis

The domain experts a performed sensitivity analysis on the full data set described above. The results of the sensitivity study show a high response for the relationship between the A input parameter and the Elastic Plateau, both shown in Figure 6.3. We wanted to confirm whether this relationship shows up using Drag and Track.

In order to evaluate, we first varied A in the input space to see if there was a correlation to Elastic Plateau shown on the interactive callouts on the output space. Second, we varied the Elastic Plateau in the output space to see if A correlated. In both scenarios, we used the interactive callouts to simply drag the variable A or the part of the curve in the output callout that represented the Elastic Plateau.

Figure 6.6 shows the results of the interactive callouts from both directions. The visual confirms the relationship between the parameter A and the Elastic Plateau feature. As A increases the Elastic Plateau also increases, which shows that the visualization is useful in confirming known information about the specific shock physics ensemble. The sensitivity analysis for these two variables took less than two minutes with our approach.

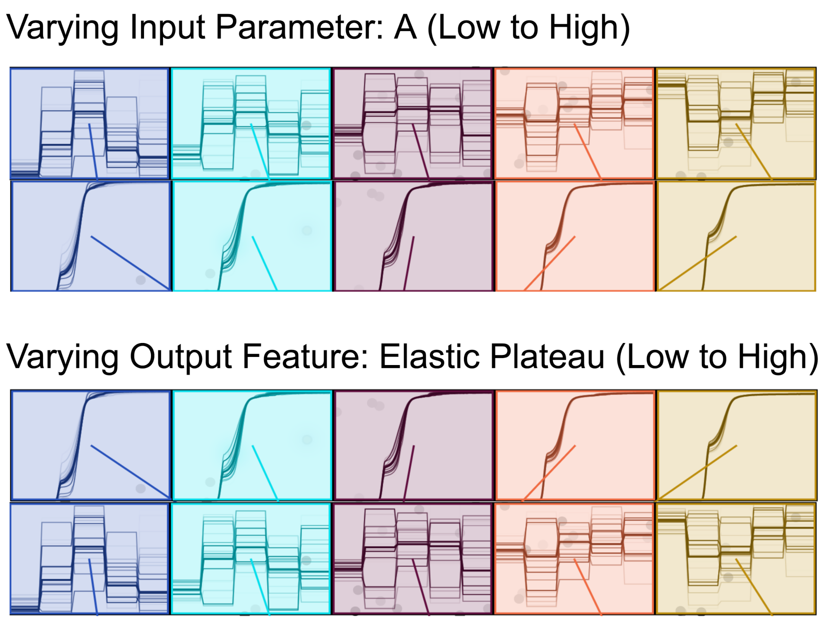


Figure 6.6: Visual confirmation of the correlation between input parameter A and the Elastic Plateau output feature discovered in a sensitivity analysis. Top: As A increases, the Elastic Plateau increases when directly manipulating multiple callouts in the input space. Bottom: As the Elastic Plateau decreases, the input decreases when directly manipulating callouts from the output space.

6.3.4 Domain Expert Evaluation

We evaluated our visualization technique based on feedback from three domain experts, who were familiar with the shock physics application described above. One expert's background was in fluid simulation and model validation, and the two other experts were material scientists who ran shock physics experiments. We used a similar task work-flow as described in Section 6.3.1 to walk the users through our approach.

The current visual analysis tasks for material scientists include many of the techniques described in Section 2.1.1. These include parallel coordinate plots, dimensionally reduced sensitivity analysis scatter plots, and superimposed 1D velocimetry curves. The subject matter experts agreed that working in dimensionally reduced spaces like principal components analysis (PCA) is abstract and hard to relate points positioned in space with actual values.

Overall the feedback was positive, and the users were excited at the possibility of using tools like ours on new data sets, especially ones that have not been explored. The ease of navigating a space by "dragging and seeing" the relationships between the input and output immediately gave our approach utility. The users explained that it is useful to compare the spread of the tracked paths as they move through both input and output spaces. For example, a path that moves through one PCA space an appreciable amount and shorter in another intuitively describes the model's parameter relationships.

The actual path, itself, was a little abstract to understand its meaning, however, the users suggested that perhaps showing how the attributes change as the path progresses would be useful in relating it to actual values. They thought that it would be useful to be able to see which parameters changed along the path while dragging a feature like the elastic plateau, perhaps in another linked view that represented a time-line or while hovering over the path.

The highlights were considered helpful in selecting similar data instances as a way of validating the uncertainty and sensitivity in the model. Again, the users suggested that a tight fit in one space with a tight fit in another, shows that the spaces correlate well, while a more scattered selection in the PCA space would show that the neighbors are not that similar from the other projection. The users liked the ability to see multiple highlighted sets from several different places on the dimensionally reduced views. This allowed them to compare the sensitivities across different criteria.

The users thought that being able to filter the ensemble from the parallel coordinates plot was vital and worked nicely in our application. They suggested that sometimes, especially with domain knowledge or intuition, there are sets of parameter values they would specifically like to study. For example, occasionally one parameter needs to be physically tight while observing how the rest of the parameters respond to the particular variable. This is one part of our application we added based on initial expert feedback, but it naturally fit into the work-flow, enabling new perspectives on the data.

The domain experts finally made two suggestions on where this application would be most useful in their current study. 1) They thought it would be great for aluminum optimization, where there is an unknown material, but we don't know the parameters that go into it. These are good approaches for optimizing a new data set. 2) They were excited about discussing how our approach had potential to solve a current inverse problem that exists in the shock physics community. Many scientists have parameterized the model described above, but most of what exists are other parameterized models. Model parameters can be compared from experimental data sets, but there is not a way to compare outputs. Our technique could be used to evaluate their models quickly. The users theorized “playing the what if game” by superimposing an experimental curve on top of the interactive output callout. Then the output could be modified to try to match the static experimental curve. For example, how close you can a user get the elastic plateau to match without the timing being off? If a model did not match the experiment then something interesting was found and the model might need to be modified. The users said our approach could potentially take the ambiguity out of this process.

6.3.5 Application to Synthetic Models

We also used our implementation to explore four other synthetic models using multidimensional scaling (MDS). These were simpler examples including a parameterized Gaussian, cosine, Catmull-Rom spline, and a Bézier curve. To increase the number of parameters in the Gaussian and cosine, we added a few output parameters like height and position offset. The splines had six input values spread across the x-axis with different heights. We were able to confirm the parameter relationships in both spaces. For Gaussian example, when we moved the mean position up on the output curve and pulled a point that was a short distance from the mean down, the curve was symmetric and the standard deviation went down in the input space as expected. For the spline examples, when we moved a control point up or down, the spline updated as expected. Similarly, when we moved a position on the spline up or down, the control point also moved accordingly. Finally, we were able to use the same input for both the Catmull-Rom spline and the Bézier curve, which allowed us to set the input callout to use the Bézier curve and the output callout to use the Catmull-Rom spline. The effect allowed us to use the Bézier curve to update the Catmull-Rom spline and the input parameters.

6.4 Discussion

Drag and Track provides us our first major result that focuses on understanding high-dimensional ensembles (Q2, C3, C4) via user expertice (virtual data instances) to explore important parameter and user-defined feature relationships (Q3) on both a local and global level (C5). One of the key contributions of Drag and Track is to analyze the implications of combining approaches that seem to be on opposite sides of a spectrum. We ask the questions, "What would an application look like that is both local-to-global and global-to-local?" (Q2, C5) or "What if output and input parameters searched the space the same way?" (Q3) With Drag and Track, we are looking at the full data set and only a few instances (Q1, C3). The direct manipulation of input and output parameters gives us both continuous (estimates) and discrete (nearest neighbors) results (Q3).

This flexibility, however, also makes our technique a bit difficult to categorize, causing confusing questions like "What is it?" We suggest that, instead of being a kitchen sink, the merger of the navigational strategies and the input and output domains helps us solve problems we might not have been able to otherwise. Our work is motivated by other visualizations like Design by Dragging [22], an interface that starts with one instance and allows users to search a space by directly manipulating it. One of the major drawbacks for Design by Dragging is the fact that users do not get a feel of how their search relates to the overarching space. In order to solve this problem, we immerse our directly manipulated queries inside DR views of the data, so there is never a question of where the query is. According to the user feedback, this is important for shock physics, where a single instance or estimate should tell us information about the space as a whole.

Although the global-to-local solution described above is useful, we also validated that our approach also applies to the local-to-global approach. As in Berger et al [18], we calculate estimates with respect to an ensemble, but we allow for multiple queries. Our analysis in Section 6.3.3, suggests that the local-to-global approach here allows us to quickly visually compare multiple estimates along with their uncertainty distribution to confirm sensitivity information. The main thing to consider here is that the same interface and technique simultaneously solves problems that are global-to-local and local-to-global.

The quality of estimation methods is directly tied to the quality of the approximation technique. There is always the question of whether an estimate should be trusted. Although we use the inverse weighted distance for our implementation, Drag and Track allows for any other method that may more adequately match the data. Since our focus is on using estimators to annotate and track changes in the parameter spaces, estimates are naturally qualified by a nearest neighbor distribution. Therefore, Drag and Track could also be used for qualitative estimator analysis or validation. The tightness of the visual relationships with the nearest neighbors provides estimator model sensitivity information.

One major limitation of Drag and Track is the ambiguity between a VDI's predicted position on the DR spaces and the user's interaction described in Section 6.2.2. When a user clicks on a DR space, a VDI is created and the query set contains the x and y values of the clicked position. After the estimation step, the predicted data instance is re-projected back to the DR space, however, this point is not guaranteed to be the same place the user clicked since we rely on an estimation method and not the details of the DR method. One proposed solution is for the estimation method to be the inverse of the DR method, which works fine for PCA, but not other DR methods that do not have a clear inverse like MDS or simple scatter plots based on two single parameters. We originally tried to display both the query point and the predicted point to represent both desire and actual result, however this approach confused the users. Our solution was to show the query point while a user is dragging in the DR space and then show an interpolated predicted point for all other interactions. Since Drag and Track is more interested in the nearest neighbor distribution produced by the VDI, users do not appear to be confused or concerned with the location of the VDI except as a way to navigate a DR space.

A more robust solution to this problem would be to combine results from the estimator while also using properties of the DR method so that the two points agree. Cavallo and Demiralp [66] use optimization to minimize the changes in an instance, while Dis-Function [41] looks at how to change the distance function used by the DR to affect the position of points. iLAMP [136] uses a backwards projection method based on local neighborhoods. Endert et al [43] propose methods for changing the DR methods like MDS to optimize user's intent. Both the similarity search step and estimation step could use these types of methods to better approach changes related to the DR space.

Drag and Track is naturally built to use similar semantic interaction techniques [44, 40, 70] to change the DR projection method based on user intent, which a few tools implement [41, 71, 36]. In our case, the virtual data instances would work the same, however, users could simply move the ensemble data points on the DR views to change the DR method. Our work adds the ability to inform which changes need to be made by providing ensemble highlights of those points. If after searching in the output space, a user sees that the input parameters are spread out, he or she can simply move those points together to affect the DR view of interest. It would be interesting to explore how the two types of interaction could improve sensitivity and uncertainty analysis.

In the next two chapters, we investigate Application Context 3, which studies our thesis (Q1, Q2, Q3) with respect to an ensemble that is both multi-dimensional (C4) and spatially large (C2). These chapters address new challenges that appear when the data has a large number of instances (C3) while having a moderately large memory footprint (C1). For example, Drag and Track does not need remote visualization since the full static ensemble can fit into memory. Similarly, the number of parameters using Bento Box is small, meaning spatial features can be analyzed in a grid using one parameter at a time. These practical simplifications are no longer possible when we need to understand relationships between multi-dimensional parameters and user-defined spatial features of many instances stored remotely on a supercomputer.

6.5 Relevant Publications

- Daniel Orban, Daniel F Keefe, Ayan Biswas, James Ahrens, and David Rogers. Drag and track: A direct manipulation interface for contextualizing data instances within a continuous parameter space. *IEEE transactions on visualization and computer graphics*, 25(1):256–266, 2018. [23]

6.6 Acknowledgements

The authors thank David Walters, Richard Sandberg, and Cindy Bolme for valuable feedback on Drag and Track as it relates to shock physics. Thanks to the artist Francesca Samsel for working with us to improve the colors for our visuals in Drag and Track. This work is published under LANL LA-UR-18-22844.

Chapter 7

Application Context 3: The CMS Toolkit - Understanding Cancer Cell Migration

In the previous chapters we have discussed how to visualize a small distribution of large unstructured spatial data sets for user-defined comparison (Application Context 1 - Chapters 3-4). We also have demonstrated using dimensionality reduction and multiple linked views to add intuition to a large ensemble that is also high-dimensional (Application Context 2 - Chapters 5-6). In this chapter we start looking at the unique problems we face when the data set is both spatial and high-dimensional. Here we lay the groundwork for our final work described in Chapter 8 by studying a cell migration application. We study the trade off between traditional scientific approaches (e.g. two dimensional scatter plots) and high-dimensional visualization approaches (e.g. dimensionality reduction), along with the limitations of each. In this application, we need to consider remote visualization since the ensemble is large and the output is also relatively large. Finally, we investigate trends in the spatial representation to add intuition and to prepare for Chapter 8, which ties everything together using the same application.

7.1 Motivation and Application Background

Scientists often understand empirical results by creating scientific models that explain the observed phenomena. Since the advent of computer science, these theoretical models are often translated into computational models (i.e. simulators), which allow scientists to test their hypotheses. If a simulation result matches the empirical data, it gives evidence to the truth of the theory. Unfortunately, there are two major problems with this approach:

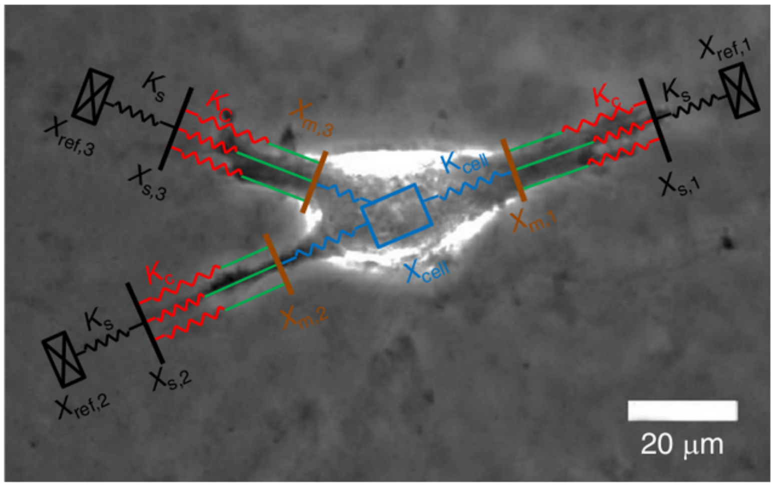
- **Forward Problem** - Computational models often have many parameters, but a scientist might not know the specific parameterization that will match the emperical data. Which parameterizations should be tested?

- **Inverse Problem** - There may be several parameterizations that match the empirical data. Which other parameterizations lead to the same output result?

A new scientific paradigm [2] suggests a data-intensive approach to scientific discovery, where computational models produce ensembles that can be analyzed statistically to both to validate hypotheses and inform theory. Unfortunately, however, there is a growing disconnect between intuitive scientific exploration and data-driven methods (described in Challenge 2 in Chapter 1). Scientists are not ready to sacrifice the intuition gained from simplified parameter exploration in order to trust black boxes. These approaches, however do not scale as we will discuss in this chapter, so we must investigate other methods for understanding data that are also able validate hypothesis and inform theory. Data-driven methods have their own limitation: they may scale, but the results are less intuitive. For example, a principal component analysis dimensionality reduction may fit all of the data on a 2D scatter plot, but we lose the ability to understand the axes.

Consider an application in biomedical engineering where researchers would like to effectively limit the spread of cancer to other parts of the body using chemical processes (i.e. drug administration) [25, 48]. The Motor Clutch Model (MCM) was developed [137, 138] as a stochastic description of the molecular dynamics causing a cell to move. Pictured in Figure 7.1, model parameters like the number of motors, number of clutches, and stiffness constants map directly onto the empirical physical environment.

(a)



(b)

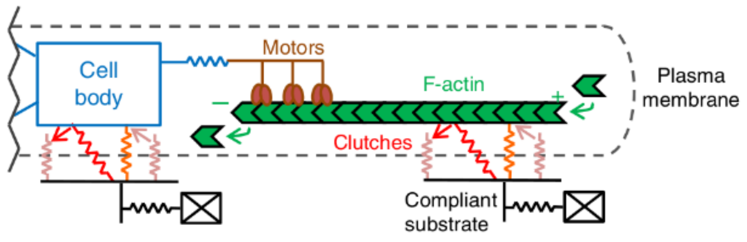


Figure 7.1: The Motor Clutch Model (MCM) (b) superimposed on an image of a cell (a). The spatial stiffness properties and molucule variables of the cell map onto the MCM. Courtesy of Bangasser et al. [48]

The molecular biology equivalents can be modified by controlling gene expression or biological processes through drug administration. For example, a drug may change the substrate stiffness or increase the number of motors, which may slow down or speed up movement. Unfortunately, the parameter space is complicated because, in practice, cells that act similar may not always have the same physical properties. One drug may slow the speed of cancer in one patient, but increase the speed of cancer in another patient. Figure 7.2 describes this concept by showing that reducing the number of motors and clutches in one part of the parameter space has the opposite result in another location.

Here we concretely see the two problems described above. First, the forward problem exists because there are many parameterizations to try before finding an output that

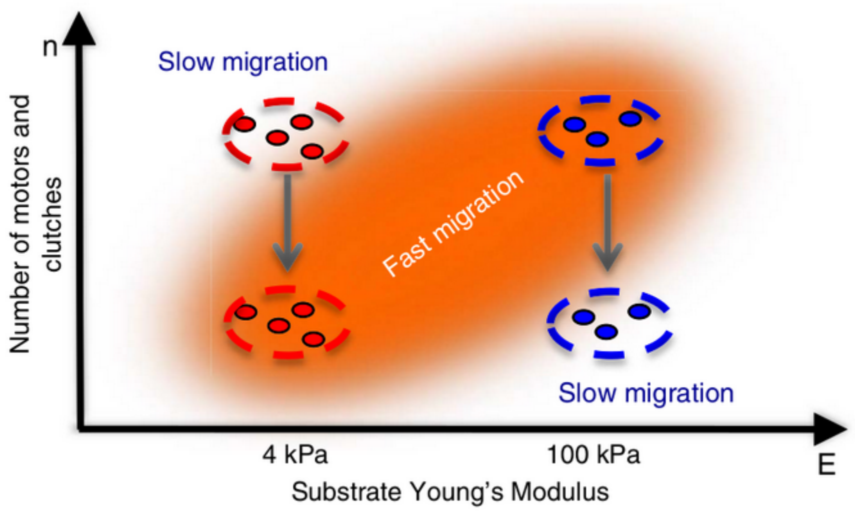


Figure 7.2: An illustration of how applying a drug in one part of the parameter space may slow down cell migration, while in another part the same drug may cause cells to migrate faster. Courtesy of Bangasser et al. [48]

looks similar to a patient's response to a drug. Second, the inverse problem exists since multiple parameterizations could describe the same patient cell behavior. In other words, it is not clear whether the input parameters match the patient. The actual parameterization may exist somewhere else in the parameter space. In this example, we would need to look at alternatives throughout the parameter space. This enables us to hypothesise multiple possible explanations for a patient's cell behavior, informing perhaps several potential clinical solutions. The state of the art in visualizing cell movement involves displaying low-dimensional scatter plots or line charts comparing two or three variables at a time (e.g. stiffness vs. cell migration), however, as we will see, these approaches rarely consider the problems described above.

In this chapter we discuss the limitations of both traditional (e.g. 2D scatter plots or line charts of model parameters) and scalable (e.g. dimensionality reduction, parallel coordinate plots, similarity metrics, etc...) approaches for for building intuition into the relationships between the input space and the output space. We show how the traditional approaches provide insight into the forward problem and how scalable approaches can help us investigate the inverse problem. We then analyze how the approaches can meet in the middle to understand the space and provide insight in scientific discovery. We conclude this chapter by providing motivation for the missing pieces that would be required to intuitively explore parameter spaces through interactive scalable visualizations.

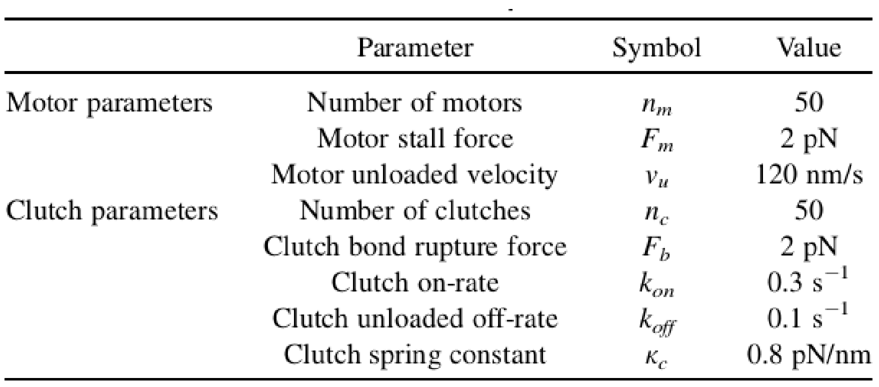
7.2 Visualization Techniques and Implementation

In this section we describe our techniques for studying both the forward and inverse problems in scientific models. We start with methods necessary to generate simulation data as well as rendering techniques for large spatial ensembles that have a large memory footprint. Then we discuss traditional (forward) approaches with scalable (inverse) approaches for studying the same parameter space. For both the forward and inverse approaches, we first look at simple methods for searching the space (2D interpolation in Section 7.2.3 and nearest neighbor search in Section 7.2.4). We then scale these methods to move beyond our local views (small multiples of 2D interpolation in Section 7.2.3 and Drag and Track in Section 7.2.4). Finally, in each context we investigate how to visually analyze the spatial cell data (Sections 7.2.3 and 7.2.4). These were chosen to analyze the limit of the state of the art in high-dimensional and spatial cell visualization.

7.2.1 Cell Migration Simulator

In order to study how to visualize large spatial ensembles, we used the Cell Migration Simulator (CMS) [48] to study how cells move. The motor clutch model described above focuses on a single F-actin bundle, commonly referred to as “one arm in a cell”. CMS is a stochastic model that combines multiple F-actin bundle systems described by the MCM in order to understand how cells move as a result of the motor clutch approach. These F-actin bundles are known as modules and intuitively represent “multiple cellular arms”. Figure 7.1 shows both a theoretical diagram of the model as well as how it relates to the physical structure of a cell. The MCM suggests that F-actin filaments, molecules that add structure to a cell, are attached to a substrate (surface or material) via molecular clutches that grip the substrate. These clutches mechanically resist the F-actin retrograde flow, a mechanism for building and destroying F-actin pictured in Figure 7.1 at the + and - sides of the F-actin. Myosin motors pull F-actin towards the center of the cell, while clutches resist this pull by attaching to the substrate.

