Scalable Visualization of Multivariate Spatiotemporal Distributions from Scientific Simulation Data

By

TYSON ALLAN NEUROTH

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Kwan-Liu Ma, Chair

Julian Panetta

Patrice Koehl

Committee in Charge

2023

To all who are interested.

Contents

Abstract

Scalable Visualization of Multivariate Spatiotemporal Distributions from Scientific Simulation Data

As computational power increases, scientists simulate complex physical systems at a larger scale. The resulting data is valuable but cumbersome due to its size and complexity. This dissertation is driven primarily by exasperating data visualization challenges in two domains: gyrokinetic particle-in-cell plasma physics simulations devoted to solving problems in tokamak fusion energy production and turbulent combustion simulations devoted to improving fuel efficiency and the formulation of new environmentally friendly fuels.

Challenges in these domains stem from data size and complexity due to high dimensionality, large numbers of simulation grid points, large numbers of particles, and the complicated joint physical and statistical interpretations of particles in particle-in-cell plasma simulations. In both cases, due to chaos and turbulence, the systems can evade predictability and exhibit emergent properties and pattern formations that are not yet well understood through causal analysis starting from earlier states and parameters.

Ultimately, the simulations produce large amounts of spatially distributed and multivariate data elements that jointly model the states of the simulations at each time step. The data is precious to researchers; however, since large-scale and complex data distributions characterize the state spaces, it is non-trivial and time-consuming for scientists to comb through and absorb the data. Furthermore, since the raw data is too large to manage with current I/O limitations, data storage systems, and networks, researchers are forced to make uncertain compromises as they reduce the data.

Our contributions ameliorate these problems through data summarization and interactive visualization. First, we develop methods and systems for visualizing particle data and phase-space particle distribution functions from tokamak fusion simulations. To accomplish this, we introduce a novel approach for the interactive visualization of large sets of spatially organized histograms. The approach is leveraged into a visualization system tailored for the study of phase-space particle distribution functions and the evolution of the statistical weights of the simulation superparticles. We then develop tools for visualizing large sets of multivariate trajectories, such as the phase-space particle trajectories in fusion simulations, as well as trajectories of particles from linear accelerator simulations.

Finally, we develop an approach for spatial statistical visualization of multivariate volume data from turbulent combustion simulations. The approach is based on a novel dynamic, nested, hierarchical spatial decomposition method based on isobands, connected components, and the restricted centroidal Voronoi tessellation. Our tessellation is restricted between level sets so that the Voronoi tessellation conforms to the boundaries of surface-based features. We leverage the tessellation in a custom visualization system for interactive local spatial statistical analysis. The system is designed in collaboration with expert combustion scientists.

Acknowledgments

First, I thank the scientists who contributed their time collaborating with us. From Princeton Plasma Physics Laboratory, I would like to thank Choongseock "CS" Chang, Stéphane Ethier, Robert Hager, Michael Churchill, and Weixing Wang, who provided data and collaborated with us on the fusion visualization research. From Sandia National Laboratory, I would like to thank Jacqueline Chen and Martin Rieth, who were gracious enough to meet with us weekly for several years to collaborate on our work focused on Turbulent combustion simulations, as well as Aditya Konduri, and MK Lee. The scientists we worked with made this work all worthwhile in the end. Moreover, I would like to thank Dave Pugmire, an excellent host during my Oak Ridge National Laboratory internship.

I thank my colleagues from VIDI labs at UC Davis, who provided advice, encouragement, inspiration, and friendship throughout the years. I would also like to thank Jessica Stoller and Alyssa Bates, the CS graduate advisors, and graduate studies advisor, Wallace Woods. Next, I thank my Ph.D. committee and my qualifying exam committee, which includes, aside from each member of my Ph.D. committee, Professor Nina Amenta and Professor Ralph Aldredge.

Next, I thank my family, including my mother, Mara, whose support was instrumental to my life's successes, and my sisters, Michelle and Erin, whom I look up to and who have inspired and motivated me.

I also thank the US Department of Energy (DOE), who funded most of this work, and Oak Ridge National Laboratory, Argonne National Laboratory, and National Energy Research Scientific Computing Center (NERSC) for providing us with computational resources.

Finally, I thank my advisor, Distinguished Professor Kwan-Liu Ma, for pushing me to succeed, making all of this work possible, and supporting me throughout the years.

Chapter 1

Introduction

Large simulations of spatial, physical systems are pushing the frontiers of modern science and engineering. However, the resulting data is often challenging to analyze due to a range of factors, such as unpredictability from chaos [1] and turbulence [2], non-linearity, high dimensionality [3, 4, 5], large dynamic ranges of scales [6, 7], multi-physics interactions [8], and the large compositions of interacting components that all together model the states of the simulated systems.

As supercomputing power, measured in floating point operations per second (FLOPS), advances and larger-scale systems become feasible to model, the increasingly large amounts of resulting data are more and more challenging to manage and visualize. For reference, the first gigascale (10^9) computer (Cray-2) released in 1985 [9], the first exascale (10^{18}) computer (Fugaku) released in 2020 [10], and the DOE's first exascale computer (Aurora) is set to release in late 2022. While exascale computing marks a rough threshold for simulating important scientific phenomena [11], exploring simulation data is still a significant challenge even at orders of magnitude smaller. While datasets are reaching the exascale, workstations used for analysis remain bottlenecked by system memory at the gigascale.

This dissertation focuses on ameliorating this discrepancy and enabling more value extraction from large spatially distributed and multivariate data through interactive visualization. The challenges we focus on stem from both human and hardware constraints. The human constraints stem from people's limited time and perceptual and cognitive abilities. Hardware constraints come primarily from the limited storage space, network bandwidth, compute power, and system memory of workstations used by researchers for visualization and analysis.

1.1 What is visualization and why is it needed?

Visualization is an interactive process that forms a feedback loop between persons, data, images, and analysis. Information is encoded through different channels, such as shape, size, and color, and familiar visual cues, such as lighting and perspective, are used to convey spatial and structural information. The visually encoded information is effectively absorbed thanks to the sophisticated visual processing systems of human beings and our cognitive abilities, which help us to integrate the information into abstract mental models and reason analytically. Further, with its exceptional general intelligence, the mind can quickly notice anomalies, patterns, and trends, including unexpected ones that algorithms coded with prior knowledge might miss.

What is gained from visualization is commonly categorized as insight. Since the insight gained depends on the persons, their prior knowledge, and intuition, it is non-trivial to precisely define and measure [12, 13, 14]. The difficulties in measuring insights attributed to visualization stem from the complexity of the human mind and the often long and complex processes of thought that steer open-ended or uncertain scientific workflows.

One comment from a collaborating scientist has stood out in its simplicity, "It is just useful to know what the particles are doing." In context, the behaviors of the particles (what they are doing) are non-trivial to grasp since there are billions of them, and they have complex interpretations as statistical superparticles. Greater awareness and stronger intuition offer extensive benefits, from helping to validate and debug the simulation to more effective brainstorming toward the formation of scientific hypotheses, to name a few.

While "seeing what happened" is an obvious goal, it is not simple in practice when the data is vast, complex, and multifaceted; many things happened. In practice, the visualization process usually needs to be guided and focused rather than aimless to be efficient yet not so narrow in focus as to suppress unexpected but insightful information.

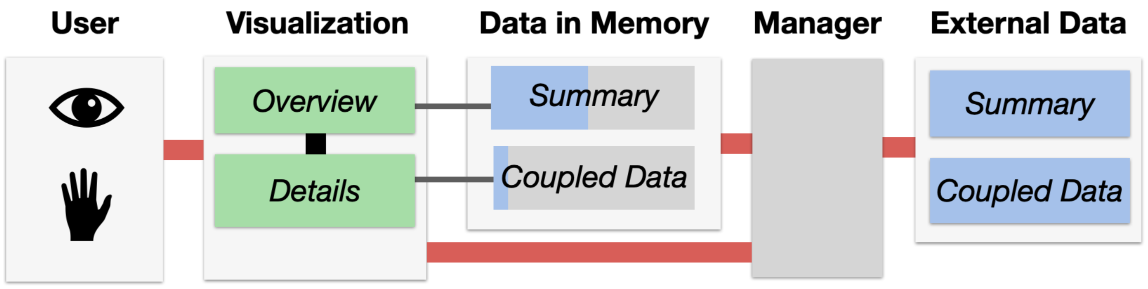


Figure 1.1. An illustrative model for scalable interactive visualization. The complete data is too large to fit in system memory and too complex to view with all its details through a single visualization. The problem is addressed through data summarization, organized and coupled data structures, and interaction through the visualization system. The visualization system relies on the *overview first, details on demand* model. Combined with the *focus+context* model, this allows the user to maintain important context to relate the current visualization views with the encompassing data space so that they can navigate and drill down into details following an intuitive and coherent workflow. The summary data is helpful to represent a large portion of the raw data in a compact size that can fit in system memory—the overviews generated from the summary data act as an interface that facilitates drilling down into details. The details, or raw data, which are much too large to fit in system memory, are selectively extracted for visualization as the user drills down. A data manager dynamically facilitates the transfer of data from external sources, such as local hard disks or remote data through networks, into system memory and manages its persistence based on the demand driven by the interactive visualization workflow. In order to ensure details can be extracted efficiently from the complete data, they are coupled with the summary data through an organized and indexed data layout.

1.2 Scalable Interactive Visualization

Our overall goal is to create visualization systems that continue to operate effectively as the size and complexity of datasets increase. In other words, the primary form of scalability we focus on is the scalability of a visualization system, in terms of effectiveness, as a function of data size and complexity. The effectiveness of the visualization is dependent on both human and hardware constraints. Towards this end, multiple forms of scalability come into play. Richer et al. surveyed the holistic concept of scalability in visualization and proposed a framework for modeling it [15]. In this dissertation, we consider *perceptual and cognitive scalability*, as well as *computational scalability*, which we characterize broadly to capture issues posed by human and hardware constraints.

The first issue is that the data is too large and complex to represent with one image;

thus, the user must explore the data through sequences of changing views. Context and interactivity provided through the visualization system are crucial for human efficiency and the coherency of the exploration workflow. The second issue is that the capacity of system memory, where visualization takes place, is dramatically smaller than the complete data. Therefore, attempting to scale the visualization workflow purely through data reduction is not feasible. Instead, the system must bring in and retain data selectively without exceeding memory limitations.

Without scalable interactive tools, the visualization process is arduous and time-consuming. Such conditions can force scientists to prioritize heavily and only investigate the data through limited vantage points based on expectations and prior knowledge. As a result, the unexpected is at risk of going without notice, and thus errors and anomalies are at risk of being undetected, and new phenomena are at risk of remaining undiscovered.

For reference, we present an illustrative model for scalable visualization in Figure 1.1. Our approaches in this dissertation combine data decomposition, local summarization, coupling of lightweight summary data with details through an organized data layout, and interactive visualization systems designed following *focus+context* [16] and *overview first, details on demand* [17] visualization models. We now delve further into the relevant concepts of scalability, the underlying problems, and the techniques and approaches applied to address them.

1.2.1 Perceptual and Cognitive Scalability

A visualization conveying too much information can confuse or mislead the user [18]. As the amount of detail and complexity in a visualization view increases, the perceptual abilities of the user are more and more overwhelmed [19, 20]. This problem has been called the perceptual scalability of visualization [21, 22]. When it comes to large-scale multivariate time-varying simulation data with a 3D spatial component, the amount of detail and complexity is too vast to be understood reliably through only a few views.

Let us consider first the more straightforward task of visualizing one variable in a 3D domain (*e.g.*, a 3D scalar field). This kind of data is usually visualized through direct volume rendering [23] or by extracting and rendering isosurfaces [24] or other features.

Due to the data being 3D, there is an inherent issue with occlusion [25]. Objects in the front of the view occlude objects in the back, and the exterior occludes the interior. Color and opacity transfer functions [26], which assign color and opacity values to each point in the 3D space as a function of the scalar value, are used to hide, reveal, emphasize, or de-emphasize different parts of the data. The user then extracts visual information through a series of views generated with different transfer functions. This process is usually most effective when interactive but can sometimes be automated based on data-driven view selection [27] and transfer function design heuristics. Another technique for revealing occluded visual information is *exploded views*, which isolates 3D features in the spatial domain and then spatially separates them in the rendered visualization view [28]. For 3D flow data, topological approaches [29] help detangle features and guide the extraction of visual information.

Overall, the visualization of 3D scalar data is a mature field with scalable solutions. However, comprehensive visualization of 3D data with more than one variable is a much more challenging task, and the difficulty increases as the number of variables increases.

To understand the challenges with visualizing 3D multivariate data, we should first consider some of the use cases. First, let us consider vector fields, where each point in the spatial domain is vector-valued. An example is a velocity field where the vectors include one velocity component for each orthogonal axis in the 3D space. More generally, we have tensor fields where each point is matrix-valued. One possible way to visualize this kind of data is through multidimensional transfer functions [30], which assign color and opacity as a function of the multivariate values. The main problems with this approach are that humans can perceive limited dynamic ranges of color and have individual differences and deficiencies in how they perceive color. Further, the mental mapping of color and opacity to vector or matrix values is generally unintuitive.

Another approach is to use *glyphs*, which encode information through shape and orientation. Since the vectors or tensors commonly carry orientation information (*e.g.*, directional information) that is coherent with the axis of the 3D space, this is an intuitive approach that allows one to grasp some of the meaning in the visualization without reference to a legend. We can thus more directly and easily scan and relate the different components in the image to understand the complete picture. For vectors, arrows are used, and for tensors, parameterized shape models are used in conjunction with shape metrics. Color can also be applied to glyphs to encode an additional information channel. The main problems with glyphs are that they take up much space since they need to be large and spaced out enough to perceive their shapes easily. Moreover, since rendering one glyph for each discrete volume element would result in far too much clutter, glyphs usually represent spatially aggregated or sparsely sampled data points. Aggregation alleviates issues from clutter and occlusion but destroys information, sometimes leaving the encoding ambiguous or misleading. For example, vectors pointing in opposite directions can cancel out.

Regarding arbitrary multidimensional volume data, glyph representations often lack intuitive encodings. Furthermore, limitations of human perception make it challenging to translate the visual properties of an abstract object to quantitative parameter values. Also, when spatially aggregating multivariate scientific simulation data, simplifying assumptions about the underlying distribution are often not justified, so simple parametric statistical models or shape models will not faithfully represent the data. Thus, more information about the distribution of the aggregated elements needs to be conveyed or at least retrievable by the user. However, due to limitations in screen space and human perceptual abilities, we must limit the amount of information displayed to the user at a given time. The more we aggregate and simplify through abstraction, the more coverage the visualization view conveys, but at the cost of reduced detail.

Due to the reasons above, visualization through the aggregation of multivariate and multifaceted data into glyphs is much more effective when the user can interactively adjust the views to reveal hidden detail. With a large and complex space of possible views or parameterizations, revealing the details in the data aimlessly requires a large amount of time and effort. It can be overwhelming as the user must rely extensively on memory and attempt to determine the relations between many different views. Visualization researchers have sought ways to steer the exploration process more effectively. Particular applications require particular designs, yet some general paradigms and approaches have become popular and proven effective: overview first, details on demand, focus+context, and view-linking. An overview or summary can cover more of the domain but with less detail. The summary or overview acts abstractly as a map, which helps orient the user and helps the user to choose a starting point for drilling down into details. As the user selectively reveals detail, overviews can be rendered alongside the details or underneath them, which helps the user stay oriented and grasp the context. View-linking further helps the user to relate different facets of the overviews and details.

These approaches are leveraged to reduce cognitive load and make the visualization process more efficient, comprehensive, and robust. In practice, the visualization designs that integrate them tend to be application specific, and the design process requires collaborative research with experts and practitioners. The reliance on collaboration with rare and busy experts and the need for data access and HPC resources have made for slow progress in bringing this research direction to widespread fruition. Moreover, the demand for interactivity to address perceptual scalability and comprehensiveness, and the demand for efficiency to achieve interactivity with large data, make implementation labor-intensive and challenging. In the application domains of focus in this dissertation, to the best of our knowledge, no available tools or visualization designs yet existed to support a comprehensive and perceptually scalable solution.

1.2.2 Computational Scalability

Having established the need for perceptually scalable visualization solutions, we now consider computational issues:

- Entire datasets are too large to store on the hard disks of desktop workstations and can even overwhelm data centers.
- Interactive visualization requires the occupation of data in system memory, which is vastly smaller in capacity than even the already overwhelmed hard disk capacities. The performance mandated for interactive transformation and rendering often requires the data also be persistent in memory during the interactions.

- The algorithms and implementations for transformation and rendering should be efficient enough for low latency throughout some interactions.
- For scalability, the systems must efficiently bring new data into memory from disk or through a network when demanded.

In terms of storage on disk, supercomputing facilities where the data is produced have strict quotas and purge user data after some time once projects end. Simulation scientists typically save the data at a reduced temporal frequency to meet the constraints. After the simulation runs, data is transferred to long-term storage systems and commonly needs to be later moved again through a network to a cluster or workstation for visualization and analysis. This issue causes long delay times to getting started. To help illustrate the hindrance to the scientific process, a fraction of one of the combustion datasets we utilize in this dissertation took approximately one month to transfer from remote storage to the systems where we processed it. Further, the visualization process can be time-consuming when suitable and scalable visualization software is unavailable for the problems at hand.

While distributed parallel rendering tools such as Paraview [31] and Visit [32] help overcome some of the limitations for 3D rendering at scale, there is little for scalable multivariate spatial statistical analysis or for building customized interactive visualization systems with linked views that are suitable for the needs of our driving applications.

1.2.3 Data Reduction and Summarization

Data reduction can be considered to be motivated by two general goals: filter out the data that is redundant or extraneous to future tasks and meet the minimal restrictions, or reduce disadvantages imposed by limited hardware resources to store and manage the data. The former, besides reducing data, also reduces complexity; region-of-interest (ROI) based reduction and feature extraction are examples. If all of the extraneous or redundant data, for all future interests, could be predetermined, then optimizing the data reduction under the constraints would be straightforward. However, problem domains are broad and uncertain in the real world; thus, determining which information can be safely thrown out can be difficult or in some cases impossible.

What is retained can be compressed, either with lossy or lossless compression algorithms, and data can also be reduced by reducing the floating point precision. Subsampling is another widely used reduction approach; in some cases, the raw data can be reconstructed from the data samples, *e.g.* based on compressed sensing theory [33]. These methods ameliorate management and storage problems for large data, but usually only by a limited constant factor, especially if the error must be controlled, and the data must be reconstructible.

For reference, Marsaglia et al. provide a survey on data reduction in scientific visualization and analysis [34]. They break down the category into five sub-categories: lossless, near-lossless, and lossy compression, mesh reduction, and derived representations.

A core part of our approach is domain decomposition and statistical summarization. While data summarization is often categorized in the literature as a form of data reduction, it is not strictly useful for reducing the total data size. We characterize it as a form of derived representation, which will likely not preserve detailed information. Due to the challenges in determining what information is redundant or irrelevant to future tasks, and the loss of information inherent in summarization, we take a conservative approach to its use as a replacement for raw or otherwise reduced data. Still, summarization may be used to one's benefit, even without reducing the total data. For example, due to the compact size, straightforward interpretation, and breadth of coverage, data summaries are helpful for interactive visualization or real-time simulation monitoring. Moreover, since summary data is useful in the overview-first, details-on-demand visualization approach, it is an excellent form of data to couple with more detailed subsets of data.

Domain decomposition can also play a role in data reduction since one can optimize the data reduction by reducing or abstracting data more or less aggressively in different parts of the domain based on saliency. By breaking down the data into smaller chunks and summarizing those chunks with bounded-scale summaries, one can scale up the total data size while controlling the parameters of the decomposition and summaries so that computational limitations for interactive visualization are not exceeded.

1.3 Driving Application Domains

With room for generality, this research's primary driving application domains are the visualization of particle-in-cell plasma simulation data and turbulent combustion simulation data. Most of the research presented in this dissertation was done in collaboration with scientists in the fields of tokamak fusion simulation and combustion simulation. Some of the work was done through more limited collaboration with scientists who simulate linear particle accelerators. Further, we apply some of our work to the visualization of a marine mammal migration dataset. The data we use is described in more detail in Appendix A. Videos which are helpful for understanding the data and some of the analysis interests are described and referenced in Appendix B.4, along with video demos of the software we produced. These videos may be helpful for the reader, to make better sense of our research as they work through it. We now briefly introduce the primary application domains.

Particle-in-cell Plasma Simulations Plasma physics simulations model systems of matter full of charged particles, such as ions and electrons. In these systems, the charged particles generate electric fields and currents and interact with each other and magnetic fields. The particle-in-cell (PIC) method [35] uses a combined Lagrangian (discrete particles) and Eulerian (fields modeled on grids) approach. Since the time scales are very small, and the number of particles in real physical systems is enormous, it is not feasible in most cases to model all of the real particles directly. Instead, superparticles representing a larger number of real particles are used in the simulation.

We utilize particle-in-cell data from simulations of tokamaks, a class of magnetically confined fusion devices. These devices work by confining hydrogen plasmas within powerful toroidal magnetic fields while heating them to the extreme temperatures required for fusion reactions. Particle-in-cell simulations are used to model the confined plasma within these devices to aid in the design and theoretical development of the underlying physics. Hopefully, this technology will one day provide vast sources of clean energy.

This dissertation utilizes data from two gyrokinetic particle-in-cell tokamak fusion simulations: The Gyrokinetic Tokamak Simulation (GTS) [36], and The X-point Gyrokinetic Code for Transport in Tokamaks (XGC) [37]. In both simulations, each particle lives in a 5D phase space. Each particle in the simulation also carries signed weights, representing contributions to an aggregate perturbation of a background distribution. The weights evolve, and it is important to the scientists we collaborated with to understand the evolution of these weights in conjunction with the phase-space particle distributions and superparticle trajectories.

Turbulent Combustion Simulation Combustion-driven processes are pervasive in the modern world, and thus, the applications are many and diverse. This dissertation utilizes data from a simulation code called S3D [38], which is a parallel direct numerical simulation (DNS) that simulates compressible reacting flows. The data is comprised of chemical mixture fractions, pressure, and velocity values for each grid point of the mesh at each time step.

1.4 Dissertation Structure

In general, our technical approach to visualization combines spatial decomposition, local summarization with statistics and distribution functions, and tailored interactive visualization systems. The visualization systems leverage the decompositional and distribution function based approaches, combined with linked views, focus+context, and overview first, details on demand techniques. Various methods are introduced to fill the gaps as we develop novel visualization systems based on these concepts tailored for the applications.

Before presenting our research, we provide background and summary of related works for relevant topics. Then we summarize related visualization work focused more specifically on our driving application domains.

Our research with tokamak scientists focuses on the visualization of particle data. In Chapter 3, we present our research on visualizing phase-space particle distributions through interactive visualization. In that chapter, we introduce a novel approach to visualizing spatially distributed velocity data through spatially organized histograms. Next, in Chapter 4, we extend the method of spatially organized histograms to a more general approach and design a visualization system based on the technique that also addresses the need for co-analysis with the particle weights and temporal analysis. Then, in Chapter 5, we introduce an interactive technique based on the distance plot, or unthresholded recurrence plot, to analyze individual multidimensional particle trajectories in combination with particle weights. Finally, in Chapter 6, we extend the capabilities for interactive temporal analysis of large sets of multidimensional trajectories. Chapters 6 and 4 also include applications to particle data from a linear particle accelerator dataset, and Chapter 3 includes an application to a marine mammal dataset.

The research with combustion scientists is confined to Chapter 7. This application domain focuses on field data stored on fixed mesh nodes rather than particle data. Again, a primary goal is the visualization of multivariate spatially distributed data; furthermore, we again take a decompositional and distribution function-based approach. However, since the collaborating combustion scientists prefer to investigate the spatial facets of the data through conditioning on complex surface features, we require a new more sophisticated spatial segmentation approach. We address this problem through level set restricted centroidal Voronoi tessellation. An interactive visualization system leveraging the tessellation for spatial statistical analysis is designed based on the goals and feedback from the collaborating scientists.

Finally, we discuss the combined work from a more holistic point of view before concluding. The software and media we developed through this work are referenced in Appendix B. Again, the data we utilize is explained in more detail in Appendix A.

1.5 Publications Stemming from This Work

The research presented in the dissertation are based on the following publications.

(Chapter 3) Neuroth, T., Sauer, F., Wang, W., Ethier, S., and Ma, K. L. (2015, October). Scalable visualization of discrete velocity decompositions using spatially organized histograms. In 2015 *IEEE 5th Symposium on Large Data Analysis and Visualization* (LDAV) (pp. 65-72). © 2015 IEEE.

(Chapter 4) Neuroth, T., Sauer, F., Wang, W., Ethier, S., Chang, C. S., and Ma, K. L. (2016). Scalable visualization of time-varying multi-parameter distributions using spatially organized histograms. *IEEE transactions on visualization and computer graphics*, 23(12), 2599-2612. © 2016 IEEE.

(Chapter 5) Neuroth, T., and Ma, K. L. (2019, April). Interactive Spatiotemporal Visualization of Phase Space Particle Trajectories Using Distance Plots. In 2019 *IEEE Pacific Visualization Symposium* (PacificVis) (pp. 232-236). © 2019 IEEE.

(Chapter 6) Neuroth, T. A., Sauer, F., and Ma, K. L. (2019, June). An Interactive Visualization System for Large Sets of Phase Space Trajectories. In *Computer Graphics Forum* (Vol. 38, No. 3, pp. 297-309). © 2019 John Wiley and Sons.

(Chapter 7) Neuroth, T. A., Rieth, M., Aditya, A. Lee, M., Chen, J., and Ma, K. L. (VIS 2022). Level Set Restricted Voronoi Tessellation for Large scale Spatial Statistical Analysis. To appear in *IEEE Transactions on Visualization and Computer Graphics*, 29 (1) 2023. © 2022 IEEE.

Chapter 2

Background and Related Work

While most of the dissertation focuses on the visualization of fusion and combustion simulation data, we draw broadly from more general concepts. Furthermore, the technical contributions that together form our solutions are related to various lines of research with a range of applications. Therefore, we begin with a more general presentation of the background and related works, focusing on those works which are most closely related to our specific methodology. We then end the chapter with background and summary of work done specifically for tokamak and combustion data.

2.1 Density Estimation and Distribution Functions

Our approach is centered on visualizing spatiotemporal data through local statistical summarization using distribution functions. We use the term *distribution function* because it is general enough to include: the particle distribution functions (PDFs) that we visualize from the tokamak datasets in Chapters 4 and 3, the probability distributions functions from Chapter 7, and the dense trajectory visualizations in Chapter 6.

Distribution functions can be modeled in several ways, with the three main classes being *non-parametric* models (such as histograms and kernel density estimation), *parametric models* such as Gaussians, or *semi-parametric models* such as mixtures of Gaussians. David W. Scott's book, *Multivariate density estimation: theory, practice, and visualization* [39], is an excellent introduction to the topic.

Parametric models are very useful if their assumptions are valid in the application do-

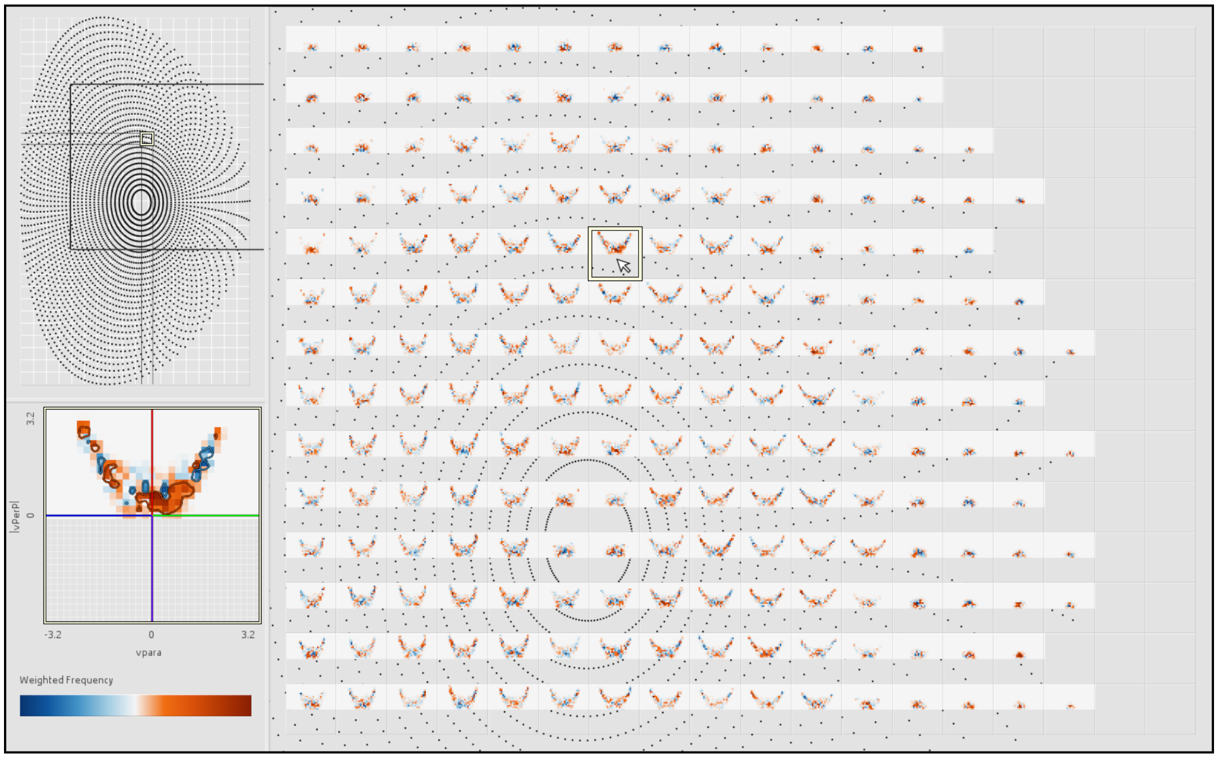


Figure 2.1. A visualization of distribution functions for the time-varying GTS data. The visualization is a screenshot of the visualization system presented in Chapter 4. These distribution functions represent signed perturbations of the phase-space particle distribution function, as explained in Chapter 4 and further in Appendix A.1.2. The purpose of this figure is to highlight some of the characteristics which motivate the use of non-parametric models.

main, largely due to their simplicity and applicability as predictive tools. Non-parametric models don't rely on as many assumptions and represent the data more explicitly; thus, they are more flexible. They tend to be more beneficial for exploratory visualization but less useful for prediction because they tend to overfit the data. Rice suggests a workflow for density modeling where one begins with an exploratory process utilizing more explicit and less smoothed models (*e.g.*, scatter plots and histograms), and then works towards the application of more smoothed models [39].

In the context of this dissertation and the related works, parametric and semi-parametric models, such as Gaussians and mixtures of Gaussians, are especially powerful when the assumptions are valid because they can be represented with only a few parameters and thus have a small storage cost. However, especially in complex spatiotemporal simulation data, one must be careful when considering the application of parametric models since

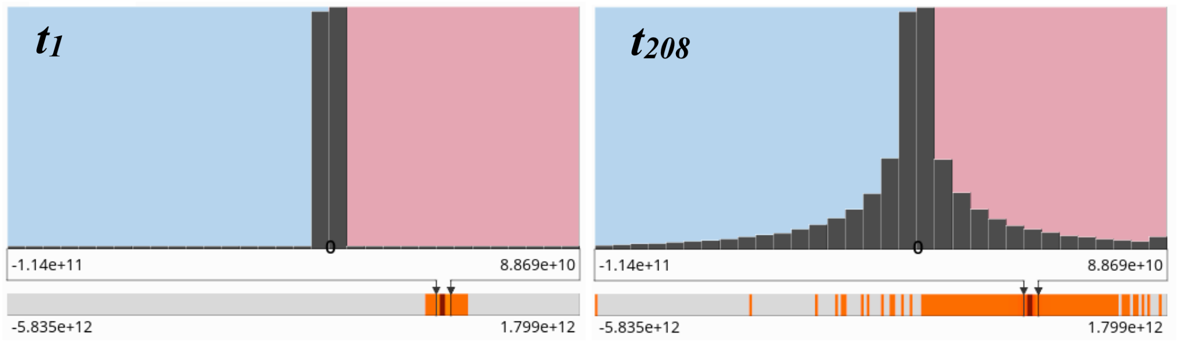


Figure 2.2. A histogram of particle weights for a subset of particles from the XGC simulation. The visualization shows the full histogram as a heatmap on the bottom (with darker color indicating higher frequency/count), and a histogram as a bar chart shows a subset of the full data corresponding to the extents/outer edge values marked by the arrows. The figure illustrates the peakiness and extraordinary long tails that develop in these distributions over time. Left) The distribution at $t = 1$. Right) the distribution at $t = 208$. The visualization is a screenshot of one of the interactive visualization views used in the visualization system presented in Chapter 4.

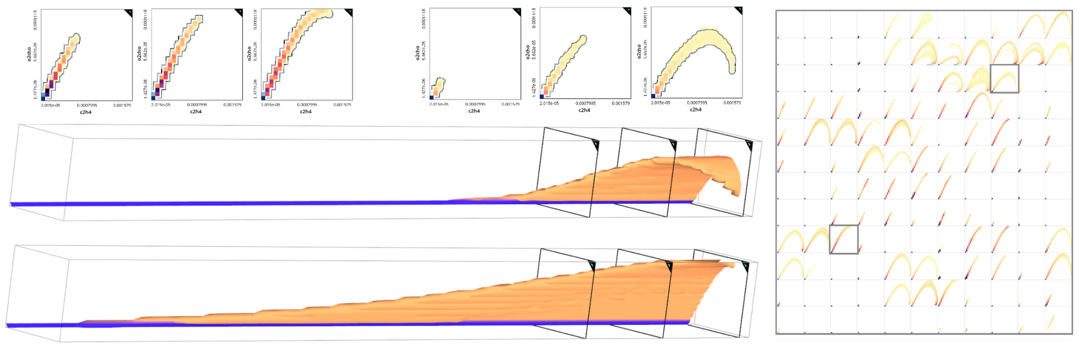


Figure 2.3. A visualization of a time-varying 2D histogram of chemical mixture fractions in the 2D autoignition data. The joint plots, computed at different time steps, are shown above in juxtaposition. The upper-left three plots correspond to the lower temporal visualization, and the upper-right three plots correspond to the temporal plot shown in the middle of the figure. These temporal views show the surface/outer edge of the joint plot extended in time. The two cases correspond to different spatial regions/subsets, and the visualization on the right shows the spatial domain with the two regions outlined in black. The purpose of the visualization is to show an example of characteristics we encounter in some of the joint distributions in our combustion data.

the data can be highly variant over time, non-Gaussian, or carry non-linear multivariate correlation structure.

Histograms are cheap to compute but come at a higher storage cost than parametric alternatives. Since they explicitly decompose the data into discrete bins, they also can easily support simple interactive techniques for drilling down into details, as demonstrated in Chapters 3 and 4. In this dissertation, we primarily focus on non-parametric exploratory analysis using histograms, but our tool in Chapter 7 also supports 2D Gaussian mixtures and kernel density estimation (KDE) [40].

For context, we highlight a few distributions from data utilized in this dissertation: Figure 2.3 shows the evolution of two histograms from a combustion dataset over time. Initially, the correlation appears linear but later becomes non-linear. Figure 2.2 shows the interactive histogram tool from our visualization system described in Chapter 4. The bar chart represents the “zoomed-in” portion of the histogram. The heat-map-based visualization below it represents the complete histogram, and the arrows show which part of the complete histogram the zoomed-in view represents. The figure highlights the peakiness, long tails, and extreme outliers in the distributions of particle weights in the XGC simulation data. Figure 2.1 shows some of the histograms generated from the GTS data, which have unusual shapes and characteristics. In these cases, it would be difficult to use a parametric model without hiding important information.

2.2 Spatial Segmentation and Summarization

Many papers have been published over the years on spatial statistical aggregation to reduce and analyze large simulation data. Chaudhuri et al. introduced scalable distributed algorithms for computing histogram-based spatial summaries of large data sets on various kinds of 3D grids [41]. Lu et al. further improved on the state of the art, with compact representations of histograms based on bit-strings and space-filling curves [42]. For simulation monitoring and anomaly detection in transonic jet engine simulations, distribution functions were used to quickly and compactly represent facets of the flow conditions so that they could be easily monitored in real-time for human-in-the-loop stall detection [43]. Ye et al. [44] decomposed the domain of combustion simulations into blocks, then computed spatial statistics used for querying Lagrangian particles through a coupled out-of-core (off memory) data structure. Many more related methods have been introduced. Gaussian mixture models (GMMs) and copula functions have provided highly minimal footprint models of joint statistics [45, 46]. Additional work has focused on utilizing spatial statistical correlation to recover spatial information from spatially aggregated data to enhance the ability to perform volume rendering directly through sampling spatially localized PDFs [47]. In terms of spatial segmentation for statistical aggregation, different approaches have been applied, including kd-trees and simple linear iterative clustering [45]. Thompson et al. [48] leverage such *in situ* (while the simulation is running) generated histograms for feature detection through a combination of statistical and topological methods.

One way to segment flow data is through topological methods. Heine et al. provide an in-depth survey [49]. Topological methods can also be used to constrain sub-sampling methods to preserve important flow features [50, 51, 52, 53]. Topological exploration is commonly done through graph-based data structures. Bremer et al. proposed merge trees to segment combustion data into hierarchical features that can be summarized through statistics and explored efficiently through their graphs [54]. Dynamic nested tracking graphs track the topological changes in isosurfaces over time and have been used to create interactive visual summaries [55]. They have also been used within an integrated visualization system that couples topological features with in situ rendered images using a cinema database [56]. Jankowai and Hotz proposed a generalization of isosurfaces to multi-fields, which inspires a new approach to topological visualization and analysis in multivariate data [57]. These are just a few examples, as topological segmentation is a rich and active area of research.

Voronoi tessellations are one of the most fundamental methods in computational geometry and related fields. The centroidal Voronoi tesselation is a Voronoi tessellation where the Voronoi sites are positioned at the centroids of their Voronoi regions. Our method, presented in Chapter 7, utilizes restricted centroidal Voronoi tessellation for spatial segmentation as a basis for spatial statistical analysis. One popular algorithm for computing a discrete CVT is Lloyd's Algorithm, which has been applied widely and studied in depth [58]. Amenta et al. introduced Voronoi algorithms for conforming tessellation of 3D surfaces [59, 60, 61]. Such tessellations are used for constructing and refining meshes [62], which has various applications. Recently, a provably conforming Voronoi meshing algorithm, VoroCrust [63], which borrows from Power Crust, has been proposed. Our tessellation in Chapter 7 is proposed as a new variation of the CVT that is more suitable for statistical analysis due to its ability to trade off conformity for spatial homogeneity.

2.3 Clutter Reduction

Many works focus on reducing clutter in flow visualization. Kirby and Laidlaw [64] used concepts from painting to design a multi-layer representation of colors and textured patterns to represent different parameters of a flow field in an easy-to-read manner. Obermaier and Joy [65] used a visualization of metric tensors to show deformations on 3D surfaces. Ellipsoid glyphs can represent properties like the velocity gradient on these surfaces. In addition, topologically-based methods, which were first introduced by Helman and Hesselink [66], can be used to extract specific flow patterns of interest and present them in a low-clutter way.

In Chapers 3 and 4, we use a "panning-window" style layout and sampling method for interacting with spatially organized histograms. Our technique can be considered a form of "Magic Window" as defined by Tominski [67]. Other methods have been used to show spatial variation and uncertainty quantities, such as vectors, in a summarized manner. For example, glyphs have been designed for visualizing uncertain flow fields by showing ranges of possible velocities [68]. Furthermore, Hlawatsch et al. [69] have designed a glyph that encodes statistical moments.

2.4 Histograms for Motion Representation

In Chapter 3, we use spatially organized histograms to visualize particle motion. Representing motion with a histogram-based format has been used before in computer vision. For example, Ihaddadene and Djeraba [70] utilize a block direction histogram to represent the overall motion of crowds in different "blocks" or fields of view. Romanoni et al. [71] utilize spatiotemporal histograms for background subtraction in cases where the camera is in motion. Another example is the use of "Histograms of Oriented Gradients" as feature descriptors. Dalal et al. [72] utilize these to detect humans in motion relative to the background accurately. Lastly, Jung et al. [73] used velocity histograms to cluster visually tracked objects. The clusters are identified based on the motion of groups of objects by identifying similar neighborhoods in the resulting velocity histograms.

2.5 Boolean Filtering for 3D Flow Data

In Chapter 6, we developed an algorithm for GPU accelerated Boolean filtering of particle trajectories. Similar work has been done by Salzbrunn et al., who formally defined the concept of pathline predicates [74]. Their work focuses on 3D flow fields and the selection of pathlines based on sets of time-dependent properties. Computing the properties is done in a non-interactive preprocessing stage, and queries based on the properties are done interactively. Shi et al. did similar work for 3D flow fields [75].

For high-performance logical filtering and querying, GPU accelerated database systems have been proposed [76]. These systems must support general database operations such as join, select, and sort. They utilize data structures and algorithms such as B-trees, indexing/tables, hashing, etc. Another technique for rapid queries on scientific datasets is Fast-bit [77]. Fast-bit uses bit-indexing based on pre-computed property flags. One can then accelerate common query patterns by looking up the pre-computed conditions in a table rather than computing them on the fly.

2.6 Focus+Context

Focus+context is a style of visualization where the data in focus is emphasized but simultaneously visualized with less emphasized data to provide context. Theory on the generalization of this technique was proposed by Hasuser [16]. In the area of Focus+Context for scientific data, Muigg et al. have introduced techniques for visualizing temporal features in large data sets [78]. Doleisch et al. designed a Focus+Context visualization for simulation data that incorporates an XML and Java-based feature description language [79]. This approach parallels our work in Chapter 6. We also design focus+context style visualizations which are classified as examples of *interactive lenses*, sometimes referred to as *magic windows*. Tominski et al. provide a recent [80] on this class of techniques.

2.7 Combustion Data Visualization

Flow phenomena are an exceptionally prominent topic of research since flow plays an important role in many spatial physical systems besides combustion. Combustion science is a subset of the more general area of reacting flows, where the system's evolution is driven strongly by flow dynamics and chemical reactions. The flow is usually modeled using the Navier-Stokes equations [81]. Flow remains a major challenge for science, partly because of *turbulence*, which is a phenomenon where flow disturbances become chaotic (unpredictable due to extreme sensitivity to small differences in the state of the system). Challenges to visualization and analysis stem from the need to understand the system's evolution in terms of both the turbulent flow dynamics in the physical space and the reactions in the *composition space*, which consists of the mixture fractions of different chemical species.

Statistical visualization of large-scale high-dimensional data is challenging because of the curse of dimensionality in high-dimensional statistics [39] and the limitation in the number of dimensions that can be visualized in one view. While, with three dimensions, we can use direct volume rendering or surface rendering, as suggested by Rice [39], beyond that, we run into the fundamental limitations of human visual perception. A scatter plot matrix showing the joint distributions of each pair of variables can be used to help understand correlation structure in high-dimensional data. Extensions, such as the binned scatter plot matrix [82], have been introduced to reduce the complexity of the visualization for large multivariate data. However, the number of plots grows quickly ($O(n^2)$ in the number of variables), and only 2D correlations are truly ascertainable from the visualization. Linked views and interactive sub-selection are used to ameliorate this problem [83]. Through this process, one can focus on a meaningful set of multivariate conditions to see how the other variables correlate in terms of their dependencies with those conditions. Furthermore, selections can be linked with visualization views showing the physical space, as done by Jones et al. [84].

In addition, spectral analysis is a major research topic for combustion data and turbulent flow in general. Spectral analysis, in this context, involves analysis of flow features, *e.g.* eddies or vortical structures, based on their length scales. In turbulence theory, energy transfers from large-scale structures into smaller-scale structures in a process called *energy cascade* [85]. The theory was pioneered by Andrey Kolmogorov [86], who introduced the Kolmogorov Turbulence Spectrum. Some combustion visualization research has focused on scale-based flow feature analysis [87, 88, 89]. One way this has been approached is through band-pass filtering in the frequency domain and then transforming back into the physical domain. By doing this, we can get local information about spectral information [90]. The main limitation here is that the Fourier transform assumes the data is periodic in each dimension of the physical domain, which is not commonly the case. Alternatively, the wavelet or curvelet transform can be used for spectral decomposition, even without periodicity assumptions. The former has been used for scale-based geometric analysis of flow features in combustion data [91].

In Chapter 7, we develop a new spatial segmentation method that is compatible with the topological approach (as an extension for more general and finer-grained exploration), combining connected component decomposition and centroidal Voronoi decomposition. We then leverage this approach to support an interactive visualization system for scalable multivariate spatial statistical analysis, a much-needed capability that was not yet adequately supported.

2.8 Tokamak Data Visualization

Tokamak simulations vary in terms of the simulation purpose and the type of data produced. The flow inside a tokamak is commonly studied through the lens of magnetohydrodynamics, the study of electrically charged fluids in the presence of magnetic fields. One topic of study focuses on the patterns and topological features that emerge in the magnetic fields. The stability of the magnetic field is essential since the magnetic fields are responsible for containing the extremely hot plasma in the device. Sanderson et al. used Poincaré plots to study recurrent patterns in toroidal magnetic fields [92]. Tricoche et al. applied these methods within a more general study of topological features in area-preserving maps[93]. In other cases, the stability of the magnetic field is assumed and modeled as a static (non-changing) field, *e.g.* the XGC data we utilize in this dissertation. While there is an evolving electric field in this simulation, our collaborative work was limited to the goal of analyzing particle data.

In one of the first major papers addressing the visualization of large-scale tokamak simulation, Crawford et al. assessed the visualization needs and their support by current technologies [94]. These include large-scale volume rendering tools, point-based rendering techniques, and hybrid point/field-based rendering methods. An approach was later introduced for visualizing particle data through point rendering within an integrated system for multivariate filtering (such as parallel coordinates plots) and for coloring the particles using interactive color transfer functions [95, 84]. Other systems and techniques that have been applied for visualizing particle data from tokamak simulations include grid-based averaging [96, 97].

The work presented in this dissertation stems from a need not adequately supported by existing tools, which was to interactively visualize thousands of local distribution functions computed locally, throughout the spatial domain, from statistical super-particles. The distribution functions, in the form of signed and weighted histograms, were being visualized one at a time by the collaborating domain scientists using standard analysis software such as Matlab. This mode of analysis is highly limiting and time-consuming. To help support the needs, we develop interactive visualization software focusing on the collective visualization of thousands of spatially organized histograms [98, 99]. We further develop software offering new capabilities and performance enhancements for more advanced interactive analysis of the multivariate particle trajectories [100, 101]. The contributions are outlined in more detail in the associated Chapters 3, 4, and 6.

Chapter 3

Visualizing Velocity Distributions with Spatially Organized Histograms

This chapter is based on material from our publication, *Scalable visualization of discrete velocity decompositions using spatially organized histograms* [98]. © 2015 IEEE.

The ability to visualize the motion of a group of objects has applications in many fields. For example, the study of fluids often uses large sets of tracer particles to represent the intricate flow patterns in both a laminar and turbulent setting. Presenting the motions of these particles in a coherent fashion can lead to a better understanding of the underlying processes involved in phenomena such as fusion [37, 36] or combustion [102]. Geospatial movement data is another area that focuses on detecting motion-based patterns of objects such as animals or motor vehicles. For example, studying trends in the movement of marine life [103] can lead to a better understanding of migration or feeding patterns. These are simply a few examples as there are many areas of study that focus on the motions of a group of objects.

Visualization techniques are often subject to a trade-off between clutter and loss of information. For example, visualizing the velocity decomposition of a group of objects using vector plots can result in a great deal of clutter. Averaging the motion into a velocity vector field can alleviate this problem, however significant information about the true underlying velocity decompositions may be lost. This chapter focuses on developing a visualization tool that can address these issues.

Velocity-based histograms work by sampling the frequency of objects that exhibit a particular velocity decomposition. In our implementation, each histogram bin is represented by a cell in a 2D grid and its associated frequency is visualized using a color map. This results in an information-dense visualization of the overall motion of a group of objects, with a nice regular structure that lends well to user interaction and efficient computation. Many of the challenges of this technique lie in the ability to present many velocity histograms to users simultaneously (each representing a different portion of the domain).

3.1 Methods

The main application of this technique is for interactive visualization of the velocity decomposition of a group of objects. The velocity of each object is sampled in order to form a set of spatially organized velocity histograms to be visualized by the user. The means of sampling and generating the histograms, as well as their visual appearance and spatial layout, are all important factors that must be considered and are described in more detail in the following sections.

3.1.1 Overview

One of the benefits of our system is the ability to interactively explore the data in real-time with various adjustable parameters and configurations. For these reasons, the histograms must be computed at runtime. The binning process is the most computationally intensive step and can be done efficiently using GPU acceleration.

3.1.2 Velocity Histograms

Figure 3.1 shows a comparison of the velocity histogram to two other techniques that attempt to visualize the motion of a group of objects. The most straightforward method, shown on the left of the figure, involves adding a velocity vector to each object. The direction and size of the arrow can be used to infer the speed and direction of motion of that particular object. The downside to this technique is that it can result in a great deal of clutter when the number of objects is large, making it difficult to visualize pat-

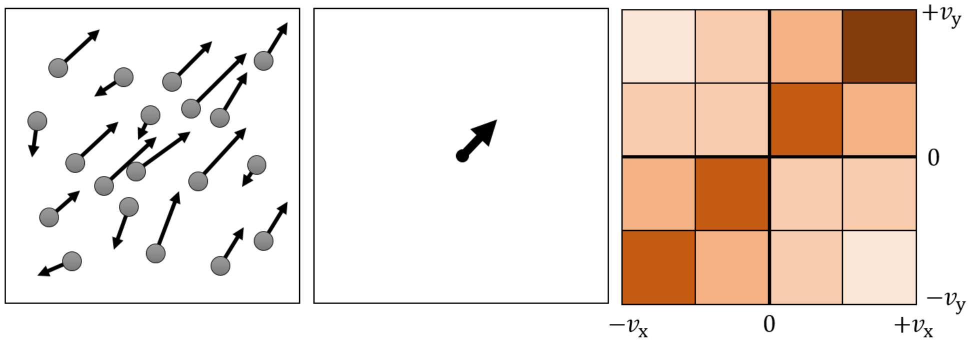


Figure 3.1. An image comparing techniques of visualizing the velocity decomposition of a group of objects. Left) Assigning a vector to each object. Middle) Averaging the velocities into a vector field. Right) Sampling into a velocity histogram with darker colors indicating a higher frequency.

terns. An alternative to this technique, shown in the middle of the figure, is to average the velocities of all objects into a vector field. This has the advantage of reducing the amount of clutter but also results in a loss of information about the underlying velocity decomposition. Another option is to use trajectories of pathlines. However, this can once again result in clutter, and becomes difficult to indicate which way along the pathline the object is moving. Furthermore, there is some temporal confusion since a drawn trajectory represents multiple points in time.

This chapter focuses on the use of velocity histograms, which are shown on the right of the figure. The velocity histogram has the advantage of being able to accurately show quantitative information about a large number of objects in a low-clutter, well-organized visualization which supports easy, interactive quantitative exploration. The velocity of each object is sampled and binned into an appropriate histogram cell. Color can then be used to indicate the frequency of objects occupying each cell. In the example shown in the figure, it is clear that the majority of objects have a velocity with a large positive x and y component (as indicated by the dark cell at the top-right of the histogram). On the other hand, few to no objects are moving to the bottom-right or top-left (as indicated by the lightly colored cells in those regions).