This tension along with the retrograde flow rate, produces a traction force that, when combined with other traction forces from other modules, causes the cell to move through a force balancing mechanism.



	Parameter	Symbol	Value
Motor parameters	Number of motors	n_m	50
	Motor stall force	F_m	2 pN
	Motor unloaded velocity	v_u	120 nm/s
Clutch parameters	Number of clutches	n_c	50
	Clutch bond rupture force	F_b	2 pN
	Clutch on-rate	k_{on}	0.3 s^{-1}
	Clutch unloaded off-rate	k_{off}	0.1 s^{-1}
	Clutch spring constant	κ_c	0.8 pN/nm

Figure 7.3: The input parameters associated with the Motor Clutch Model. Courtesy of Bangasser et al. [25]

The base parameters for the motor clutch model are pictured in Figure 7.3. These primarily include motor and clutch parameters [25], however for the CMS model, other parameters like maximum retrograde flow rate, substrate stiffness, and max number of modules are also inputs into the system [48]. As the simulator and model improve, there is an expectation that the number of parameters will continue to increase. The cell migration simulator that is used in this chapter is a C++ implementation. It uses random seeds to add variability into the system. In other words, it is possible to run a simulation with the same parameterization, but get different results. This enables a statistical analysis of the variability of the same cell parameterization.

CMS outputs include a 2D spatial-temporal representation, time series data, and several output values including cell motility (movement), traction forces, and actin flow rates. Scientists are interested in how changing stiffness and the other parameters affect the output values (forward). They are also interested in which parameters describe a set of output values from patient data (inverse).

7.2.2 Data Sampling and Rendering

For the purpose of studying CMS in this chapter, we focus on the four input parameters: substrate stiffness (substrate), number of motors (mpool), number of clutches (cpool), and the max polymerization rate (maxpoly - the F-actin growth rate). Each of these parameters is logarithmically scaled in the sampling space, meaning that the distribution of samples is exponential. For example, the stiffness would be sampled at 0.01, 0.1, 1, 10, 100 instead of 1, 25, 50, 75, 100. Each parameterization with the cpool less than 750 is stochastically run 10 times, producing a distribution of data to calculate uncertainty at each point in the parameter space. Due to the time and resources it takes to calculate more than one simulation with cpool greater than 750, we only have 1 or two simulations for each parameterization with the cpool above 750.

We employ three separate sampling strategies to sample the CMS parameter space:

- **Systematic Sampling** - We use a Cartesian sampling for the four parameters described above. This turns out to be 8*5*5*5 = 1000 samples equating to 6400 simulations (since there are 10 simulations for cpool less than 750).
- **Random Sampling** - We sample the space by randomly choosing the parameters within the range for each parameter.
- **Latin Hypercube Sampling** - We define discrete parameter ranges and ensure we sample uniquely in those ranges to more densely sample this space.

After sampling, we have three separate ensembles we can use to run our analysis. Each ensemble stores the results in a shared disk space on a supercomputer at the Minnesota Supercomputing Institute (MSI). Each simulation represents 6 hours of spatial-temporal data stored at the granularity of 1 second. Each simulation takes between 10MB and 20MB of space depending on a dynamic number of modules that can be created, with an average of 14MB. The total amount of data for the systematic sampling alone is approximately 110GB.

Since the collective data sizes in the ensemble are much larger than we can efficiently send over the network or render on a single GPU, it is important that we (1) do not move the data for analysis, and (2) have an efficient mechanism for remote visualization and interactive communication. We created a server on the supercomputer that communicates to a remote client via web sockets. A user can use any modern browser, navigate to the server's URL, and communicate directly with the server to query and interact with the data in real-time. The server can run complicated analysis algorithms without moving the data while enabling the ability to run more simulations. Since we are using web sockets, a bidirectional communication link is established using JSON as serialization. This also enables multiple users in multiple remote locations without requiring the expert knowledge needed to interact with a supercomputer.

We created an application specific library that allows for efficient data access and the calculation of summary statistics. This includes the specification of a binary input and output file format for CMS, which allows us to read and write data in real-time for efficient storage and navigation of simulations. It also enables spatial feedback for viewing cell dynamics and possible in-situ analysis. Summary statistics are calculated based on the binary output format, providing a mechanism for analysis on the raw data. Therefore, we do not need to re-run the simulation if there are additional analysis needs.

7.2.3 Forward Approaches

Forward approaches sample the parameters for a model and inspect the output. For example, if we wanted to study movement of a specific type of cell for different substrate stiffnesses, we could fix the parameters and run a simulation only varying the stiffness parameter. Here, we are trying to find relationships in the data by asking questions directly through the model's input parameters using an ensemble.

Parameter Exploration through Interpolation

(

The domain experts are especially interested in understanding how changing a parameter with respect to another parameter can show trends in the data. For example, it is important to know how a cell's motility (i.e. migration speed) changes when we both increase substrate stiffness and increase the number of clutches. This involves fixing the other parameters so we can adequately understand how the two parameters relate to each other with respect to the output.

We created the CMS Viewer tool, shown in Figure 7.4, a tool that allows users to customize scatter plot axes, filter data through an interactive parallel coordinate plot,

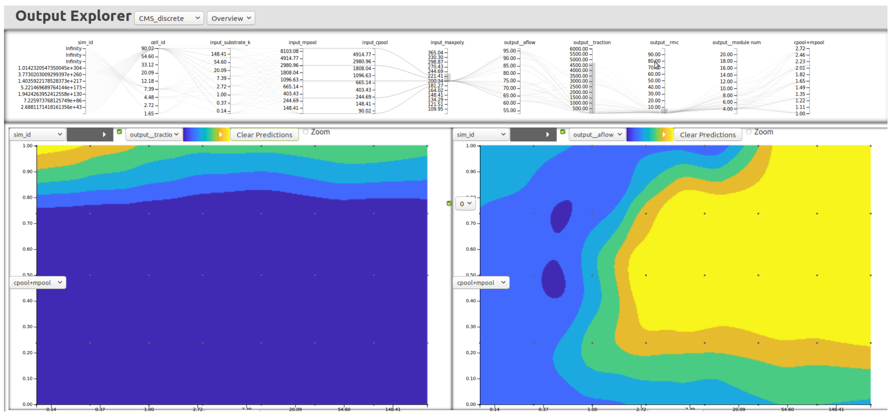


Figure 7.4: The CMS Viewer tool allows users to dynamically filter an ensemble using the parallel coordinates plot and view user-defined axes along with 2D bicubic interpolation to explore a parameter space.

and view 2D interpolated color maps representing a third scalar value. Two scatter plots are displayed so that users can look at the relationships between multiple output values (e.g. simultaneously viewing motility and traction). Users can interactively change axis variables, color maps, and filter the data to investigate the data. This allows users to explore the parameter space by simply changing the parameter they would like to view. The 2D interpolated map uses 2D bicubic interpolation, since it is a common interpolation method the domain experts use.

Small Multiples

Although the CMS Viewer tool allows users to see localized views of the parameter space with respect to two parameters, it is also important to investigate how more than two parameters affect the system. In other words, we need to scale the 2D interpolation approach to a higher number of dimensions. To do this, we created the Small Multiples tool, a tool that displays multiple 2D bicubic interpolation plots as they vary across two additional parameters. Figure 7.5 shows how we can view the dynamics of an output parameter (motility) across four parameters: substrate stiffness, number of clutches, number of motors, and the maximum polymerization. Each miniature in the figure is a 2D bicubic interpolation of motility as it relates to stiffness verses the number of clutches at a different location in the parameter space. For our study, we are looking

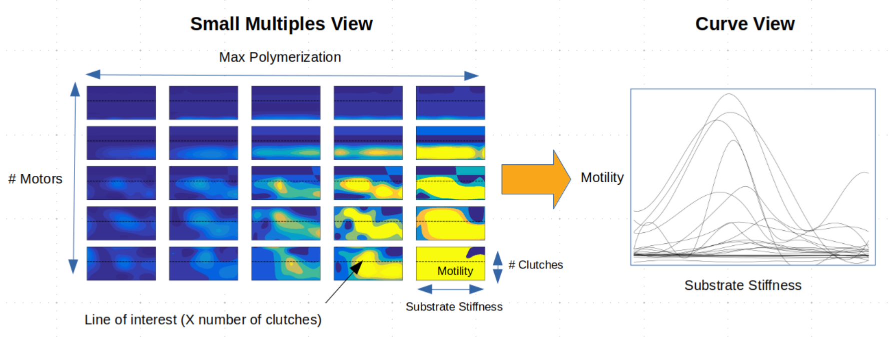


Figure 7.5: The Small Multiples tool displays a grid of 2D bicubic interpolation maps in order to show relationships between 4 parameters simultaneously. This tool has a dual curve view, which visualizes the curves represented by linked horizontal lines across the small multiples.

at a four dimensional parameter space, so this view allows us to study the CMS system and relationships between the four parameters, two of which are continuous and two are discrete.

This tool has a dual curve view, which visualizes linked lines across each small multiple as shown in Figure 7.5. In the small multiples view, each horizontal line represents fixing one of the continuous parameters so we can sample across each small multiple. These lines are visualized in the curve view showing how the output value changes with respect to the chosen horizontal parameter. In the figure, each small multiple shows how the motility changes with respect to both the number of clutches (y-axis) and the substrate stiffness (x-axis). In the curve view, the superimposed motility curves (cooresponding to the horizontal line in each small multiple) show a distribution describing the variability that is possible within the parameter space.

Spatial Visualization

Since the CMS model is closely related to the spatial representation of a cell as shown in Figure 7.1, we have an opportunity to add intuition here. In this case it is important to view the shape of a cell, forces associated with each module, and the motility.

We visualize the spatial representation by focusing on the cell path and the module lengths. Figure 7.6 shows an animated view of a cell as it's shape changes and travels along the simulated path. As the simulation progresses over time, users can relate shape

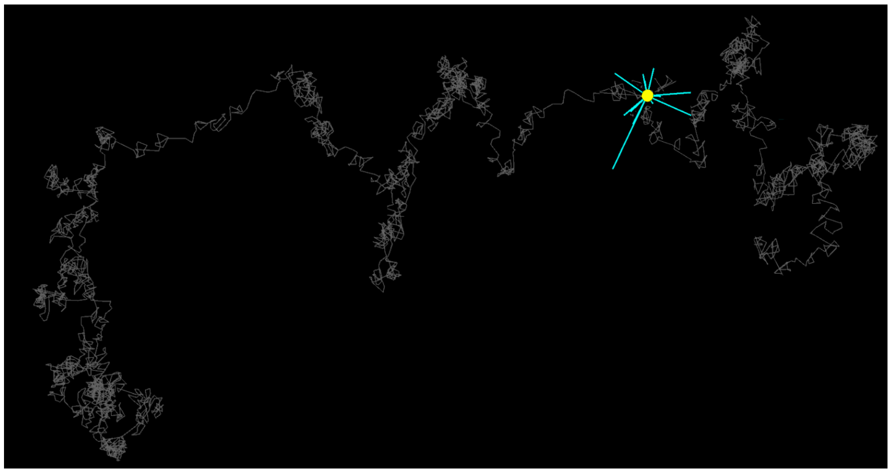


Figure 7.6: An animated view of how a cell's fundamental shape changes according to the Cell Migration Simulator (CMS) along the simulated path.

to speed and distance (e.g. cell motility). This view allows scientists to study the basic expression of a parameterized cell over it's lifetime.

7.2.4 Inverse Approaches

Our approach so far has been a forward search approach, which means we have started with the parameters and searched for trends in the data. We would like to start looking at inverse approaches, which start with the trends and search for parameters. The forward approach allows us to study and analyze a parameter space in detail and statistically validate our hypothesis. The inverse approach, however, helps us know where to look in the data and create hypotheses.

Curve Closest Match

A driving force behind the CMS model is to understand how to slow the spread of cancer of patients. Experimental data consists of motility curves that plot the random motility coefficient (RMC) of a patient's cancer cells as they traverse across different stiffnesses. For example, cells might move faster on a less stiff substrate surface (e.g. muscle) than on a stiffer substrate surface (e.g. bone). This property produces an experimental motility curve profile.

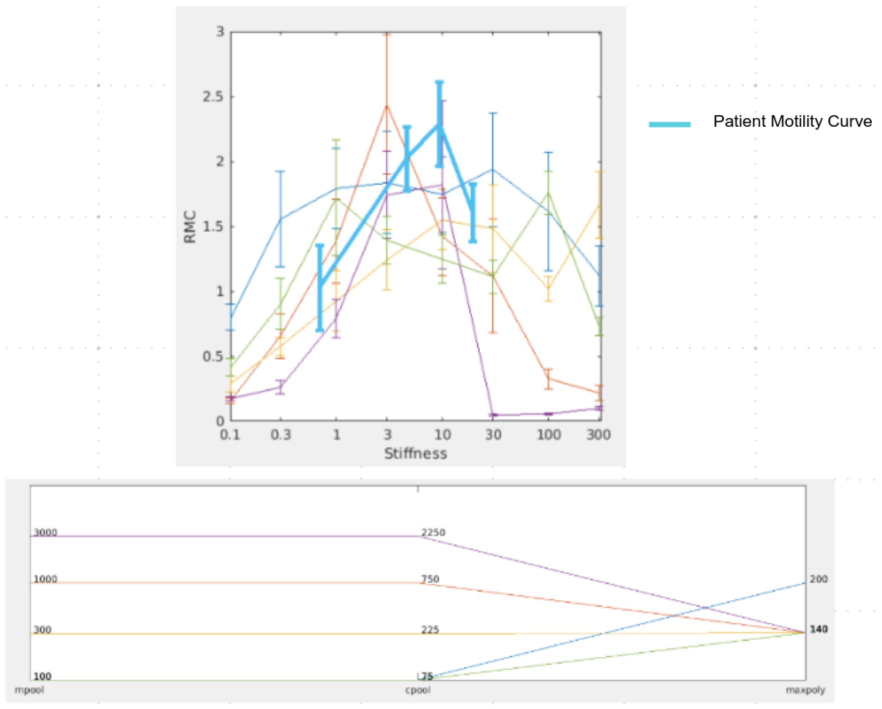


Figure 7.7: A tool that allows users to enter a patient motility curve and finds the five closest matches within the parameter space. Results show the five curves superimposed with the patient curve (top) and the parameter distribution for the simulated curves (bottom).

As part of the CMS Toolkit we created a Curve Closest Match tool, which takes a patient's motility curve and finds similar curves in the simulated ensemble. This couples the experimental data with the model's parameter space. The tool searches the space of simulated curve possibilities based on a user specified distance function. The closest matches from available CMS samples are then displayed along with the distribution of possible parameters. Figure 7.7 shows the patient curve along side the top 5 closest matches and the associated parameters of the simulated curves. This tool is considered an inverse tool since it starts with the output and produces an input distribution. This tool merely reports the nearest neighbors, but more work needs to be done to show how the results are related to other curves in the global context. For example, the curves in the figure do not look very similar, but we need to evaluate this in the context of the full data set. We need to ask questions like "How many curves are in the whole dataset?"

and “What do other curves look like?” To investigate these types of questions, we used the Drag and Track application applied to the same parameter space.

Drag and Track

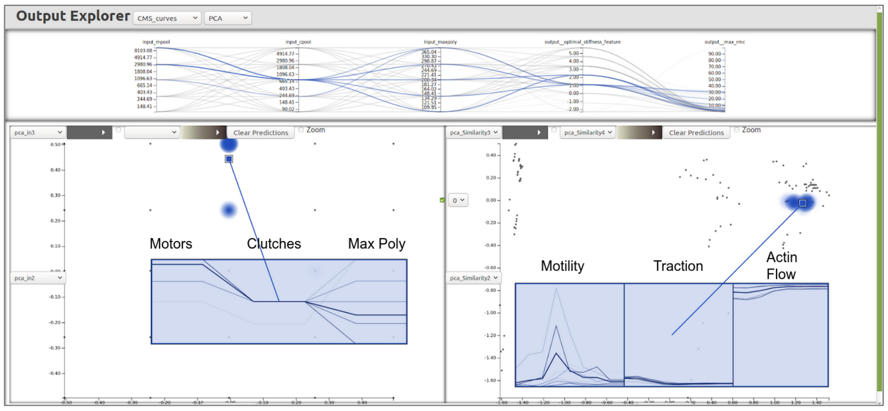


Figure 7.8: The Drag and Track application allows users to explore profiles of motility, traction, and actin flow curves both in the input space and the output space through virtual data instances (VDI).

In order to dynamically explore the input and output spaces in the CMS model, we adapted the sampled data into a format that would work in the Drag and Track [23] application. Figure 7.8 shows the CMS data as a set of output curves (motility, traction, and actin flow) verses substrate stiffness. The left side of the plot shows the input space and its parameters (motors, clutches, and maximum polymerization) and the right side shows the output space. Using dimensionality reduction (PCA of both spaces) we are able to see clusters of similar points, especially in the output space. Points that are close to each other have a similar set of output curves. Using virtual data instances described in Chapter 6 we can investigate the local uncertainty of the clusters and their related input parameters. This technique is also considered an inverse technique that uses dimensionality reduction to cluster the output in order to understand the possible input distribution.

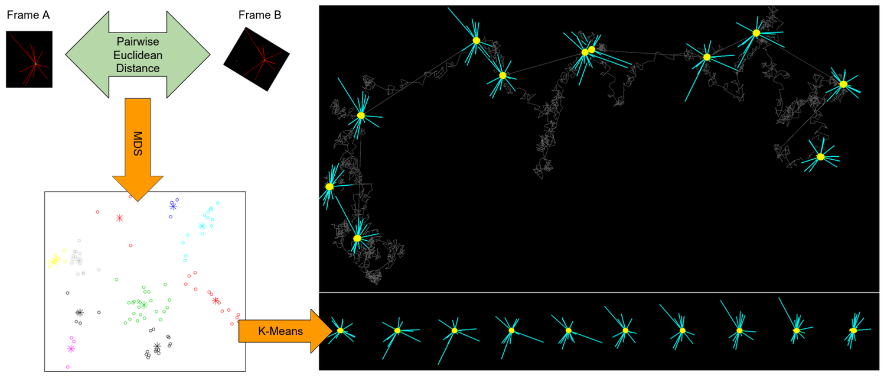


Figure 7.9: The algorithm used for defining key frames based on common similarity features in a cell's shape data across time. The frames are first aligned and compared pairwise via a euclidean distance. These distances are used in a multidimensional scaling algorithm (MDS) and clustered to find common trends based on the similarity metric.

Spatial-Temporal Key Frames

The CMS Toolkit includes one final inverse technique that analyzes the output in the spatial domain by determining key frames of interest. Figure 7.9 shows a set of key cell shape changes along the cell path over time. The algorithm starts by first aligning each simulation frame based on shape or spatially relevant variable (e.g. force) so that it can be compared. After the pairwise alignment (alignment algorithm described below), we calculate the euclidean distance for each pair. The figure shows a multidimensional scaling algorithm (MDS) applied to the pairwise frame distances to reduce the number of dimensions. We then cluster similar frames with the K-Means algorithm in order to pick the K key frames that best describe the simulation. As we will see in Section 7.3 the key frames are dependent upon the calculated distance, which could be different depending on the variable of interest and distance function.

In order to align the cell frames, we use an algorithm detailed in Figure 7.10. Since the cell shape is periodic, being circular, we can apply the cross correlation to determine the phase shift where cell frames are most similar. First we create, normalize, and smooth a periodic 1D signal for both frames, allowing us to analyze a function representing the variable of interest (e.g. length of each module or force at each module). For efficiency we translate to the frequency domain where we compute the cross correlation and then translate back to the time domain. The peak at the result gives

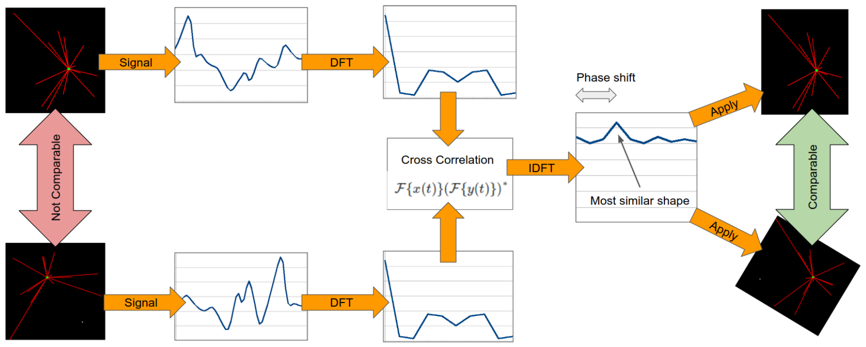


Figure 7.10: In order to align two cell frames, we use the cross correlation to find the phase shift / rotation for the most similar shape. Signals are normalized and the cross correlation is computed in the frequency domain for efficiency.

the phase shift for the periodic function, and therefore a rotation of a cell that has the most similar signal fit. Since we normalize, the actual distance function used in the key-frame generation uses a different distance function (e.g. euclidean distance) based on the aligned original signal.

7.3 Results

To understand the CMS model, we sampled the C++ implementation of the CMS Simulator as described in Section 7.2.2. We first compare the sampling methods in light of our visual analysis. Then we confirm known trends about the data set using our visualization tools. Third, we give an account of how the CMS Toolkit relates to experimental data. Finally, we provide results related to the spatial-temporal key frame generation.

7.3.1 Sampling Methods

In order to evaluate how the sampling methods affect our visual analysis we used the CMS Viewer Tool (see Section 7.2.3) to visualize the 2D bicubic interpolation results across sampling methods. Figure 7.11 shows these 2D bicubic interpolation plots for clutches verses substrate stiffness as the number of motors increase. The random sampling did not produce enough results to produce an adequate bicubic interpolation map when constraining to motor parameter ranges. In Figure 7.11 it is possible to see that the Latin Hypercube sampling matches the basic shape of the systematic sampling with some variation. The variation is due to the fact that the Latin Hypercube sampling is looking at a range of motor values rather than a single value.

7.3.2 Verifying Known Trends

According to Bangasser et al [25] there is an inverse relationship between the traction and retrograde actin flow output variables as detailed in Figure 7.12. In order to reproduce this relationship, we used the CMS Viewer to visualize and compare two bicubic interpolated maps (see Figure 7.12). Both maps had the substrate stiffness on the x-axis and the number of clutches on the y-axis. The map on the left shows the traction force and the right map shows the retrograde actin flow. The contours are inverses of each other, verifying the known trend in the model using the sampled data and 2D bicubic interpolation. In Figure 7.13, we also used the Small Multiples tool to confirm that this relationship exists throughout the parameter space when varying the parameters. Here we see a correspondence between each small multiple on the left and right. For each small multiple pair, when the traction force is low (blue), the actin flow is high (yellow). Similarily, when the traction force is high (yellow), the actin flow is low (blue).