Such a histogram is a powerful method of visualizing overall trends of the motion of a group of objects. We can then further extend this technique to also study the differences in motion in various parts of the domain. This is done by decomposing the domain into

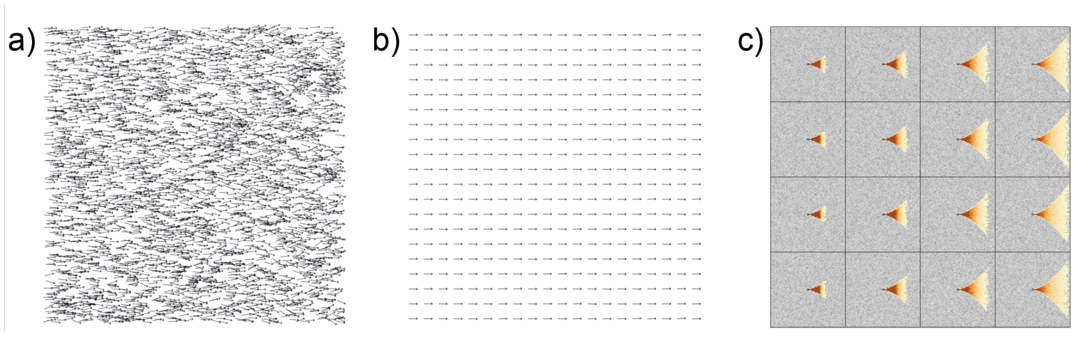


Figure 3.2. A synthetic particle dataset designed to show the advantages of using the histogram-based representation. a) Attaching a velocity vector to each particle results in a great deal of clutter. b) Averaging particle velocities into a vector field eliminates important flow patterns. c) The histogram-based view provides a low-clutter decomposition of the velocities in each part of the domain.

the desired number of subsections, each with its own velocity histogram. All the objects within a particular subsection are then sampled to form a velocity histogram for that region. A comparison between the histograms in different parts of the domain can show spatial variations in the motion of the objects. Moreover, adjusting the resolution of this domain decomposition can highlight general trends that permeate larger portions of the domain, as well as small-scale details. This is discussed further in Section 3.1.3.

Figure 3.2 shows an example of a synthetic particle dataset designed to highlight the advantage of the velocity histogram-based view. It consists of $\sim 200,000$ randomly distributed particles whose velocity in the x-direction (horizontal) is a positive random number and increases with the x position for some particles. The velocity in the y-direction (vertical) is a positive or negative random number that increases with x position for all particles. The y-direction velocity has an equal chance of being positive or negative. The fact that many particles are moving in opposite directions throughout the domain helps demonstrate the advantage of the histogram-based view: a) An arrow is attached to each particle to indicate its velocity. Drawing only 2% of the full data still results in an abundance of clutter. b) Averaging the particle velocities into a vector field causes the opposite motions of particles in the y-direction to be averaged out and become lost. Moreover, the increase in velocity in the x-direction is also difficult to notice since only a few particles display this variation. c) In the histogram-based view, each histogram highlights the overall decompositions of velocities in each portion of the domain. Darker colors represent a higher frequency of particles. The center of each histogram represents no movement whereas a highlighted bin towards the top-right of the histogram, for example, would represent fast motion in the positive x and y directions. Since the histograms represent distributions, many of the subtle patterns that would normally become lost via averaging are easily seen.

Flows like the one used in this example, where particles move differently from one another, even when occupying the same location in the domain, are commonly found in certain applications, such as fusion science. The plasmas that flow through fusion reactors contain oppositely charged particles that behave uniquely from one another in the presence of an electromagnetic field. This makes many traditional visualization techniques limited. The histograms used in our visualization tool are 2D in nature. This is because occlusion makes visually representing a 3D histogram very difficult. As a result, we employ a number of interactive techniques so that users can explore 3D domains to a limited extent. This is discussed in more detail in Section 3.1.3.

3.1.2.1 Generation from Particle Data with GPU Acceleration

In our algorithm, the texels of a single channel texture store the values of the histogram bins. The object velocities and positions are transferred to the GPU in the form of a vertex buffer. Interactive parameters, such as the position and structure of the sampling grid, the histogram resolution, and the normalization factors, are transferred to the GPU as well. The vertex shader then maps each object to a histogram bin based on its position and velocity. The result is a texel index which must then be converted into the correct position in normalized device coordinates before being input to the fragment shader. With additive blending enabled, the fragment shader then increases the value of the texel corresponding to the mapped histogram bin. The end result is a texture storing the computed histograms that can be queried as necessary to support interactive features before being rendered to the screen using an adjustable transfer function.

3.1.3 Visualization System

Our system provides interactive control over the spatial organization of the histograms, the resolution of the histograms, and any velocity/frequency thresholding. While the user interacts with the visualization, the relevant information is automatically displayed according to the selection of the user. In addition, several features are included to extend the capabilities of the system and to aid in navigating the visualization. Lastly, the system provides basic control over 2D slice extraction from 3D datasets, as well as time step selection and animation.

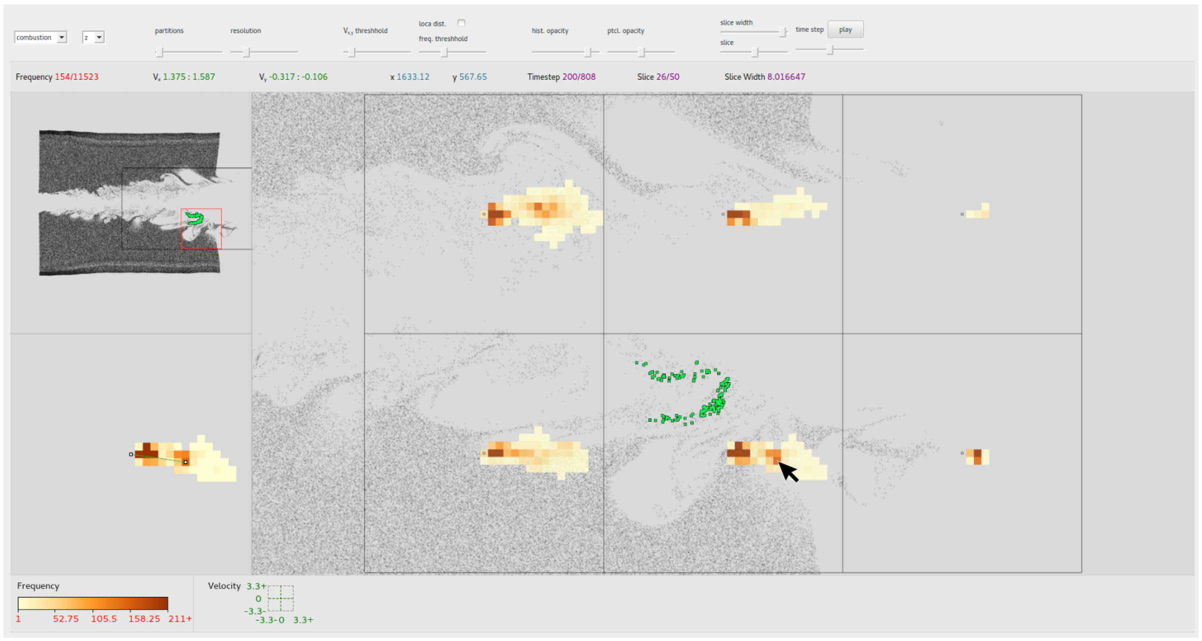
3.1.3.1 The Histogram Layout

Different spatial histogram layouts can be selected by the user depending on the application. The spatial layout defines how the domain is partitioned into multiple local sampling regions. These local regions also define the location, size, and orientation of the visual representation of their associated histograms. Reference points are drawn at the centers of the histograms to mark the bin that corresponds to a velocity of zero.

Our primary layout uses a panning window approach. The sampling region is partitioned into a rectangular grid while a slider controls the number of partitions. When the user drags the mouse over the view space or zooms in, the particles move relative to the screen, while the grid remains fixed. We also employ a layout that remains fixed with respect to the particles as the user pans and zooms. Such a layout has the benefit that zooming and panning can be done without changing the sampling region sizes. However, the panning window approach gives more control over the locations of the sampling regions with respect to the particles. In addition, custom layouts can be made to suit specific applications. An example of such a layout can later be seen in Figure 3.6 where the histograms are positioned and aligned along a magnetic flux surface.

3.1.3.2 Special Features

An example of the user interface can be seen in Figure 3.3 and highlights many of the features discussed in this section. As the user mouses over a histogram bin, corresponding particles are highlighted in green. This feature enables the histogram to double as a powerful selection tool for inspecting particle locations according to velocity and en-



Frequency 154/11523 V_x 1.375 : 1.587 V_y -0.317 : -0.106 x 1633.12 y 567.65 Timestep 200/808 Slice 26/50 Slice Width 8.016647

Figure 3.3. The user interface. The particles associated with the moused-over histogram bin are shown in green. The moused-over histogram is shown in more detail in the bottom left corner of the screen, and the mini-map is shown in the top left corner.

ables enhanced pattern and trend finding capabilities. Additionally, the histogram as a selection tool can be easily extended to trigger the display of other information about the corresponding particles, depending on the needs of the domain expert, e.g. temperature, trajectories, or velocity parallel to the viewing angle. Empty histogram cells are left transparent in order to minimize the occlusion of the particles within the associated sampling regions. For the case when the entire histogram is filled with values, the opacity of the histograms can also be manually adjusted. In addition, the user can adjust the opacity of the particles themselves, to strike a balance between visibility and distraction.

The color vs. frequency mapping is, by default, globally assigned to all histograms in order for users to see the relative differences in frequencies between regions. This global thresholding has the advantage of being able to reduce the influence of large peaks that would otherwise suppress subtle patterns in the histogram. The majority of the figures in this chapter use the global thresholding scheme with the colormap shown in Figure 3.4. In addition, we also provide the option to locally scale the colormap normalizing with the maximum frequency in each histogram. This results in a separate color scale for each

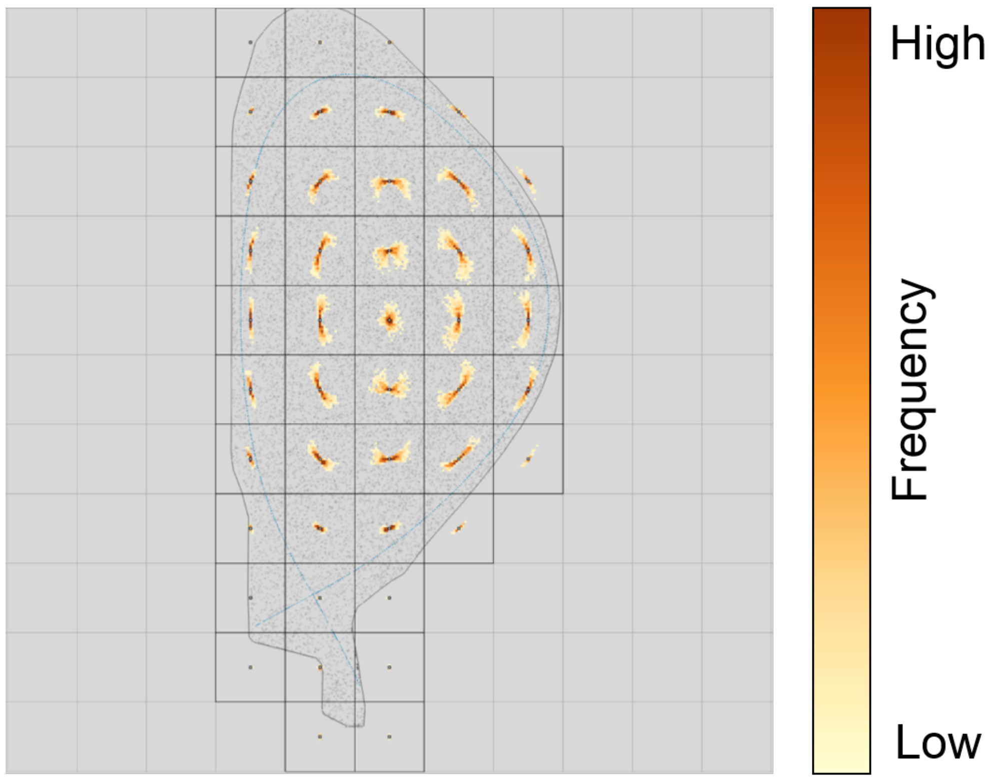


Figure 3.4. An overview of the histogram visualization using the C-MOD dataset showing the general patterns throughout the domain. The colormap used for the frequencies of each bin is shown as well and matches the colormap for all following figures.

histogram which can be referenced by individually selecting the corresponding region. This has the advantage of clearly representing the distribution regardless of the number of particles that are sampled in that region.

To help the user navigate the visualization, we provide a mini-map that shows both the outline of the grid layout as well as the outline of the selected histogram. This feature gives users domain-level context when exploring the data at high zoom levels. In addition, when the user selects a sampling region, the associated histogram is drawn larger at the side of the view space for easier viewing. This feature also allows the user to more accurately select specific histogram bins of interest when looking to identify additional information such as the exact value of the bin. In addition, a line is drawn from the center of the histogram to the selected bin to help the user perceive the direction of the associated velocity.

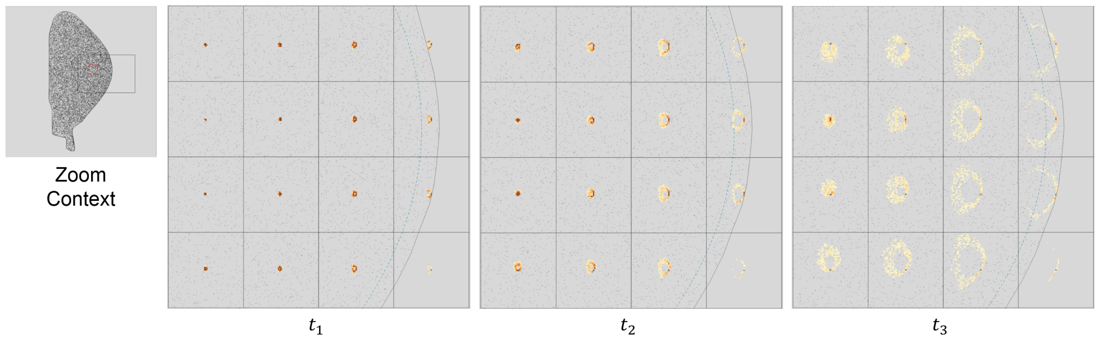


Figure 3.5. A zoomed view of the C-MOD dataset showing three select timesteps. Over time, unique ring-like patterns begin to form showing that slower-moving particles tend to move in a vertical direction and faster-moving particles tend to move towards the left. The presence of these rings seems to cycle on and off throughout the duration of the simulation.

Other relevant information is automatically displayed as well. This includes information such as the velocity range corresponding to the selected histogram bin and the number of particles in the sampling region. Additional domain-specific information can also be displayed as needed. When applicable, controls are present for 2D slice extraction from 3D data sets. The user can control the slice thickness and position via sliders as well as the axis of the orthogonal view. A slider is also used to adjust the current timestep. Additionally, the system can cycle through timesteps automatically, freeing up the mouse to control other interactive parameters.

3.2 Results

We test our visualization tool with real-world datasets to demonstrate its usefulness.

3.2.1 Fusion Data

We use the Alcator C-MOD data described in the Appendix, Section A.2. The particle subset that we were provided from this simulation consists of $\sim 50,000$ particles per time step. For this dataset, we visualize displacement vectors based on the time series of particle data.

The simulation domain of fusion devices represents a complex torus-like shape. However, scientists are often interested in the motion of the plasma towards or away from chamber walls (in the direction of the "minor radius" of the torus). As a result, many visualizations (including our own) focus on presenting information on a 2D "poloidal" slice where the less interesting motion in the third dimension is hidden. We project all particles throughout the torus onto the slice view so that the resulting histogram-based visualization, while 2D in nature, represents information from the entire 3D domain.

Appendix 3.4 shows an example of the histogram-based visualization for the C-MOD dataset. In this case, we directly sample any particles that fall into a histogram region (as denoted by the black grid). From this zoomed-out overview image, we can see general patterns that permeate the domain. Since the center of each histogram/region represents a zero velocity, we can see that many particles, heavy ions, in this case, are moving around the slice in both a clockwise and counterclockwise direction. While the majority are moving slowly (as shown in a dark color), smaller groups of particles are moving quickly (as shown in a lighter color). Particles near the top and bottom of the slice are either slow-moving or stationary.

After obtaining an overview of the general patterns throughout the data, users can explore details by zooming into certain regions and adjusting the resolution of the histogram grid and the histograms themselves. Appendix 3.5 shows a zoomed view of a select set of timesteps throughout the simulation. From the histograms, we can tell that these particles, electrons, in this case, begin by moving very slowly. Over time, their displacement vectors begin to increase in magnitude and ring-like patterns begin to form in the histograms. These rings show that the slower moving particles tend to move vertically while the faster moving particles tend to all move towards the left. Animating through the simulation shows that the presence of these ring-like structures cycles on and off at key timesteps.

Lastly, we demonstrate the ability to deform the histograms into non-Cartesian structures of interest. One such structure, the separatrix, represents an important magnetic flux surface in the tokamak geometry. In this case, we use a layout scheme that places histograms along the separatrix surface (defined in the acquired dataset) and transform the velocities into components parallel and perpendicular to the surface. This allows users to analyze the motion of particles relative to this curve. Appendix 3.6 shows an overview

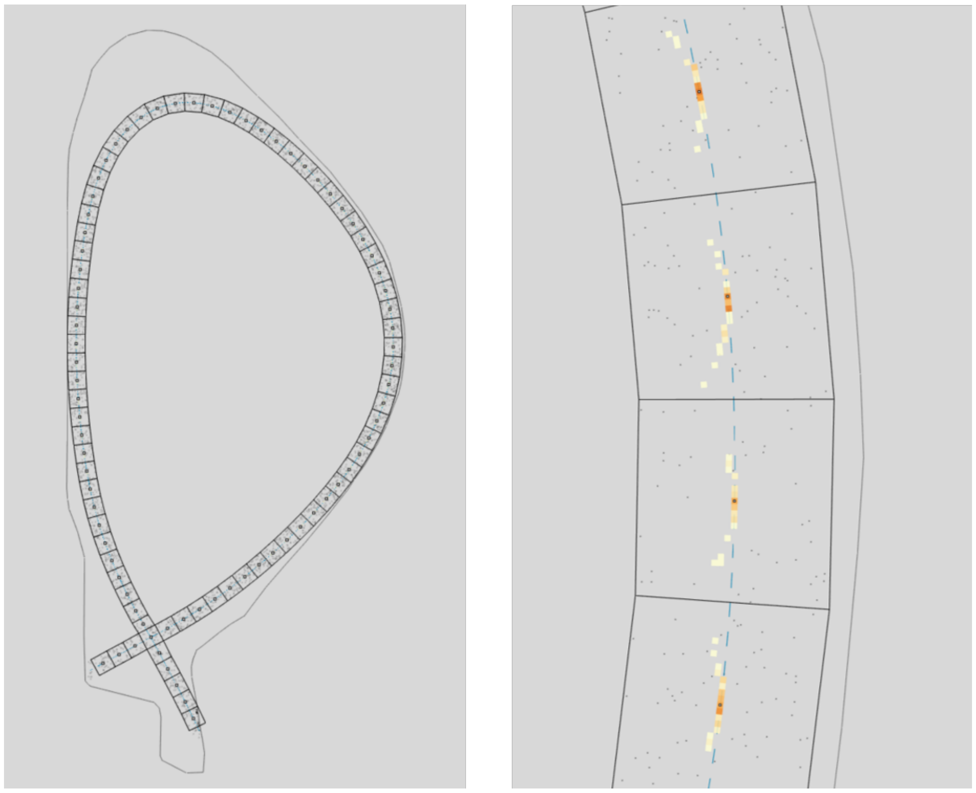


Figure 3.6. Left) An overview showing the histogram layout on the separatrix surface. Right) A zoomed view showing that only higher-velocity particles have a perpendicular velocity component to this surface.

of this new layout, as well as a zoomed-in version. Just like the Cartesian layout, the position of the histogram bin relative to the center of the histogram denotes the velocity direction that the bin represents. We can see that most of the particles tend to move parallel to the separatrix surface except for some higher-velocity particles which have a slight perpendicular component towards the center of the slice.

3.2.2 Combustion Data

The combustion dataset we use is described in the appendix A.3.1.1. In this dataset, being able to explore all three dimensions of the domain is important to combustion scientists. Since our histogram-based visualization is 2D in nature, we use the interactivity of the system to explore different aspects of the domain. For example, users can choose to view the data from multiple orthogonal axes. In addition, the histograms can represent particles sampled from a slice in the data whose position and thickness are both adjustable

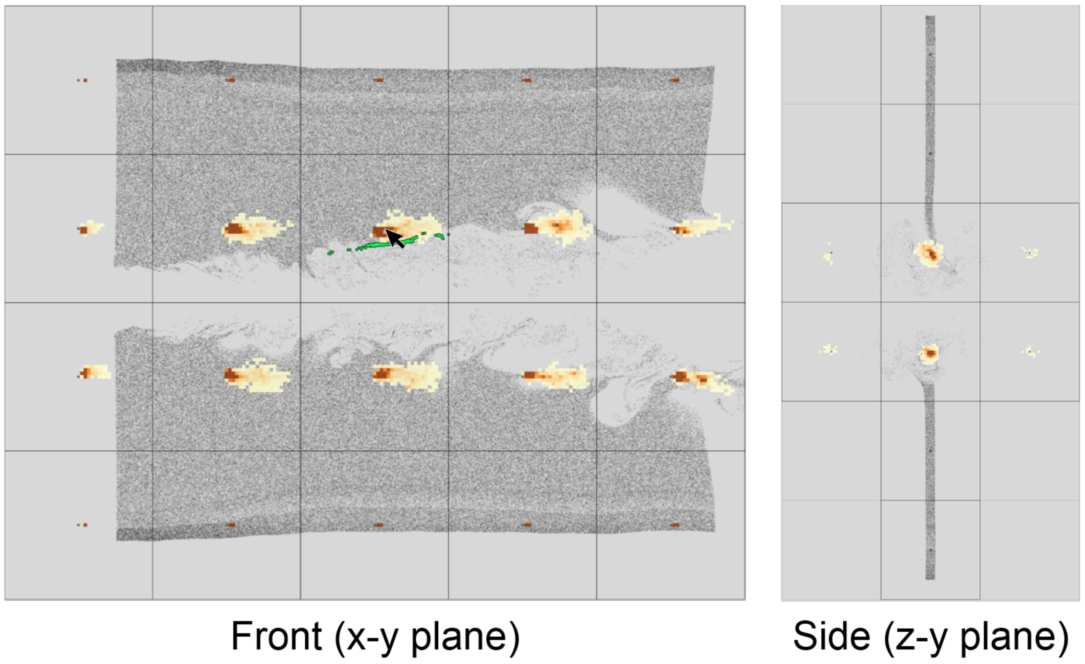


Figure 3.7. Two images showing the combustion dataset as viewed from different orthogonal directions. From the x-y plane, it is clear that particles near the center of the combustion jet are moving both quickly and slowly to the right. Particles near the top and bottom of the images are fairly stationary. A set of particles is highlighted in green when mousing over a particular histogram bin of interest.

in the user interface.

From the histogram-based view in Appendix 3.7, we can see the velocity decomposition of the particles from multiple orthogonal viewing angles. Viewing from the front, one can see that particles are moving towards the right both quickly and slowly near the center of the jet, whereas particles near the top and bottom of the image are stationary. In addition, mousing over a particular histogram bin highlights the corresponding set of particles in green. Viewing from the sideshows similar patterns but reveals an additional dimension of motion. Upon careful inspection, there is a greater variation in the velocities in the top half of the image when compared to the bottom half.

3.2.3 Marine Life Movement Dataset

The last data we test comes from the Tagging of Pelagic Predators (TOPP) [103] dataset, Appendix A.4.2. This is a geospatial movement dataset that tracks the motion of marine life throughout the Pacific Ocean. Datasets like these are essential in understanding

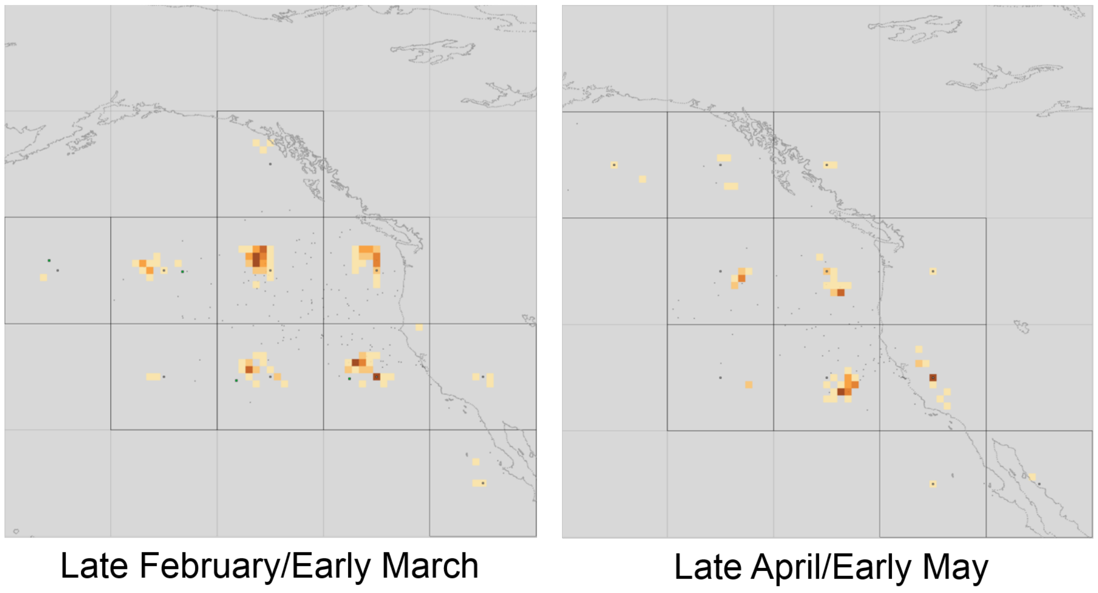


Figure 3.8. The velocity decomposition of elephant seals at two different times of the year near the California coast. Left) The direction of movement is primarily towards the northwest as the seals migrate into the Pacific Ocean to feed. Right) The direction of movement is primarily to the southeast as the seals return to the coast.

animal behavior in terms of migratory or feeding patterns. We choose this dataset to show that our histogram-based visualization tool has applicability in areas outside of flow visualization.

Appendix 3.8 shows an example of our visualization applied to this geospatial movement dataset. More specifically, we look at the migratory patterns of elephant seals that inhabit the California coast. On the left, we can see that the primary direction of motion is towards the northwest in late February and early March. This is a result of the seals migrating into the Pacific Ocean to feed after a long breeding period on the coast. A few months later in late April and early March, the direction of motion reverses towards the south-east when many of the seals return to the coast.

3.3 Discussion

Overall, we present an interactive system which utilizes the advantages of velocity-based histograms to visualize and explore the motion of a group of objects. While many traditional techniques are subject to a trade-off between visual clutter and loss of detail, a histogram-based representation can provide a clear description of the velocity decomposition of a system of study. In the next chapter, we extend our approach to also use an in situ generation scheme which can be used for extreme-scale applications.

Chapter 4

Visualizing Weighted Particle Distribution Functions

This chapter is based on material from our publication, *Scalable Visualization of Time-varying Multi-parameter Distributions Using Spatially Organized Histograms* [104]. © 2016 IEEE.

In the previous chapter, we introduced an approach to visualize the movements of discrete objects based on spatially organized histograms. In this chapter, we expand on this approach, with an in situ processing scheme to compute histograms during the simulation where the full data is available. We then leverage the same zoomable panning window approach as in the previous chapter. We also extend the visualization system with a number of features, such as the support to visualize weighted particle distributions, hierarchically drilling down into the particle weight distributions, and linking with phase plots. Additionally, we add a linked temporal view and build in support for visualizing the temporal evolution of a selected 2D distribution function using temporal stacking and then rendering the time-extended 2D+1 data stacks through 3D isosurface rendering. Different projections of the spatially organized histograms are explored to enhance the ability to select regions of interest, and capabilities to derive variables through an embedded DSL for mathematical expressions and using them to define new plots to use in the different views are included. We present case studies using this approach with the XGC data

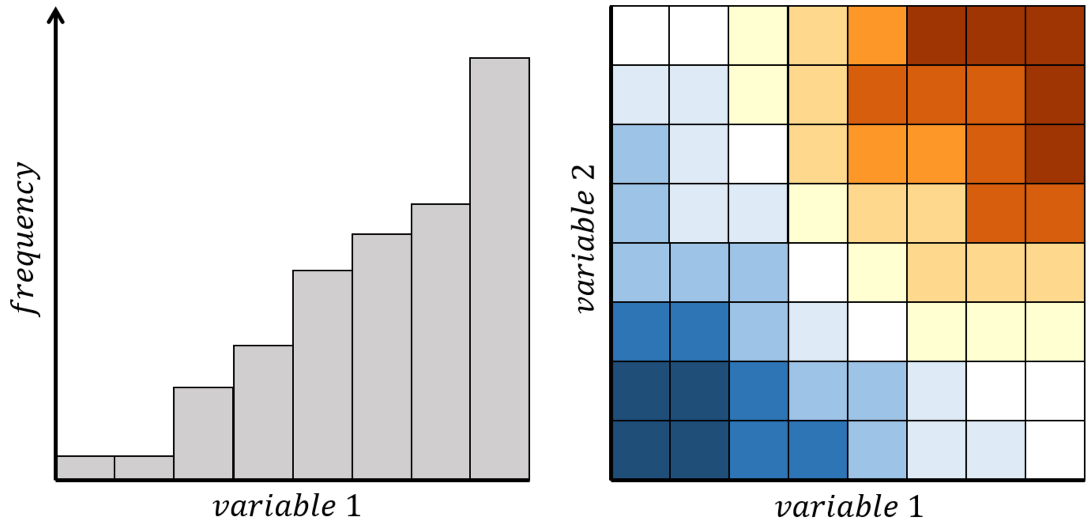


Figure 4.1. An image depicting the visual representation used for 1D and 2D histograms in this work. Left) A 1D histogram discretized into bins. The frequency of entities sampled into each bin is represented by the height of the bar. Right) A 2D histogram with each bin represented by a colored cell. Darker colors represent bins with higher frequency, and the hue can differentiate between positively (red-orange) and negatively (blue) weighted samples.

described in Section A.1.1, the GTS data in Section A.1.2, the Ethylene Flame data from Section A.3.1.1, and the AECP3 Accelerator data from Section A.2.1. Performance results for the on-the-fly scheme from the previous chapter are included and compared against the in situ scheme presented in this chapter.

4.1 Preliminaries

Before we describe our methods, we introduce some preliminary background knowledge about the types of histograms we can make use of as well as the data types that our methods can support.

4.1.1 Weighted Histograms

In some cases, it is necessary or useful to weight the contributions of each sampled object. We compute our weighted histograms as follows, where $H(i,j)$ gives the value of the bin at row and column (i,j), $bin(i,j)$ represents the set of objects mapped to it, and w_k is the weight for the sampled object, O_k.

$$H(i,j) = \sum_{O_k \in bin(i,j)} w_k$$

In various particle-in-cell fusion simulations such as XGC [37] and GTC [36], the simulation particles each represent a variable number of real-particles. In this case, scientists need to use this value as a weight in order to see the proper distributions corresponding to the modeled physical system. Further, these weights represent perturbations from a Maxwellian background, and thus can be negative or positive. Still, visualizing the distributions of unweighted simulation particles can be useful for making sense of how the simulation is behaving.

Alternatively, one could use any variable in the data as a weight when generating the histograms. This allows users to view additional information in the limited 1D or 2D spaces since the frequency counts are now being adjusted by a separate variable. However, it must be considered that the "frequency" of a certain bin can be caused by a large number of samples with a small weight, or a small number of samples with a large weight (see Section 4.2.3.4). As an analogy, suppose you want to see distributions of campaign contributions in a local election. You could compute an unweighted histogram showing how many people contributed to a campaign. Alternatively, you could weight each contribution by its amount in order to see how money/influence is distributed. Weighting contributions to one party negative and the other positive allows one to see how deviations from a neutral position are distributed. See Figure 4.1 for an illustration.

4.1.2 Applicable Data Types

Any set of data points embedded in a mathematical space can be sampled in order to summarize their distributions. In this chapter, we focus on particle data from fusion and accelerator simulations. However, we could also operate on data associated with discrete grid points. Furthermore, as we can also sample entities in non-physical spaces, we can visualize and study many different combinations of variables in a single visualization. Thus the techniques in this chapter can be applied to many multi-parameter datasets, and are not limited to just scientific simulation data.

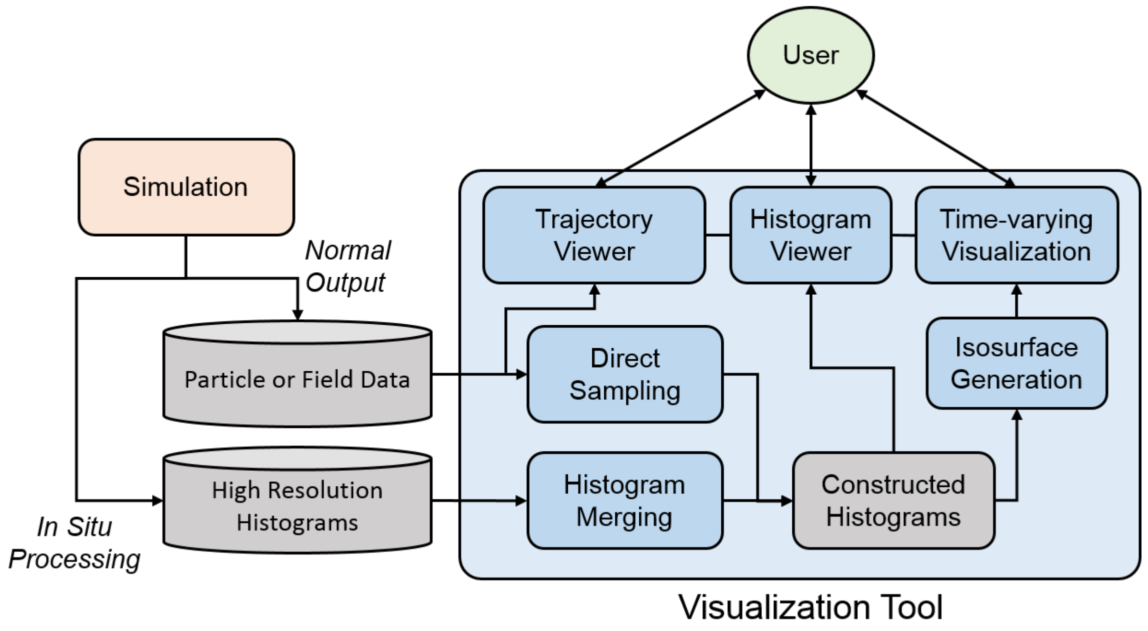


Figure 4.2. An overview of our workflow. Histograms can be constructed on-the-fly directly from particle data or from pre-generated high-resolution histograms that were computed in situ. A user interfaces with the visualization software and can explore the data using three linked views: a histogram viewer which shows the spatially organized histograms, a trajectory viewer which shows the trajectories of particles corresponding to selected bins, and a time-varying visualization which uses isosurfaces to show temporal patterns of a selected histogram.

4.2 Methods

4.2.1 Overview

One important aspect of such a technique is the ability to interactively explore the data through a variety of adjustable parameters and configurations, (*e.g.*, chosen variables, size/distribution of sampling regions, histogram resolution, etc.). As a result, the histograms displayed to a user need to be constructed at runtime; however, this is one of the most computationally expensive steps. We, therefore, provide two schemes for histogram generation each designed to handle different dataset sizes as shown in Figure 4.2. For smaller datasets, we utilize GPU acceleration to efficiently sample the raw simulation data directly. This scheme has been described already in Chapter 3. For large-scale datasets, we provide an alternate method that can emulate this same interactivity. Instead of sampling the raw data values directly, a set of high-resolution histograms are

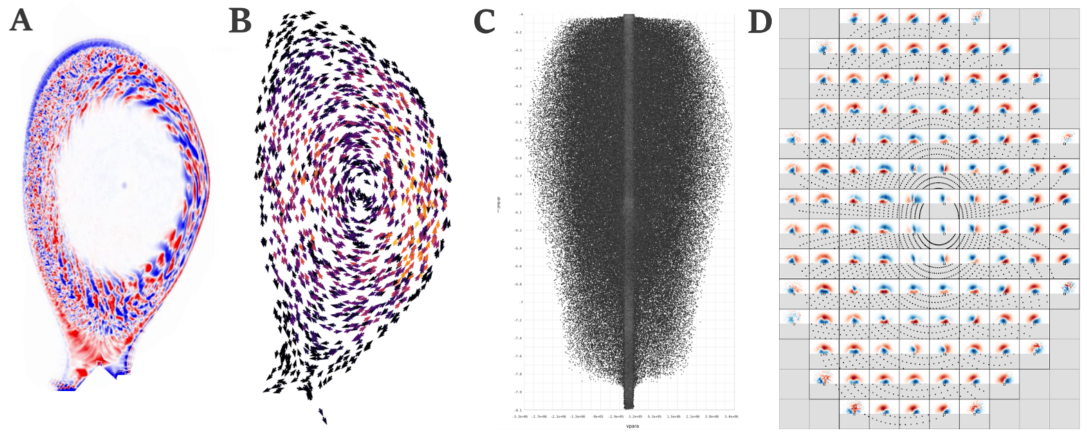


Figure 4.3. A) Color can be used to highlight spatial variations throughout the domain but does not show trends between variables. B) A vector plot showing particle motion (color mapped to magnitude) results in clutter, even when plotting only 0.5% of the full data. C) A scatter plot can be used to show trends between two different variables in the particle data but does not show any spatial variations. D) Spatially organized histograms can display both trends between different variables and spatial variations throughout the domain in a low-clutter manner.

constructed in situ and saved to disk. These can then be sampled in real-time according to the user-defined parameters.

The visualization tool itself provides multiple linked views which can be used to explore different aspects of the data. A trajectory viewer is used to directly view raw simulation data and provides spatial context as well as detailed information on demand. The histogram viewer presents a set of spatially organized histograms that can present major and minor trends in the data in a low-clutter manner. This view focuses on exploring spatial differences in the distributions of a sample. Lastly, a time-varying visualization is provided to highlight temporal variations in histogram distributions. Histograms of interest are stacked into a 3D volume, and isosurfaces are used to visually represent major trends.

4.2.2 Spatially Organized Histograms

The main advantage of using spatially organized histograms is illustrated in Figure 4.3. Traditionally visualizations can expose spatial variations in the data by plotting values directly and using color to represent variable values (part A). While this makes spatial variations clear, it is difficult to compare multiple variables at once without introducing confusion, over-plotting, and clutter. Furthermore, using vector plots to describe particle motion can result in a great deal of clutter (part B). To compare trends between multiple variables, phase-space plots may be used where each axis represents a particular variable (part C). However, such a view does not show spatial variations throughout the domain. A spatially organized set of histograms can be used to represent both spatial variations as well as trends between variables simultaneously in a low-clutter manner (part D).

4.2.2.1 In Situ Generation and Sampling

The interactive capabilities available through an on-the-fly sampling method are a major facet of our visualization tool. However, the computational overhead of histogram generation in larger datasets limits the responsiveness of the system, impeding real-time exploration. As a result, we implement an alternative in situ scheme which can be used to handle large-scale datasets while minimizing any loss of functionality in the visualization tool. In this scheme, raw data values are directly sampled and transformed into a histogram-based representation during the runtime of the simulation allowing us to accumulate the statistical information of objects at much larger scales.

Each of these pre-generated histograms is saved so that they can be sampled in real-time by the visualization tool during post-processing analysis. Each histogram is sampled onto a mesh, dependent on the type of simulation, where each grid point represents the center of a 3D volume from which objects are sampled. Since the size of each sampling region can vary, especially in unstructured grids, the volume of the sampling region is also saved. This will in turn be used to more accurately sample histograms in the visualization tool. The histograms themselves represent distribution functions that are normalized using a normalization factor according to their relative intensities. Users can control a balance between the temporal resolution of output steps, the spatial resolution of the sampling mesh, and the phase-space resolution (number of histogram bins) to suit their needs.

Our interactive system then samples this more manageable data representation for interactive exploration. Because the grid sizes can be dense and unstructured, it is often effective to visualize an overview at a reduced level of detail. For this reason, we partition the domain into disjoint sampling regions, as we do in the on-the-fly scheme, and sample

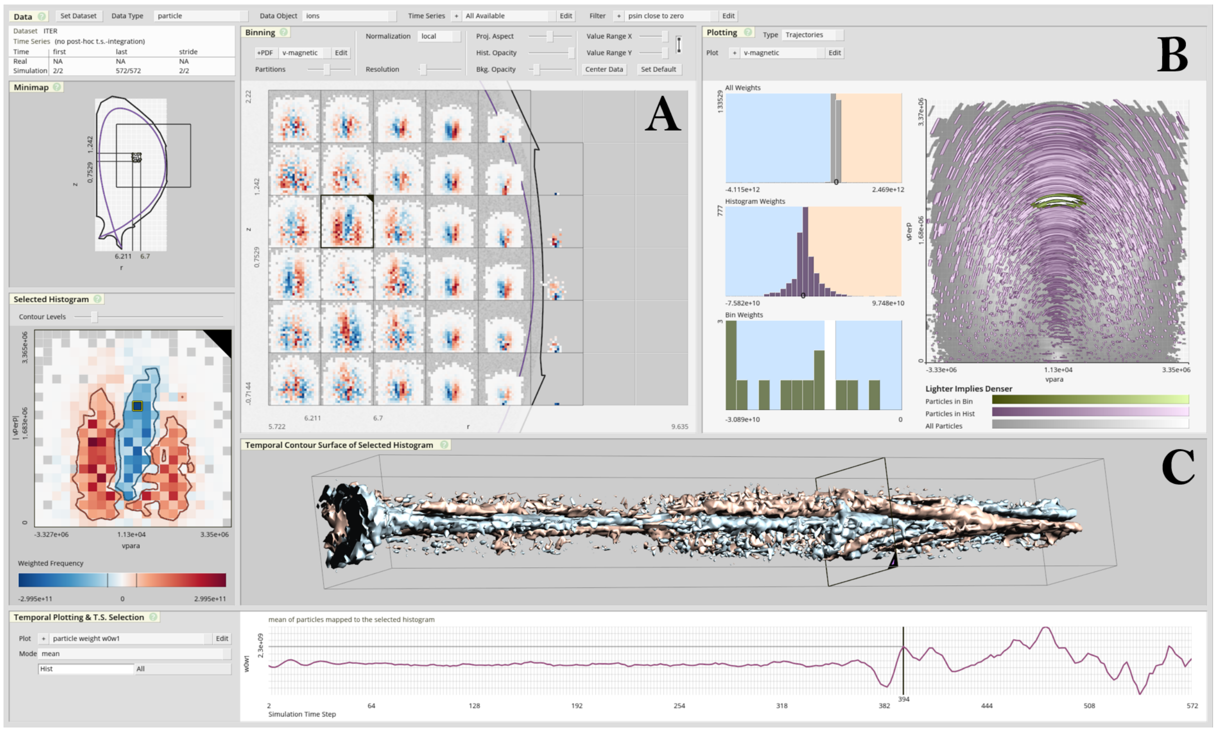


Figure 4.4. A screenshot of the user interface. The interface consists of three main views: the histogram view (A), the trajectory view (B), the time-varying view (C). Additionally a minimap depicting the zoom and location of the histogram view is shown on the top-left and a detailed view of the selected histogram is shown on the bottom-left.

grid points (their associated raw histograms) by region. The histogram values of grid points that fall into the same sampling region are merged using the normalization factor, N, and the net sampling volume, $\sum_{k=0}^{M-1} V_k$, where k is the grid point index, and M is the number of grid points. The normalization factor is equal to the inverse of the total phase-space volume where particles are sampled from and is computed during the simulation. The resulting histogram values are computed as follows:

$$H(i,j) = \frac{N}{\sum_{k=0}^{M-1} V_k} \sum_{k=0}^{M-1} h_k(i,j)$$

where $h_k(i,j)$ is the 2D histogram of a sampled grid point with index k and $H(i,j)$ is the resulting merged histogram, and M is the number of grid points.

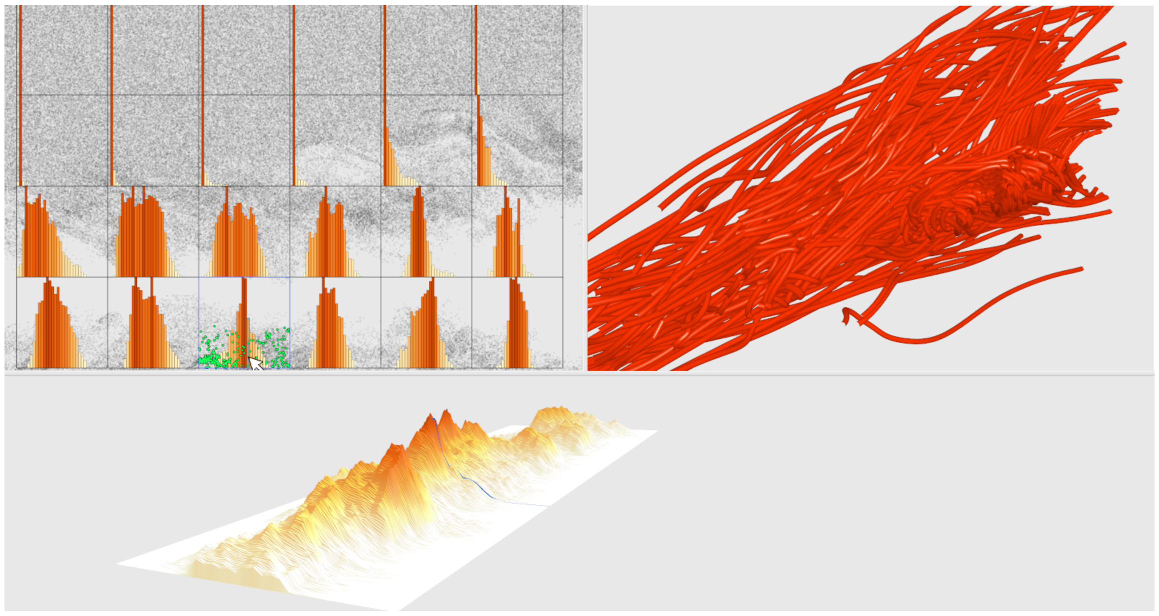


Figure 4.5. An alternate example of the three main visualization views. A 1D histogram is generated from the particle data in the histogram view (top-left). Trajectories for particles corresponding to a moused-over bin are shown in 3D in the trajectory view (top-right). A spectrogram-like plot shows the evolution of the distribution in the selected histogram over time in the time-varying view (bottom).

4.2.3 Visualization System

The visualization system ties together three main interactive views: a histogram view, a trajectory view, and a time-varying visualization. Each representation presents a different perspective on the data and allows for real-time exploration. Furthermore, each view is linked so that selections and interactions in one view simultaneously affect the other. This multi-faceted approach allows users to investigate multiple aspects of the data at the same time. Figure 4.4 shows an overview of the user interface of the visualization tool.

4.2.3.1 The Histogram View

The histogram view presents the 1D or 2D distribution functions computed using either the on-the-fly or in situ sampling schemes. By comparing the relative frequencies of different histogram bins, users can explore trends between multiple variables and spatial regions throughout the simulation domain simultaneously. Figure 4.5 shows the use of 1D histograms in the histogram view.

By default, this view uses a panning window approach. The screen space is partitioned into a regular grid of sampling regions. Entities within each sampling region are used to construct histograms in real-time based on the user-defined parameters. The user interface can be used to control these parameters, such as the number of partitions, the resolution of the histograms, or the variables used for sampling. When the user drags the mouse over the view space or zooms in, the sampled objects move relative to the screen, while the grid remains fixed. We also employ a layout that remains fixed with respect to the particles as the user pans and zooms. Such a layout has the benefit that zooming and panning can be done without affecting the sampling partitions. However, the panning window approach gives more control over the locations of the sampling regions with respect to the particles.

One potential disadvantage of partitioning the space into rectangular sampling regions is that such regions may not conform well to the geometries underlying the data. This can be alleviated by projecting the data into different spatial layouts. For example, the Tokamak device geometry is based on the magnetic flux surfaces that act to confine the plasma. The two spatial variables of interest in this space are the magnetic radius and the angle about the center of the poloidal plane. By projecting the data from this space into Cartesian coordinates, such that the poloidal angle is mapped to the x-axis, and the magnetic radius is mapped to the y-axis, the user can easily select regions that conform to tokamak geometries of interest as in Figure 4.6. Furthermore, we can scale the projection in each dimension separately, allowing us to sample from larger ranges of one of the variables relative to the other.

4.2.3.2 The Trajectory View

The trajectory view is used to show the overall motion of sets of particles (or other objects) in either physical or phase space. Conveniently, the histogram view provides a powerful and unique way of selecting desired particle subsets. Due to the simple and regular structure of the spatially organized histograms, the user can apply mouse interactions over a sampling region and a bin in the associated histogram, and the system can unambiguously and efficiently extract the corresponding sampled objects. This process ultimately implements a 2-level range-based selection, with the first level corresponding to a 2D spatial range (sampling region/histogram) and the lower level a 1D or 2D value

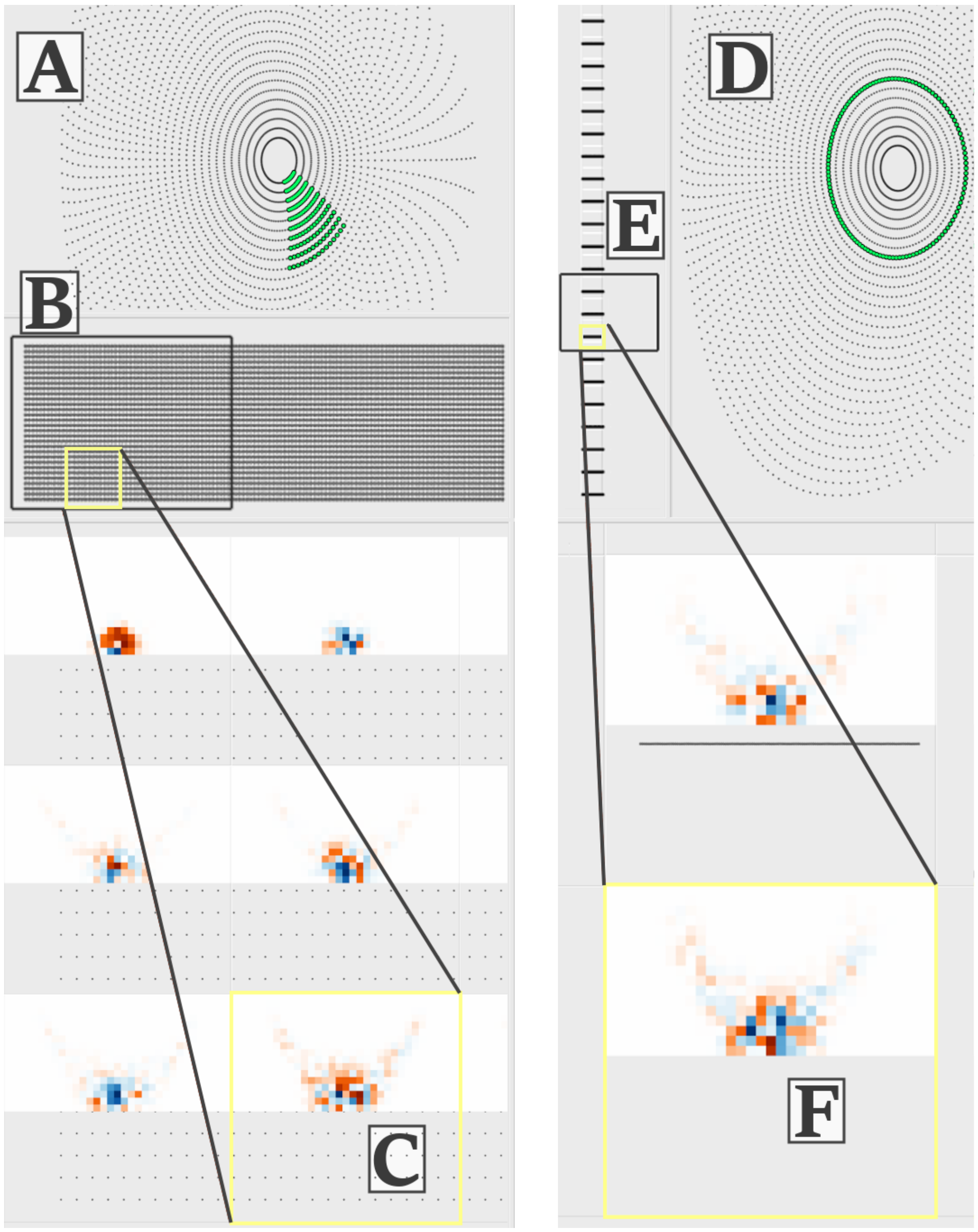


Figure 4.6. Projecting curved geometry into a Cartesian viewport for easier visualization and interpretation. A,B,C) Projecting sampling points into a Cartesian x-y layout based on radius and sweeping angle. A) The selected grid points in another spatial context, B) The mini-map view, C) The histograms overlaid over the projection. D,E,F) The same type of projection with one dimension scaled to an extreme in order to favor sampling the points based on one of the spatial variables over the other. D) The selected grid points, E) The mini-map, F) The associated histograms.