Bangasser et al [25, 48] also describe the complexity of the CMS space suggesting that there are dual parameter relationships in the model, where increasing both the motors and the clutches affect the motility of a cell. We also investigated this relationship using the CMS Viewer Tool pictured in Figure 7.14. The left image shows an illustration of how this type of relationship might work. In the CMS Viewer we set the x-axis to the substrate stiffness variable and the y-axis to the an aggregate variable that increases as both the number of motors and clutches simultaneously increase. The results show a semi-diagonal pattern, suggesting a dual parameter relationship with cell motility. Interestingly, the map shows an inverse diagonal shaped like an L. This is likely due to the fact that illustration visually describes what a dual parameter relationship would look like, showing a shift in the optimal stiffness (the stiffness where the motility is at a maximum). The L shape may be because the increasing of the number of motors and clutches may need to be calibrated as they increase at different rates. We do, however, observe the optimum stiffness shifting on a diagonal from upper left to lower right.

In order to investigate this further, we used the Small Multiples tool described in Section 7.2.3. Each horizontal line on the multiples represents a motility curve in the curve view. Figure 7.15 shows how we can define the optimal stiffness in a set of curves and project the point back onto the motility maps. In the figure the optimal stiffness is represented by red in both views. Each 2D bicubic interpolation map represents stiffness on the x-axis and the number of clutches on the y-axis. Here we observe that when the number of motors is low (top rows of the small multiple views), there is no clear shift in optimal stiffness. However, as we increase the number of motors, we see the optimal stiffness (red) increase as the number of clutches decrease. This is seen as a diagonal red curve traveling from upper left to lower right. The diagonal shift implies that there is a dual parameter relationship, where increasing both the motors and the clutches affects the optimal stiffness. Additionally, in this view we can also look at the relationship between increasing motors and increasing the maximum polymerization rate. In this case, there is no clear difference in the optimal stiffness when we move from left to right, implying that these two parameters do not have as strong as a relationship when it comes to slowing down cell movement.

Finally, in order to get an overall understanding of the CMS model we used the inverse features in the CMS enabled Drag and Track application described in Section 7.2.4. Figure 7.16 shows results from creating virtual data instances at the clusters from a PCA of the output space cell profile curves. After quickly observing and visually organizing the visual callouts of the virtual data instances, we observed three major trends in the data, also theorized by the scientists. We observe that the output profiles are related to the relationship between the number of motors and the number of clutches. If the number of motors are proportionally less than the number of clutches (A), there is little or no motion as seen in the output curves since there is nothing to overcome the clutch forces. If the number of motors is proportionally more than the number of clutches (C) there is little traction and high actin flow, mainly because the motors are recycling the actin filaments with nothing to stop them. This would be like a car driving in high gear on the ice with little traction. The final case (B) shows the case where the number of motors is proportional to the number of clutches. Here we observe the dual parameter relationship between motors and clutches and the shift in the optimal stiffness. In this case, the motors and the clutches create a force balance, causing the cell to move based on the substrate stiffness.

Surprisingly, these three trends were quickly discovered (within 5 minutes) by a user who did not have knowledge about these cases before hand. This user was the developer of the Drag and Track application who knew the system well. To find these trends, the user simply pointed and clicked to create virtual data instances and arranged the visual callouts based on intuition.

7.3.3 Exploring Experimental Data

One major motivation for the CMS model is to inform pathological treatment from understanding cell movement. For example, in order to save lives, understanding the mechanisms behind cell migration can help inform which drugs might slow down the spread of cancer. As we have seen, however, the same drug that may slow down the spread of cancer may also speed it up depending on the patient and cell type. In our investigation, we considered a patient specific motility curve pictured in Figure 7.17. Here the motility was the maximum when the stiffness 9.3 kPa, defining the optimal stiffness at 9.3 kPa.

In order to understand the dynamics of cell motility for experimental data, specifically the patient, we used the Curve Closest Match tool (Section 7.2.4) to search for the parameters that are at play. We entered the patient specific motility curve along with the patient variability (i.e. SEM) and used a distance function that analyzed the similarity between the patient curve and the simulated curves. The output is shown in Figure 7.7, which shows the top 5 curves and their parameterizations. The results imply that the most important variable here is the maximum polymerization rate which is distributed around 140 with one curve having a value of 200. Interestingly, we see the dual parameter relationship again where the number of motors and the number of clutches are coupled and increase at similar rates.

7.3.4 Spatial-Temporal Key Frame Analysis

For time varying data it is often useful to view snapshots at key time frames to summarize a set of actions [139]. These sample the data to show an overview of important states. They can also be used as points of interpolation if needed. For the CMS model, we would like to see where a cell might change from one state to another state. Differences across the time varying cell show changes in gene expression.

Figures 7.9 and 7.18 show the results of applying the spatial-temporal key frame generation when using cell shape as the comparison metric. We highlight some of the interesting features we see in Figure 7.18. First, we can see how shape appears to find key places where the cell changes direction and speed. Second, we see places where the shape is considered different, but does not make a huge impact in cell motility. Third, there is movement that the key frames do not take into account. The second and third point implies that shape is perhaps not the only feature we should consider when analyzing motility.

In order to investigate this, we looked at the key frame generation using three separate distance metrics: comparing shape, comparing force, and comparing shape combined with force. The results are pictured in Figure 7.19 along with the highlighted region that the shape approach did not discover key frames. Notice that each of the distance metrics discovered different features upon the curve. This implies that the distance metric is important for combining aggregated cell shape features with path features. It also motivates more investigation into designing the distance metric. More work needs to be done to understand which distance metric to use. The desired key-frame generation depends on the question being asked and should be evaluated by expert user intuition.

7.4 Discussion

The CMS Toolkit gives us a glimpse into fairly common and traditional ways to understand scientific problems through visualization. The techniques both helped us understand the CMS model from a forward perspective and the inverse perspective. We were able to verify and discover known trends and used the toolkit to inversely tie patient data to the parameter space model. Finally, we started a spatial investigation into cell motility.

At first glance it would appear that the CMS model can be adequately represented by the tools in the CMS Toolkit. Surprisingly, this is not the case. Our investigation in this chapter was limited to four parameters: substrate stiffness, number of motors, number of clutches, and maximum polymerization. The CMS model, however, is fairly complex with more than 10 parameters including those listed in Figure 7.3. In addition, the number of parameters continues to grow as the model becomes more complex to match observed cell migration patterns.

Unfortunately, the forward visualization approaches do not scale beyond four dimensions. For example, the CMS Viewer allows users to view 2D bicubic interpolation maps that represent three dimensions each. The Small Multiples tool allows users to visualize up to only four dimensions at a time. The spatial visualization also only allows one local view of the parameter space at a time, meaning it is difficult to know how the parameters relate to each other across the space. It might be possible to add a third and fourth dimension to the CMS Viewer using time and a third spatial dimension, however, it is difficult to comprehend anything more. Perhaps using this 4D bicubic map for the small multiples would increase the number of dimensions to six. Adding a grid of small multiples grids could increase the number of dimensions to eight. Each time a dimension is increased in the forward approaches, the complexity of the visual increases exponentially, quickly loosing its ultimate utility and simplicity, which is the primary reason that scientists like to work in this domain.

Interestingly, inverse approaches do scale with the number of dimensions. There would be no change to either the Curve Closest Match tool, Drag and Track, or the Spatial Key Frames feature. All these approaches start with the output space, which is invariant to the number of dimensions, and use these output features to understand the input parameters. For example, applying a larger number of dimensions in Drag and Track simply requires organizing the output clusters through virtual data instances and analyzing the resulting input parameter distributions. With the Curve Closest Match tool, we simply provide the patient motility curve and retrieve the input parameter distributions associated with the most similar curves. The problem here is that these approaches rely heavily on distance metrics and the results are not as easy to intuitively understand (e.g. What do the PCA axes mean?). Also, these approaches act more like black boxes that give answers without explanation.

Increasing the number of dimensions also makes it hard to rely on systematic sampling as the number of samples required also increases as the number of dimensions increase. For example, in order to visualize 10 dimensions using the CMS Viewer and systematic sampling, would require $8 \times 5^9 \times 10 = 156,250,000$ simulations to be run. Since simulations take on average 15 minutes to run, this would require at lease 9.3 years using 20 compute nodes across 24 cores with the current technology. It would also require 2.2 Petabytes of memory to store the data. Fortunately, our results show that the Latin Hypercube sampling provides promise for helping us relax this restriction.

In summary, as the number of dimensions and the amount of simulation data increases, it can become quickly infeasible to solely rely on forward approaches. Both visually and computationally, it is hard to find trends in the large parameter space. On the other hand, the inverse approaches, which help us to know where to look, are difficult to validate or understand. Our discussion suggests that it is important to combine these two approaches. We can use the inverse approaches, which include dimensionality reduction and machine learning algorithms (i.e. k-nearest neighbors interpolation), to find regions in the parameter space worth studying. Then we can use the forward approaches to understand these regions of interest in the detail needed for scientific analysis and intuition.

This chapter motivates our work in the next chapter, where we remove the restriction on the number of parameters and investigate the higher dimensional space (Q2,C4). This chapter adds an interesting local (C5) view of "What if?" questions, where we can analyze the parameter space dynamics using 2D bicubic interpolation plots. We can quickly see the trends in a lower dimensional space as parameters change, while statistically quantifying those results. This, however, hardly approaches the global dynamics (Q3,C5) in the space due to the small number of dimensions and the rigid types of questions that can be asked (e.g. parameter vs. parameter).

We are now equipped to extend our investigating into using user-driven (Q3) "What if?" questions in both the local and global space (C5). In the next chapter, we describe this as parameter space dynamics, that are possible to investigate when we combine the visualization with real-time sampling of the parameter space (Q2).

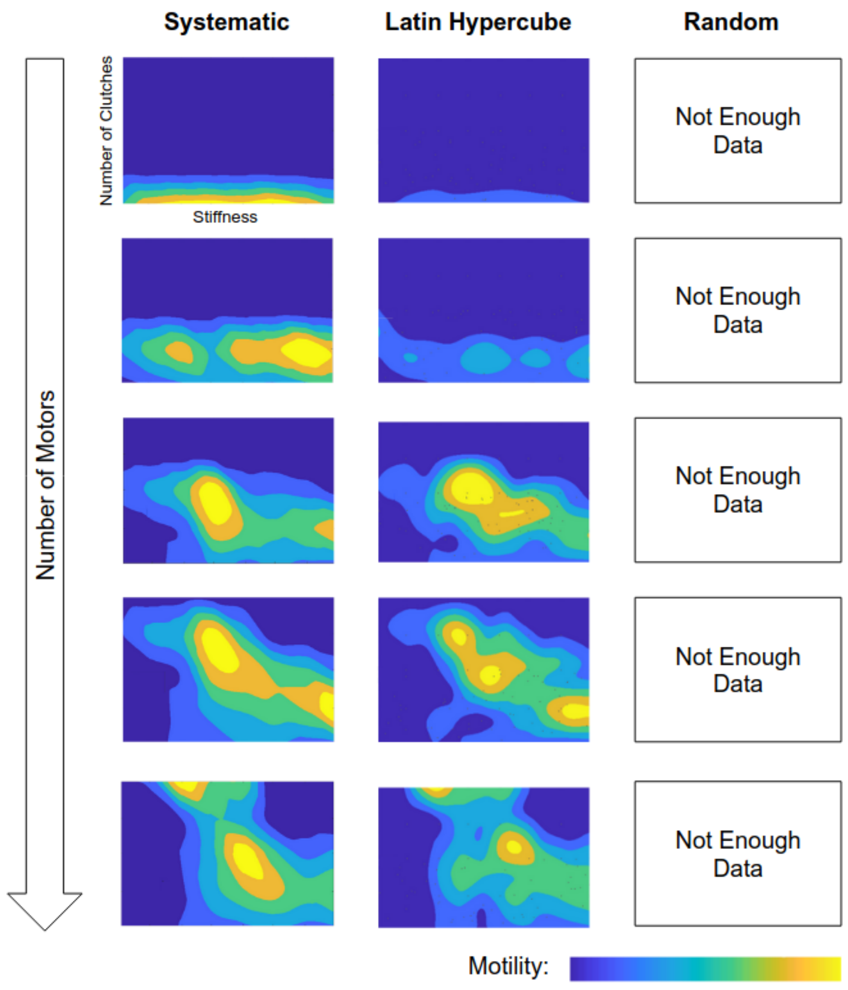
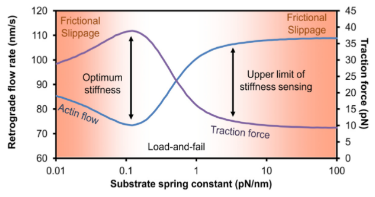


Figure 7.11: A visual comparison of sampling methods. The random sampling did not provide enough local samples to visualize using bicubic interpolation. The systematic sampling and Latin Hypercube sampling have similar regions of high and low motility. For example, as the number of motors increase, we see an area of high motility follow a diagonal from upper left to lower right.



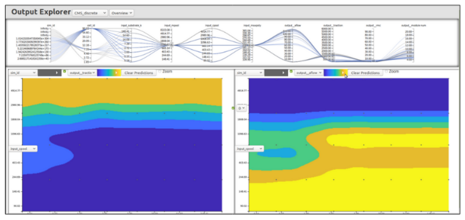


Figure 7.12: Left: A known trend in the MCM model, where there is an inverse relationship between traction force and actin flow. Courtesy of Bangasser et al [25]. Right: Verification of the inverse relationships using the CMS Viewer.

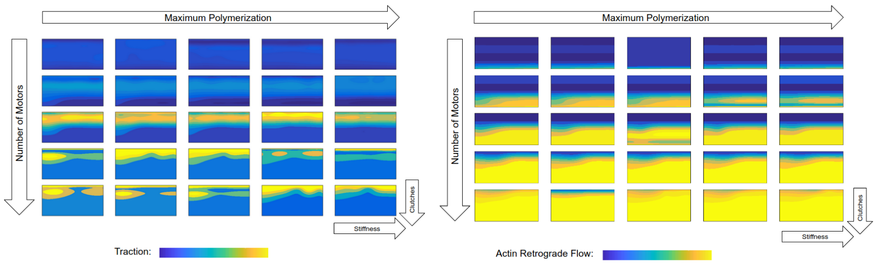
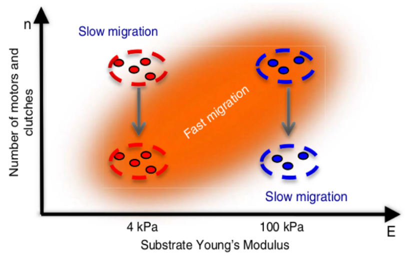


Figure 7.13: Verification of the inverse relationship between traction force and actin flow throughout the parameter space using the Small Multiples tool.



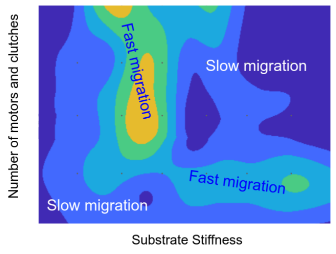


Figure 7.14: Left: An illustration of how the same drug may result in opposite effects in different mechanical environments. This figure descirbes an example dual parameter relationship where simultaneously increasing motors and clutches produces a shift in migration across stiffness. Courtesy of Bangasser et al [48] Right: Visual of a similar shift in migration using the CMS Viewer in another part of the parameter space (low motors and low clutches).

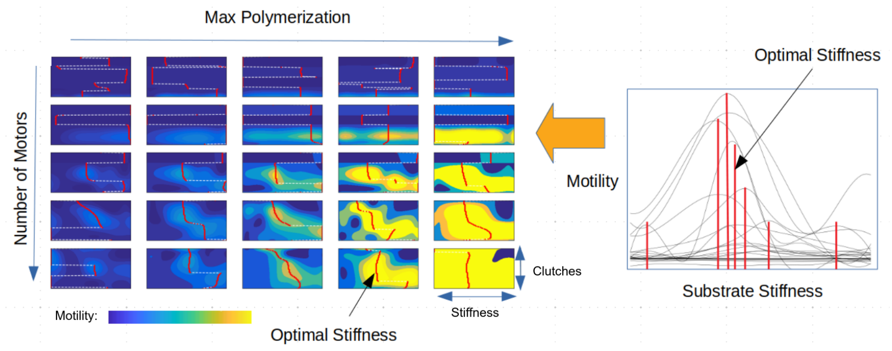


Figure 7.15: Verification of the dual parameter relationship and the shift in optimal stiffness as both the number of clutches and the number of motors increase using the Small Multiples tool.

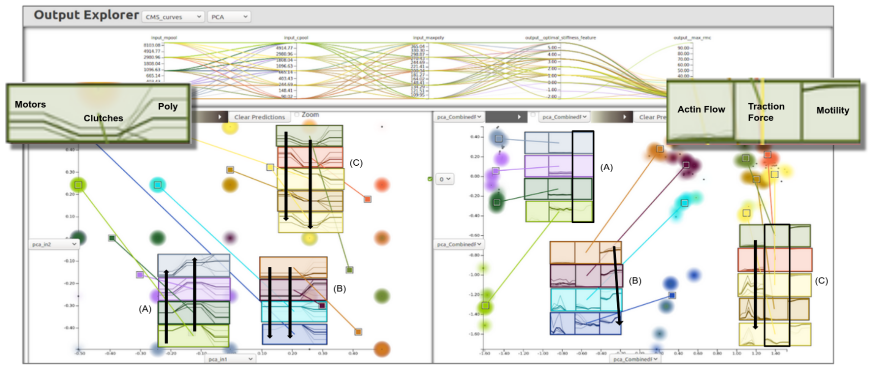


Figure 7.16: Three major trends were found in the data by creating and organizing virtual data instances in the output space around the clusters produced by a PCA of the output curves. (A) Cell cannot move if there are proportionally less motors than clutches. (B) The motility is shifted when both the motors and clutches increase simultaneously. (C) If the number of motors is proportionally greater than the number of clutches, the traction is low and the actin flow increases.

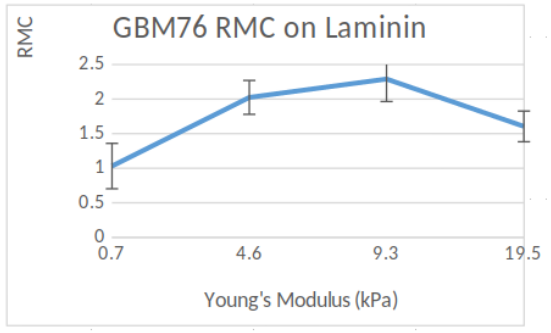


Figure 7.17: The results of a patient motility curve used in our analysis.

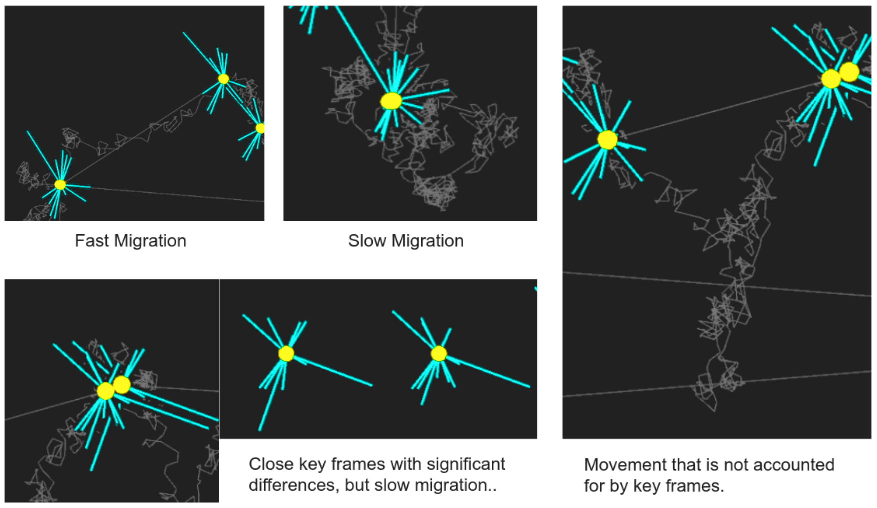

Fast Migration
Slow Migration
Close key frames with significant differences, but slow migration..
Movement that is not accounted for by key frames.

Figure 7.18: Visual analysis of the spatial key frames generated based on shape and their relationship to the underlying cell path.

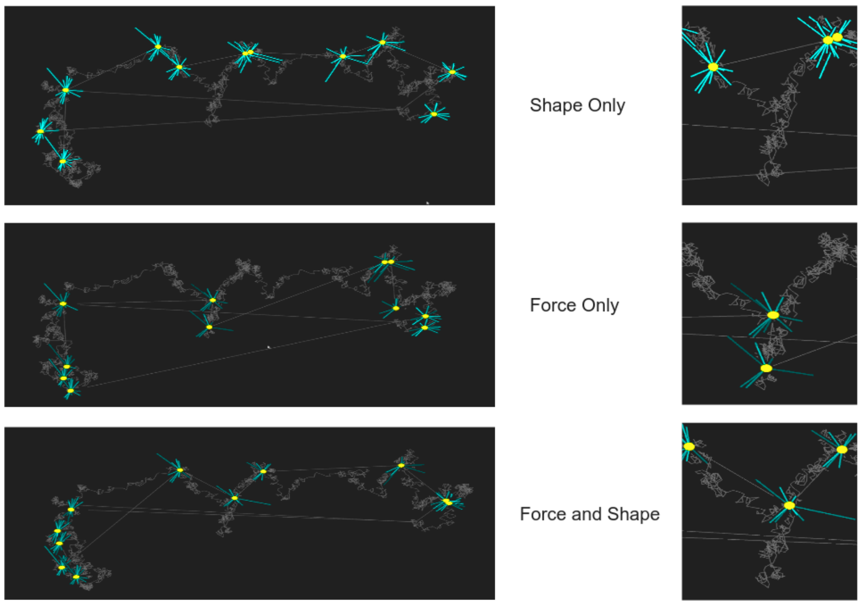


Figure 7.19: Key frame generation for the same cell based on different similarity metrics: comparing the shape only, the forces only, and the forces combined with the shape.

Chapter 8

Application Context 3: Virtual Experiment - Steering a Simulation Ensemble through User Interaction

This dissertation has investigated three challenges pictured in Figure 1.2. The research described in this chapter is a logical culmination, synthesizing lessons from the previous research to investigate how to steer the sampling of an ensemble (Challenge 2) that is both spatially large and has many multi-dimensional instances (Challenge 1) in order to understand user-defined feature dynamics in the context of uncertainty (Challenge 3).