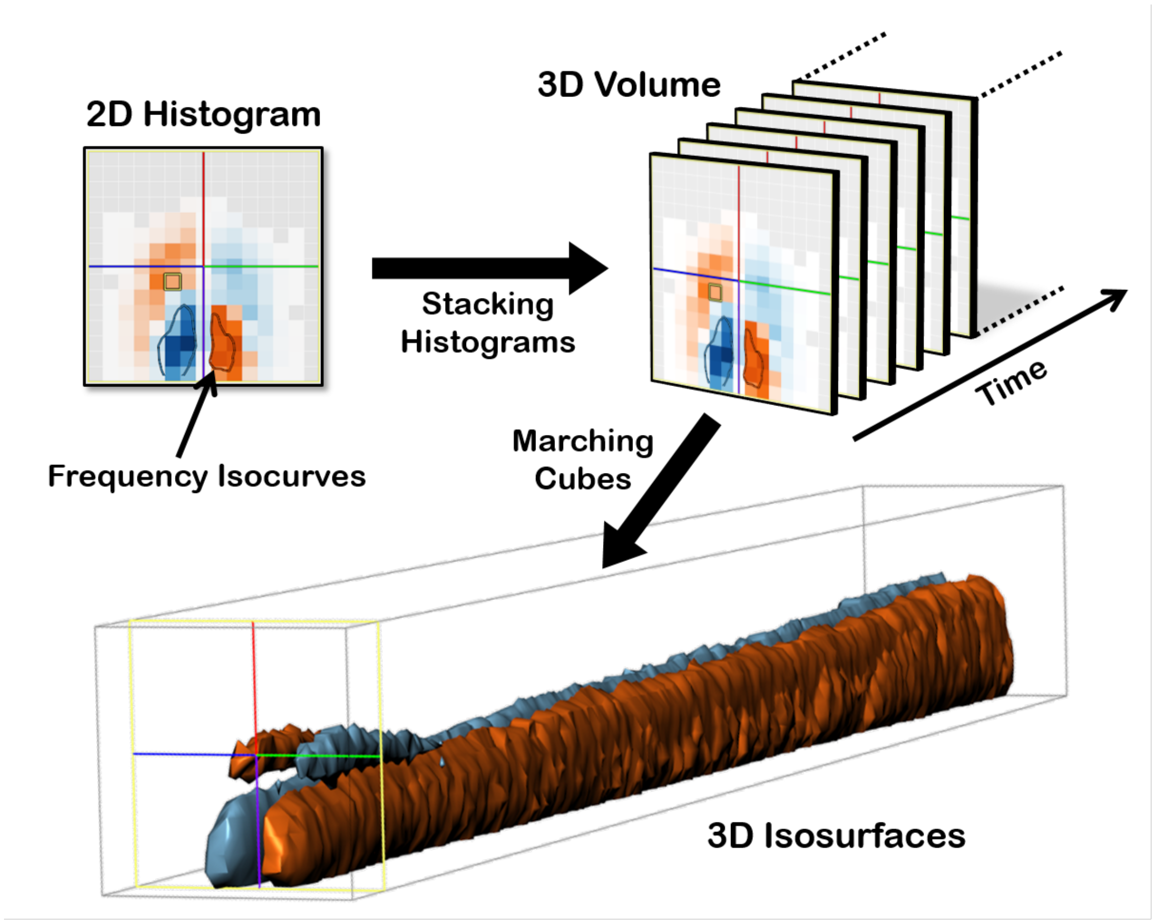


Figure 4.7. The steps involved in generating the isosurfaces in the time-varying view. First, a frequency isovalue is chosen and displayed as isocurves on the currently selected histogram. Next, histograms using the same sampling parameters are generated for all available time steps and stacked into a 3D volume. Lastly, an isosurface is generated with the selected isovalue using marching cubes.

range (bin of a 1D or 2D histogram). After the user makes such a selection, the particle trajectories are constructed and displayed in the trajectory view.

Since this process occurs in real-time, users can interactively explore the motion of particle subsets in either physical space, phase space, or both. An example of a selection can be seen in Figure 4.4B. In this case, the extent of all trajectories is drawn in gray, the trajectories corresponding to the selected histogram are drawn in purple, and the trajectories corresponding to the selected bin are drawn in green. The lightness/darkness of the color represents the density of trajectories at that location. To the left of the trajectory view, bar charts show the distribution of weights in the dataset as well as the selected histogram and bin. This feature provides useful information in the case where the histograms are weighted and also provides an interface for additional weight-based levels of particle selection. See Section 4.2.3.4 for more details.

4.2.3.3 The Time-varying Visualization

While the histogram view is effective for comparing trends between variables as well as spatial differences throughout the domain, it's not effective for visualizing trends over time. As a result, we implement a time-varying view that can display the temporal properties of a selected histogram. Occlusion and over-plotting make it difficult and confusing to view time-varying patterns for each bin in a 2D histogram. Instead, we rely on isosurface techniques to extract surfaces from a volume that represents the histograms values at each time step. We choose isosurfaces because they are a simple and intuitive way of exploring a 3D volume. While a more flexible direct volume rendering approach could be another option, it would rely on careful selection of the transfer function, which can be a burden on the user and can result in a more complex and difficult to interpret visualization.

Figure 4.7 describes the process through which we generate the time-varying view. First, a histogram of interest is selected. A user then chooses a frequency isovalue. This forms a set of contours that separates the 2D histogram into regions where the bins have a frequency higher than the isovalue and regions where the bins have a frequency lower than the isovalue. Next, additional 2D histograms over a series of time steps are generated using the same set of parameters (sampling region size, number of bins, etc.). These are then stacked into a 3D volume, where two dimensions represent each of the histogram variables and the third dimension represents time.

As the construction of the time-varying visualization requires computing histograms over many time steps, it often represents the primary performance bottleneck of the overall system. To accelerate this process we use the GPU. To maximize performance, we want high saturation of computational resources and a balanced workload between GPU threads. We must also prevent multiple threads from attempting to update the same bin value (at a single memory location) simultaneously. These considerations affect our choice of how to parallelize the computation, and which GPU computing platform to use.

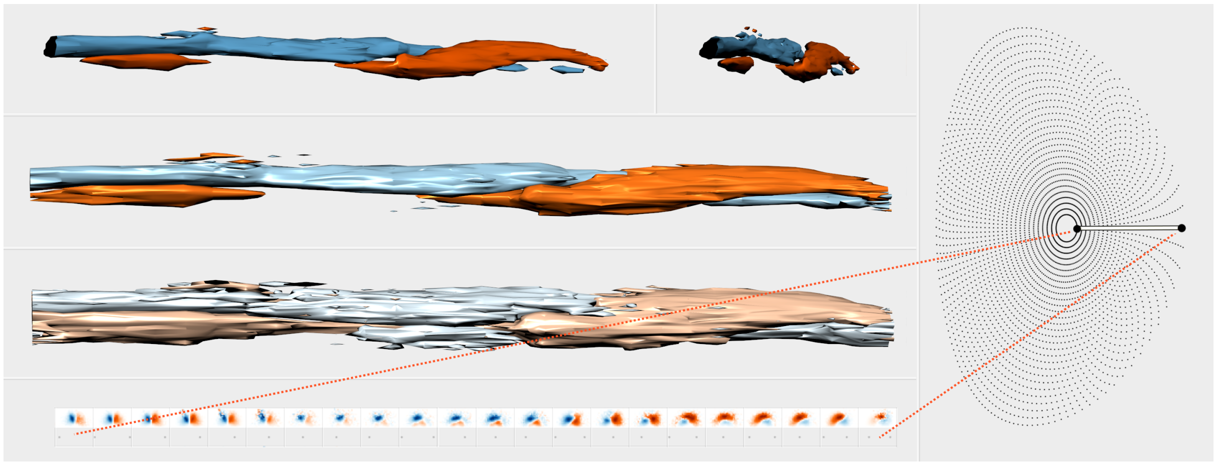


Figure 4.8. An image demonstrating the ability to use the time-varying view to construct isosurfaces over a spatial path rather than through time. Histograms are sampled along this path (shown at the right). These are then stacked into a 3D volume to be visualized by isosurfaces. The views on the top left show isosurfaces from various isovalues and viewing angles and reveal distinct trends in the data.

One option would be to assign each GPU thread a separate chunk of time steps. This has the benefit that concurrent writes to the same memory address are implicitly avoided as the computation for each time step is independent of each other. However, because the number of time steps is often small relative to the number of cores in modern GPUs and each time step may correspond to a large chunk of data, this can lead to a load balancing problem. Another option is to assign threads work based on particle id. Because the number of particles is typically much larger than the number of time steps as well as GPU cores, this approach may result in better load balancing. The drawback is that different particles handled by different threads may be mapped to the same bin, and therefore some locking mechanism is required to ensure that only one thread updates a bin's memory address at a time.

Modern GPU architectures, *e.g.* NVIDIA's compute architectures since Kepler [105], support efficient hardware-based atomic operations on floating-point data. Specifically, we can use CUDA's `atomic_add(float*, float)` function to increment a bin within the GPU kernel. Because we found parallelization over the particle ids using atomic operations to be much more efficient than parallelization over the time steps, we choose to use this method.

We must also consider that all of the data needed to compute the temporal view may not fit in GPU memory at once. In this case, chunks of the data need to be processed one at a time, which means that large amounts of data need to be transferred to the GPU each time the histogram parameters have changed and the visualization needs to be recomputed. In the tests, we used a Titan X graphics card with 12GB of memory, which could safely store a little over 500 time steps $\times$ 1M particles $\times$ 5 32-bit floating-point variables persistently in memory. In addition to the particle data, GPU memory is allocated for the volume/histogram stack, but because the volume is typically very small relative to the size of the particle data, its memory footprint is insignificant.

When utilizing the in situ scheme, histograms have already been computed per grid point and thus typically represent a smaller data size than the raw particles. In addition, the grid points do not move over time, so the mapping of a grid point to a sampling region needs to be done only once. As a result, this procedure can be done reasonably quickly using only CPU thread parallelism along with CPU vector instructions.

Once the 3D volume has been computed, we perform marching cubes [106] to generate the isosurface. In general, the time it takes to construct the isosurface geometry is insignificant compared to the time it takes to generate the 3D volume. Performance tests for histogram stacking and isosurface generation can be found in section 4.3. Note that it is possible to select multiple isovalues resulting in the generation of multiple isosurfaces.

Once the isosurface has been generated, users are free to pan, rotate, or zoom around the constructed mesh. An intersecting slicing plane indicates the currently selected time step and helps to orient the viewer. Users can also adjust any histogram or isovalue parameter and receive real-time feedback on how the mesh changes form.

When visualizing a 1D histogram over time, we use a waterfall plot. In this case, we can incorporate the values of each bin into the visualization. The 1D histograms are stacked and connected into complex polygons depth-wise over time. An example of this can be seen in Figure 4.5.

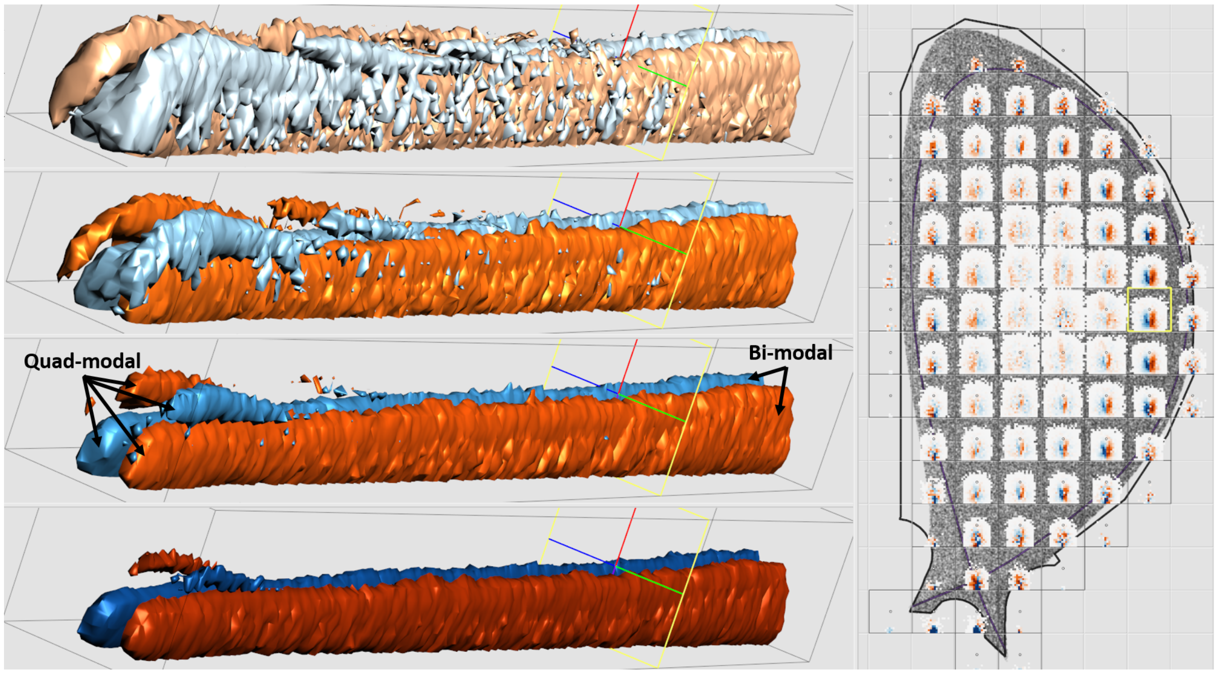


Figure 4.9. Using the time-varying visualization to study temporal trends for a selected histogram in the XGC dataset. The horizontal histogram variable represents the velocity parallel to the magnetic field, while the vertical histogram variable represents the velocity perpendicular to the magnetic field. Left) The temporal view is shown using 4 different isovalues, low to high from top to bottom. Right) The histogram view. A global threshold was used to map color to frequency, and the volume was not normalized per-time step. This way the visualization highlights the differences in overall weighted particle frequency in different areas of the domain as well as over time. One interesting aspect shown in this visualization is how the distribution starts out quad-modal and evolves to become bi-modal.

4.2.3.4 Other Features

As previously described, a particular bin in a weighted histogram could represent different numbers of discrete particles depending on their weights; a small number of high-weight particles and a large number low weight particles could map to the same "weighted frequency". As a result, we also provide a visualization of the distribution of weights from a selected bin or histogram, as well as from the full particle data. This can be seen in the left portion of Figure 4.4B as 1D histograms. Selecting a subset of particle weights from the distribution highlights the bar and any corresponding trajectories in white.

An additional feature is the ability to stack histograms into a 3D volume based on a physical path through the domain rather than through time. In this case, histograms are generated along a discrete set of points along the path. Each point represents the center of a sampling region from which particles are sampled. In this case, the isosurfaces represent continuous variations in the distribution between variables along this physical path. An example of this can be seen in Figure 4.8 where grid points are sampled along a line from the center of the poloidal plane to the edge of the high field (right) side. The histogram stack is ordered by radial distance and shows how the distributions change according to this spatial variation.

The normalization of histograms is another factor that users can adjust interactively. Sometimes it is useful to visualize variations in bin frequency, or overall sample sizes, across multiple histograms at once. In such cases, it is important to ensure that the color mapping used is globally consistent across every region and time step. However, this can significantly reduce the usable color range for histograms with relatively low maximum bin frequencies, and as a result, can hide patterns and make variations in-perceivable for certain individual histograms. For this reason, the user interface includes a slider to apply a global normalization factor in order to amplify unpronounced features. We also provide a local normalization feature that allows the full range of color to be used in each histogram independent of its maximum frequency relative to the other histograms. Local normalization is favorable for showing how the variables are distributed within each region individually. When the user chooses local normalization, the histogram view is affected as well as the temporal volume, in where the histograms will be normalized on a per-histogram, per-time step basis.

The system also features an embedded mathematical expression DSL, based on the *exprtk* mathematical expressions library [107]. The user can define sets of constants, build new constants from existing ones, and derive variables based on the raw variables in the data as well as the constants. The system compiles the expressions on the fly, and signifies when the expression is valid. The derived variables can then be used in user defined phase plots, projections, and time plots. Figure 4.11 shows the interface for the timeline plots. These features are demonstrated in the video demo for the Baleen software system listed in Section B.4.

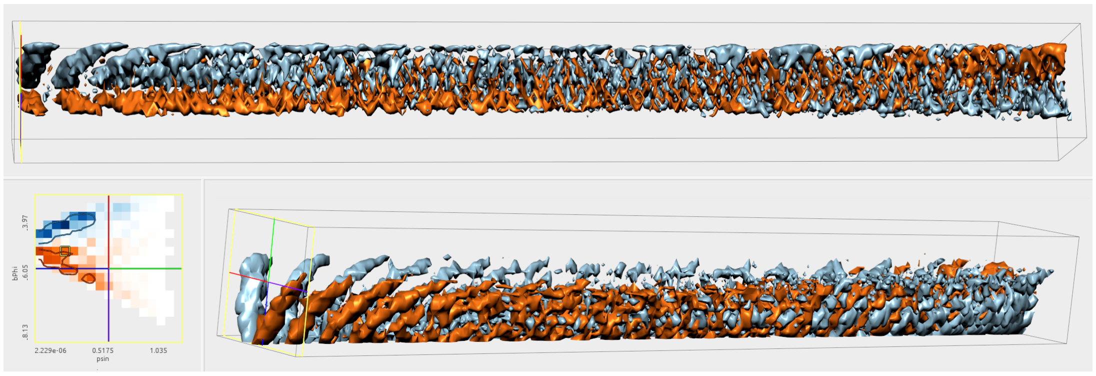
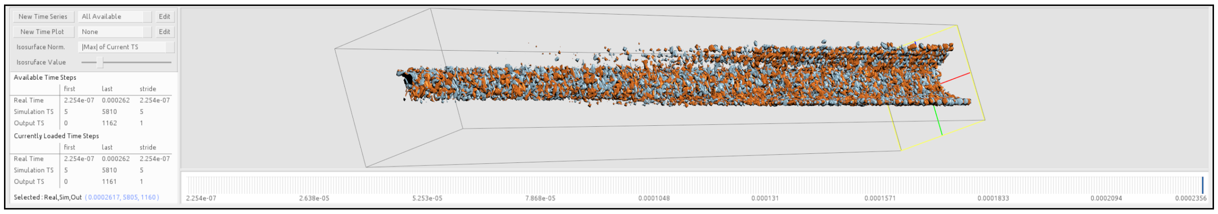


Figure 4.10. Using the time-varying visualization to study temporal trends for a selected histogram in the XGC dataset. The horizontal histogram variable represents the magnetic radius, while the vertical histogram variable represents the magnetic field strength in a direction that runs around the torus geometry. The isosurfaces reveal complex wavelike patterns which tend to swap places with their positively and negatively weighted counterparts.



New Time Series	All Available	Edit
New Time Plot	None	Edit
Isosurface Norm.	|Max| of Current TS	
Isosruface Value		

Available Time Steps

	first	last	stride
Real Time	2.254e-07	0.000262	2.254e-07
Simulation TS	5	5810	5
Output TS	0	1162	1

Currently Loaded Time Steps

	first	last	stride
Real Time	2.254e-07	0.000262	2.254e-07
Simulation TS	5	5810	5
Output TS	0	1161	1

Selected : Real,Sim,Out (0.0002617, 5805, 1160)

Figure 4.11. The timeline view with the interface for defining new temporal plots or selecting from existing ones.

4.3 Results

We demonstrate the effectiveness of our system using a set of real-world datasets in the fields of fusion and particle accelerator research. We use results from two different fusion simulations to test the on-the-fly and in situ sampling schemes provided by our system and demonstrate the unique patterns that the visualization can highlight. We then test the system using a particle accelerator dataset to show its applicability to other fields of science. Lastly, we provide performance results to justify the interactivity of our technique when using either sampling scheme.

4.3.1 Fusion Datasets

The fusion datasets include the XGC and GTS datasets described in the Appendix A.1.1 and A.1.2 respectively. Their spatial domains and coordinate systems are illustrated in Figures A.1 and A.2. As in the previous chapter, the data is toroidally integrated and

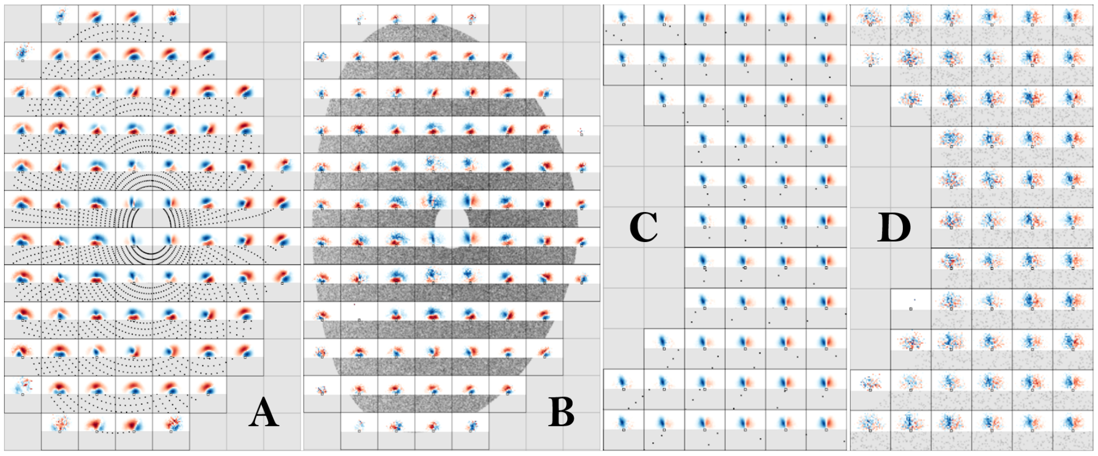


Figure 4.12. A) Visualizing the in situ generated histograms by merging histograms from the simulation grid points within each sampling region. B) Comparing the same simulation using histograms generated on-the-fly from particles directly. C/D) Zooming in the domain from A and B respectively highlights the advantages of using the in situ method. When the sampling regions have fewer particles to sample (D) the histograms become noisy. This is not an issue in the in situ case (C) since it was able to sample a much larger number of particles during the simulation.

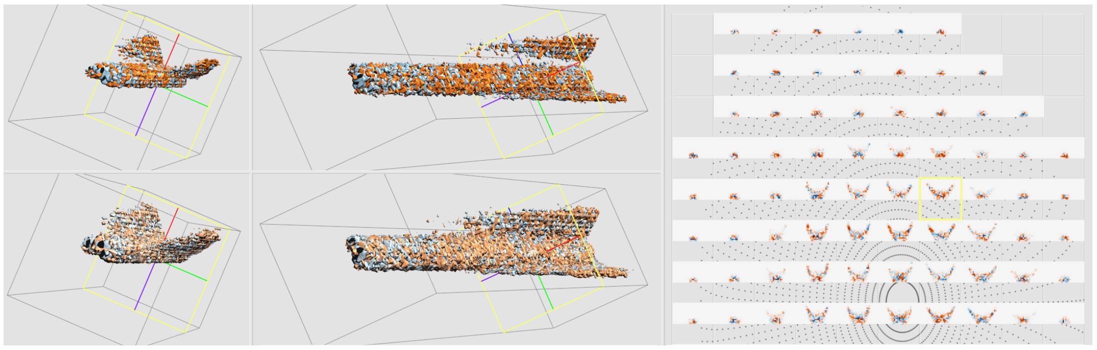


Figure 4.13. An image depicting the in situ generated histograms from the GTS dataset. The horizontal histogram variable represents the velocity parallel to the magnetic field, while the vertical histogram variable represents the velocity perpendicular to the magnetic field. The right side shows spatial variations in the distributions throughout the simulation domain. The left side shows two time-varying isosurfaces of the selected histogram over time at different isovalues. This reveals a unimodal to trimodal evolution of the distribution.

projected onto the poloidal plane.

4.3.1.1 On-the-fly generation with XGC

Figure 4.9 shows an example of the system employing the on-the-fly sampling scheme to visualize the XGC dataset. In this case, the weight of the particle describes the perturbation from the background (Maxwellian) distribution. Positive/negative weights describe an increase/decrease of particle population from the background. The histograms were computed using these weights, and a divergent color map was used to differentiate negatively (blue) and positively (red) valued bins. From the figure, it can be seen that at the beginning of the simulation, the distribution is quad-modal alternating between negative and positive and eventually becomes bi-modal. Additionally, one can see how the weights grow over time since the surfaces associated with fixed bin frequencies/isovalues expand in size as the simulation progresses.

When using our visualization tool, physicists have described that it can easily investigate the perturbed distribution function and the particle trajectories which are responsible for the perturbation. They mention that one very useful example is monitoring the growth of particle weights. In particle simulations, excessive growth of particle weight could induce statistical noises and degrade the accuracy of the simulations. Hence, monitoring the growth of particle weights and identifying the cause is important to regulate the statistical noises.

Another example can be seen in Figure 4.10. In this example, the horizontal histogram variable represents the magnetic radius, while the vertical histogram variable represents the magnetic field strength in a direction that runs around the torus geometry. The time-varying isosurfaces reveal complex wavelike patterns forming in distinct positively and negatively weighted groups. Furthermore, these groups tend to swap positions over time occupying different regions of phase space.

4.3.1.2 In Situ Generation with GTS

Next, we test the in situ sampling scheme using data from the GTS simulation. Histograms were generated during the simulation with access to the full resolution particles. These histograms are then sampled based on the user-defined parameters of the sampling regions. The values in the histograms represent energy distributions in that the particles are weighted by energy relative to a background value (which allows them to be positive or negative).

The horizontal axis of each histogram represents the velocity of the particle parallel to the direction of the magnetic field, whereas the vertical axis represents the magnitude

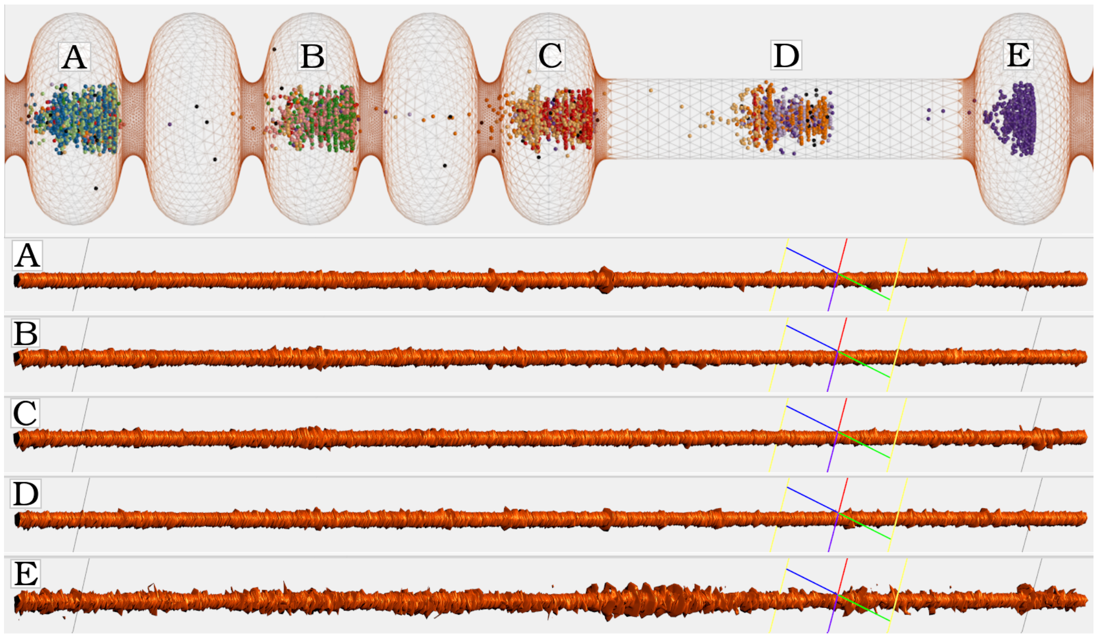


Figure 4.14. The top of the image shows a visualization of clusters of particles in the cryomodule device with the mesh depicting the shape of the device/individual cavities. In this case, they were colored categorically according to which cavity they were emitted from. Temporal histogram stack isosurfaces are computed for each cluster based on momentum in the x and y directions (perpendicular to the beam) and are shown side by side for comparison. As opposed to the other clusters, the leading cluster (E) is made up entirely of particles that were emitted from the same cavity. Additionally, the particles that make up this cluster experienced a higher than normal level of instability, especially during a time segment starting about halfway through the available time steps.

of the velocity perpendicular to the magnetic field. Since the perpendicular velocity is represented as a magnitude, it is always positive and results in only the top half of our histograms bearing values. These types of velocity plots are commonly used when studying gyrokinetic simulations.

Figure 4.12 shows an example of the in situ sampled histograms using our visualization tool (A/C) vs. a comparison to the on-the-fly particle-based sampling scheme (B/D). Due to I/O limitations, the simulation is only able to dump a small subset of the full particle data, forcing the on-the-fly method to sample only 3% of all particles. The points in the background show the grid points from which the pregenerated histograms were merged. From the images, it is clear that the in situ version was able to capture more detail since it represents statistical information from a much larger sample size. This is further exacerbated when zooming into the domain; as fewer particles are sampled per histogram, eventually we observe excessive degradation of statistical quality(D).

Figure 4.13 shows an alternate run of the GTS simulation and focuses on studying the time-varying properties of the histograms. The left side of the figure shows three different isosurfaces of the selected histogram (which is outlined in yellow) with time increasing towards the right. Each of these isosurfaces shows a unimodal distribution towards the start of the run with nearly all particles exhibiting a small parallel and perpendicular velocity. This evolves into a trimodal distribution consisting of a small parallel and perpendicular velocity, a large positive parallel and large positive perpendicular velocity, and lastly a large negative parallel and large positive perpendicular velocity. This is what forms the distinct "V-like" shape in the images.

We can also look at spatial variations within a single time step (near the end of the simulation, as indicated by the yellow indicator in the 3D views) as shown on the right side of the image. We can see that these "V-like" distributions are more prevalent near the center of the tokamak cutting plane. This gradually transforms to the unimodal distribution near the edges which only contain particles with small parallel and perpendicular velocities. Furthermore, there seems to be an even influence between positively and negatively weighted particles in both space and time.

4.3.2 Accelerator Dataset

Our next example uses data from a simulation called ACE3P [108] which is used to study the electromagnetic dynamics within particle accelerators. The specific device this simulation run studies is called a cryomodule which uses a set of resonating cavities to accelerate the particles. A better understanding of certain processes, such as dark current, in which charged particles become emitted from the cavity surfaces and enter the accelerating beam, can lead to an improved design of the device.

In this application, we sampled particles from the accelerator data over a projection of r (radial distance) and z (along the length of the cryomodule). However, in this case, we were interested in examining the behavior of different clusters of particles over time rather than the behaviors of particles in specific regions. For this reason, we used the spatial layout only to make our initial selections based on the cluster, and then used the computed histograms, for the particles in those clusters, at each time step to construct the temporal histogram stack. The histograms here represent momentum in the x and y directions (perpendicular to the length of the device). The visualization was made from about 15,000 particles over about 3,000 time steps. This gives an overview of the movement of the particles perpendicular to the beam and easily highlights segments in time where apparent extreme or abnormal behavior occurs.

Figure 4.14 shows a comparison of 5 different clusters of particles (labeled A-E). In the time-varying view, time is increasing towards the right. Since each isosurface represents momentum perpendicular to the motion of the beam, the thickness of the isosurface tube represents a point in time where instabilities are occurring which can cause particles to exit the beam and become deposited into a cavity downstream potentially damaging the device. Comparing each of the clusters, it is clear that cluster E contains the most amount of instabilities over the entire time window, which may explain why it has fewer particles than clusters A-D.

4.3.3 Performance Results

The performance results for generating spatially distributed 2D histograms for a single time step can be found in our previous work [109]. This section will instead focus on the performance results for generating the 3D temporal volume of stacked histograms for both the on-the-fly and in-situ methods. This generation step is the largest bottleneck as performing marching cubes to construct an isosurface is much faster in comparison considering the typical sizes of the associated volumes. For example, a volume representing a 32×32 bin 2D histogram over time could include over 16,000 time steps (much larger than the typical use case) before exceeding the size of a 256^3 volume (which is relatively small by today's standards). For testing, we used an Intel i7-5939K processor with 6 cores at 3.5 GHz and a Titan X graphics card with 12 GB of memory.

Figure 4.15 shows the timing results for the on-the-fly method. The graph shows the time it takes to construct the 3D volume as a function of the number of time steps in the data. Each curve represents a different number of particles that are sampled. Since the number of particles in each sampling region can vary over time, we artificially ensured that each particle would be mapped to the selected sampling region, which represents the worst-case computationally. In practice, the performance should be higher on average as in the intended use case the selected sampling region will contain only a fraction of the full data.

The graph itself shows a general linear increase in the time it takes to compute the 3D volume with a distinct jump that occurs at the points where all of the particle data does not fit into GPU memory. At that point, chunks of data need to be transferred to the GPU and processed one at a time whenever the volume needs to be recomputed. The red curve shows the most extreme test case where we were able to process over 22 GB of particle data (1,000,000 particles $\times$ (2 spatial variables + 2 histogram variables + one weighting variable) $\times$ 4 bytes) into a volume in less than 2 seconds. As the number of particles increases, the user may need to reduce the number of time steps and vice-versa, however the visualization can remain interactive with reasonably large data sizes. For example, with 1 million particles per time step, we can generate volumes representing 500 time steps at a time and still get reasonably high frame rates. This delay can be alleviated further with the use of additional GPUs which could process each of the chunks simultaneously.

Figure 4.16 shows the performance tests of the in situ versions. In this case, each curve represents a different number of grid points that are within the selected sampling region. Because the grid points remain fixed over time (unlike particles), mapping them to sampling regions only needs to be done once, each time the sampling layout changes and the rest of the computation is just a summation, which can be done very efficiently even without using GPU acceleration. These timing results are based on our parallel CPU implementation (using OpenMP) which also utilizes CPU vector operations to further increase performance.

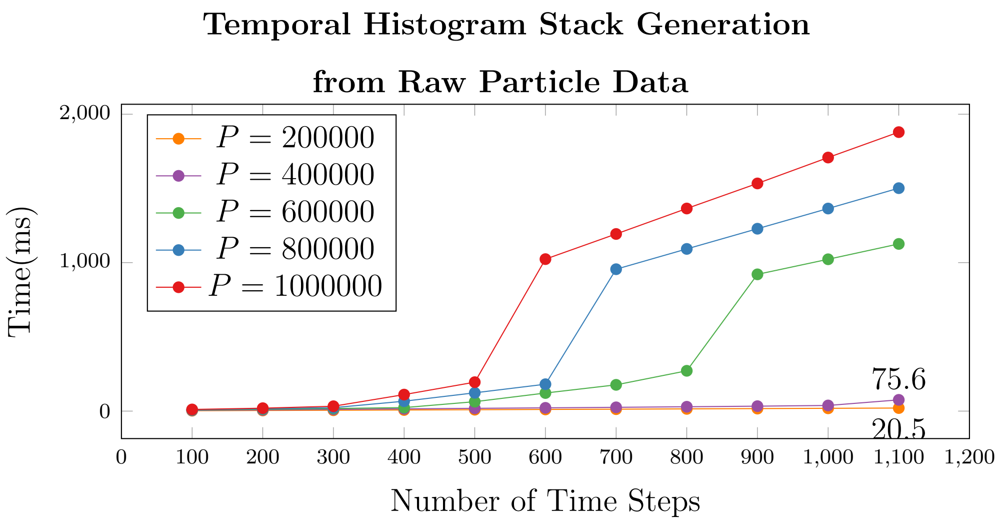


Figure 4.15. Timing results for the time-varying visualization using the on-the-fly method with varying numbers of particles (P). A jump in time occurs at the point where the GPU can no longer keep all of the loaded time steps in memory. Sampling from 1M particles, we were able to achieve fluid interaction while computing up to 500 time steps per volume.

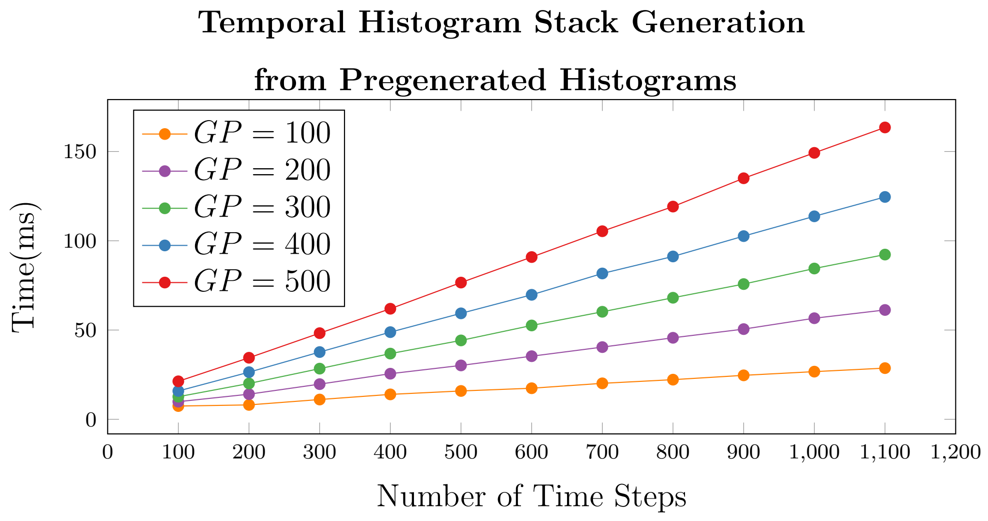


Figure 4.16. Timing results for the time-varying visualization using the in situ method with varying numbers of grid points/pregenerated histograms sampled (GP). Since each thread is simply computing a summation over the number of grid points, for disjoint sets of time steps, it scales linearly and is very computationally efficient.

4.4 Discussion

The results demonstrate the ability of our system to not only visualize trends between variables in conjunction with spatial trends but also to analyze time-varying properties as well. Maintaining generality allows the system to be applicable to a large variety of data types while employing multiple data processing schemes allows the system to handle various data scales.

4.4.1 Scalability

Using the on-the-fly scheme with GPU acceleration, the data scales that desktop computers can handle depend on the number of data entities (samples $\times$ time steps). In general, the system can handle up to $\sim$ 10 billion data entities before there is a several-second delay when requesting a time-varying representation from a particular sampling region. Limits can be pushed further by utilizing multiple GPUs or even a distributed setting that can process chunks of particles simultaneously.

For larger data scales, the system can utilize the in situ sampling method. Since histograms are pregenerated during simulation time, they can represent information from massive numbers of particles. While the in situ sampling method cannot be used to select trajectories (since the particle data is not available), the larger statistical sample size provided can be a huge advantage.

Chapter 5

Visualizing Multivariate Trajectories with Recurrence Plots

This chapter is mainly based on material from our publication, *Interactive Spatiotemporal Visualization of Phase Space Particle Trajectories Using Distance Plots* [104] © 2019 IEEE, but also contains material from our paper, *An Interactive Visualization System for Large Sets of Phase Space Trajectories* [101]. © 2019 John Wiley and Sons.

The distance plot (or unthresholded recurrence plot) has been shown to be a useful tool for analyzing spatiotemporal patterns in high-dimensional phase-space trajectories. In this chapter, we incorporate this technique into an interactive visualization with multiple linked phase plots and extend the distance plot to also visualize marker particle weights from particle-in-cell (PIC) simulations together with the phase-space trajectories. By linking the distance plot with phase plots, one can more easily investigate the spatiotemporal patterns, and by extending the plot to visualize particle weights in conjunction with the phase-space trajectories, the visualization better supports the needs of domain experts studying particle-in-cell simulations. We demonstrate our resulting visualization design using particles from the XGC ITER tokamak fusion simulation described in Section A.1.1.

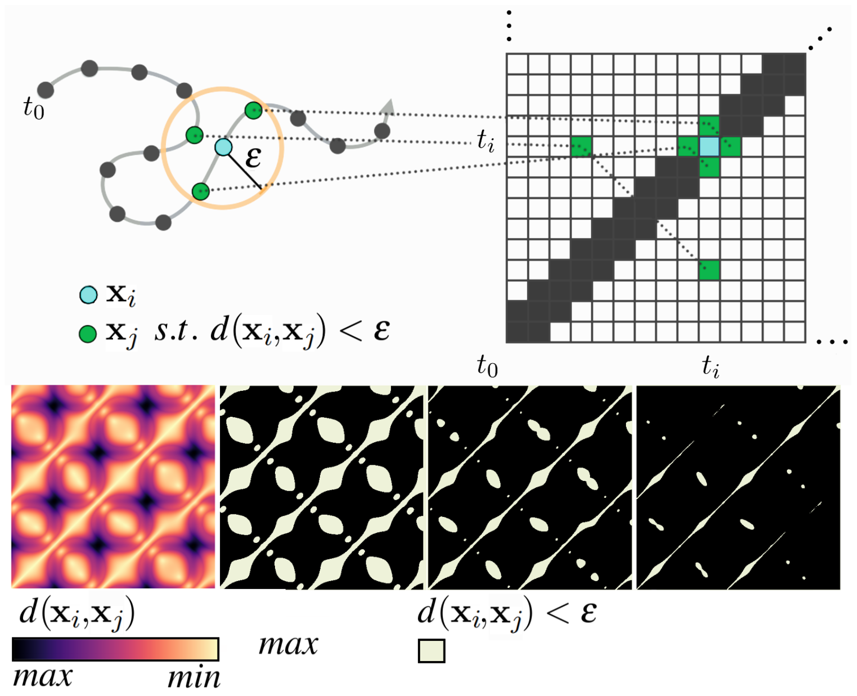
Distance plots and recurrence plots [110] are useful for highlighting recurrent patterns and dynamical transitions in phase-space trajectories from multidimensional complex systems. Despite an embrace by communities in domain sciences and applied mathematics, the visualization community has yet to robustly investigate their use as an interactive tool for qualitative visualization. Specifically, in many-particle systems (especially PIC simulations where particles commonly carry time-varying weights), there are several issues that should be addressed if one wishes to utilize these techniques. Standard procedures for applying recurrence quantification analysis (RQA) are error-prone and tedious even for simpler single component systems. The significance of a single component (*e.g.*, particle) in a many-component system such as a PIC tokamak simulation, the overall complexity of the entire system and its boundary conditions, and the complex meaning of a trajectory that carries weight, makes the setting highly non-trivial. A typical application of RQA to these kinds of particle trajectories would be problematic. Despite these limitations, it is clear that recurrence plots, and especially the distance plot, can be useful for understanding the spatiotemporal patterns of these trajectories. In this chapter, we show how this can be done through the combination of a novel distance plot visualization, and linked phase-space views.

Our main contribution is the extension of the plot to also visualize the weights of marker particles in PIC simulations. This is important for researchers studying tokamak fusion simulations since the particles often carry time-varying weights which identify each particle's contribution to the overall particle distribution. While interactive recurrence plots have been used for information visualization [111] [112], linking them with phase plots for studying scientific simulation data appears to be novel in application. A small additional novelty to our visualization is the incorporation of marks on the distance plot to indicate occurrences of events. We apply these methods to study phase-space trajectories of particles from the XGC tokamak fusion simulation [113]. Our preliminary results show interesting patterns related to little-understood dynamical interactions between the phase-space trajectories and evolution of particle weights, and correlations between the weights and trajectories in terms of their (quasi) periodicity.

5.1 Background

The phase space of a dynamical system includes those variables which are necessary to uniquely describe the system's state. A trajectory of a dynamical system, therefore, represents an evolution of the system's state over time from a particular initial condition. A single particle in a many-particle system can be viewed as a component (or subsystem) of the overall system, that follows its own phase-space trajectory.

In practice, it is common to refer to a plot in which time is a parameter rather than an axis as a phase plot, regardless of whether each axis is technically one of the actual phase-space variables of the system.



t_0
ε
t_i
$\mathbf{x}_i$
$\mathbf{x}_j$ s.t. $d(\mathbf{x}_i,\mathbf{x}_j) < \varepsilon$
t_0
t_i
$d(\mathbf{x}_i,\mathbf{x}_j)$
max
min
max
$d(\mathbf{x}_i,\mathbf{x}_j) < \varepsilon$

Figure 5.1. Top) An illustration of a recurrence within a dynamical system. Each point represents a phase-space position (or state) $\mathbf{x}$ along a phase-space trajectory. Recurrence occurs when the distance $d(\mathbf{x}_i, \mathbf{x}_j)$ between a pair of states at discrete time steps (i, j) is within a threshold, ϵ (the recurrence threshold). Bottom) An example recurrence plot computed from the phase-space trajectory of an ion in a tokamak fusion simulation. On the left is the un-thresholded recurrence plot (or distance matrix). To the right are the distance plots derived from it using progressively smaller thresholding parameters.

5.1.1 Distance and Recurrence plots

A recurrence plot is a symmetric binary matrix where the columns/rows represent time steps, and the cells (i, j) represent the mapping $R(i, j) = 1$ if $||\mathbf{x}_i - \mathbf{x}_j|| < \epsilon$ else 0, where $\mathbf{x}_t$ is phase-space vector of the system at time t. Similarly, the unthresholded recurrence plot (or distance matrix) is the matrix with each cells (i, j) representing the distances $||\mathbf{x}_i - \mathbf{x}_j||$. Figure 5.1 demonstrates visually how the recurrence plot is computed and what it represents.

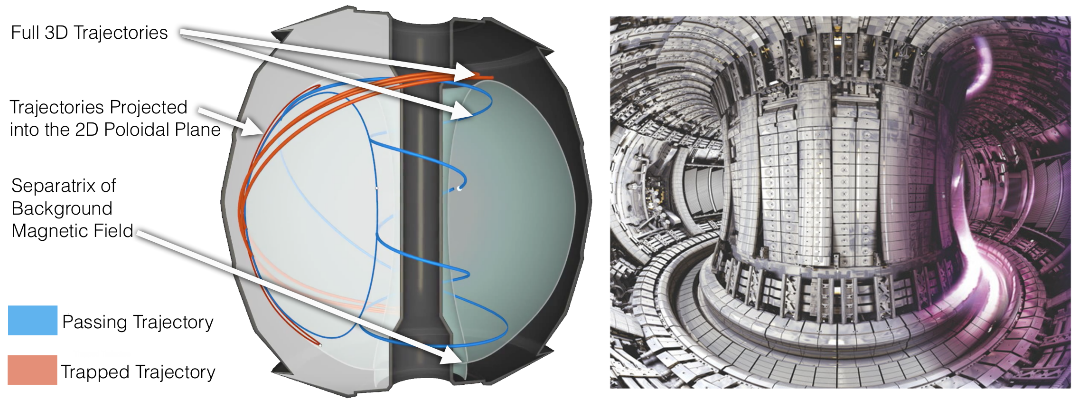

Full 3D Trajectories
Trajectories Projected into the 2D Poloidal Plane
Separatrix of Background Magnetic Field
🟦 Passing Trajectory
🟥 Trapped Trajectory

Figure 5.2. A depiction of the tokamak fusion device. Right) An image of the inside of a real device. Left) A cross section depicting the poloidal plane, and two important types of particle modes: trapped (blue), and passing (red).

5.1.2 Requirements

Figure 5.2 depicts the tokamak device as well as the two important types of particle trajectories (trapped and passing) that are featured in our analysis. Our design is motivated by three primary challenges. First, the trajectories of interest are typically at least 5D. One approach is the use of multiple linked views to visualize different sub-spaces. However, we wish to identify patterns in the higher dimensional space since they are meaningful for domain scientists to understand the system's dynamics. Distance plots are useful for finding high dimensional patterns in dynamical systems, yet they lack spatial context. Second, physicists who develop and study XGC stress the importance of analyzing the particle weights in conjunction with the phase-space trajectories. Third, we would like to support the study of a phenomenon of interest. These factors motivated our design which includes an extended distance plot that also shows the particle weights, markers indicating occurrences of discrete events that cue specific phenomena and includes multiple linked 2D phase plot views.

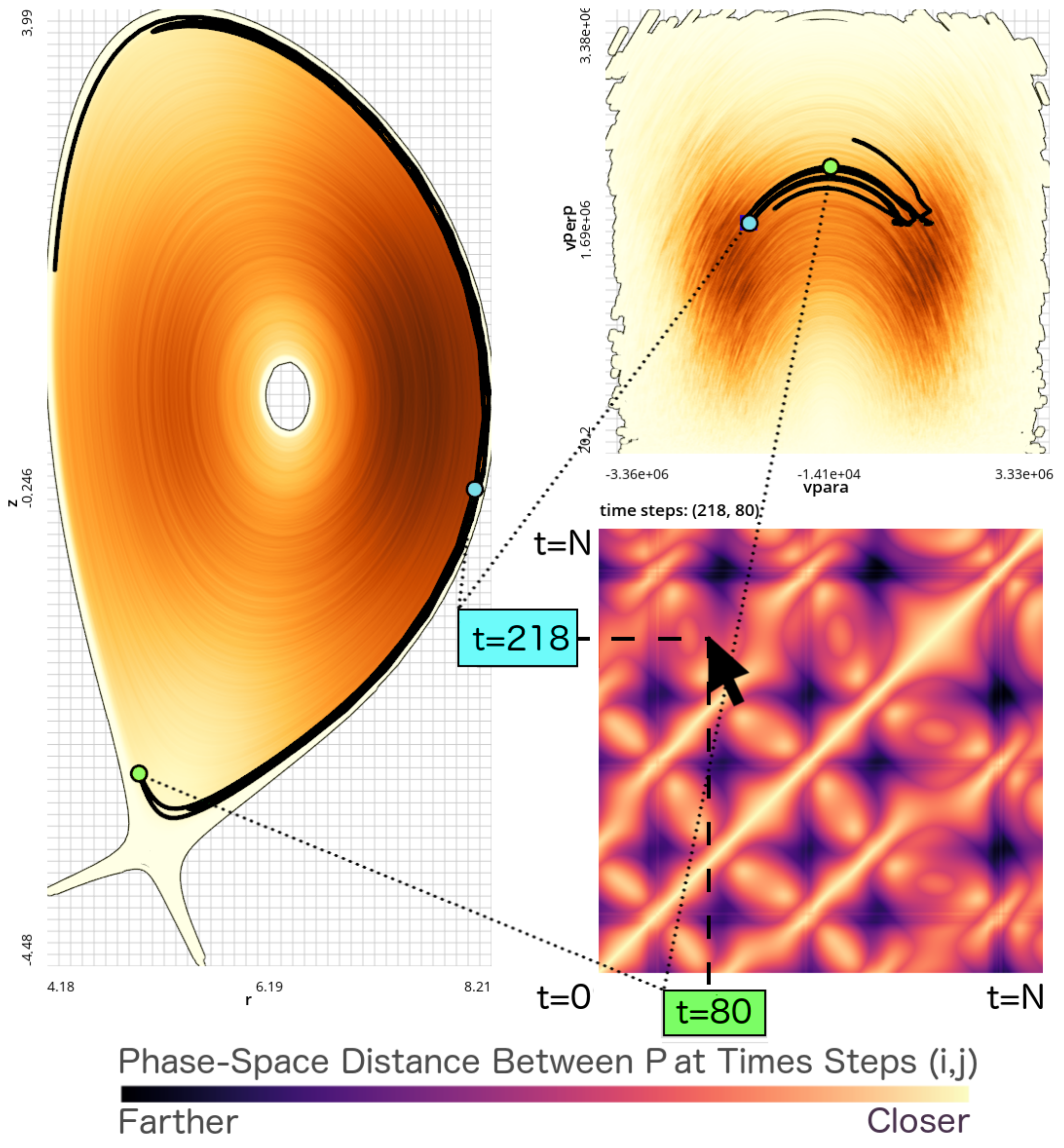


Figure 5.3. Phase plots in the poloidal plane (left) and velocity space (upper right) with a selected trajectory (black). Its distance plot is shown in the lower right. A pair of time steps that correspond to a recurrent state is selected with the mouse. The points/states at these two time steps are shown in pink and green dots along the trajectory in each of the phase plots.

5.2 Related Work

The more general problem of how to visualize multivariate temporal scientific data has been studied extensively. Some approaches include dimensionality reduction, linked and coordinate views, glyphs, and 2D or 3D plots or volume visualizations with additional variables mapped through color [114]. Other techniques, specifically for analyzing multidimensional trajectories or pathlines, include similarity analysis [115], clustering [116], and flow feature-based pathline attributes [75]. The problem with these methods that we address with this work, is the difficulty of exploring recurrent patterns in high dimensional spaces.