As we have discussed in this dissertation, Ensembles are composed of discrete samples of a continuous multi-dimensional parameter space. As the number of dimensions in the parameter space increases, the volume (the number of possible values) of the space increases exponentially making the sampling sparse. This leads to a phenomena called the "curse of dimensionality" [30], which suggests that the amount of sampled data that is needed for methods that require statistical significance increases exponentially as the number of dimensions increases. In other words, with a sparse sampling, it becomes increasingly hard to prove anything statistically. This is because we cannot feasibly collect enough samples to achieve the desired density on a global level. This is a fundamental problem for high-dimensional parameter spaces, and therefore, important

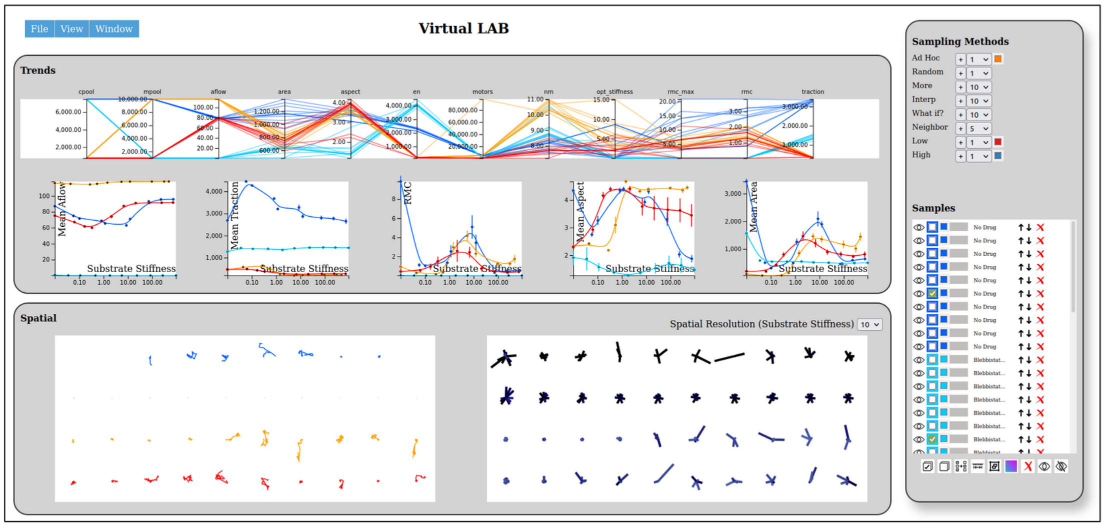


Figure 8.1: The Virtual Experiment user interface that allows scientists to ask questions interactively and steer the sampling of an ensemble. As users manipulate the experimental control, simulations are run providing real-time feedback about the question of interest. The output explores global relationships between input parameters and the output data. This visual shows an experiment that studies a dual parameter relationship in the Cell Migration Simulator (CMS) model.

for us to address in our study.

In this chapter, we ask the question, “How can we leverage user-interaction to effectively sample a parameter space while providing statistically significant results both locally and globally?” (Q2, Q3) In addition, we want to build large (Q1, C1, C3) ensembles that are both spatial (C2) and high-dimensional (C4). In this case, we can work towards focusing the sampling on a user’s question instead of sampling the parameter space in areas that are invalid, not interesting, or not important to the user. In order to accomplish this, we propose coupling user interaction with real-time sampling, allowing the ability to answer high-dimensional questions (Q2, C4) that cannot be answered by more traditional global sampling methods. For example, imagine that an engineer would like to study which parameters of the Motor Clutch Model (MCM) [137, 138] are important for increasing migration when also increasing the number of motors and the number of clutches. This practical question can be sampled in a subspace of the full parameter space, making it possible to statistically analyze the local and global implications of the question. As several questions are asked, subspace samples can be aggregated to create an ensemble that has been tuned for the application domain based on expert user intuition.

Interestingly, being able to control the sampling, allows us to investigate another question relevant to our thesis: “How can we understand the local and global dynamics of a parameter space?” (Q3, C5) In our study, the question of parameter space dynamics shows up as “What happens to parameter A if I increase parameter B while decreasing parameter C?” or “What input parameters causes this output feature to change shape?” For example, in the cardiac lead application engineers ask “How does the fluid flow change as I increase the lead stiffness or length?” For the Cell Migration Simulator (CMS), scientists are interested in seeing how a cell’s optimal stiffness changes as the number of motors or clutches change so they can know what types of drugs to apply to slow down a cell’s migration. Parameter space dynamics asks “What if?” questions, which are questions that investigate how data instances change when parameters or features change. These types of questions are illustrated above. To study parameter space dynamics, we can strategically sample the subspace, creating an ensemble with many different instances. We can then modify the instances, changing only the parameters of interest, and re-sample the space. This provides finite differences, allowing users to analyze how changes in a feature are related to parameter changes (i.e. sensitivity) and other features both locally and globally.

8.1 Motivation and Application Background

In Chapter 7 we discussed the mechanics of cell migration. We also presented the results from several tools that provided insight into CMS. One of the major problems with these approaches, however, is that the tools rely on static ensembles. We must assume that the answers to our questions exist within the ensemble. This was not a problem since we only studied four dimensions and were able to densely sample the ensemble. In this chapter, we want to scale beyond four dimensions. Unfortunately, because of the “curse of dimensionality”, it is not possible to densely sample the space. For example, in Section 7.4 we calculated that in order to visualize 10 dimensions using the CMS Viewer and systematic sampling, it would require $156,250,000$ simulations to be run.

Since cell types and gene expression have a profound impact on motility, it is important to be able to statistically analyze these features in detail, meaning we need techniques that allow users to run the Cell Simulator more densely in specific regions in the parameter space. At the same time, we need to understand other parts of the parameter space that may also have an impact on cell motility. To make the situation a bit more complex, there is a spatial representation of the cell that is important for intuitively understanding the cell dynamics. To address these challenges, we extend several techniques from Chapter 7. We are specifically interested in adding detail to the spatial visualization (see Figure 8.3) and tightly integrating the spatial visualization with parameter and output data analysis (see Figure 8.7) .

We are also interested in studying the sparsity problem. Although useful for understanding parameter spaces, ensembles suffer from two major statistical sampling problems that exist at opposites ends of the sampling spectrum:

- **Global Sparsity** - Ensembles that sample the full parameter space are often sparse due to their high-dimensional nature. Therefore, it can be difficult provide the statistical significance necessary to focus on a specific problem due to phenomena like the "curse of dimensionality" [30].
- **Local Focus** - On the other hand, for ensembles that focus on specific parts of the parameter space, it is often impossible to apply the generalizations necessary for understanding the space as a whole.

Ensemble visualization and visual parameter space analysis tools aim to provide insights into these complex spaces while helping users explore relationships interactively. Surprisingly, however, these tools rarely focus on providing users with control over the sampling strategies. In a survey of parameter space analysis tools, Sedlmair et al [32] describe that most of the tools surveyed used a *systematic sampling* (e.g. Regular Cartesian or Latin Hypercube Sampling), while very few used an *integrated sampling* approach, which allows users to generate additional samples or adapt sampling strategies. Some tools suggest that *integrated sampling* should be part of the system, but this component is usually presented as a potentially useful black box [22].

Our goal is to investigate whether integrated sampling enables us to scale beyond the few dimensions studied in the CMS Toolkit. Ideally, scientists would be able to create virtual experiments that allow them to interactively ask “What if?” questions via simple sampling methods and spatial interaction that define the domain of interest. Recall from the introduction that “What if?” questions ask how data instances change when user-defined parameters or features change. Bangasser et al [48] describes an experiment where four separate cell configurations were compared: (1) no drug, (2) applying a drug that increased the motors, (3) applying a drug that increased the clutches, and (4) applying both drugs together. In a virtual experiment, each of these configurations would be a separate set of running simulations that the user could explore in-situ (while the simulation is running). Several additional design goals would aid in user-driven parameter space exploration. In order to dynamically sample areas of interest, users should be able to add more samples , interpolate between the different configurations, and explore nearby neighbors. To add intuition, users should be able to change propertiesparameters in real-time to create and analyze new samples based on their interaction, exploring the spatial properties of simulations.

Traditional simulation workflows often are major bottlenecks for visualizing and understanding parameter spaces due to long simulation completion times, large remote data sets, and cumbersome data processing workflows. Scientists and engineers often need to wait days to gain any intuition into their results. These approaches are called *Informed Trial and Error* strategies in the literature [32]. An approach would need to scale to a large number of simultaneously running simulations. This would have the potential to enable a unique environment where creating and deleting simulations would be the the norm rather than the exception. Asking questions of the space, therefore, could be as simple as adding many additional strategically placed samples.

We are also interested in whether we could ask more general questions about the dynamics of parameter spaces. Can we approach the idea of global sensitivity by asking the same question in multiple parts of the parameter space? For example, “What if we increase the number of motors and the number of clutches throughout the space?” Perhaps we can determine whether changes in the output features can be said more generally..

8.2 Visualization Techniques and Implementation

To explore the potential of integrated sampling in ensemble visualization and to meet the design requirements for high-dimensional and spatial parameter spaces (e.g. cell migration), we built the Virtual Experiment Framwork (VEF). This framework is designed to extend the techniques from the CMS Toolkit (described in Chapter 7) and investigate the high-dimensional sparsity limitations. To work towards this goal, the VEF combines visual parameter space analysis with real-time simulation sampling. Users can control both the sampling of the space and the data transfer via a simple communication protocol described below.

8.2.1 Overview and Definitions

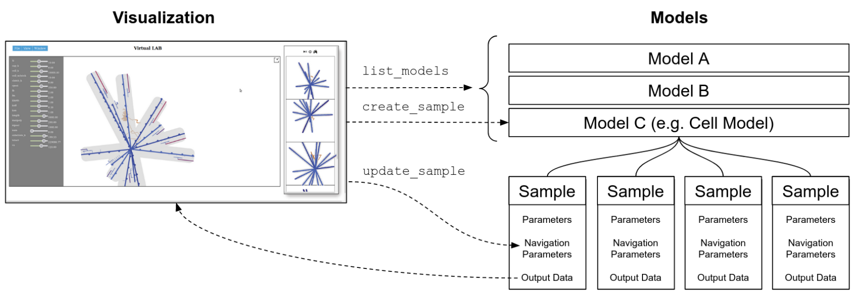


Figure 8.2: Conceptually, the Virtual Experiment Framework (VEF) is a visualization system that can query models, create samples from the models, and update the samples interactively.

Figure 8.2 provides the basic conceptual overview for the Virtual Experiment Framework. The visualization is able to create samples from a list of available models. These samples can then be updated through navigation parameters while the resulting output data is visualized. Below we define our terms and describe the types of commands that can be performed using our communication protocol.

Input-Output Model. Our framework assumes that at least one parameterized input-output model is defined. The input can be any number of model parameters, which usually include a set of numerical values. The Cell Migration Simulator is an example of a model, specifically a simulation. A model can also include statistical models (e.g. regression, predictive, machine learning models, etc...).

Model Sample. A model sample is the result of applying a set of parameters to a model. Each model sample is therefore represented by a set of parameters and the corresponding output data, including complex and unstructured data types. Samples are considered dynamic, meaning that they can be manipulated by a separate set of **navigation parameters** defined by the model. Navigation parameters are not part of the model input parameter set, but they are typically used to navigate through large data sets. An example of such a parameter would be time. It does not make sense for time to be part of the model parameters because it is not an input variable. Instead users could change the time to navigate through the output data, only retrieving timesteps of interest. This saves valuable network bandwidth and limits the need to store all the output data on the client. Note that according to Figure 8.2, the navigation parameters are used when updating a sample. The result of changing the navigation parameters triggers a change in the retrieved output data.

Example Input-Output Model. In our analysis we consider multiple models. The simplest example is the Cell Migration Simulator, which includes several input parameters listed in Figure 7.3. Each sample is a simulation of a cell defined by a unique parameterization. One of the parameters represents a random seed, allowing stochastic sampling of the same parameters with different outputs. Users can update the cell data by changing the time navigation parameter, directing the simulation to move forward in time. Users can also toggle a boolean spatial navigation parameter to let the simulation whether to send updated spatial data needed for visualization.

Composite Models. For our methods, we make use of several composite models. Composite models are defined as models that contain or make use of other models. In the study of cell migration, changes across different substrate stiffnesses is of upmost importance. We created a model that contains multiple cell samples of increasing substrate stiffnesses. This is accomplished by using the example Cell model described above. In this case, we have samples that contain other samples from a different model. In Section 8.2.5 we define a completely different composite model that is used for real-time simulation steering.

Virtual Experiment Protocol (VEP). We define a protocol for communication between visualization applications and computational models. Models can be registered and users can employ them to create, update, and destroy samples. Below are the protocol commands that can be called anywhere within the framework including computational nodes and visualization applications. A few of these commands are illustrated in Figure 8.2:

- `register_model(model):` Applications can register models. Usually this is called from a computational node on a supercomputer cluster.
- `deregister_model(model):` Applications can notify the visualization that the model is no longer available.
- `list_models():` Clients can query the list of available models that can be used to create samples.
- `get_parameters(model):` Describes a default set of parameters that can be used to create samples for the specific model.
- `create_sample(model, parameters):` Creates a sample from a specific model and a given set of parameters.
- `delete_sample(sample):` Deletes a sample.
- `update_sample(sample, navigation):` Updates the sample data based on the navigation parameters.

8.2.2 User Interface

Figure 8.1 shows an overview of the visualization tool with three main components: the experimental control on the right, a view of the output trends on is the top left, and an overview of spatial trends is on the bottom left. This is the primary application view, however, there are two additional tabs that include the detailed spatial view (Figure 8.3) and a dynamics view (Figure 8.11). Figure 8.1 shows a virtual reproduction of an experiment where a dual parameter relationship is analyzed via drug administration [48].

Experimental Control

Most of the user interaction on the main tab is done using the experimental control, which acts as a virtual input device enabling users to control the experimental sampling and visual features. Section 8.3.1 walks through a detailed example of how to use the experimental control as well as several sampling methods to quickly ask questions. The "Sampling Methods" section of the experimental control allows users to configure and create new samples, often based on the available samples (see Section 8.2.3). For example, users can create a sample in any part of the space by pressing the plus next to the "Ad Hock" sampling method. A dialog box opens asking the user to enter the desired parameters and a name for the sample. It is possible to create many samples simultaneously for statistical analysis using the numerical drop down for each sampling method.

The bottom part of the experimental control provides mechanisms for sample selection, coloring, grouping, deleting, and organizing. It also reports the current progress for each sample (e.g. how much time has elapsed in the simulation). The experimental control is also linked to both the trend and spatial views. Therefore, selecting a sample in the control will highlight and show the selected sample in the linked views. This works in reverse as well, so users can highlight and select curves in the line or parallel coordinate charts that represent different samples.

Trends View

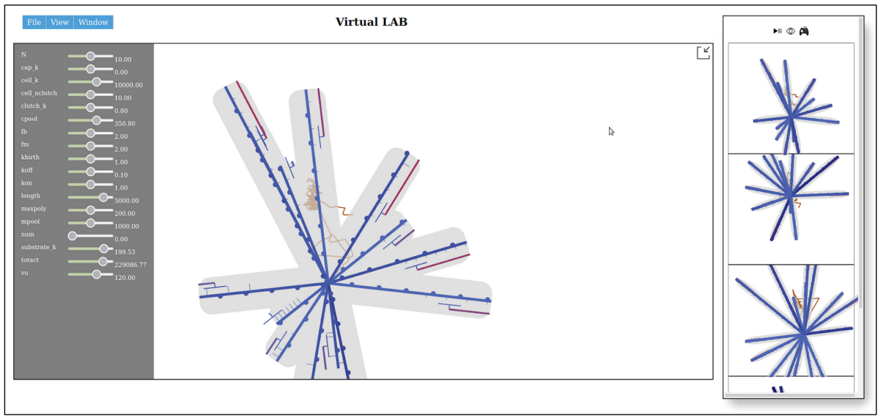
The Trends View shows relationships for important experimental input parameters and output variables. In the case of the Cell model, pictured in Figure 8.1, it is important to show several output variables (e.g. actin flow, traction force, RMC, aspect ratio, and area) with respect to the substrate stiffness. Much of the cell migration literature uses these line charts as a basis for understanding the dynamics of a cell. Each sample is composed of a set of simulations as a composite model, which is described in Section 8.2.1. The line charts can visualize any sample that can be represented by a 1D curve, a common representation for output ensemble data [14, 13, 126, 11, 23]. Here, samples that are represented as curves, are superimposed for comparison. These can be grouped together by the sample name to show uncertainty with error bars to remove visual clutter.

The parallel coordinates plot in the the trends section shows the relationships between input parameters and output data. Each line represents a separate sample, similar to the line charts. Users can define a color for each sample using the experimental control. This allows users to visually annotate trends and relationships based on their own intuition. For example, outliers can be colored red, deleted, or set to invisible. Alternatively, users can define a color gradient across samples to better see the relationships within the parallel coordinate chart.

Spatial View

Finally, the spatial visualization section at the bottom of the main view shows unstructured trends in the output data. For example, we can see in the Cell model from Figure 8.1, that both the cell path (left) and cell shape (right) are visualized for comparison using small miniatures. In this case, each row represents a separate sample and each column is a different substrate stiffness (increasing from left to right). Major spatial trends across the space can be noticed between miniatures.

Detailed Spatial View



N	10.00
cap_k	0.00
cell_k	10000.00
cell_nclutch	10.00
clutch_k	0.80
cpool	350.80
fb	2.00
fm	2.00
kbirth	1.00
koff	0.10
kon	1.00
length	5000.00
maxpoly	200.00
mpool	1000.00
num	0.00
substrate_k	199.53
totact	229086.77
vu	120.00

Figure 8.3: The Detailed Spatial View enables users to investigate and compare spatial properties of samples. The left toolbar is for interactively manipulating samples when in interactive mode.

Users can select several miniatures by using the control button and the left mouse click to view in the detailed spatial view pictured in Figure 8.3. The detailed spatial view has several additional key features. First, it has a large detailed view of the selected spatial data and a set of smaller selected spatial representations on the right for comparison. Clicking on a smaller spatial view will swap the smaller view into the larger view for detailed inspection.

In the upper right of the detailed spatial view, users can toggle several different states. These include pausing the simulation, alternative spatial view configurations, and the interactive mode. Pausing the simulation allows users to view details without needing to track changes in movement. For the Cell model we have provided two separate spatial view configurations depending on features of interest. The default view shows the cell shape, path, and traction force. The mechanics view includes the mechanical model showing spring forces, engaged clutches, and actin flow animation. These view configurations are highly dependent on the spatial representation of the model. The system, however, can utilize other javascript visualization modules if provided. In our development we have tested and integrated several technologies that are useful for spatial visualization including SVG, D3, Three.js, HTML Canvas, and P5.js. The type of spatial visualization depends on the type of sample. In this chapter, we have focused specifically on the Cell model spatial view.

Interactive Mode

Interactive mode is enabled by clicking on the joystick model at the upper right hand toolbar in the detailed spatial view. This enables users to directly manipulate the parameters of a sample in real-time. Figure 8.3 shows a new toolbar on the left with a set of sliders, each representing a separate parameter. Users can directly modify the parameters and see how the selected samples change spatially. Interacting with the sample means users can ask questions in real-time about the parameters and quickly navigate to any area of the parameter space. Section 8.2.5 describes how this process works on black box models using the results from many simultaneously running simulations. Users can further explore the current query by clicking the "Create Sample" button and adding the location to the current set of samples. In addition there is a "Create Question" button that utilizes the recorded user interaction to define a custom "What if?" sampling method. The new samples or sampling methods can be used in the primary workflow for analyzing in the virtual experiment, creating a intuitive exploration and scientific analysis loop.

Dynamics View

The parameter dynamics view (Figure 8.11) shows the local and global parameter space dynamics as a result of the "What if?" experimental questions. The view shows how parameters or output data change as the experiment changes. Global dynamics enables us to understand how changes in parameters or the output data affect different parts of the parameter space. We can, therefore, analyze the parameter space more generally.

This view shows each output variable as a separate line chart where the domain represents the user defined change and the range is the corresponding data value. For example, in Figure 8.11, the domain is defined as increasing both the number of motors and the number of clutches. Notice how the number of motors increase and aspect ratio decreases across over the domain of interest. Each line represents an answer to the question at a location in the parameter space. Users can define these locations by using the "Interpolation" or "What if?" sampling methods discussed in Section 8.2.3.

8.2.3 Sampling Methods

In the VEF, we introduce several simple sampling methods. This section describes how each of these methods work. The key for each method is to determine which parameters to pass to the `create_sample(model, parameters)` function of the Virtual Experiment Protocol.

Ad Hock. The Ad Hock method enables users to sample at any location in the parameter space. When this method is clicked, a dialog box appears with textboxes containing the default value for each parameter. Users can change these values if desired and submit to create a sample, and the visualization calls the create function with exactly the specified parameters.

Random. The Random method creates samples in random parts of the parameter space by randomizing the values for each parameter. Each parameter's random value will be calculated as a normalized value between the minimum and maximum values. This method enables users to look at parts of the parameter space that may not have been previously explored.

More. This sampling method enables users to collect more samples of an existing sample for statistical analysis. The “More” method will find the first checked sample and copy its parameters. In addition it will increment the random seed value in the parameter set to ensure a new simulation. The random seed is simply another parameter in the model that is configured differently for each application.

Interpolate. The interpolate method calculates a linear interpolation between each of the parameters using the top two selected samples. The resolution of the linear interpolation is chosen using the numerical drop down beside the method. Choosing 10 will create 10 samples linearly distributed between the two selected samples in the parameter space.

What if? The “What if?” sampling method enables users to quickly apply a dynamics question to all selected samples. For example, users may want to ask, “What if I increase the number of motors and increase the number of clutches at the same time?” When a user clicks on this method, a dialog box will appear with a drop down for each parameter. The selection options are “none”, “increase”, or “decrease”. In the example above, a user will chose “increase” for both the number of motors and the number of clutches. Once submitted, the sampling method iterates through the selected samples.

For each sample, a linear interpolation between the minimum and maximum values for the selected parameters is created. All other parameters remain the same, allowing us to ask the same question in different parts of the parameter space. If 10 samples are selected and the resolution is 5, then 50 new samples will be created.

Neighbor. This sampling strategy creates samples in the neighborhood of the first selected sample. Each parameter is normalized and changed by a small constant difference in the positive or negative direction. Here we can ask more globally relevant questions about the parameter space starting with a valid configuration and its parameter space neighbors.

Custom Methods. In Section 8.2.2 we describe the interactive mode, which enables users to directly modify parameters for a live simulation. In this view, the “Create Sample” button acts as a sampling method similar to the Ad Hock method. A new sample is simply created based on the current location in the parameter space the user has navigated to. When the “Create Question” is pushed, a new custom sampling method is created based on the order of interactions. Each time the user changes a parameter, the change is recorded as part of the question. For example, if a user lowers the number of clutches and then increases the number of motors, the specific parameter values are saved as a way to reproduce the interaction. When a user clicks on the new user defined sampling method, samples are created by applying the parameter changes to all selected samples. This enables the global exploration across the parameter space by asking the same interactive question across many locations.

8.2.4 Technical Architecture

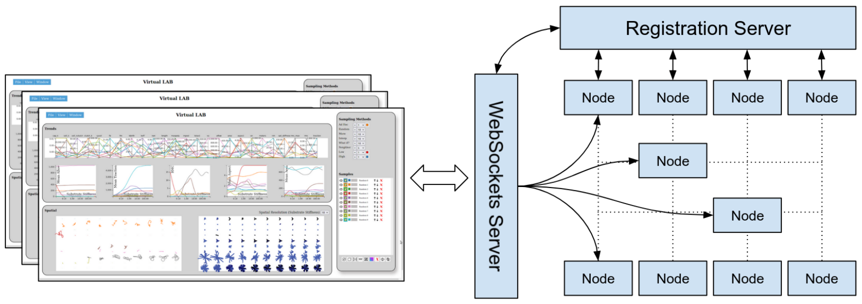


Figure 8.4: This figure shows the infrastructure of the virtual experiment, which connects a web based client to a server that deploys simulations on a supercomputer and provides real-time feedback needed for the design of the experiment.

The virtual experiment architecture is illustrated in Figure 8.4, where it shows a web based visualization connected to nodes through a WebSockets server. As shown in the figure, the WebSockets server communicates directly with nodes that are running simulations. The infrastructure is almost completely decentralized with the exception of a registration server that is used to query model locations. When a node starts, it will register the running model with the registration server. When the a new client connects to the WebSocket server, it will query the registration server for available nodes for the models of interest. There can be multiple WebSocket servers querying the same registration service. The decentralized nature of the system makes processing and communication both scalable and efficient.

Figure 8.5 details the communication between the visualization, WebSockets server,

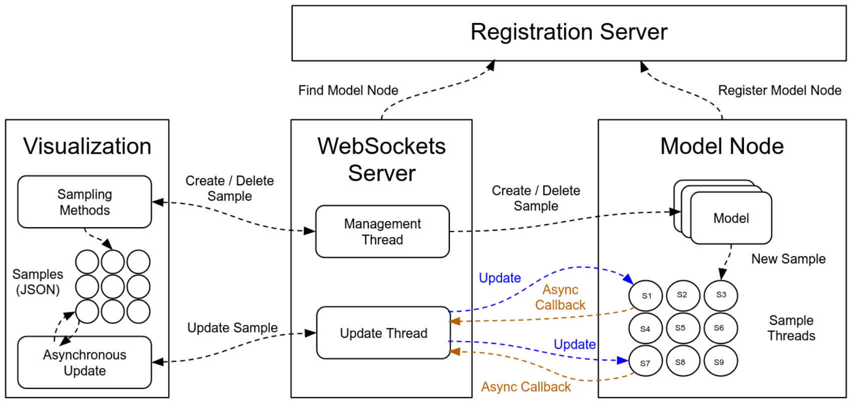


Figure 8.5: The visualization communicates with a WebSockets server to create and update new samples. The WebSockets server finds registered models using the Registration Service and communicates with the samples on separate threads.

the registration server, and a model node. The visualization enables users to create samples via the sampling methods described in Section 8.2.3. A JSON object representing the parameters for the new sample is sent to the WebSockets server's management thread, where it finds an available node to create a sample on. Currently we use a simple round robin approach for choosing which node to allocate a new sample. The management thread is used for creating and deleting samples. This thread is synchronized with the update thread when these actions occur so an update doesn't happen when deleting a sample. The create request is sent over TCP/IP sockets to the node, which uses the selected model to create a new sample. A new sample update thread is created at this point to handle asynchronous callbacks, removing the need for samples to wait for each other. This feature is key for real-time feedback as some samples take longer to complete an update step. It also allows the node to take full advantage of multiple processing units.

The visualization is notified of the sample's creation and a new JSON sample proxy object is created in the UI. Whenever the sample proxy object is updated, a new update request is sent to the WebSockets server. Here the update thread sends an asynchronous update request to the node and returns so that the visualization can continue asynchronously. On the node, the request is received and the specific sample thread is notified and begins working on the request. Once finished, the sample thread sends the updated data back to the WebSockets server. Meanwhile, the update thread on the WebSockets server monitors for any completed update callbacks from any of the samples. When these are received in any order, the visualization is immediately notified with the corresponding updated sample data. The JSON proxy objects are updated and the samples are ready for another round of asynchronous updates.

8.2.5 Real-Time Black Box Steering

In Section 8.2.2 we described an interactive steering technique for quickly maneuvering throughout the parameter space. For many applications it is possible to change the parameters directly in a live simulation, which our system can support through the navigation parameters. However, for samples similar to the Cell Migration Simulator, this approach is not easy. Specifically for CMS, the parameters that are defined at the beginning of a simulation cannot be easily changed in-situ without reworking the complex physics behind the model. Therefore, we must treat the CMS model as a black box, which cannot be changed as the simulation is running. This poses a major problem if we want to steer the simulation interactively.

Our approach interpolates between many concurrently running simulations, quickly reacting to user input and resampling as necessary. Our black box approach, therefore, creates and updates many potentially relevant samples in order to explore one sample in real-time. Here we use another composite model as defined in Section 8.2.1. We call this the **Interactive Model**, which contains another model. When a sample from an interactive model is created, it immediately allocates a set of simulations for each numerical parameter. The samples for each parameter are distributed linearly across the parameter domain by a constant resolution. For example, if there are 10 parameters at a resolution of 5, there will be 50 active samples. Each parameter would therefore be represented by 5 samples extending from a local place in the parameter space. Intuitively, it would be like sampling across 10 orthogonal axes from a point in space. If desired, users may provide an interpolation function that shows a continuous view between samples. It is important to remember, however, that accuracy is based on the sampling resolution since these points represent actual samples.

When a user changes a parameter, the interactive sample uses interpolation if provided to calculate the output data between the existing samples. Otherwise, the output data for the closest sample is used. Once the user commits to a change (e.g. releases a parameter slider), the values represented by the other parameters are no longer valid since the center has moved. Therefore, the current samples are deleted and new samples are calculated based on the new center. At any time, the number of samples in the composite interactive sample is the number of parameters multiplied by the resolution.

Fortunately, this scales well on distributed systems (e.g. consider a cluster with 10 nodes having 24 processors each). However, we have also implemented some optimizations to improve performance depending on the application, parameter count, or resolution. First, since each sample exists on its own thread, when a sample is not being updated, there is very little overhead. In this case, we run a sample until it reaches a specific ramp up time, and then we sleep until the thread is needed by a user interaction. Second, since a user is only sliding one parameter at a time, only those samples representing the specific parameter need to be updated in real-time. Third, to optimize for interactive mode, we pause any active simulations and divert resources to the real-time query. These three improvements allow for quick interaction throughout the parameter space. In addition, since the center is recalculated when a parameter changes, the result is as accurate as the simulation itself. In Section 8.4 we discuss alternatives as well as the relationship to the broader visual parameter space analysis context. In summary, real-time black box steering is a novel approach that enables user interaction of a model without changing the model itself.

8.2.6 Implementation

The Virtual Experiment Framework was written as a C++ backend using the libwebsockets library and standard TCP/IP sockets for client-server communication. Libwebsockets is a flexible, lightweight, and efficient C implementation of the WebSockets protocol. The front end client was written in Javascript using the D3 and JQuery libraries. The VEF provides a small set of header files that contain lightweight pure virtual abstract classes for Models (Model.h) and Samples (ModelSample.h). In order to provide a new model to the system, developers provide implementations for these to interfaces and register them using a Server object that is also exposed in a header file (Server.h).

The Cell Migration Simulator is written as a standalone C++ library, completely decoupled from the VEF. A cell simulation is an C++ object that contains a step method for updating the time via a dt. Other than providing an initial set of parameters, the simulator acts as a black box. In order to enable simulation steering, we created a separate C++ application that implements the Model and ModelSample classes, where each ModelSample contains a CMS simulator object. When the application runs, it registers itself as a model with the Registration Service, which itself is a simple server application written in C++ for passing node location information. The CMS model application also is a server that clients can connect to to create and update samples. Fortunately, as in the CMS example, the simulation code does not need to be coupled to the VEF, and therefore, can be a completely separate library.

8.3 Results

In this section we evaluate the Virtual Experiment Framework. First, we provide example workflows for using the system and describe tasks a user would accomplish. Second, we analyze the utility of our visual analytics approach in Section 8.3.2. Here we use the VEF to analyze an existing experiment using three novel workflows. Third, we show how the VEF enables scientists to explore both high-dimensional and unstructured data accurately and efficiently (Section 8.3.3). This result investigates the usefulness of parameter interaction and spatial visualization. Finally, we evaluate the performance of the system running on a supercomputer (Section 8.3.4).

All experiments were completed on the Minnesota Supercomputing Institute (MSI) Mangi supercomputer using the amdsmall partition. For all except the performance evaluation, we used 10 nodes with 24 cores per node, a totol of 240 available cores. The visualization ran on a GPU machine on the MSI network that could be accessed remotely via the dcv2 protocol. This machine had 2 cores and 8GB or RAM, and it ran both the WebSockets Server and the Registration Service. We used the Chromium Web Browser when running the web-based visualization.

We used the composite CMS model, where each sample contains 10 cell simulations of increasing substrate stiffnesses. Since each simulation is executed on one thread, the composite sample uses 10 threads on the same node. For analysis, scientists are interested in the properties of a cell after 6 hours (21600 seconds) of simulation time.

We used an update timestep of every 10 seconds of simulation time.

Please note that all of the results in this chapter are preliminary results based on user interaction by the first author of this dissertation. This user is a computer scientist who built the system and understands it well. Additional work needs to be done with scientists and engineers who study cell migration. This involves gathering expert feedback and validating simulation results.

8.3.1 Examples

This section describes example usage scenarios of our system and the tasks that would be used to evaluate its effectiveness.

Scenario 1: Dual Parameter Relationship

Dr. Poppy is a biomedical engineer who would like to understand an interesting dual parameter relationship observed in a cell migration experiment [48]. Cells appear to move faster for certain substrate stiffnesses when increasing both the number of molecular motors (e.g. myosin) and clutches (e.g. integrins). In the experiment, Drug A reduces the number of motors and Drug B reduces the number of clutches. The dual parameter relationship is studied by applying each drug separately and then applying both at the same time. She would like to reproduce the experiment virtually using the Cell Migration Simulator through the Virtual Experiment Framework.

Poppy starts by clicking the plus symbol next to the “Ad Hock” sampling method in the experimental control after setting the desired sample color to blue. She accepts most of the parameter defaults, but makes sure that she uses a high number of motors (e.g. 10000) and a high number of clutches (e.g. 7500), and names the sample “No Drug” as the control sample. From there she creates three more samples using the Ad-Hock method: (1) “Drug A”, by setting the number of motors to 1000, (2) “Drug B”, by setting the number of clutches to 750, and (3) “Both Drugs”, by setting the number of motors to 1000 and clutches to 750. Immediately, results start returning from nodes on the supercomputer, providing real-time feedback about the experiment.

Already, she can see expected differences between the curves in the relevant line charts. For example, the familiar inverse relationship between actin flow and traction force is immediately obvious (see Section 7.3.2). Since the CMS model is stochastic, Poppy would like to analyze the results from several samples rather than just one. In the samples section of the experimental control, she checks the sample that she labeled “No Drug” and then clicks the plus sign next to the “More” sampling method asking for 10 more samples. Immediately 10 more samples are added to the sample list labeled “No Drug”. The blue colored lines in the line charts, representing the “No Drug” sample, shows error bars detailing the standard error of the mean (SEM) across different substrate stiffnesses. Poppy does the same thing for the other labeled samples and observes the different curves converge to their final values in real time.

Unfortunately, it is obvious that the two simulations that have a high number of clutches will take a much longer to run, but this is not surprising since these are much more expensive for CMS to compute. She decides to let the experiment continue to run and opens up a new tab to run a different, but related, experiment in parallel. Fortunately, she knows that her current supercomputer allocation includes 20 nodes, each with 24 cores, so she has plenty of bandwidth to ask more questions. In the new tab she decides to look at the problem using smaller numbers of clutches and motors to confirm the theory with more immediate results. This time she creates the “No Drug” and the “Both Drugs” samples, but uses lower values for the “No Drug” sample (5000 motors and 2000 clutches).

Poppy thinks it would be interesting to investigate the changes between the “No Drug” and “Both Drug” samples. She checks both sample checkboxes and clicks the “Interpolate” sampling method requesting a resolution of 10 samples. Immediately, 10 more samples appear whose parameters move from the “No Drug” state to the “Both Drug” state. Here she hopes to find a smooth transition between the two in the output data. Toggling the ability to show the individual curves rather than the error bars, it is obvious that the inverse actin flow and traction force transitions as the parameters transition. Using the parallel coordinates chart she can highlight and see how the parameter transitions relate to the output data as well.

She then decides to check on her other experiment to see how it is going. Poppy notices that the actin flow for one of the simulations is a very large number skewing the mean. This usually indicates that the specific random seed of the simulation has reached an invalid state rendering the data from that sample useless. Poppy decides to create a replacement sample by clicking on the curve on the actin flow chart to select the sample. She then clicks the “More” sampling method configured for 1 additional sample. Finally, Poppy clicks the red X beside the sample to delete it and correct the chart.

While the other experiments are running, Poppy wants to evaluate her question from one more perspective. Based on the interpolation results, she can see that there is a clear shift in the predicted optimal stiffness as the simulations near completion. In addition, she sees a clear pattern in the cell’s spatial path and shape from the spatial view to consider later. This is promising, but she would also like to know if this relationship exists generally for similar cell types. She opens a new tab and creates the “No Drug” sample with a lower number of motors and clutches (750 clutches and 1000 motors) to explore a different part of the space. She checks the sample and decides to investigate similar cell types by looking at parameterizations in the local neighborhood by clicking the “Neighbor” sampling method. These are cells that are similar, but with small changes in each of the parameters.

It is obvious that the cells look and act quite differently based on the line charts and the spatial visualization. She checks all the neighbors and clicks the “What if?” sampling method to ask the same question for each sample. From the dialog, she sets the drop down values to increase the number of motors and to increase the number of clutches. For each neighbor, a set of samples are created that vary both parameters at the same time while keeping the other parameters the same (see Figure 8.10). As the simulation is running, Poppy switches to the dynamics view, where she observes a line chart for each output variable (see Figure 8.11). From here she can see in general what happens as both the number of motors and the number of clutches increase at the same time throughout the subspace of similar cell types.

As the simulations run, she begins to see several trends. Immediately it becomes clear what happens as the number of motors and clutches increase. She observes that the clutch engagement number, the number of modules, and the traction force all increase. It is also interesting that the aspect ratio decreases, meaning that the cell shape becomes more uniform. She switches back to the spatial view and confirms the aspect ratio shift, seeing a diagonal pattern. In this process, she notices that the diagonal pattern also exists in the cell path, which suggests that the shape is perhaps related to movement.

Poppy shifts back to the dynamics view and notices that the optimal stiffness output variable is increasing, providing evidence that the dual parameter relationship is true more generally in the selected subspace. Although most of the samples are nearing completion, a few will need hours to finish, so Poppy decides to let them run overnight and review in the morning. Poppy exports the data from each tab so she can recreate experiment experiment or load the computed data into a future experiment.

Scenario 2: Exploring a Parameter Space

Arlo is a computer scientist who recently started working with biomedical engineers to assist on creating new computational models for cell migration. He would like to start with the CMS model. However, since he is not a biologist or engineer, the parameters are abstract and the concepts are completely new. Arlo opens the Virtual Experiment visualization to interactively explore the parameter space. He starts by creating a sample using the “Ad Hock” sampling method using the default parameter settings and clicks on one of the cells to open the detailed spatial view.

To interact with the cell he clicks on the joystick icon in the upper right, which, in turn, expands a toolbar on the left for directly modifying the simulation parameters via sliders (see Figure 8.3). Arlo moves the slider for the number of clutches back and forth, noticing that two interesting changes occur. First, as the number of clutches decrease, the traction force colored on the cell arms decreases. Second, the cell stops moving when the number of clutches in the pool reaches about 130 clutches. He finds a similar spatial relationship by moving the number of motors. Besides a similar relationship with traction force, he observes that the cell stops around 220 motors. There is also an obvious visual change in cell shape, where the cell area increases as the number of motors decreases. In order to explore more relationships, Arlo sets the number of motors to 220 and sees if he can increase the cell’s velocity using the number of clutches. Interestingly, the cell does not move at the clutch extremes of 75 and 750, but the cell immediately begins moving again when it reaches 250 clutches.

Arlo would like to investigate further, so he switches to the mechanical model view and moves the clutch slider to the three points. When the number of clutches is small, he observes that the actin is moving, but there are not enough clutches to grab onto the substrate and pull each arm. When the number of clutches is large, there are many clutches attaching to the substrate, but the actin does not appear to be moving, meaning that perhaps there are not enough motors to pull the cell anywhere. Finally, when he sets the number of clutches to 250, both the actin is flowing and the clutches are connected to the substrate and causing a directional force to be exerted. Arlo clicks the “Create Sample” button to officially create a sample at the the current parameter space location for future investigation.

Arlo then decides to investigate the relationship between the cell and the amount of available actin. Moving the total actin slider back and forth reveals that cells with less actin are small and have only a few arms, while cells with more actin have larger area and many arms. He also notices that when there is less actin, the path of the cell is less continuous, suggesting that it is perhaps less stable of a configuration. In order to test this theory, he clicks the “Create Question” button on the bottom of the left toolbar and creates a new “What if?” sampling strategy that is added to the main view. He returns to the main view and creates 10 samples using the “Random” sampling strategy. He then selects all the samples and chooses his new strategy (based on his interaction) in order to analyze the dynamics across the parameter space. As expected, according to the parameter space dynamics view, the area and the number of modules increased based on the interactive “What if?” question about actin flow for the random locations in the parameter space. Perhaps more samples are needed to confirm this as a more general rule, but Arlo considers it a hypothesis worth testing.

8.3.2 Visual Analytics

To gather preliminary results, we applied our approaches to an existing experiment. The experiment is detailed in Bangasser et al [48], where a shift in the optimal stiffness (substrate stiffness with the largest motility) was observed when two different drugs were applied to U251 glioma cells. One drug, Blebbistatin, inhibited (decreases) the amount of myosin II motors (molecular motors) and the other drug, Cyclo(RGDfV), partially inhibited integrin-mediated adhesions to the substrate (bound clutches). We used the Cell Migration Simulator, a stochastic generalization and simulation of the Motor Clutch Model (MCM), which treats each cellular arm as a separate MCM module. In CMS, the force balance produced from several modules causes a cell to move in the direction of the force. The theory suggests that cell migration patterns are highly dependent upon the substrate stiffnesses (the stiffness of the surface that a cell is traveling over).

We loaded the CMS model into the Virtual Experiment Framework to ask the same exact questions that were asked in Bangasser et al [48]. We used three different virtual experiments to try and reproduce known facts about the Motor Clutch Model [25]. In the first virtual experiment, we explored similar trends in the simulation data for the dual parameter experiment using our sampling methods. In the second virtual experiment, we interpolated between samples to study the dynamics of the MCM model [25] using the CMS model. In the third virtual experiment, we investigated the dual parameter relationship more generally by looking at a neighborhood of similar cells in the parameter space.

Virtual Experiment 1: Sampling

Our approach for recreating Bangasser et al experiment was to sample the parameter space directly. In order to understand the uncertainty, we needed to stochastically sample in the same location multiple times. Here we used the Ad Hock sampling method with a resolution of 10 (specified by the adjacent dropdown - see Sampling Methods in Figure 8.1). Using the Ad Hock sampling method we created the following samples (40 total):

- No Drug: 10000 motors, 7500 clutches, 10 samples
- Blebbistatin: 1000 motors, 7500 clutches, 10 samples
- Cyclo(RGDfV): 10000 motors, 750 clutches, 10 samples
- Both Drugs: 1000 motors, 750 clutches, 10 samples

After sampling the space, it took approximately 12 hours for all samples to complete on 10 nodes with 24 cores each. Since each sample contains 10 simulations and there are 40 samples, 400 simulations were run. The "No Drug" and the "Blebbistatin" samples were more expensive simulations than the "Cyclo(RGDfV)" and "Both Drug" samples. This is because the simulation slows down significantly for clutch numbers over 750. The "Cyclo(RGDfV)" and "Both Drug" samples were 80% complete 1.5 hours into the simulation freeing up processing for the other two sets of samples. We could see the results of running simulations at any time. Figures 8.1 and 8.6 show the visual analytics output for this dual parameter experiment.

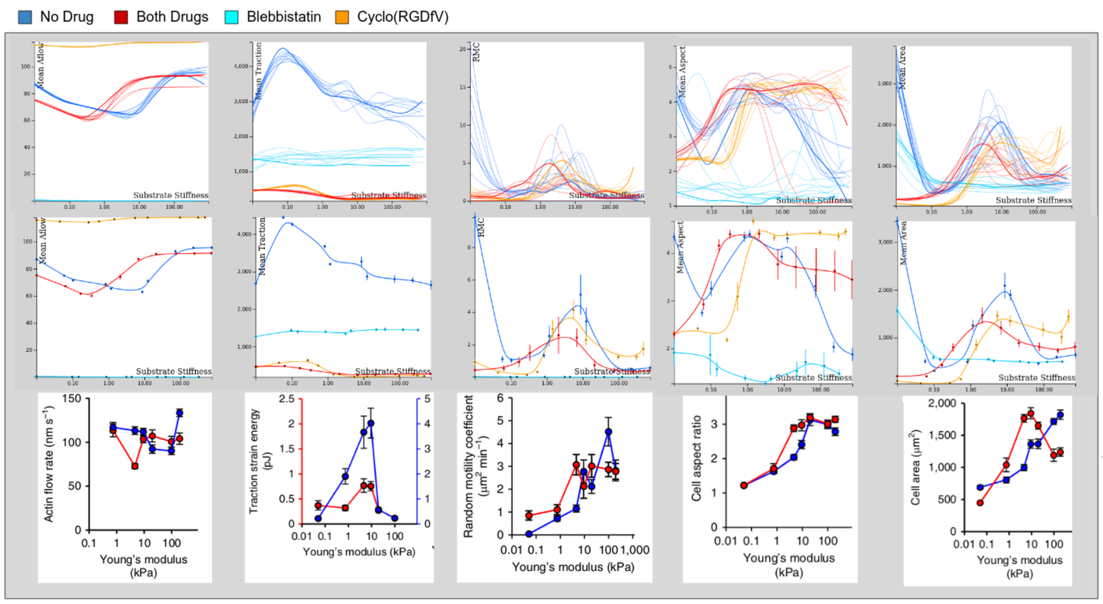


Figure 8.6: The result of the Dual Parameter Virtual Experiment. The top two line charts show the same expected properties of several output variables. The bottom row shows the actual experimental results. Bottom row, courtesy of Bangasser et al. [48].

We were able to use the VEF to verify several known trends about the CMS model. As the simulation was running, some results were available almost immediately, while other output values took longer to compute. For example, within the first 5 minutes, the inverse relationship between actin flow and the traction force was observed (see Section 7.3.2). In the visual, both the actin flow and traction force curves for the "No Drug" and "Both Drugs" resemble the results of the real experiment. The random motility coefficient (RMC) curves, however, took much longer to verify. Even 1 and 2 hours into the simulation, the optimal stiffness shift was still uncertain. This type of uncertainty is expected according to subject matter experts who suggest that the RMC calculation must be at least 66% complete or 4 hours into the simulation (simulation timescale). Eventually, the RMC curve converged. Figure 8.6 shows that the visual results of the output match the curve relationships presented in Bangasser et al [48]. The results from our virtual experiment confirm a similar shift in the optimal stiffness in the output curves.