Distance plots and recurrence plots have been proven useful for these purposes [110]. Extensions of the recurrence plot include fuzzy recurrence plots [117], which use cluster centers as an alternative to discrete states, as well as cross recurrence plots [118] and joint recurrence plots [119], which can be used in limited circumstances to analyze separate trajectories together. Additionally, Poincaré plots and Poincaré maps are also used to study recurrence in dynamical systems. Sanderson et al. used them to study recurrent patterns in toroidal magnetic fields [92], and Tricoche et al. applied these methods within a more general study of topological features in area-preserving maps[93].

In terms of interactive visualization, and domain-specific extensions, a "conceptual recurrence plot" was proposed by Agus et al. for finding patterns in human discourse [111]. Additionally, Demiralp et al. applied recurrence plots within an interactive system for studying eye movements associated with visual-cognitive tasks [112].

However, existing recurrence plots do not support visualizing the combination of particle weights along with the phase-space trajectories, and interactive use to complement multiple coordinated phase-space views has not been explored. Previous work addressed visualization of tokamak particle distributions through weighted, spatially organized velocity histograms [99]. While useful for visualizing local changes in the particle distribution over time, they lacked the ability to analyze patterns in the particle trajectories.

5.3 Methods

5.3.1 Linking Distance Plots with Phase Plots

Since distance plots primarily show temporal patterns, and trajectory plots primarily show spatial patterns, we use an interactive linking between them to enable a more intuitive and

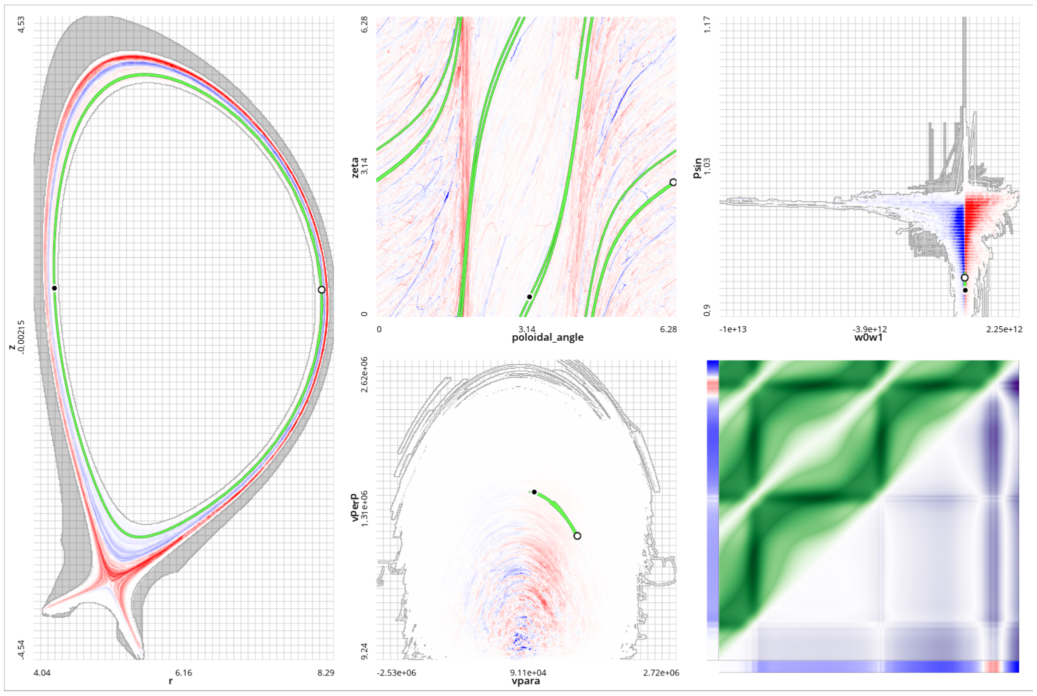


Figure 5.4. A group of linked phase plots together with our custom distance plot. The left plot represents the poloidal plane, the lower middle plot represents velocity space, the upper-middle plot represents the two angles of rotation (toroidal and poloidal), and the upper right plot represents the particle weight and magnetic radius. The background of each of these phase plots shows a heatmap-based aggregation of all of the weights of the particles, while the specific trajectory being singled out is shown in bright green. This trajectories distance plot is shown in the lower right corner. The user has selected a cell in the distance plot and the states corresponding to this pair of time steps are plotted in each of the phase plots (black point with white border and white point with black border). Unfortunately the screenshot application has not captured the mouse, so the selected cell is not marked.

detailed analysis of the phase-space trajectories. By hovering the mouse over cells within the distance plots, the associated pairs of time steps are selected and their corresponding points are then plotted in each of the phase plots, as in Figure 5.3.

One procedure that one could apply to analyze the recurrence patterns in this manner, is to first select an interesting time step (*e.g.*, based on an event, change, or anomaly) and then inspect the profile about the time step as defined by Schultz [120]. The profile about the time step corresponds to the window preceding and superseding the time step, and can be investigated by moving the mouse along the "anti-diagonals" (perpendicular to the main diagonal). Other uses are simply pointing out which actual states are recurring

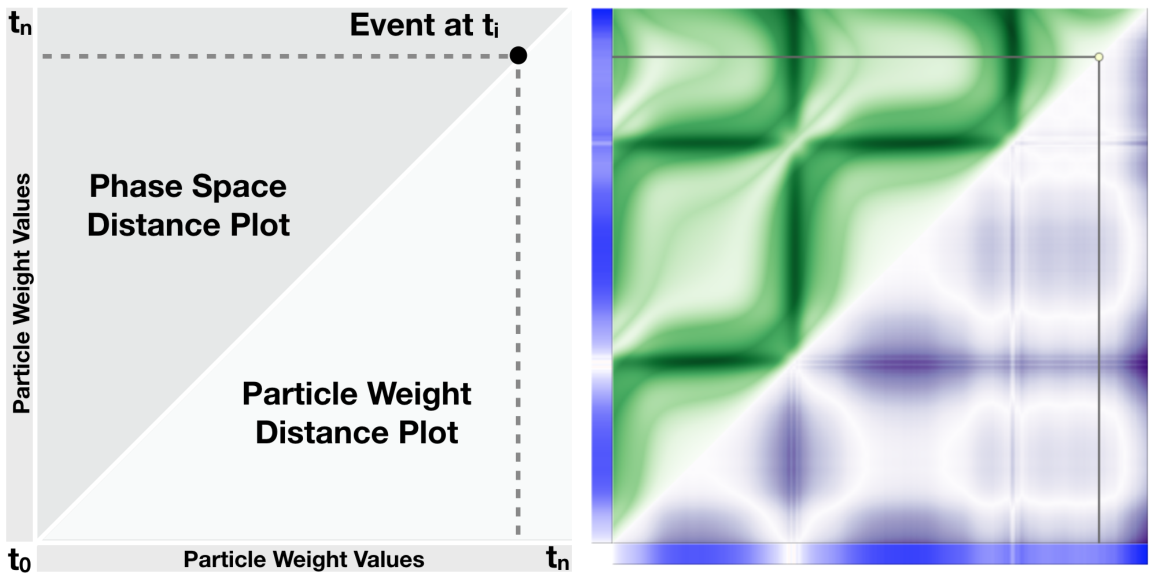


Figure 5.5. The design of our distance plot-based visualization incorporating events and particle weights. the upper-left portion of the matrix represents the phase-space distance plot, while the lower-left represents the particle weights distance plot. Different color maps are used for each to distinguish them. In each case, lighter means smaller distances. The red/blue bands along the axis show the values of the weights over time with red indicating positive weight and blue negative weight. The points along the diagonal represent the events.

(in the phase space) and demystifying new patterns found in the distance plots.

5.3.2 Incorporating Events

Scientists are commonly able to define meaningful events that may occur within their system of study. By incorporating these events into an interactive visualization, one can more easily investigate particular aspects of the data they are currently interested in. We incorporate events in our visualizations for both visual context, and selection. For visual context, points in time are recorded when the event of interest has occurred. The state of the system during these recorded events is marked as points and rendered over each of the linked phase-space visualizations, as well as along the diagonals of the distance plots (where $t = i = j$, corresponds to an event occurrence at time t). For selection, the user can directly select groups of events based on the state through interactions on the phase plots and then explore the associated trajectories and their correlations with the events.

The particular event we use in our analysis is the change of direction of rotation about the center of the poloidal plane. This is an indication of trapped particle modes. Passing particles are defined as particles that rotate fully about the center of the poloidal plane without turning around. Figure 5.2 illustrates these two types of trajectories. Research on this topic is ongoing and important for designing more stable tokamak devices.

5.3.3 Incorporating Particle Weights

Since our collaborating physicists wish to analyze the particle weights in conjunction with the trajectories, we split the otherwise symmetric phase-space distance plot into two halves, one for showing recurrent patterns in the phase-space trajectory, and the other for showing recurring patterns of the particle weights. Second, we plot the time series of the weights along the axis of the distance plot to give additional context for gaining insight into the patterns. Figure 5.5 illustrates the design of the resulting distance plot, as well as the placement of event-based markings.

Figure 5.4 shows a group of phase plots together with the custom distance plot. The phase plots show one selected trajectory (bright green) with a weight-based aggregation of all of the particles beneath it for context. Read the caption of the figure for more details.

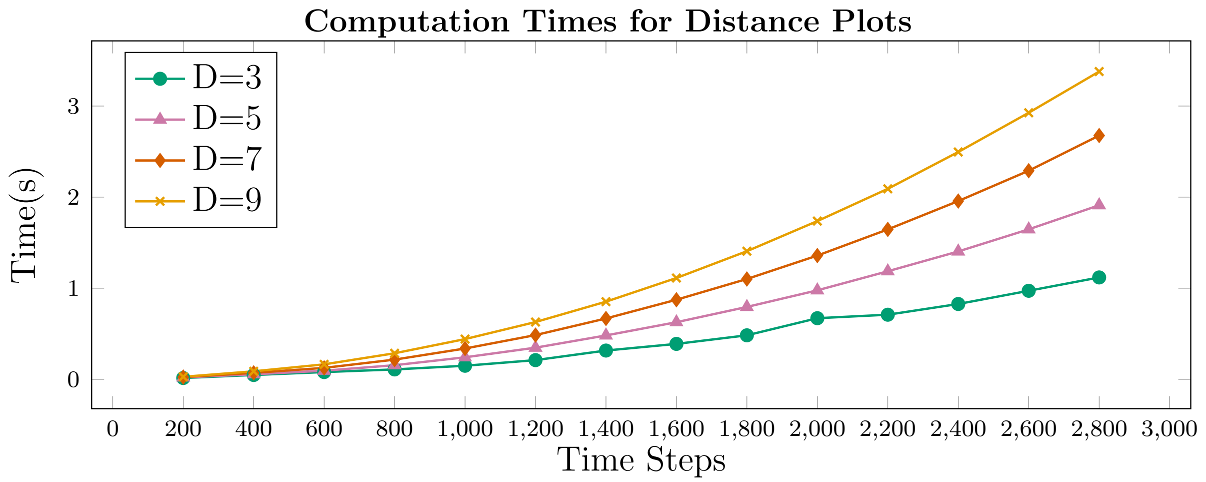


Figure 5.6. Computation times for a distance plot from trajectories of different dimensions and number of time steps.

5.4 Performance

The distance plot requires N^2 pairwise distance calculations on vectors in $\mathbb{R}^M$ where N is the number of time steps and M is the dimension of the trajectory. The computation is embarrassingly parallel and can be easily mapped to the GPU. Our implementation performs the calculation on the CPU with thread-level parallelism using OpenMP which we found sufficient for our application. We tested the performance on an Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz. Results for a range of time series lengths and dimensionalities are shown in Figure 5.6, and demonstrate sufficient efficiency for interactive visualization.

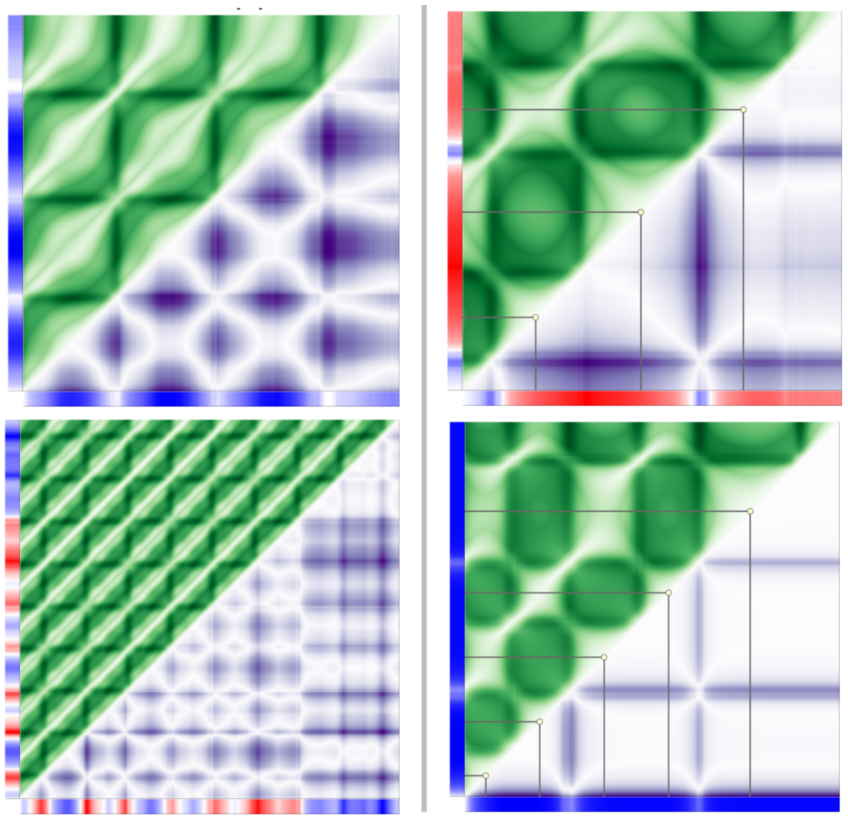


Figure 5.7. Four examples of the dual phase-space, particle weight distance plots. The left two plots are from passing particle trajectories, while the two on the right are from trapped particle trajectories and include marks representing the direction change events.

5.5 Results

Figure 5.7 shows some examples of the distance plots for different particles. An interesting pattern is an apparent correlation between the quasi-periodicity of each of these aspects

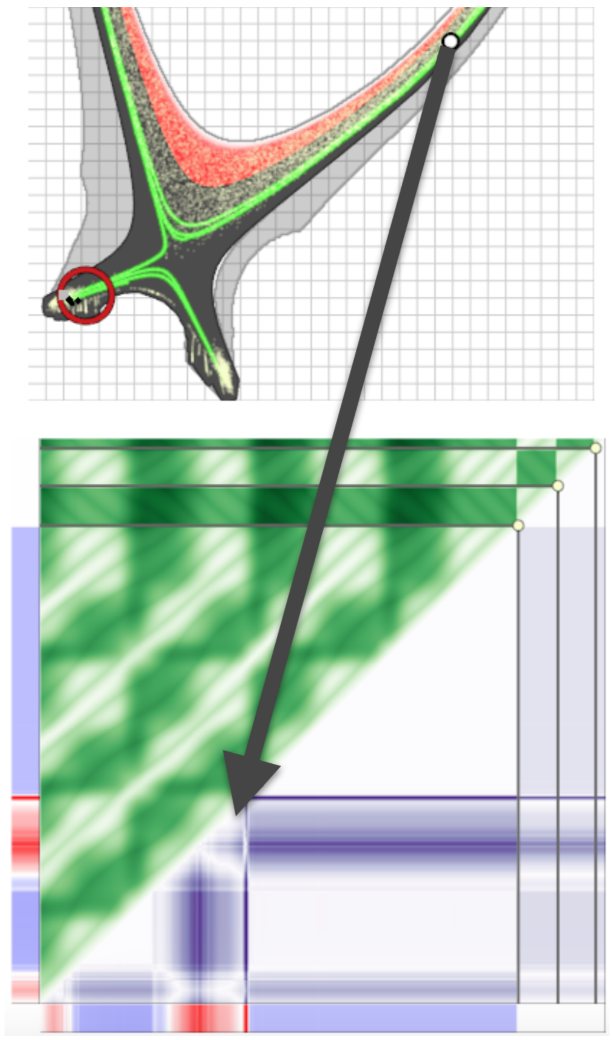


Figure 5.8. A particle on its transition from magnetic confinement to escape and absorption by the heat load diverter. The arrow shows that the point along the trajectory in the upper view (in the poloidal plane) corresponds to the time step where an anomalous change in the evolution of the particle's weight occurred. It appears this marks onset of the transition, before the subsequent loss of confinement.

of the particle. That is, it seems that the weights tend to oscillate with the orbits of the particles in phase space. While this seems to be a tendency, it is not a strict rule, and variations of interesting sub-patterns seam to occur which may loosely or tightly follows this trend. While the presented images only depict a few of many trajectories, the finding seems to be consistent with the larger set of trajectories as well (based on a visual inspection of a large sample). A more robust investigation in close collaboration with expert physicists will be done in the future to uncover the greater significance of

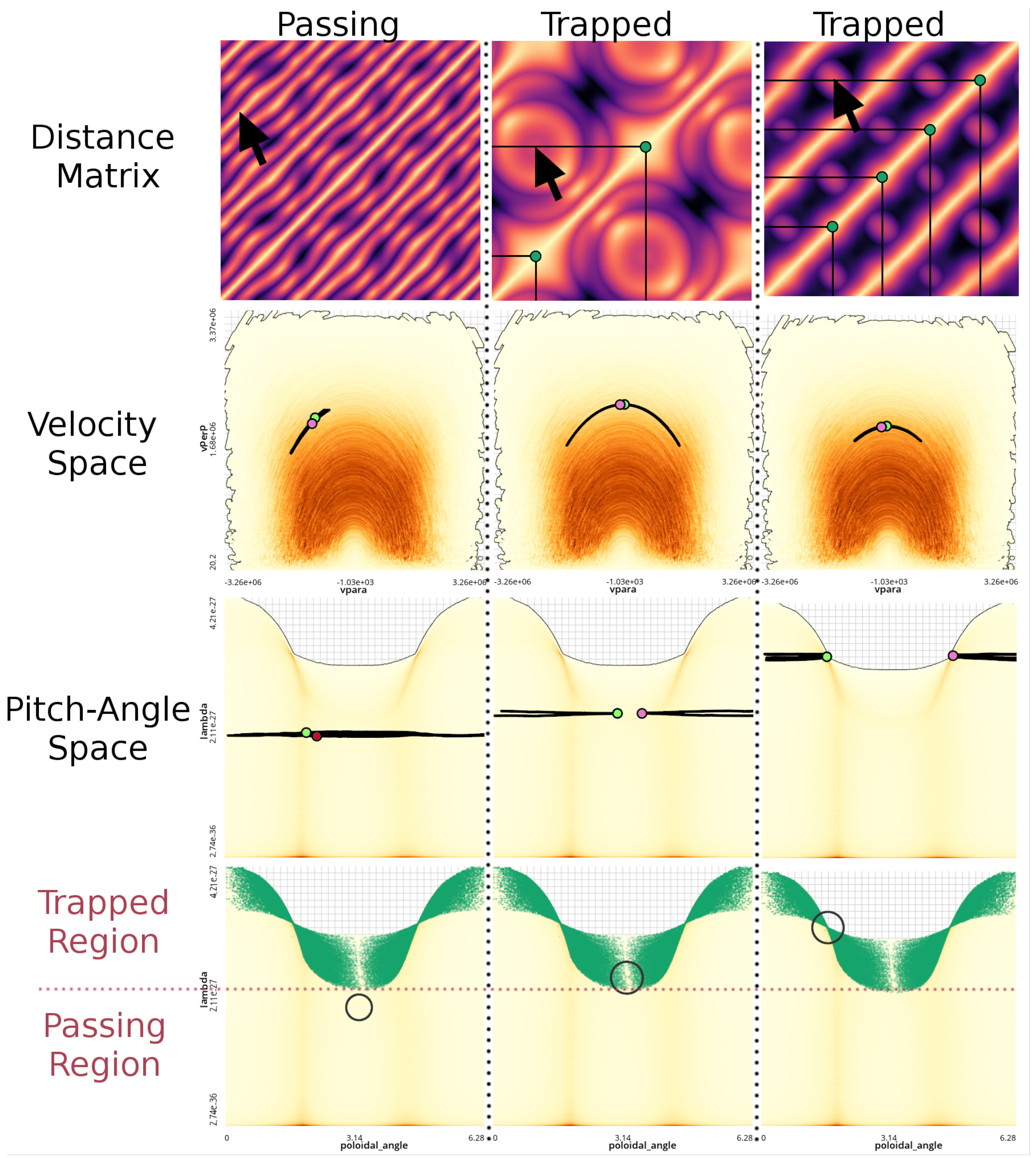


Figure 5.9. Visualizations of distance above the corresponding linked pitch angle plots, and velocity plots, for both trapped and passing particles. The green points in the phase plots represent direction changes. The green points in the distance plots represent the timestep of the direction change events for the selected trajectories.

these insights.

Figure 5.8 shows another visualization. In this use case, the transition from magnetic confinement to escape and subsequent absorption by the tokamak's heat load diverted is studied. The focus is on the relationships between weight fluctuation and the phase-space transitions from confinement to loss. This use case is interesting since particle flux near the outer portion of the toroidal device is strongly associated with particle loss, and a major driving force for weight evolution as the simulation progresses. Understanding this process can help to design better simulation techniques in addition to the physics. Our finding highlights instances in which sudden anomalous weight changes are observed before it is obvious (based on the trajectory) that the particle is escaping. In the figure, the anomaly in the distance plot is pointed to by the black arrow, as the dark vertical band (representing a narrow spike in the particle's weight). A linked phase plot of the poloidal plane is shown above the distance plot, and the point at which the anomaly occurred is pointed out as well. The direction change events plotted on the recurrence plot correspond to the point where the particle impacts the heat load diverter.

Figure 5.9 shows examples of trapped and passing particles in the recurrence plots, alongside linked views. This visualization helps to be able to see the multivariate trajectories of the trapped and passing particles without clutter. Mouse hover interactions on the recurrence plot support zeroing in on specific time points of interest in the linked phase plots.

5.6 Discussion

This work stands as a good starting point for a line of new research for advanced interactive visualization of complex many-particle dynamical systems such as PIC tokamak simulations with particle weights. Still, there are several challenges, limitations, and areas for further improvement and future work.

While applicable in many cases, distance plots are not always useful. For example, some systems have recurrence rates that are so far apart that analyzing them is impractical. Additionally, we apply distance plots as linked visualizations to deeper analyze phase-space trajectories, yet in many cases, they are used to study 1D observational time series data and the underlying phase space and associated trajectories are not available. In these cases, recurrence is studied by first "reconstructing" a phase space through time-delay embedding. This workflow requires special stages in which one must be careful to properly reconstruct the phase space, and attempt to extract dynamical features without direct knowledge of the actual phase space of the system or its trajectories. Marwan provided a guide to avoiding common pitfalls in recurrence quantification analysis [121]. Note that in our application, we do not attempt recurrence quantification analysis since it would be problematic in our setting, but instead utilize the distance plot (unthresholded recurrence plot) for qualitative analysis. While thresholding is primarily used to support the process of RQA, in the setting where RQA is not the goal, the distance plot is considered a valuable representation for qualitative analysis. Still, thresholding has value in this setting, since it allows to clarify which states are the most recurrent (have the closest pairwise distances).

While a single recurrence plot only shows a single trajectory, one could explore a large set of trajectories by using the patterns within the recurrence plot as a basis for trajectory comparison and clustering as well as feature extraction and search. One possible direction is to use deep learning, for example, neural networks/auto-encoders to learn latent features that are invariant to subtle, or uninteresting differences such as phase alignment, or small differences in frequency.

We leave it as future work to robustly investigate the potential for these ideas. One additional challenge is scalability if distance plots for each separate trajectory need to be computed and stored at once. The straight-forward implementation will be computationally expensive in both time and space when dealing with a very large number of trajectories. To store each of the plots alone would require $N \times T^2$, where N is the number of trajectories, and T is the number of time steps. This is too much space to use in the case of large-scale simulation data which can include thousands of time steps and hundreds of thousands to billions of particles. For this reason, the distance matrices will likely need to be computed on the fly and used as needed.

Chapter 6

Visualizing Multivariate Trajectory Distributions

This chapter is based on material from our publication, *An Interactive Visualization System for Large Sets of Phase Space Trajectories* [101]. © 2019 John Wiley and Sons.

Many scientific efforts focus on complex multi-component systems that are intensive to model and difficult to understand. Some examples include space or confined plasma, molecular and fluid dynamics, and astronomy. With modern computational resources, capabilities to simulate such systems are becoming more feasible. Still, assimilating the resulting large and complex multivariate trajectory data presents non-trivial visualization and analysis challenges due to the enormous amounts of information, and computational intensity to filter, transform, and visualize the data interactively.

Since the output from scientific simulations can be large and complicated, visualizing all of the details at once is not feasible. Visualizing overall trends through aggregation-based visualizations provides useful information, yet hides many details and variations within the data that may shed light on important physical phenomena. As an analyst sifts through the data from the high to low-level details, it is important that they don't lose context or miss the intermediate levels of detail that bridge the gaps and give confidence about the importance and representativeness of the lower-level details. It is well established that data categorization and Focus+Context techniques fit these problems well [114].

Subsetting the data is a pragmatic starting point for categorization, drilling down into details, and exploring conditional hypotheses. Furthermore, interactively combining subsets through set-wise/Boolean operations is a useful approach since it allows one to simplify the creation of new subsets, and organize them as meaningful selections that have concrete probabilistic/statistical interpretations. To support this approach interactively for large sets of trajectories, we design a novel GPU-based algorithm for computing/selecting the Boolean combinations as they are defined interactively. The method works by transforming the specified Boolean formulas into bitwise expressions, and then inserting them into dynamically generated and compiled GPU kernel code. This results in a highly efficient interactive logical filtering feature that can operate on multiple gigabytes of data in less than a second.

Based on this core approach, we design and implement a Focus+Context visualization system. Our system is motivated largely by two specific application domains, particle-in-cell tokamak plasma simulations, and particle accelerator simulations. However, within these constraints, we have designed our system generally for visualizing large sets of phase-space trajectories from complex dynamical systems. Central components of the Focus+Context visualizations are trajectory aggregations that provide overall statistical descriptions of the trajectories. To compute these aggregations, we use high-performance GPU rendering techniques. Unique to our trajectory aggregation approach is the option to use resident time weighting (from spatiotemporal sampling theory). Based on the goals of our system, we also incorporate interactive lensing operations. Our system is implemented using C++, GLSL, Cuda NVRTC, and OpenMP. The following are our main contributions:

- the design and implementation of a Focus+Context visualization system for the analysis of complex dynamical systems,
- an efficient GPU algorithm for computing Boolean combinations of data subsets that supports interactive user specification,

- and a GPU algorithm for computing trajectory density plots that incorporates time-residency weighting.