It is important to compare our approach with the traditional CMS data collection and visualization workflow. In Chapter 7 most of our data was collected from batch jobs that were run on a supercomputer. The workflow requires that a user log onto a Linux system, configure the script for the simulations that need to be run, and submit a job request. Here, it important to be as efficient as possible to take advantage of available resources. Allocating the correct number of nodes and processors for each separate experiment can be complicated. Unfortunately, it is not always clear how long a simulation might run, so resources will often need to be over allocated, which often implies that the job waits longer in a queue. Once a job is submitted and run, users can monitor the output data files to track progress. Finally, after a job has completed without errors, the data is downloaded and processed for visualization. These manual processes require special knowledge on how to run batch job and access remote systems. The workflow is error prone, takes hours to setup and collect data, and is very complicated.

In contrast, our virtual experiment required 8 button clicks and 10 variable changes. The whole process took less than 5 minutes to setup, Most of the setup time was spent picking colors and making sure that Blebbistatin and Cyclo(RGDfV) were spelled correctly. While running, we monitored the simulation, visualizing the cell shape, path, and output variables. When the experiment finished, we exported the data in both CSV and JSON formats for further analysis using the VEF or other tools.

Virtual Experiment 2: Interpolation

For the next experiment, we analyzed the dynamics between samples using the Interpolation sampling method. To start, we imported the saved data from Virtual Experiment 1. We also created a “Increase Both Drugs” (75 clutches, 100 motors) sample to look further into the parameter space. The new theoretical sample further inhibits the clutches and motors.

We first selected one of the “No Drug” samples and one of the “Both Drugs” samples. To study what happens between samples, we clicked the Interpolate sampling method, which created 10 samples that linearly interpolate between the parameters of the two samples. Next we repeated these steps, but for the “Both Drugs” and “Increase Both

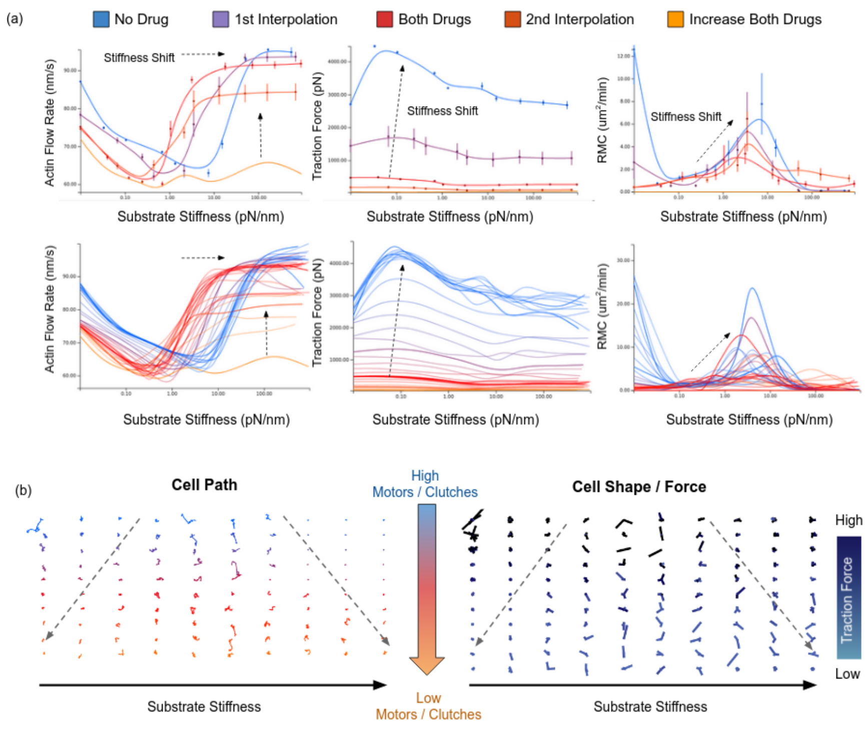


Figure 8.7: Interpolation between samples as the number of clutches and motors decrease. (a) Line charts showing interpolation between samples for actin flow, traction force, and motility. (b) The cell path and cell shape / force as the number of motors and clutches decrease.

Drugs" samples. The results, shown in Figure 8.7, illustrate an increase in the optimal stiffness as motors and clutches increase. The figure also shows the spatial trends based on the interpolation. Our preliminary simulation results show a correlation between the cell path and the cell shape, suggesting there is perhaps a link between shape and motility. They also show that the traction force decreases as we move to lower clutches and motors. To configure the experiment, we clicked 4 checkboxes and 2 buttons.

We also studied the motor parameter. We created a "Low Motor" sample (750 clutches, 100 motors) and a "High Motor" sample (750 clutches, 1000 motors) using the Ad Hock sampling method for each. We selected both of these and clicked the Interpolate sampling method. The results, displayed in Figure 8.8, visualize an interesting nonlinear relationship, where between 850 and 900 motors, a state change occurred as suggested by the jump in actin flow. We used the graph to select the two extremes and

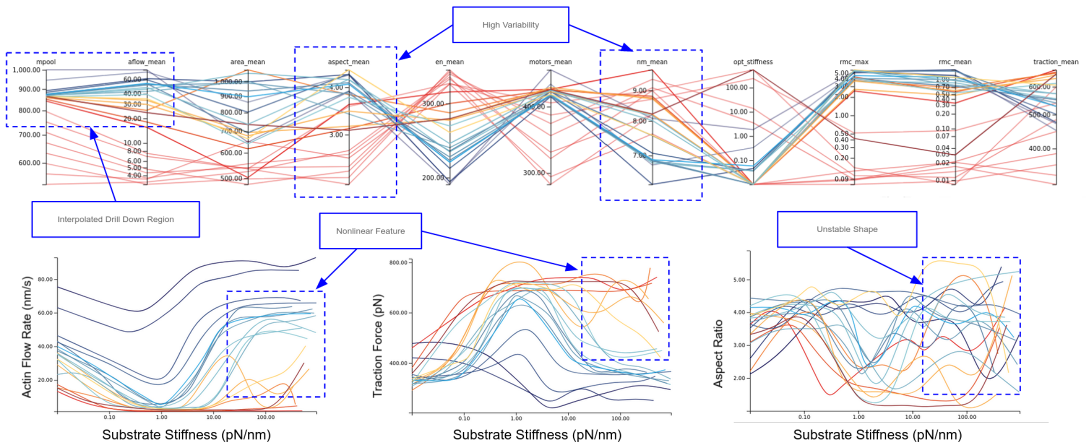


Figure 8.8: Using interpolation between samples of high and low motors revealed a non linear relationship for high substrate stiffnesses. Drilling down using interpolation between samples suggests a discontinuity and two distinct states in the parameter space.

further drill down into the dynamics of this region of interest. We completed 3 drill down interpolation actions each at an interpolation resolution of 5. This was done while the other simulations were running, producing 15 more samples. As displayed in Figure 8.8, the difference in samples between the two states suggests a possible discontinuity at this location in the parameter space.

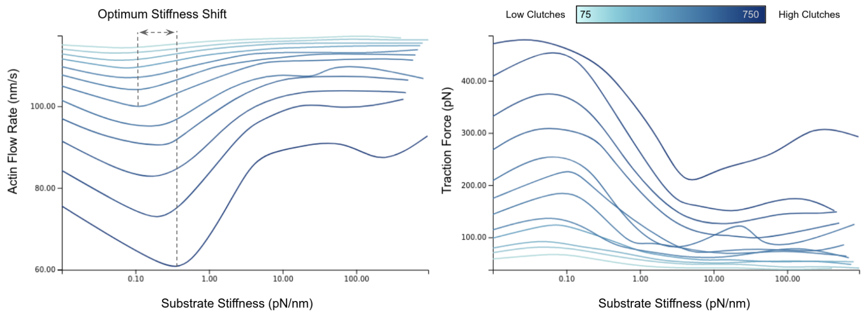


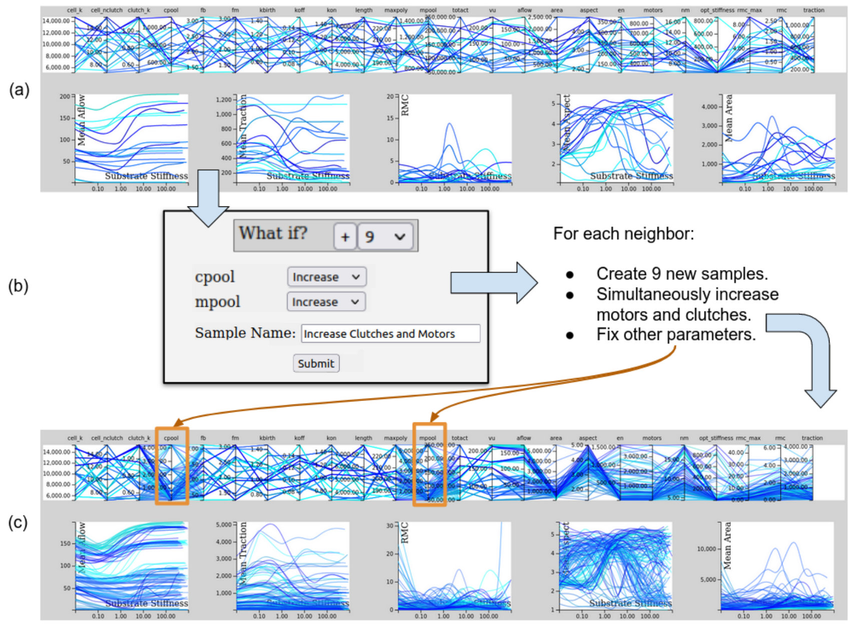
Figure 8.9: Interpolating between samples of high and low clutches has confirmed the inverse relationship between actin flow and traction force.

We performed the same analysis for the number of clutches between "Low Clutches" (75 clutches, 1000 motors) and "High Clutches" (750 clutches, 1000 motors). Figure 8.9 shows how the interpolation has reproduced the known inverse relationship between actin flow and traction force as detailed in Bangasser et al [25].

Each of these sub-experiments were executed simultaneously in separate web browser tabs. All simulations completed within 6 hours.

Virtual Experiment 3: Neighborhood and Dynamics

In the third virtual experiment, we used the “Neighbor” and “What If?” sampling methods to explore the parameter space more generally. First we created an Ad Hock sample using the default parameters for the simulation (750 clutches, 1000 motors, 200 max polymerization). Then we created 20 nearby neighbors using the “Neighbor” sampling method. These samples are near the default sample, but each parameter is projected a random distance within 10% of total normalized distance between the maximum and minimum. Figure 8.10-(a) shows the local neighborhood variability of the default sample for the CMS model.



cpool Increase
mpool Increase
Sample Name: Increase Clutches and Motors

For each neighbor:

- Create 9 new samples.
- Simultaneously increase motors and clutches.
- Fix other parameters.

Figure 8.10: (a) We can sample a local neighborhood using the Neighbor sampling method and visualize the variability. (b) Selecting the neighbors and asking a “What if?” question creates multiple samples for the domain of interest. (c) Traditional mechanisms for visualizing dynamics questions is difficult due to occlusion and visual clutter.

In order to investigate whether the dual parameter relationship extends to the local neighborhood, we selected all the neighbors and clicked the “What If?” sampling method. We selected increase for motors and increase for clutches and created a question that would be asked for each selected sample. The question resolution was 8, therefore, giving us 160 new samples since there were 20 neighbors. The samples ran for 4.75 hours. We removed several samples due to bad data. We removed an entire set of samples for one of the neighbors, leaving us with only 19 answered questions. Due to the large number of samples, it was difficult to understand the output data using traditional approaches as seen in Figure 8.10-(c).

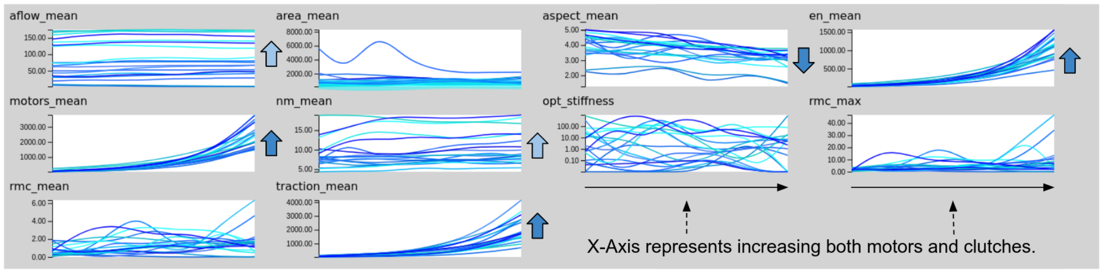


Figure 8.11: The dynamics view for understanding the dual parameter experiment within a local neighborhood.

Switching to the dynamics view makes it much easier to see the trends for each output data value. Figure 8.11 shows the trends in the local neighborhood as both the number of motors and the number of clutches increase. Here, for each output, there are 19 lines (1 for each neighbor). Each line is the same “What if?” question asked in a different part of the space. From the figure, we can hypothesize some more general and global insights about the question, “What if we increase the number of motors and the number of clutches?” Our initial analysis based on preliminary simulation results is as follows:

- The output data for motors, engaged number of clutches (en), and the traction force all increase.
- The aspect ratio shows a decrease.
- The actin flow appears to increase slightly. The visual also shows two distinct

states, one above 100 and one below. Increasing the motors and clutches does not cause samples to change their state.

- The number of modules (mn) stays constant or increases.
- A shift in the optimal stiffness is not completely obvious. We would expect the optimal stiffness to increase based on previous dual parameter analysis. The visual, however, does appear to contain complex patterns that may perhaps be worth further investigation. This may require additional sampling using the same method.

8.3.3 Parameter Space Exploration

This section details a preliminary investigation into the interactive real-time black box sampling and spatial visualization. As with the other results in this chapter, this experiment was completed by the first author of this dissertation. Therefore, in order to better validate this approach we need additional feedback from subject matter experts. Some initial feedback about this approach is discussed in Section 8.4.

Ultimately, scientists and engineers are interested in solving real world problems, not just understanding a model. For example, one application for CMS is to predict and understand how to slow the spread of cancer for specific patients. One of the difficulties, however, is that a drug that slows down cell migration for one cell type might speed it up for another. This problem, also illustrated in Figure 7.2, is fundamental to understand the concept of locality within a parameter space.

To investigate the VEF's ability to explore the parameter space, we asked whether the interactive mode could help us find the situation described above. In other words, we asked the question: "Can we find two separate cell types (locations in the parameter space), where, after adding the same drug to both cells (the same change), one speeds up and the other slows down?" We were also interested in whether we could better understand the relationships between parameters and the spatial output. Here we considered a theoretical drug that inhibits the number of clutches, similar to applying the drug Cyclo(RGDfV), which inhibits clutch activity,

We loaded the default CMS cell into the detailed spatial view and opened interactive mode. Simply changing the number of clutches up and down did not give us our desired

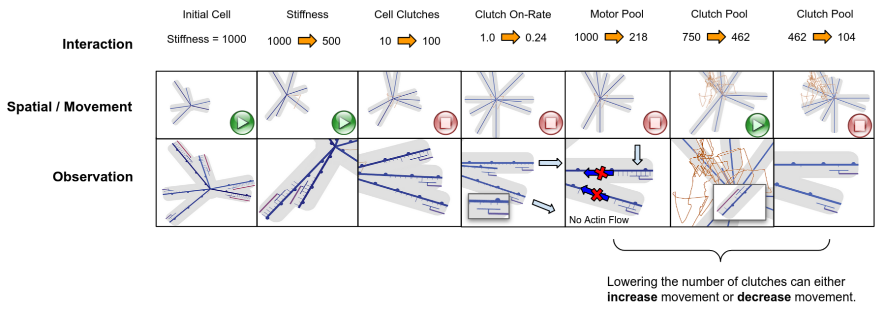


Figure 8.12: Using the interactive mode, preliminary simulation results show the potential of searching the parameter space for locations of spatial interest. The interactive steps are listed along with the relevant spatial observations.

result. We had to create a situation where first lowering the clutches would allow the cell to move, and then lowering them further would cause the cell to slow down. Figure 8.12 details our exploration and observations as we directly manipulated the parameters. We started with the default cell and a high substrate stiffness (1000). The moving cell showed flowing actin, many clutches bound, and high traction force. First we reduced the stiffness and saw a small change in the bound clutches Figure 8.12 shows bound clutches as small tick lines on the cell arms that touch a parallel line representing the substrate. When the actin flows and clutches are bound, a force will be exerted on the substrate (visualized as the outer parallel line). Unbound clutches are not connected and will not cause this force.

In order to create a drag force on the cell center, we added 100 cell clutches. These are not bound to the arms, but the cell body itself, suggesting that the arms need to work harder to make the cell move. We observed that the cell stopped here and that there were fewer bound clutches. We also lowered the clutch on-rate, decreasing the the number of bound clutches and creating a free flowing system where motors quickly break clutch bonds. In the figure, you can see that the cell grows larger and there are very few bound clutches. In order to cause the clutches to bind, we lowered the number of motors, which in turn caused the actin flow to stop.

Fortunately, this created the environment that was ripe for a solution. Reducing the clutches from 750 to 462 caused the actin to flow and the right number of clutches to be bound, producing a force imbalance and the cell to move again. The location for the numbers was found by moving the clutch parameter slider until the maximum speed was observed. Finally, we reduced the number of clutches to 104, which caused few bound clutches and a drag force that was too strong for the motors to move the cell. In other words, according to preliminary simulation results, if we apply the drug to the location with 750 clutches, it will speed up, and if we apply the drug to the location with 462 clutches it will slow down. The interactive search enabled us to observe two locations where this scenario could be true, but more work needs to be done to investigate further.

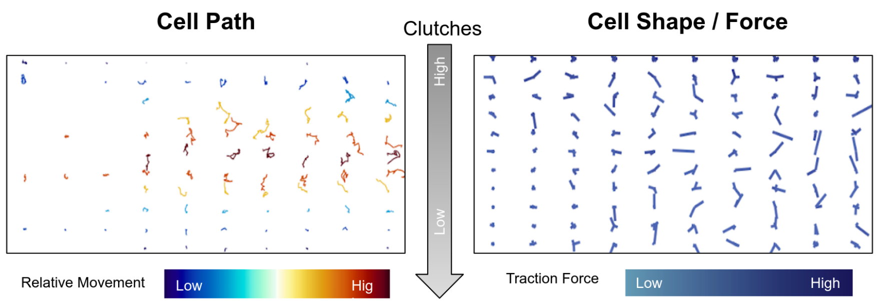


Figure 8.13: Our spatial assumptions from the interactive query were confirmed by creating relevant samples and interpolating between them. The cell appears to move and then stop as the number of clutches decrease in this part of the parameter space.

In order to apply visual analytics to the situation, we moved the clutch slider and clicked the "Create Sample" when the clutches were high, middle, and low (e.g. 750, 462, and 104). We exited interactive mode and returned to the main visualization where our samples were already running. We then used the "Interpolate" sampling method to create samples between the high, middle, and low samples. The final results for this analysis is pictured in Figure 8.12. Here, we see that the relative movement first increase and then decreases as he number of clutches decrease. This provides further evidence for our hypothesis generated from the interactive mode. It details how the interactive mode can work together with the visual analytics mode.

8.3.4 Performance

We evaluated the performance of VEF on the MSI supercomputer using an allocation of 12 nodes with 24 cores per node, a total of 288 available cores. Each simulation was executed on a single thread, meaning each composite sample used 10 threads on the same node. We tested our system by running multiple samples of the default cell configuration, each with the same random seed. For analysis, scientists are interested in the properties of a cell after 6 hours (21600 seconds) of simulation time. We used update timesteps of 10 seconds and 600 seconds of simulation time. One completed simulation takes about 9 minutes to run on a single node on the MSI supercomputer.

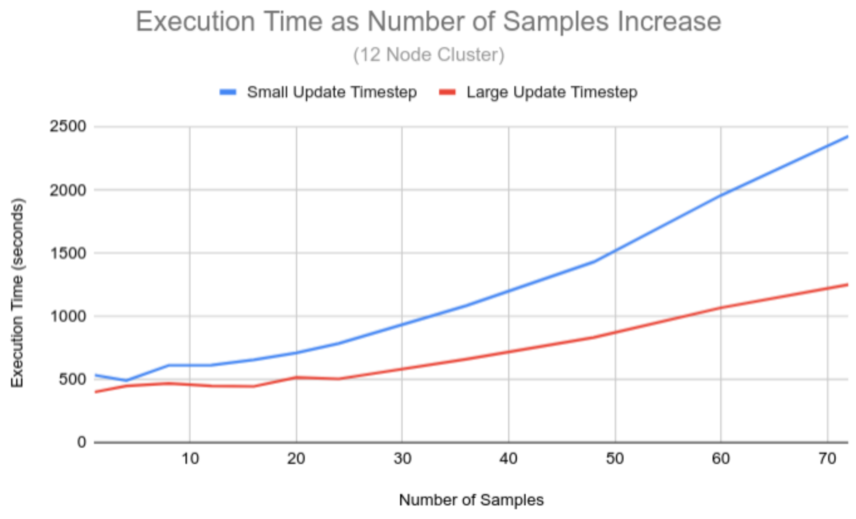


Figure 8.14: The two lines show that execution time scales linearly as we increase the number of samples. The higher slope for the low timestep suggests that there is higher visualization and data overhead due to more updates.

Figure 8.14 shows the scalability of our approach as the number of simulations increase, updating every 10 seconds (small update timestep) or every 600 (large update timestep) seconds. The trend in the completion time is linear as the number of samples increase. Since each node has 24 processes and each sample uses 10 threads, 12 nodes should be saturated at 24 samples. After 24 samples are created, processing time should increase linearly as a function of the number of cores. There should be little to no processing slow down before 24 samples. However, the small update timestep curve shows a linear increase before 24 samples. This would suggest that there is computational overhead in visualizing each node that causes a small delay in the asynchronous update. The amount of data transferred across the network to the WebSockets server is larger with lower update timesteps.

We tested this with a larger update of 600 seconds (every 10 minutes of simulation time), which prioritizes node processing over visualization update speed. In this case, we do not see a slope increase before 24 nodes. Larger timesteps, therefore, reduce the data load on the WebSocket server and the amount of processing that the visualization needs to do. Fortunately, the CMS simulation uses its own internal timestep, meaning that accuracy is not lost by changing the visualization update timestep.

8.4 Discussion

In this chapter we investigated our first DDEI that has the potential to handle the high-dimensional sparsity problem (Q2). The Virtual Experiment Framework enables users to visually interact with many multidimensional simulations (Q2, C3, C4) that are also large and spatially complex (Q1, C1, C2). The VEF enables users to steer simulations through user interaction (Q3) and via the exploration of both local and global dynamics (C5). In this section, we analyze our results with respect to spatial features (Section 8.4.1), high-dimensional sparsity (Section 8.4.2), and the state of the art in cell mechanics (Section 8.4.3).

8.4.1 Parameter Space Analysis of Spatial Features

One of our goals was to extend many of the techniques from the CMS Toolkit (Chapter 7). Specifically, we were interested in coupling the spatial representation with parameter space exploration. Our preliminary simulation results show promise in this area. We can visualize how changes in the parameters correlate with changes in output features. For example, Figure 8.7-b shows spatial relationships as a result of the dual parameter relationship by interpolating from a high number of clutches and motors to a low number of clutches and motors (e.g. the Bangasser et al experiment). From these initial results, we can see potential correlations between parameters and output variables like motility and traction force. Here we notice that shorter paths map to similar cell shapes. The traction force also appears to decrease as the number of motors and clutches decrease. More work needs to be done to verify this type of relationship with subject matter experts.

Another example of this type of spatial analysis is shown in Figure 8.13. Preliminary simulation results show that if we decrease the number of clutches we can see the cell path grow and then shrink. This appears to be a potential nonlinear relationship between the clutch parameter and cell movement for higher substrate stiffnesses. We can start to develop hypotheses for future investigation.

Interestingly, this potential local spatial relationship was noticed as a result of using our new Real-Time Black Box Steering approach (Sections 8.2.2, 8.2.5, and 8.3.3). Figure 8.12 details an example interactive mode exploration session for changing five different parameters. The purpose here was to gain intuition and build hypotheses about direct relationships between parameters and spatial features. When parameters changed, we could see spatial changes. For example, when the number of cell clutches went from 10 to 100, the cell appeared to stop. Similarly, when the motor pool parameter changed from 1000 to 218, the actin flow no longer moved. We hypothesize that these user-defined interactions have the potential to build intuition into unstructured spatial data sets. In Chapter 9 we propose a user study to investigate this ability to improve the understanding between input and output relationships.

The Real-Time Black Box Steering approach, however, is not without its limitations. Initial feedback from subject matter experts expressed both excitement and concern. One potential problem with this approach is that new simulations often have period of computational overhead. CMS, for example, needs a certain amount of ramp-up time to reach a steady state in order to be useful in analysis. Therefore, when a user is navigating, we need to use mature simulations so that results are not misleading. A simple approach would be to only allow a user to change parameters whose simulations are in steady state. Users could be notified with a visual indicator after a short ramp-up delay while moving throughout the space. Alternatively, it may be possible to preemptively ramp-up additional simulations in relevant parts of the space, building a readily available steady state ensemble of active simulations. In Chapter 9 we describe future work that could address these challenges by investigating user-driven sampling and anticipating user interaction.

8.4.2 Addressing High-Dimensional Sparsity

Another goal for this chapter was to work towards addressing the high-dimensional sparsity problem (Q2) for both local and global (C5) questions. For all previous chapters we focused on static ensembles, which as we have seen in Chapter 7, may not scale beyond a few dimensions for traditional visualization methods (like those used in the CMS Toolkit). The amount of samples needed to statistically analyze a result from a static ensemble grows exponentially due to the “Curse of Dimensionality”. Fortunately, the Virtual Experiment Framework uses a dynamic ensemble, so it is possible to sample the parameter space as needed. This means we only need to gather information that is relevant to the question of interest. We do not need to sample in areas of the parameter space that are unhelpful or uninteresting, saving valuable resources.

In the CMS Toolkit we systematically sampled four parameters: the number of clutches, the number of motors, the maximum polymerization, and the substrate stiffness. In this chapter we started exploring other parameters including the number of cell clutches and the clutch on-rate (see Figure 8.12). Informally, as hinted in Example Scenario 2 (Section 8.3.1), we explored the spatial relationships for the parameters total actin, maximum polymerization, clutch bond rupture force, and the motor unloaded velocity. Although not explicitly mentioned in this chapter, all the parameters were as accessible as the four parameters that were available through the CMS Toolkit.

Preliminary simulation results show that we are able to confirm experimental results (Section 8.3.2) and known trends in the CMS model (Section 8.3.2). These experiments had no requirement on the number of dimensions or resolution of dimensions. In fact, it was possible to drill down further in the parameter space using interpolation between output curves (see Figure 8.8), something that is not possible to do with a static ensemble.

Finally, we started to explore the parameter space dynamics using the “What if?” sampling method. Looking at a local neighborhood, we can start to investigate what happens if we simultaneously change multiple parameters at the same time. More work needs to be done here, but this work motivates the potential for discovering global trends in the data (see Figure 8.11). We explored changes made to all the parameters, scaling to a much higher dimensional space than the four dimensional space studied by the CMS Toolkit. To explore this further, Chapter 9 describes future work for visually mapping the dynamics of a parameter space and optimizing paths through the space based on user-defined questions.

8.4.3 Comparison with the State of the Art

In the area of molecular cell dynamics, CytoSIM [140] is a "cytoskeleton simulation suite designed to handle large systems of flexible filaments with associated proteins such as molecular motors." [141]. The toolkit enables users to visualize a live simulation and aggregate the results from multiple simulations into an HTML page to easily view images from all simulation results. During the live mode, users can reload a configuration file to update simulation parameters and see the results in-situ. Mechanisms for running many simulations on the supercomputer are available through command-line scripts as batch jobs.

As with CytoSIM, the VEF utilizes many supercomputing techniques like starting and stopping simulations. They both also enable viewing in-situ and offline data. For CytoSIM, visual interaction is limited to one local simulation, while the VEF can actively view simulations running on a supercomputer and does not require batch job configuration (see 8.3.2 for a comparison between the VEF and batch job processes). The main difference, however, is that CytoSIM is a cell mechanics model, while the VEF enables the exploration of models. In other words, we can use the VEF to explore the CytoSIM model. Consider that both CytoSIM and CMS are parameterized simulations with a set of inputs and spatial outputs. This means that the VEF could could potentially use CytoSIM as the underlying model instead of CMS, or in addition to CMS.

8.4.4 Summary

In summary, we investigated the ability to intuitively interact with a simulation while applying statistical analysis. Initial results show promising future directions for understanding cell migration and high-dimensional parameter spaces that are also spatial. More work needs to be done to further investigate the exciting potential of DDEIs.

Chapter 9

Conclusion and Future Work

In this dissertation we investigated data-driven exploratory interfaces and how they contextualize parameter spaces through user interaction. Our work takes a critical and logical first step toward addressing the large spatial data problem (Q1), the high-dimensional problem (Q2), and the meaningful interaction problem (Q3). This investigation provides a framework for further studying how to incorporate human intuition into Big Data applications using interactive visualization. In this chapter we summarize our contributions and discuss several follow-up research directions.

9.1 Summary of Contributions

9.1.1 Data Sampling and Rendering Techniques for Large Spatial Data

In order to scale several applications to meet the data-driven exploratory interfaces standards, we developed several key data sampling and rendering techniques. In Chapters 3 and 4 we discovered new techniques for interactively visualizing multiple large data sets, each with a memory footprint much larger than the available memory on the GPU. We introduced data variable streaming techniques in order to optimize data transfer based on a user's region of interest. We also reported results that inform how to optimize the use of render and compute GPUs on the same machine [49]. Our glyph-based rendering techniques and unstructured fluid flow sampling allowed us to reuse GPU memory efficiently, enabling us to render a large number of particles across many regions of interest and many simulations [9, 28, 49].

In Chapter 8, we enabled interactive in-situ sampling that scales to a large number of simultaneously running simulations. This contribution enables the steering of ensemble sampling based on user-interaction. Here we used similar streaming techniques as we did for the Cardiac Lead Application to send data over the network if it is needed for visualization. This reduces the amount of data we need to store and transfer, enabling a scalable interactive system. The results show that our sampling approaches scale linearly as a function of the number of distributed nodes (Section 8.3.4). They also show that streaming only variables of interest for some nodes improves performance, which is ideal as proposed in Bento Box, where we can see useful spatial trends using only a local neighborhood or a few key instances.

9.1.2 Searching a Parameter Space through Direct Manipulation

One of the most exciting contributions is our ability to extend the state of the art in regards to using direct manipulation for exploring a parameter space. Design by Dragging [22] is a direct manipulation interface that enables users to change input and output values on an instance to search for and navigate to the most similar relevant instance. Chapter 6 shows how we extend these ideas to the continuous space by adding direct manipulation techniques for predicting the gaps between ensemble instances. These predictions, or virtual data instances, are simultaneously validated by an uncertainty distribution of nearest neighbors. Therefore, two key contributions are (1) the intuitive exploration of a continuous space and (2) the ability to ask dynamic “What if?” questions via predictions [23].

Chapter 8 enables the real-time exploration of parameter spaces by integrating the ability to directly manipulate simulation samples. We also closely tied the sampling process to the visualization. These two together extend the work in Drag and Track (Chapter 6) to create an intuitive exploration and interactive real-time analysis loop. Therefore, we move beyond static ensembles, producing results based on user-defined questions that have the potential to be analyzed in context.

9.1.3 Annotating Parameter Spaces for Contextual Understanding

In Visual Parameter Space Analysis [32] a fundamental problem is that parameter spaces are continuous and ensembles are discrete. Since ensemble visualization and comparative visualization are major tools for exploring these spaces, we need to generate results that are understood in the context of uncertainty. Our work, as detailed in Chapters 4, 5, 6, and 8, moves toward using an ensemble to understand or annotate the parameter space. For example, in Chapter 4, we illustrated how to study a user-defined region of interest or unstructured feature across an ordered local neighborhood of samples [9, 28]. In Chapter 5, we used ensemble filtering, multiple linked views, superposition, and juxtaposition to generate unique views into specific shock physics questions [126].

Chapter 6 extends traditional methods to include dimensionally reduced spaces, enabling new methods for exploring and annotating continuous subspaces. Intuitively, users can explore the parameter space in locations between ensemble data instances. These predictions that are linked by nearby samples across views provides insights about the relationship between multiple subspaces. Highlighting these samples provides a new interactive way to visualize the uncertainty of user-defined questions [23].

Finally, Chapter 8 describes how we can realize and statistically analyse the uncertainty by sampling a neighborhood at these locations. We use the ensemble generated by the sampling as a mechanism for both visualizing the uncertainty and insight into where to sample next. Our sampling methods make it easy to recursively gather additional samples in areas of high uncertainty. In other words, users can visualize the global uncertainty of their questions throughout the parameter space, and then drill down to the appropriate level of detail, where necessary, through sampling.

9.1.4 Exploring Parameter Space Dynamics through Integrated Sampling

In Chapter 6 we started exploring how to understand parameter space dynamics by tracking changes across linked dimensionally reduced views. Users could directly modify a user-defined feature and follow the path of the predicted change, similar to sensitivity analysis, except for changes over a user-defined domain. Each track was a local view into the dynamics of a specific "What if?" question. Our results demonstrated that we can ask these types of questions when our interaction verified known trends in the model [23] for a user-defined domain.

The Virtual Experiment (Chapter 8) took this further by introducing the idea of global dynamics, which asks the same user-defined question throughout the parameter space. Using integrated sampling, we were able to run the simulations starting at seed points within the parameter space, similar to seeding particle paths in a fluid flow visualization (Chapters 3 and 4). Our preliminary simulation results show promise as we look towards studying relationships between the user-defined domain and the output data (Section 8.3.2).

9.1.5 Methods for Exploring Several Application Domains

While studying data-driven exploratory interfaces, we applied our techniques to cardiac lead design [9, 28, 49], shock physics [23, 126], and cell migration (Chapters 7 and 8). Our technical contributions helped us understand parameter spaces from multiple disciplines. For the cardiac lead design application, we *scaled* the rendering to handle large spatial simulations, viewed *together in context*, to understand the dynamics of *user-defined* features. We studied a high-dimensional problem in shock physics using the *scalability* of dimensionality reduction combined with *direct manipulation* to intuitively understand a parameter space analyzed in the *context of uncertainty*. Finally, we explored the cell migration problem using high-performance computing to *scale* the sampling and in-situ exploration of parameter spaces. This provided insight into specific *user-defined "What if?" questions* that can be *statistically evaluated* through further sampling. Our results across a wide variance of applications shows the practical utility of using DDEIs.

9.2 Future Research Directions

Further investigation into cell migration with subject matter experts. All the results from Chapter 8 were based on one user, the first author of this dissertation who created the virtual experiment application. The results, therefore, represent a preliminary investigation into the utility of our approaches. In visualization, however, it is important to validate a user interface or visual technique with the actual users of the application domain. Additional work needs to be done with experts in cell migration in order to validate our hypotheses. We anticipate that this would require several rounds of iteration with detailed feedback to make it useful for the intended users. Initial results need to be analyzed and compared with known trends, and we should evaluate the tool based on insights gained by expert intuition.

User study to evaluate the effectiveness of DDEIs. It is important to systematically show that DDEIs improve the understanding of input $\longleftrightarrow$ output relationships. The best way to study how humans use interfaces to understand ideas is to run a user study. We propose using the Cell Migration Simulator within the Virtual Experiment Framework. First, however, we would need several rounds of expert feedback to further develop the tool and questions of interest. For the user study, we would start by training users on both cell migration and use of the VEF. From there, we would lay out a series of tasks similar to the one described in Chapter 8. An example task might be, “Can you find this scenario within the parameter space by modifying multiple parameters?” We could analyze the task with followup questions including, “Do you have a better understanding of the parameters that you used to find this location?” Much more work would need to be done to develop such a user study.

Visually mapping the dynamics of a parameter space. This dissertation builds a framework for technically asking dynamics questions, however, it is not completely clear how to visualize the dynamics of a system. We used multiple line charts for each output variable (Section 8.3.2), but there is an opportunity to develop novel visualization techniques to reduce the dimensions of these high-dimensional trajectories. From here, we can cluster and develop a lower dimensional map of these relationships with respect to a domain of interest. This would be similar to seeding particles in a 2D space and following their trajectories to understand the flow of a system. In the lower dimensional space we could use techniques similar to Hummel et al [13] and [12] who cluster fluid flow trajectories in a classification space (e.g. principle components) to visually compare trends in a dynamic system. Although these techniques were created for spatial 2D and 3D data, it would be interesting to see the equivalent in higher dimensional spaces.

Optimizing paths through a parameter space. An interesting question worth asking about any space is, “How do we get from here to there?” For example, “What is the quickest way to the store?" or "How can I stop the spread of cancer for this patient?". These are pathfinding search questions, which may or may not have answers. There may be more than one answer for some questions, and for other questions there may not be a valid path. In Section 8.3.2 and Figure 8.8 we showed an example of these types of discontinuities. There may not be an obvious continuous path between states, however, it may be possible to go around the obstacle via another path. If we have a definition for invalid configurations, it may be possible to use configuration space techniques like the Probabilistic Roadmap [142] or Rapidly-exploring Random Trees [143]. This is especially true now that we have integrated sampling into the analysis workflow.

Inverse methods for searching a parameter space. Both Design by Dragging [22] and Drag and Track [23] employ inverse prediction methods for exploring a space using statistical methods. These methods are based on nearest neighbor searches of a previously sampled ensemble. Unfortunately, this does not work in a space that has not yet been sampled. We can definitely use an ensemble as a starting place for where to look, but key instances may be missing from the analysis. In fact, this is a major problem with machine learning methods, which rely and are optimized for existing data.

For example, in the case of cell migration (Chapter 8) there is no obvious way to enable inverse queries for real-time black box steering. One solution may be to look at a local neighborhood as a set of samples. From there we can use existing ensemble techniques or optimization and integration. We can perhaps calculate directional derivatives to the nearest neighbors and use a minimization technique like least squares to calculate a high dimensional gradient for the question of interest. From there, we can integrate and iterate along an optimized path. This may also be a method for finding local minima or obstacles in the parameter space that can be used along with route planning techniques to arrive at a desired destination.

User driven sampling. When sampling is integrated into a visual system, we are able to collect data on their queries. This will tell a system which parameters and output data are important, creating an ensemble of relevant questions. Here we can use this information to optimize the sampling to focus on what is important, therefore reducing the need for useless sampling in an irrelevant part of the space. It is important to also study any computational bias that could be introduced into the system. When does user driven sampling remove user exploration and reinforce known ideas?

Anticipating user interaction. Similar to user driven sampling, we are able to use data to analyze user interaction. Using statistical models, we may be able to predict what a user will do in certain situations. This would be similar to the Google search auto-complete, where we could provide options that would help the user explore the space. At the same time, in anticipating the user's next move, we could add more relevant samples to the system to analyze possible outcomes. This is similar to our real-time black box steering where we start multiple simulations in case the user navigates to the location in the parameter space. In fact, we could already use the real-time black box steering to provide more information to the use on which parameters might cause a decrease or increase in multiple output variables.

9.3 Concluding Remarks

We are reaching the edge of a new frontier as automation becomes tightly integrated into our society. The future relationship between the human and the computer is both exciting and uncertain, as with any worthy journey into the unknown. This dissertation investigates Data-Driven Exploratory Interfaces, suggesting that automation and human intelligence can work together to solve problems. We show that DDEIs overcome many limitations of traditional visualization approaches, enabling scalable and meaningful interaction in the context of uncertainty. We look forward to the future, anticipating that our proposed methods will enable new intuitive views into otherwise ill-defined and complex phenomena.

References

[1] Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A survey. *Mobile networks and applications*, 19(2):171–209, 2014.

[2] Anthony JG Hey, Stewart Tansley, Kristin M Tolle, et al. *The fourth paradigm: data-intensive scientific discovery*, volume 1. Microsoft research Redmond, WA, 2009.

[3] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[4] Megan Garcia. Racist in the machine: The disturbing implications of algorithmic bias. *World Policy Journal*, 33(4):111–117, 2016.

[5] Gina Neff and Peter Nagy. Automation, algorithms, and politics— talking to bots: Symbiotic agency and the case of tay. *International Journal of Communication*, 10:17, 2016.

[6] Sara Hajian, Francesco Bonchi, and Carlos Castillo. Algorithmic bias: From discrimination discovery to fairness-aware data mining. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2125–2126, 2016.

[7] Executive Office of the President, Cecilia Munoz, Domestic Policy Council Director, Megan (US Chief Technology Officer Smith (Office of Science, Technology

Policy)), DJ (Deputy Chief Technology Officer for Data Policy, Chief Data Scientist Patil (Office of Science, and Technology Policy)). *Big data: A report on algorithmic systems, opportunity, and civil rights.* Executive Office of the President, 2016.

[8] Tamara Munzner. *Visualization analysis and design.* CRC press, 2014.

[9] Seth A Johnson, Daniel Orban, Hakizumwami Birali Runesha, Lingyu Meng, Bethany Juhnke, Arthur Erdman, Francesca Samsel, and Daniel F Keefe. Bento box: An interactive and zoomable small multiples technique for visualizing 4d simulation ensembles in virtual reality. *Frontiers in Robotics and AI*, 6:61, 2019.

[10] Harald Obermaier and Kenneth I Joy. Future challenges for ensemble visualization. *IEEE Computer Graphics and Applications*, 34(3):8–11, 2014.

[11] Kristin Potter, Andrew Wilson, Peer-Timo Bremer, Dean Williams, Charles Doutriaux, Valerio Pascucci, and Chris R Johnson. Ensemble-vis: A framework for the statistical visualization of ensemble data. In *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*, pages 233–240. IEEE, 2009.

[12] Florian Ferstl, Kai Bürger, and Rüdiger Westermann. Streamline variability plots for characterizing the uncertainty in vector field ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):767–776, 2016.

[13] Mathias Hummel, Harald Obermaier, Christoph Garth, and Kenneth I Joy. Comparative visual analysis of lagrangian transport in cfd ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2743–2752, 2013.

[14] Mahsa Mirzargar, Ross T Whitaker, and Robert M Kirby. Curve boxplot: Generalization of boxplot for ensembles of curves. *IEEE transactions on visualization and computer graphics*, 20(12):2654–2663, 2014.

[15] Harald Obermaier, Kevin Bensema, and Kenneth I Joy. Visual trends analysis in time-varying ensembles. *IEEE transactions on visualization and computer graphics*, 22(10):2331–2342, 2016.

[16] Harald Piringer, Wolfgang Berger, and Jürgen Krasser. Hypermoval: Interactive visual validation of regression models for real-time simulation. In *Computer Graphics Forum*, volume 29, pages 983–992. Wiley Online Library, 2010.

[17] Thomas Torsney-Weir, Ahmed Saad, Torsten Moller, Hans-Christian Hege, Britta Weber, Jean-Marc Verbavatz, and Steven Bergner. Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1892–1901, 2011.

[18] Wolfgang Berger, Harald Piringer, Peter Filzmoser, and Eduard Gröller. Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. In *Computer Graphics Forum*, volume 30, pages 911–920. Wiley Online Library, 2011.

[19] Dany Vohl, David G Barnes, Christopher J Fluke, Govinda Poudel, Nellie Georgiou-Karistianis, Amr H Hassan, Yuri Benovitski, Tsz Ho Wong, Owen L Kaluza, Toan D Nguyen, et al. Large-scale comparative visualisation of sets of multidimensional data. *PeerJ Computer Science*, 2:e88, 2016.

[20] Johannes Kehrer and Helwig Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE transactions on visualization and computer graphics*, 19(3):495–513, 2012.

[21] James Ahrens, Berk Geveci, and Charles Law. Paraview: An end-user tool for large data visualization. *The visualization handbook*, 717, 2005.

[22] Dane Coffey, Chi-Lun Lin, Arthur G Erdman, and Daniel F Keefe. Design by dragging: An interface for creative forward and inverse design with simulation ensembles. *IEEE transactions on visualization and computer graphics*, 19(12):2783–2791, 2013.

[23] Daniel Orban, Daniel F Keefe, Ayan Biswas, James Ahrens, and David Rogers. Drag and track: A direct manipulation interface for contextualizing data instances within a continuous parameter space. *IEEE transactions on visualization and computer graphics*, 25(1):256–266, 2018.

[24] Shusen Liu, Dan Maljovec, Bei Wang, Peer-Timo Bremer, and Valerio Pascucci. Visualizing high-dimensional data: Advances in the past decade. In *Proc. Eurographics Conf. Visualization*, pages 20151115–127, 2015.

[25] Benjamin L Bangasser, Steven S Rosenfeld, and David J Odde. Determinants of maximal force transmission in a motor-clutch model of cell traction in a compliant microenvironment. *Biophysical journal*, 105(3):581–592, 2013.

[26] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation*, volume 89. siam, 2005.

[27] Kyungyoon Kim, John V Carlis, and Daniel F Keefe. Comparison techniques utilized in spatial 3d and 4d data visualizations: A survey and future directions. *Computers & Graphics*, 67:138–147, 2017.

[28] Daniel Orban, Seth Johnson, Hakizumwami Birali Runesha, Lingyu Meng, Bethany Juhnke, Arthur Erdman, Francesca Samsel, and Daniel F Keefe. Comparison of multiple large fluid-structure interaction simulations in virtual reality. In *2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)*, pages 110–111. IEEE, 2018.

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

[30] Gerard V Trunk. A problem of dimensionality: A simple example. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (3):306–307, 1979.

[31] Michael Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[32] Michael Sedlmair, Christoph Heinzl, Stefan Bruckner, Harald Piringer, and Torsten Möller. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2161–2170, 2014.

[33] Hakizumwami Birali Runesha, Bogdan Florin Tanasoiu, Georgi Subashki, Arthur G Erdman, and Daniel F Keefe. Fluid–structure interaction simulation of

cardiac leads in the heart: Developing a computational model for use in medical device design. *Journal of Medical Devices*, 10(3):030959, 2016.

[34] Melanie Tory and Torsten Moller. Human factors in visualization research. *IEEE transactions on visualization and computer graphics*, 10(1):72–84, 2004.

[35] Alex Endert. Semantic interaction for visual analytics: inferring analytical reasoning for model steering. *Synthesis Lectures on Visualization*, 4(2):1–99, 2016.

[36] Alex Endert, Patrick Fiaux, and Chris North. Unifying the sensemaking loop with semantic interaction. In *IEEE Workshop on Interactive Visual Text Analytics for Decision Making at VisWeek 2011*, 2011.

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

[38] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE symposium on visual languages*, pages 336–343. IEEE, 1996.

[39] 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. *IEEE transactions on visualization and computer graphics*, 20(12):2644–2653, 2014.

[40] Alex Endert, Lauren Bradel, and Chris North. Beyond control panels: Direct manipulation for visual analytics. *IEEE computer graphics and applications*, 33(4):6–13, 2013.

[41] Eli T Brown, Jingjing Liu, Carla E Brodley, and Remco Chang. Dis-function: Learning distance functions interactively. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 83–92. IEEE, 2012.

[42] Dong Hyun Jeong, Caroline Ziemkiewicz, Brian Fisher, William Ribarsky, and Remco Chang. ipca: An interactive system for pca-based visual analytics. In *Computer Graphics Forum*, volume 28, pages 767–774. Wiley Online Library, 2009.

[43] Alex Endert, Chao Han, Dipayan Maiti, Leanna House, and Chris North. Observation-level interaction with statistical models for visual analytics. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 121–130. IEEE, 2011.

[44] Alex Endert, Remco Chang, Chris North, and Michelle Zhou. Semantic interaction: Coupling cognition and computation through usable interactive analytics. *IEEE Computer Graphics and Applications*, 35(4):94–99, 2015.

[45] Wenbin He, Junpeng Wang, Hanqi Guo, Ko-Chih Wang, Han-Wei Shen, Mukund Raj, Youssef SG Nashed, and Tom Peterka. Insitunet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):23–33, 2019.

[46] James Ahrens, Sébastien Jourdain, Patrick O'Leary, John Patchett, David H Rogers, and Mark Petersen. An image-based approach to extreme scale in situ visualization and analysis. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 424–434. IEEE Press, 2014.

[47] Adriana Schulz, Jie Xu, Bo Zhu, Changxi Zheng, Eitan Grinspun, and Wojciech Matusik. Interactive design space exploration and optimization for cad models. *ACM Transactions on Graphics (TOG)*, 36(4):157, 2017.

[48] Benjamin L Bangasser, Ghaidan A Shamsan, Clarence E Chan, Kwaku N Opoku, Erkan Tüzel, Benjamin W Schlichtmann, Jesse A Kasim, Benjamin J Fuller, Brannon R McCullough, Steven S Rosenfeld, et al. Shifting the optimal stiffness for cell migration. *Nature communications*, 8:15313, 2017.

[49] Bethany Tourek, Daniel Orban, Bogden Tanasoiu, Hakizumwami Birali Runesha, Daniel F. Keefe, and Arthur G. Erdman. Poster: Inverse design process: New

methodology to design medical devices with big data. Minnesota Supercomputing Institute Research Exhibition, April 2016.

[50] Le Liu, Alexander P Boone, Ian T Ruginski, Lace Padilla, Mary Hegarty, Sarah H Creem-Regehr, William B Thompson, Cem Yuksel, and Donald H House. Uncertainty visualization by representative sampling from prediction ensembles. *IEEE transactions on visualization and computer graphics*, 23(9):2165–2178, 2017.

[51] Ayan Biswas, Guang Lin, Xiaotong Liu, and Han-Wei Shen. Visualization of time-varying weather ensembles across multiple resolutions. *IEEE transactions on visualization and computer graphics*, 23(1):841–850, 2017.

[52] Ayan Biswas, Soumya Dutta, Han-Wei Shen, and Jonathan Woodring. An information-aware framework for exploring multivariate data sets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2683–2692, 2013.

[53] Junpeng Wang, Xiaotong Liu, Han-Wei Shen, and Guang Lin. Multi-resolution climate ensemble parameter analysis with nested parallel coordinates plots. *IEEE transactions on visualization and computer graphics*, 23(1):81–90, 2017.

[54] Alexander Kumpf, Bianca Tost, Marlene Baumgart, Michael Riemer, Rüdiger Westermann, and Marc Rautenhaus. Visualizing confidence in cluster-based ensemble weather forecast analyses. *IEEE transactions on visualization and computer graphics*, 24(1):109–119, 2018.

[55] Dane Coffey, Fedor Korsakov, Marcus Ewert, Haleh Hagh-Shenas, Lauren Thorson, A Ellingson, D Nuckley, and Daniel F Keefe. Visualizing motion data in virtual reality: Understanding the roles of animation, interaction, and static presentation. In *Computer Graphics Forum*, volume 31, pages 1215–1224. Wiley Online Library, 2012.

[56] Martin Steiger, Jürgen Bernard, Sebastian Mittelstädt, Hendrik Lücke-Tieke, Daniel Keim, Thorsten May, and Jörn Kohlhammer. Visual analysis of time-series similarities for anomaly detection in sensor networks. In *Computer graphics forum*, volume 33, pages 401–410. Wiley Online Library, 2014.

[57] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10:66–71, 2009.

[58] John Patchett, Galen Gisler, Boonthanome Nouanesengsy, David H Rogers, Greg Abram, Francesca Samsel, Karen Tsai, and Terece Turton. Visualization and analysis of threats from asteroid ocean impacts. Supercomputing, 2016.

[59] Joe Marks, Brad Andalman, Paul A Beardsley, William Freeman, Sarah Gibson, Jessica Hodgins, Thomas Kang, Brian Mirtich, Hanspeter Pfister, Wheeler Ruml, et al. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 389–400. ACM Press/Addison-Wesley Publishing Co., 1997.

[60] Divya Banesh, Joseph A Schoonover, James P Ahrens, and Bernd Hamann. Extracting, visualizing and tracking mesoscale ocean eddies in two-dimensional image sequences using contours and moments.

[61] MJ Enenhofer. Spatial-query-by-sketch. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 60–67. IEEE, 1996.

[62] Mehmet Ersin Yumer, Siddhartha Chaudhuri, Jessica K Hodgins, and Levent Burak Kara. Semantic shape editing using deformation handles. *ACM Transactions on Graphics (TOG)*, 34(4):86, 2015.

[63] 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.

[64] National Research Council et al. Committee on mathematical foundations of verification, validation, and uncertainty quantification; board on mathematical sciences and their applications, division on engineering and physical sciences, national research council: Assessing the reliability of complex models: Mathematical and statistical foundations of verification, validation, and uncertainty quantification the national academies http://www. espc. oar. noaa. gov/sites/espc/documents. *NAS% 20Report% 20Uncertanty% 20Quantification*, 2013395, 2012.

[65] Andrea Saltelli. Global sensitivity analysis: an introduction. In *Proc. 4th International Conference on Sensitivity Analysis of Model Output (SAMO'04)*, pages 27–43, 2004.

[66] Marco Cavallo and Çagatay Demiralp. A visual interaction framework for dimensionality reduction based data exploration. *ACM Human Factors in Computing Systems (CHI)*, 2018.

[67] Tom Ngo, Doug Cutrell, Jenny Dana, Bruce Donald, Lorie Loeb, and Shunhui Zhu. Accessible animation and customizable graphics via simplicial configuration modeling. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 403–410. ACM Press/Addison-Wesley Publishing Co., 2000.

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

[69] Dominik Sacha, Leishi Zhang, Michael Sedlmair, John A Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C North, and Daniel A Keim. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE transactions on visualization and computer graphics*, 23(1):241–250, 2017.

[70] Alex Endert, Patrick Fiaux, and Chris North. Semantic interaction for sensemaking: inferring analytical reasoning for model steering. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2879–2888, 2012.

[71] Ji Soo Yi, Rachel Melton, John Stasko, and Julie A Jacko. Dust & magnet: multivariate information visualization using a magnet metaphor. *Information visualization*, 4(4):239–256, 2005.

[72] Bahador Saket, Hannah Kim, Eli T Brown, and Alex Endert. Visualization by demonstration: An interaction paradigm for visual data exploration. *IEEE transactions on visualization and computer graphics*, 23(1):331–340, 2017.

[73] 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. 24(2):76–85, March/April 2004.

[74] Akira Kageyama, Yuichi Tamura, and Tetsuya Sato. Visualization of vector field by virtual reality. *Progress of Theoretical Physics Supplement*, 138:665–673, 2000.

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

[76] Andrew S Forsberg, David H Laidlaw, Andries Van Dam, Robert M Kirby, George E Karniadakis, and Jonathan L Elion. Immersive virtual reality for visualizing flow through an artery. In *Proceedings of the conference on Visualization'00*, pages 457–460. IEEE Computer Society Press, 2000.

[77] Bret Jackson. Minvr, 2017.

[78] CIBC, 2016. ImageVis3D: An interactive visualization software system for large-scale volume data. Scientific Computing and Imaging Institute (SCI), Download from: http://www.imagevis3d.org.

[79] Alexander Bock, Emil Axelsson, Carter Emmart, Masha Kuznetsova, Charles Hansen, and Anders Ynnerman. Openspace: Changing the narrative of public dissemination in astronomical visualization from what to how. *IEEE computer graphics and applications*, 38(3):44–57, 2018.

[80] JM Huang, Soh-Khim Ong, and Andrew YC Nee. Visualization and interaction of finite element analysis in augmented reality. *Computer-Aided Design*, 84:1–14, 2017.

[81] Ridha Hambli, Abdessalam Chamekh, and Hédi Bel Hadj Salah. Real-time deformation of structure using finite element and neural networks in virtual reality applications. *Finite elements in analysis and design*, 42(11):985–991, 2006.

[82] Cheng Tzong-Ming and Tsung-Han Tu. A fast parametric deformation mechanism for virtual reality applications. *Computers & Industrial Engineering*, 57(2):520–538, 2009.

[83] Michele Tonutti, Gauthier Gras, and Guang-Zhong Yang. A machine learning approach for real-time modelling of tissue deformation in image-guided neurosurgery. *Artificial intelligence in medicine*, 80:39–47, 2017.

[84] Moohyun Cha, Soonhung Han, Jaikyung Lee, and Byungil Choi. A virtual reality based fire training simulator integrated with fire dynamics data. *Fire Safety Journal*, 50:12–24, 2012.

[85] Seth Alan Johnson. *Palpable Visualizations: Techniques for Creatively Designing Discernible and Accessible Visualizations Grounded in the Physical World.* PhD thesis, University of Minnesota, 2020.

[86] Thomas Richter. *Fluid-structure Interactions: Models, Analysis and Finite Elements*, volume 118. Springer, 2017.

[87] Christoph Garth and Kenneth I Joy. Fast, memory-efficient cell location in unstructured grids for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1541–1550, 2010.

[88] Michael N Katehakis and Arthur F Veinott Jr. The multi-armed bandit problem: decomposition and computation. *Mathematics of Operations Research*, 12(2):262–268, 1987.

[89] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices.* John Wiley & Sons, 2011.

[90] Bob Nagler, Brice Arnold, Gary Bouchard, Richard F Boyce, Richard M Boyce, Alice Callen, Marc Campell, Ruben Curiel, Eric Galtier, Justin Garofoli, et al. The matter in extreme conditions instrument at the linac coherent light source. *Journal of synchrotron radiation*, 22(3):520–525, 2015.

[91] Camelia V Stan, Christine M Beavers, Martin Kunz, and Nobumichi Tamura. X-ray diffraction under extreme conditions at the advanced light source. *Quantum Beam Science*, 2(1):4, 2018.

[92] Bob Nagler, Andrew Higginbotham, Giles Kimminau, William Murphy, Thomas Whitcher, Justin Wark, James Hawreliak, Dan Kalantar, Richard Lee, Hector Lorenzana, et al. Prospects for using x-ray free-electron lasers to investigate shock-compressed matter. In *AIP Conference Proceedings*, volume 955, pages 1333–1336. AIP, 2007.

[93] Akel Hashim. Visar analysis theory and mathematical formalism. 2016.

[94] U Zastrau, M McMahon, K Appel, C Baehtz, E Brambrink, R Briggs, T Butcher, B Cauble, B Chen, H Damker, C Deiter, J Eggert, K Falk, L Fletcher, S H Glenzer, S Göde, M Harmand, A Higginbotham, Z Konôpková, D Kraus, H.-P. Liermann, M Nakatsutsumi, Pelka A., G Priebe, R Redmer, A Schropp, R Smith, P Sperling, I Thorpe, and S Toleikis. Conceptual Design Report: Dynamic Laser Compression Experiments at the HED Instrument of European XFEL. (REPORT-2017-004. XFEL.EU TR-2017-001), 2017.

[95] Cindy Bolme, Andy Mackinnon, and Siegfried Glenzer. Fourth user workshop on high-power lasers at the linac coherent light source, 2017.

[96] Anders Madsen. Conceptual design report: Scientific instrument mid. (XFEL.EU TR-2011-008), 2011.

[97] L. M. Barker and R. E. Hollenbach. Laser interferometer for measuring high velocities of any reflecting surface. *Journal of Applied Physics*, 43:4669–4675, November 1972.

[98] DD Bloomquist and SA Sheffield. Optically recording interferometer for velocity measurements with subnanosecond resolution. *Journal of Applied Physics*, 54(4):1717–1722, 1983.

[99] PM Celliers, DK Bradley, GW Collins, DG Hicks, TR Boehly, and WJ Armstrong. Line-imaging velocimeter for shock diagnostics at the omega laser facility. *Review of scientific instruments*, 75(11):4916–4929, 2004.

[100] CA Bolme and KJ Ramos. Line-imaging velocimetry for observing spatially heterogeneous mechanical and chemical responses in plastic bonded explosives during impact. *Review of Scientific Instruments*, 84(8):083903, 2013.

[101] Quintin Johnson, A Mitchell, R Norris Keeler, and L Evans. X-ray diffraction during shock-wave compression. *Physical Review Letters*, 25(16):1099, 1970.

[102] Quintin Johnson, Arthur Mitchell, and L Evans. X-ray diffraction evidence for crystalline order and isotropic compression during the shock-wave process. *Nature*, 231(5301):310, 1971.

[103] Quintin Johnson, Arthur C Mitchell, and L Evans. X-ray diffraction study of single crystals undergoing shock-wave compression. *Applied Physics Letters*, 21(1):29–30, 1972.

[104] DH Kalantar, JF Belak, GW Collins, JD Colvin, HM Davies, JH Eggert, TC Germann, J Hawreliak, BL Holian, K Kadau, et al. Direct observation of the α- ε transition in shock-compressed iron via nanosecond x-ray diffraction. *Physical review letters*, 95(7):075502, 2005.

[105] Stefan J Turneaure, YM Gupta, K Zimmerman, K Perkins, CS Yoo, and G Shen. Real-time microstructure of shocked lif crystals: Use of synchrotron x-rays. *Journal of Applied Physics*, 105(5):053520, 2009.

[106] LB Fletcher, HJ Lee, T Döppner, E Galtier, B Nagler, P Heimann, C Fortmann, S LePape, T Ma, M Millot, et al. Ultrabright x-ray laser scattering for dynamic warm dense matter physics. *Nature photonics*, 9(4):274, 2015.

[107] AE Gleason, CA Bolme, HJ Lee, B Nagler, E Galtier, D Milathianaki, J Hawreliak, RG Kraus, JH Eggert, DE Fratanduono, et al. Ultrafast visualization of crystallization and grain growth in shock-compressed sio 2. *Nature communications*, 6:8191, 2015.

[108] M. G. Gorman, R. Briggs, E. E. McBride, A. Higginbotham, B. Arnold, J. H. Eggert, D. E. Fratanduono, E. Galtier, A. E. Lazicki, H. J. Lee, H. P. Liermann, B. Nagler, A. Rothkirch, R. F. Smith, D. C. Swift, G. W. Collins, J. S. Wark, and

M. I. McMahon. Direct observation of melting in shock-compressed bismuth with femtosecond x-ray diffraction. *Phys. Rev. Lett.*, 115:095701, Aug 2015.

[109] Stefan Mangold. Fully automated beamline control system for xas beamlines. *Journal of Synchrotron Radiation*, 25(4), 2018.

[110] Santiago José Alejandro Figueroa, Douglas Bezerra Beniz, Junior Cintra Mauricio, James Rezende Piton, Stephen A Parry, and Giannantonio Cibin. Steps towards xafs beamline automation and remote access. *Journal of synchrotron radiation*, 25(4), 2018.

[111] Siniša Veseli, Nicholas Schwarz, and Collin Schmitz. APS *Data Management System. Journal of Synchrotron Radiation*, 25(5):1574–1580, Sep 2018.

[112] Owen Arnold, Jean-Christophe Bilheux, JM Borreguero, Alex Buts, Stuart I Campbell, L Chapon, Mathieu Doucet, N Draper, R Ferraz Leal, MA Gigg, et al. Mantid—data analysis and visualization package for neutron scattering and μ sr experiments. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 764:156–166, 2014.

[113] Brian H Toby and Robert B Von Dreele. Gsas-ii: the genesis of a modern open-source all purpose crystallography software package. *Journal of Applied Crystallography*, 46(2):544–549, 2013.

[114] Luca Lutterotti, S Matthies, and HR Wenk. Maud: a friendly java program for material analysis using diffraction. *IUCr: Newsletter of the CPD*, 21(14-15), 1999.

[115] AP Hammersley. Fit2d: a multi-purpose data reduction, analysis and visualization program. *Journal of Applied Crystallography*, 49(2):646–652, 2016.

[116] Clemens Prescher and Vitali B Prakapenka. Dioptas: a program for reduction of two-dimensional x-ray diffraction data and data exploration. *High Pressure Research*, 35(3):223–230, 2015.

[117] Jonathan C Roberts. Exploratory visualization with multiple linked views. In *Exploring geovisualization*, pages 159–180. Elsevier, 2005.

[118] Daniel A Keim and Hans-Peter Kriegel. Visualization techniques for mining large databases: A comparison. *IEEE Transactions on Knowledge & Data Engineering*, (6):923–938, 1996.

[119] Jonathan Cox and Michael Lindell. Visualizing uncertainty in predicted hurricane tracks. *International Journal for Uncertainty Quantification*, 3(2), 2013.

[120] Cameron Tauxe, David Rogers, Dan Orban, and Terry Turton. Cinema components. https://github.com/cinemascience/cinema_components, 2018.

[121] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *IEEE Transactions on Visualization & Computer Graphics*, (12):2301–2309, 2011.

[122] David Rogers, Jon Woodring, James Ahrens, and John Patchett. Cinema database specification dietrich release v1.1. https://cinemascience.org/index.php/getting-started/, 2017.

[123] D. Damiani, M. Dubrovin, I. Gaponenko, W. Kroeger, T. J. Lane, A. Mitra, C. P. O'Grady, A. Salnikov, A. Sanchez-Gonzalez, D. Schneider, and C. H. Yoon. Linac Coherent Light Source data analysis using *psana*. *Journal of Applied Crystallography*, 49(2):672–679, Apr 2016.

[124] Giannis Ashiotis, Aurore Deschildre, Zubair Nawaz, Jonathan P. Wright, Dimitrios Karkoulis, Frédéric Emmanuel Picca, and Jérôme Kieffer. The fast azimuthal integration Python library: *pyFAI*. *Journal of Applied Crystallography*, 48(2):510–519, Apr 2015.

[125] P. Hart, S. Boutet, G. Carini, A. Dragone, B. Duda, D. Freytag, G. Haller, R. Herbst, S. Herrmann, C. Kenney, J. Morse, M. Nordby, J. Pines, N. van Bakel, M. Weaver, and G. Williams. The Cornell-SLAC pixel array detector at LCLS. In *2012 IEEE Nuclear Science Symposium and Medical Imaging Conference Record (NSS/MIC)*, pages 538–541. IEEE, oct 2012.

[126] D Orban, D Banesh, C Tauxe, CM Biwer, A Biswas, R Saavedra, C Sweeney, RL Sandberg, CA Bolme, J Ahrens, and D Rogers. Cinema: Bandit: a visualization application for beamline science demonstrated on xfel shock physics experiments. *Journal of Synchrotron Radiation*, 27(1), 2020.

[127] Fernando Vieira Paulovich, Danilo Medeiros Eler, Jorge Poco, Charl P Botha, Rosane Minghim, and Luis Gustavo Nonato. Piece wise laplacian-based projection for interactive data exploration and organization. In *Computer Graphics Forum*, volume 30, pages 1091–1100. Wiley Online Library, 2011.

[128] Bing Wang and Klaus Mueller. The subspace voyager: Exploring high-dimensional data along a continuum of salient 3d subspaces. *IEEE transactions on visualization and computer graphics*, 24(2):1204–1222, 2018.

[129] Stephan Pajer, Marc Streit, Thomas Torsney-Weir, Florian Spechtenhauser, Torsten Möller, and Harald Piringer. Weightlifter: Visual weight space exploration for multi-criteria decision making. *IEEE transactions on visualization and computer graphics*, 23(1):611–620, 2017.

[130] JM Boteler and DP Dandekar. Dynamic response of two strain-hardened aluminum alloys. *Journal of applied physics*, 100(5):054902, 2006.

[131] Gordon R. Johnson and William H. Cook. A Constitutive Model and Data for Metals Subjected to Large Strains, High Strain Rates and High Temperatures. In *Seventh International Symposium on Ballistics*, pages 541–547, The Hague, The Netherlands, apr 1983.

[132] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524. ACM, 1968.

[133] Donald E Burton. FLAG, a Multi-Dimensional, Multiple Mesh, Adaptive Free-Lagrange, Hydrodynamics Code. In *Nuclear Explosives Code Developers Conference*, 1992.

[134] Donald E Burton. Lagrangian Hydrodynamics in the FLAG Code. In *Symposium on Advanced Numerical Methods for Lagrangian Hydrodynamics*, Los Alamos, NM, USA, LA-UR-07-7547, 2007.

[135] E J Caramana, D E Burton, M J Shashkov, and P P Whalen. The Construction of Compatible Hydrodynamics Algorithms Utilizing Conservation of Total Energy. *Journal of Computational Physics*, 146:227–262, 1998.

[136] Elisa Portes dos Santos Amorim, Emilio Vital Brazil, Joel Daniels, Paulo Joia, Luis Gustavo Nonato, and Mario Costa Sousa. ilamp: Exploring high-dimensional spacing through backward multidimensional projection. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 53–62. IEEE, 2012.

[137] Clarence E Chan and David J Odde. Traction dynamics of filopodia on compliant substrates. *Science*, 322(5908):1687–1691, 2008.

[138] Tim Mitchison and Marc Kirschner. Cytoskeletal dynamics and nerve growth. *Neuron*, 1(9):761–772, 1988.

[139] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. Action synopsis: pose selection and illustration. *ACM Transactions on Graphics (TOG)*, 24(3):667–676, 2005.

[140] Francois Nedelec and Dietrich Foethke. Collective langevin dynamics of flexible cytoskeletal fibers. *New Journal of Physics*, 9(11):427, 2007.

[141] Francois Nedelec. Cytosim, 2021.

[142] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[143] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.