6.1 Design Goals

We begin by briefly describing several high-level design goals based on the theory and application of complex dynamical systems, and collaborations with domain scientists.

DG1. Categorical and Time Dependent Properties: Besides temporally static categorizations, our system also should be able to effectively visualize time-dependent behaviors (time-varying properties of points/states along trajectories that can change categorically over time), *e.g.* dynamical regimes, phase transitions, etc. In tokamak physics specifically, trapped and passing particle modes are an important area of analysis [122].

DG2. Interactive Subset Combination: Since these types of categorizations do not necessarily split the data into mutually exclusive sets, our system should interactively support subset operations to define Boolean combinations. Moreover, the available visualizations should not only show the overall qualitative behaviors/properties of these subsets/combinations, but they should also show statistically relevant properties that could be further quantified by the domain expert.

DG3. Events: Besides static and behavioral categorizations, we also wish to support an event-driven approach to exploring a system. Some examples include particle interactions and phase transitions. For accelerator physics, important events include particle emissions and impact/absorption. For tokamak particles, the direction change event (which is associated with particle trapping) is found to be useful.

DG4. Interactivity and High Performance: Systems, or ensembles of systems, often include many multivariate, time-varying trajectories. Our system should be efficient enough that the analyst can efficiently explore large amounts of information to tease out knowledge from the data. In particular, we aim to support in depth conditional analysis of the multivariate and time-dependent behaviors of the particles. Besides both high-performance manual/mouse-based selections and rendering, an important goal of our system is to support run-time/interactive specification/language support for applying the filtering conditions needed to select Boolean combinations of time-dependent subsets.

DG5. Visual Occlusion and Clutter Management: Our system needs to handle the analysis of events in combination with heatmaps of data subsets and supersets, as well as additional manual trajectory selections. These elements play important roles for our particle analysis system, yet together they bring a great deal of clutter and occlusion to the visualizations. This clutter and occlusion must be well managed to gain the benefits of using the different elements together within multi-faceted, Focus+Context visualizations.

DG6. Flexible Layout Supporting High Aspect Ratio Plots: Beforehand, we do not know how many plots the analyst will need to link/use in a single session, nor do we know what their aspect ratios should be, or how large they should each be relative to one another. The plot layout must be easily configurable. Furthermore, of particular importance is supporting the exploration of large aspect ratio spaces, since this is a requirement for particle accelerator analysis.

DG7. Statistical and Physical Trajectory Analysis: Gyrokinetic particle-in-cell codes (that are commonly used to model plasma/fusion simulations), often use marker particles with joint statistical and physical interpretations. While the trajectories may follow physical laws, they also represent weighted statistical samples/contributions of the overall phase-space particle distribution. The particle's weights are time-varying and can also be negative, representing a decrease in phase-space density near its phase-space location. Analyzing the joint statistical and physical behaviors requires plots that aggregate the trajectories into meaningful statistics, together with an analysis of the spatial motion of the trajectories through time.

Besides cases where the particles have a joint physical and statistical interpretation, they may also carry momentum, and be involved in complex interactions such as collision, which limits the use of many traditional flow visualization techniques that are based on mass-less tracer particles. The techniques we employ by default should be usable and have clear interpretations, no matter what are the physical properties and interactions of the particles within the dynamical system.

6.2 Methods

According to the aforementioned design goals, we have devised methods to address all the essential issues. Section 6.2.1 covers the technique used for interactive specification and computation of Boolean combinations of time-dependent subsets. This technique is used to help support **DG1**, **DG2** and **DG4**. Section 6.2.2 covers the plots that are used as context to help support joint statistical and spatial analyses in support of **DG7**. High-performance algorithms are also given in that section to help support **DG4**. Section 6.2.3 covers the methods used to help support an event-based analysis approach in support of **DG7**. Section 6.2.4 describes the interactive lenses we use to help manage clutter and occlusion in support of **DG5**. Section 6.2.5 covers the manual/direct selection tools, and how one investigates spatiotemporal details of individual trajectories on top of the statistical/aggregate plots. This helps support both **DG1** and **DG7**. Section 6.2.6 describes the overall system, as well as the layout scheme for supporting high aspect ratio spaces, and flexible layouts/configurations of the multiple linked phase plot views (supporting **DG6**).

6.2.1 Subset Management and Boolean Combination

As described in **DG1**, subsets of interest are commonly time-variant. Thus we design our approach accordingly and allow subset memberships to include sub-paths along the trajectories. With each subset representing a condition, one has a collection of modular selection components that can be combined to form more complex selections for conditional analysis. This approach also lends well to statistical quantification of the visualization results.

To compactly store subset memberships for each point along the trajectory, we encode them using bit-flags. Each base subset corresponds to a bit-index and is assigned a unique symbol representing it. Analysts specify Boolean combinations of the subsets interactively using set operations. As they define them, the combinations go into lists from which they can be selected. The specified Boolean combination is then converted into an expression composed of bitwise operations, inserted into GPU code, and compiled dynamically at runtime. The resulting GPU programs can then test the membership of the Boolean combinations by directly evaluating the expressions. This enables highly efficient modification or creation of a Boolean combination without necessitating the movement of data and allows the resulting Focus+Context visualizations to be rapidly updated throughout the analyst's exploration process, as parameters are changed and details toggled. This is the core of our support for **DG4**.

Without loss of generality, we describe our implementation in terms of time-varying particle data. We denote the phase-space vector for particle i at time-step t as $\mathbf{x}_t^i$. We denote X to be the full set of loaded particles, such that $X(i,t) = \mathbf{x}_t^i$. Subsets will be denoted $S_k \subseteq X$, where k is the subset id, as $\{\mathbf{x}_t^i \text{ s.t. } f_{S_k}(\mathbf{x}_t^i) = 1\}$ where $f_{S_k} : X \longrightarrow \{1, 0\}$ maps phase-space vectors to membership in S_k.

Given subsets, $S_1, S_2, ..., S_m$, an m-bit bitmask b_t^i is computed for each $\mathbf{x}_t^i$, such that $b_t^i[k] = 1$ if $\mathbf{x}_t^i \in S_k$ else 0. With this, we test membership of $\mathbf{x}_t^i$ in S_k with the Boolean expression, b_t^i `&` 2^k `!=` 0, where `&` is the bitwise AND operator, or equivalently (b_t^i `&` 2^k) in languages where 0 evaluates to false and not 0 evaluates to true. Note that the number 2^k, in binary, has a 1 in digit k (indexed from 0), and has 0's in all other digits.

An analyst specified Boolean combination, $C \subseteq X$, *e.g.* $(\overline{S_1} \cup S_2) \setminus (S_3 \cap S_4)$, is defined by an associated Boolean combination function f_C mapping each particle $\mathbf{x}_t^i \in X$ to membership in C. We compute these functions on the GPU to support efficient exploratory analysis of the data subsets. To accomplish this, we generate GPU code through text replacement of a placeholder symbol, @, by the formulated Boolean expression operating on b_t^i to compute f_C, and then compile the GPU program during runtime.

An example conversion of the Boolean combination into a computable expression, is given in Equation 6.1. First, the set difference operations in Expression 6.1a are converted, using the property $S_a \setminus S_b \equiv S_a \cap \overline{S_b}$, to produce Expression 6.1b. Then we replace the remaining set operations with their corresponding logical Boolean operators AND, OR, and NOT respectively, which are denoted `&&`, $\|$, and `!`, to produce Expression 6.1c. Lastly, we derive Expression 6.1d by replacing the symbols for each S_k with the bitwise expression for computing membership, $(b\&2^k)$, where b is a variable input into the program representing b_t^i. Finally we have an expression that we insert into our GPU source code to compute f_C. While not shown in this example, symmetric difference($\triangle$) is converted using the property, $S_a \triangle S_b \equiv (S_a \cup S_b) \setminus (S_a \cap S_b)$, and then an application of the set difference rule.

$$(\overline{S_1} \cup S_2) \cap (S_3 \setminus S_4) \tag{6.1a}$$

$$(\overline{S_1} \cup S_2) \cap (S_3 \cap \overline{S_4}) \tag{6.1b}$$

$$(!S_1 \parallel S_2)\&\&(S_3\&\&!S_4) \tag{6.1c}$$

$$(!(b\&2^1)\|(b\&2^2))\&\&((b\&2^3)\&\&!(b\&2^4)) \tag{6.1d}$$

Algorithm 6.1: The procedures underlying our subset and Boolean combination selection process. In the GenProgram procedure, l denotes left operand and r denotes right operand. @ is a symbol within the template program that is replaced with the formula for testing membership in the Boolean combination.

```
Define InSubset(bitmask b, BitIndex k):
    return $(b\&2^k) \neq 0$// check if bit-k is 1
Define SetCombFlag(bitmaskRef b, BitIndex k):
    $temp \leftarrow b\ \&\sim (2^k)$ // set bit-k to 0
    $b \leftarrow temp \mid (int(@) \times 2^k)$ // set bit-k to 1 if @ is true
Define GenProgram(Program p, Expression e):
    e.replace( $l \cap r$, $(l\&\&r)$ )
    e.replace( $l \cup r$, $(l\|r)$ )
    e.replace( $l \setminus r$, $(l\&\&!r)$ )
    e.replace( $l \triangle r$, $((l\|r)\&\&!(l\&\&r))$ )
    e.replace( $\overline{r}$, $!r$ )
    for $S_k \in e$
        e.replace( $S_k$, $bool(b\&2^k)$ )

    p.replace(@, e)
    return p
```

Note that both & (bitwise AND), and && (logical AND) are used in these expressions and are not to be confused. Based on the generated expression, the GPU program computes memberships for each data point to choose which aggregations should include them. See Algorithm 6.1.

6.2.1.1 Alternative Implementations

For computing Boolean combinations, one alternative method is the use of symbolic programming and abstract syntax trees. In this approach, a symbol table can be made to map each subset symbol in the expression to one or more memory references (at least one per parallel thread of execution). The references are replaced by the memory of each data point, one at a time, as the expression is iteratively evaluated. If the function that evaluates the Boolean expression is pre-compiled, then it must be a function to compute a general Boolean expression. This requires an algorithmic process to parse the expression as a string, or another representative form, each time it is evaluated. Optimization can be made to make maximize the efficiency of this process. However, if the expression is known at compile-time, it can be converted to a single line of code (as we have demonstrated with our method). The limitation is that one needs to use an environment/language in which they can compile newly generated code at runtime. We use Cuda/NVRTC to achieve this. GPU processing has obvious performance benefits for this process compared with CPU code since the computation is embarrassingly parallel. An added benefit of using the GPU is that the data already needs to be processed through the GPU for graphics rendering. Thus, the data can reside on the GPU and can be shared between the rendering and Boolean filtering codes.

One limitation with our Boolean filtering implementation is that it depends solely on Cuda, which is only able to use NVIDIA GPU hardware. Additionally, we also have not integrated a method to achieve runtime compilation of CPU code as a backup. Since OpenCL also supports runtime compilation, is cross-platform, and can be compiled to both CPU and GPU code, we may integrate an OpenCL back-end in the future to increase the portability of our system. We expect that our algorithm would be efficient on a CPU as well, and it may be preferable to use the CPU instead of the GPU on some systems if the GPU memory is too small compared to the CPU memory.

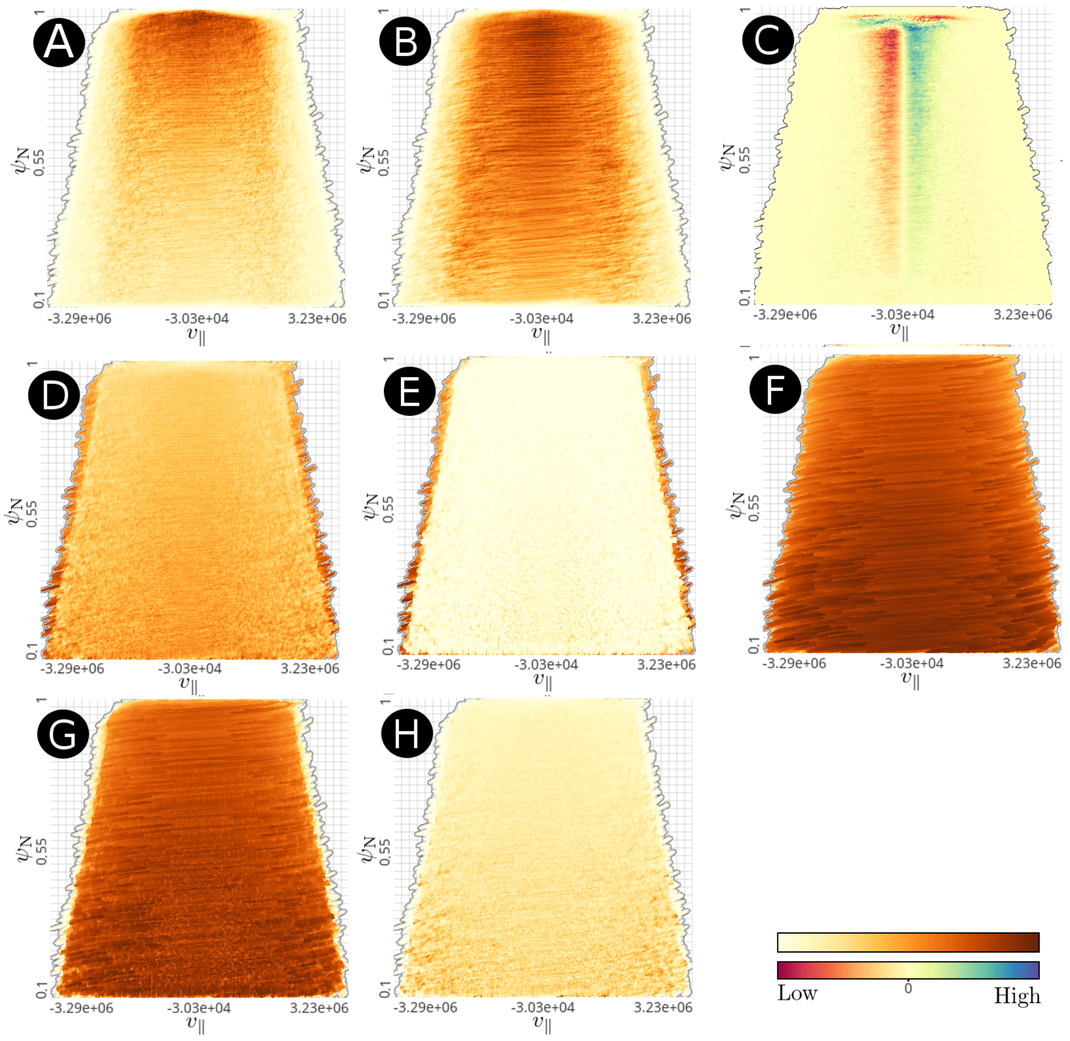


Figure 6.1. Examples of the aggregated phase plots described in Section 6.2.2. These plots represent particle statistics from a tokamak dataset. The axes are $v_{||}$ and ψ_{N}. The dataset and variables are described in Section 6.4.1. A) shows time-averaged particle density. B) shows path/curve density. C) shows weighted particle density (using the simulation weights from XGC). D, E, F, G, and H) show mean, min, max, and range and variance of $v_{\perp}$ respectively. The color maps are shown at the bottom of the figure. For plot C, a divergent color map that is centered at 0 is used since the particle weights can be negative. The other plots are colored based on the uppermost color legend at the bottom of the figure. In both cases, darker means high magnitude.

6.2.2 Aggregate Phase-Space Plots

We use 2D aggregate trajectory plots as the primary contextual elements within our Focus+Context design. Their purposes are to provide insight into overall/statistical trends and direct further selections/refinements while drilling down into the details. While many different ways of aggregating phase-space trajectories are possible, we begin with several aggregation methods that are chosen for their straightforward mathematical meaning, ability to be applied generally, and efficient computability. These include plots that bin statistical measures such as mean, variance, range, and min/max. We also use a path density plot (in the spirit of curve density estimations [123]), and a time-averaged particle density plot.

$$\rho_s(T, i, j) = \frac{1}{N_t} \sum_{t=1}^{N_t} N_p(c_{i,j}, t)/A(c_{i,j}), \tag{6.2}$$

Equation 6.2 shows how the density plot is computed. $c_{i,j}$ represents the 2D spatial cells in the heatmap defined over the subspace space s, T is a discrete time window (series of time steps), $A(c_{i,j})$ gives the area of each of the cells (respective of the particular spatial dimensions of the bins), N_t is the number of time steps, and $N_p(c_{i,j}, t)$ is the particle count within the cell at time t. Figure 6.1 shows examples of several variants of the heatmap plots used.

Hardware rastering is leveraged to efficiently compute the aggregations. This is done by setting the OpenGL blend function to `glBlendFunc(GL_ONE, GL_ONE)`, and then rendering graphics primitives to a single-channel 32-bit floating-point frame buffer. We can set the blend equation to `GL_FUNC_ADD` for summations, or use `GL_MIN` or `GL_MAX`. Note that hardware rastering and additive blending were also used by Scheepens et al. [124]. Since we are aggregating trajectories in the form of discretely sampled time-series data, we wish to interpolate between the points to reconstruct the missing data points and achieve smoother, more continuous results. To accomplish this, we could apply a constant rate over-sampling by rendering a fixed number of discrete points (`GL_POINTS`) in-between the original points. While this results in each particle being sampled at the same frequency, it will still leave discontinuities between the points. Instead, we draw lines between the points (`GL_LINE_STRIP`), to guarantee connected trajectories. However, this means that particles with larger displacements will be sampled into more bins/texels over the same period of time, which will lead to incorrect time-integrated densities and statistics. To solve this issue we use resident time weighting, a technique applied to spatiotemporal sampling

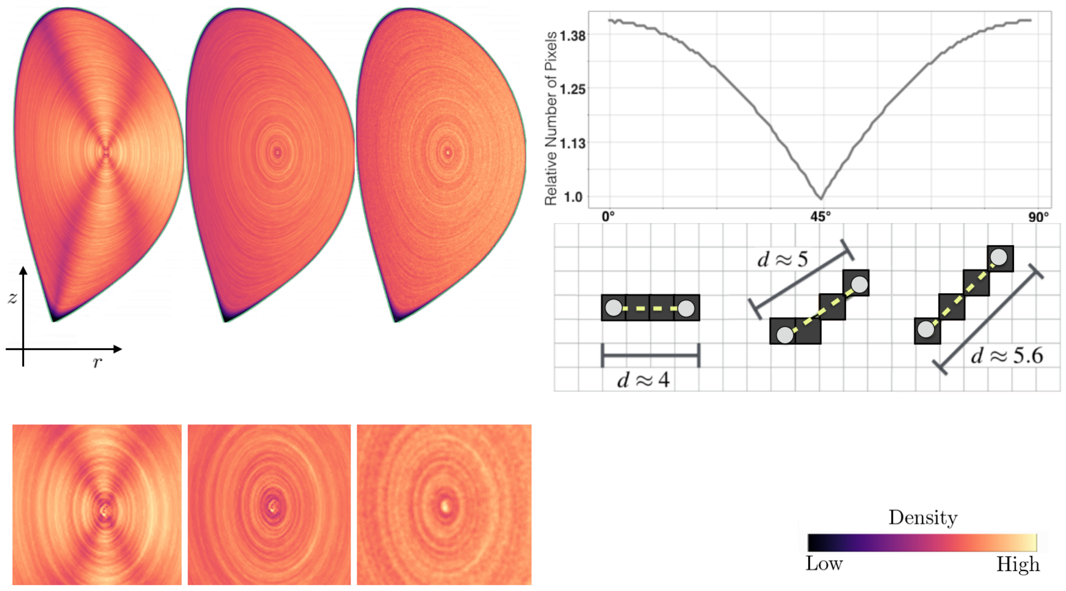


Figure 6.2. When rendering lines into a heat map using the graphics pipeline, perceptually significant inaccuracies can occur in the result due to varying numbers of pixels used by lines of different orientations. Top right) A table showing the relative number of pixels by angle. This was computed by rendering many lines at different angles and then counting the non-zero pixels. Top left) A depiction of the problem: each of the three lines is drawn with the same number of pixels even though they have different distances. Bottom left) The rendering artifact: when rendering the time-averaged particle density heat map using lines, without accounting for angle, the regions where the particle paths are nearer to 45° are under-sampled (producing the x-shaped darker region). Bottom left, middle) lines are used and the artifact is corrected. Bottom left, right) Points are rendered without interpolation.

in dynamical systems and particle flows when using physical measuring devices/probes [125] ch. 5.2.3. Each particle has a fixed net contribution between time steps, and the contribution is divided between each of the bins that the particle had visited.

Another issue in utilizing line rendering and the graphics pipeline for aggregation is that there are inconsistencies in the number of pixels used to raster lines of the same length depending on their angles. This can give rise to significant artifacts/errors in certain cases, as demonstrated in Figure 6.2. To solve the problem, we pre-compute a lookup texture of normalization factors, based on the average relative number of pixels used to raster lines by angle.

The implementation uses GLSL shaders, including a geometry shader. The lines are rendered as a `GL_LINE_STRIP`. In the geometry shader, the two points corresponding to each line can be accessed for computing the angle in image-space and the displacement in texels. The angle is then used to look up the pre-computed normalization factors in an attached texture. The geometry shader then applies them and outputs normalized values to be aggregated through the fragment shader. Since we render separate trajectories in one graphics rendering call (`glDrawArrays`), we also need to break the lines between them. Since we already are using bit-flags to store subset memberships, we dedicate one of the bits to mark the endpoints, so that we can omit the lines between different trajectories within the geometry shader. Another issue that we address is rendering trajectories in spaces where an axis wraps around. For example, in tokamak research, plots, where one axis represents an angle, are commonly used. We also handle this in the geometry shader by splitting lines crossing the wrap-around point into two separate lines.

While we compute the trajectory plots using OpenGL, we also compute statistics for each of the variables using Cuda. Since these computations all read the same data, we use Cuda OpenGL inter-operation to share GPU memory resources. The OpenGL vertex buffers are registered and mapped to Cuda device pointers.

6.2.3 Events

To support **DG3**, we incorporate events into the system. Events serve as visual elements (plotted as points), but also as selection targets for operations. The interface includes a list of these events, from which the analyst can activate, or deactivate each of them. The analyst can also engage in an event-based selection mode, which changes the trajectory probe tool to capture only trajectories that undergo any of the active events within the selected range.

6.2.4 Interactive Lenses

Since our phase plot visualizations involve multiple layers and visual elements, the resulting clutter and occlusion must be managed. While interactive subset combination, selection, and toggling of the overlaid visual elements help to address these issues (**DG5**) we take a step further with the use of interactive lenses.

First, since the aggregations favor stronger trends and leave less prominent trends

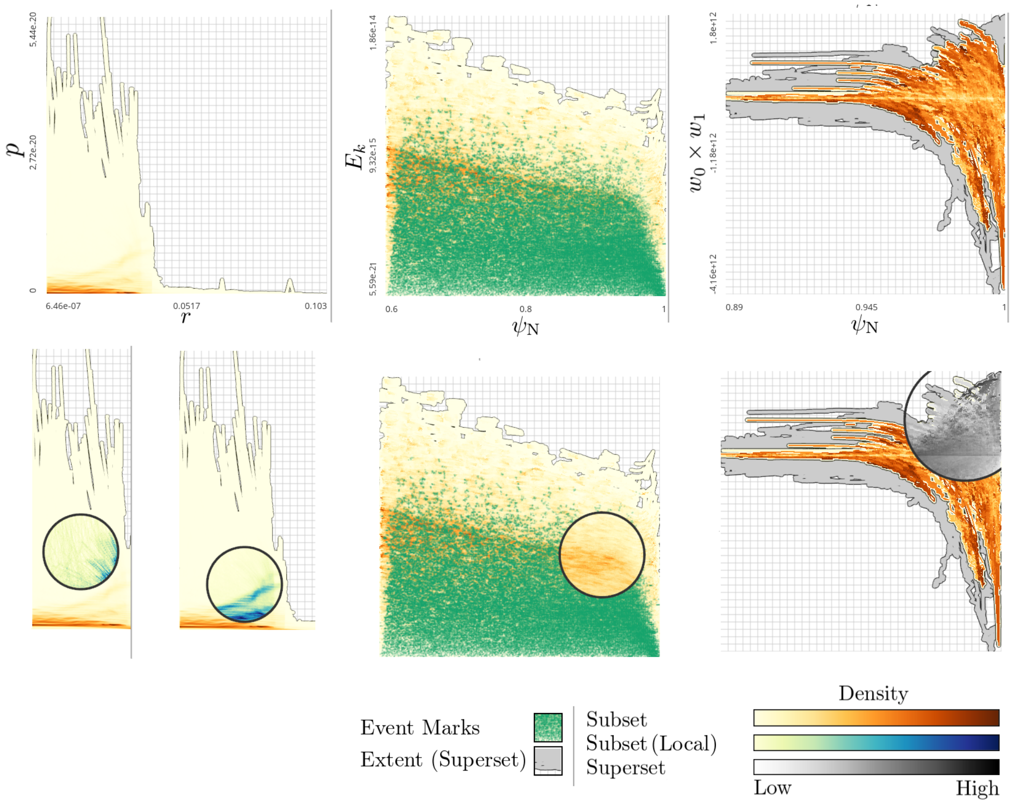


Figure 6.3. Examples of interactive lenses used in our design. Left column) an aggregate phase plot showing momentum (p) vs. radius (r) from a particle accelerator simulation. The dataset and variables are described in Section 6.4.2. Faint features in the plot on the left are brought out by a color-map localization lens (middle and right). Middle column) a kinetic energy (E_k) vs. ψ_{N} plot from a tokamak simulation. Direction change events are plotted over it (green points). A "de-occlude" lens is used (right) to look through the plotted points. Right column) mean $v_{\perp}$ in ($w0 \times w1$) vs. ψ_{N} space from a tokamak fusion simulation (Section 6.4.1). A subset representing particles with lower energy is shown in the forefront, while the extent of the full data is in the background. An interactive lens (below) brings the aggregation of the full data to the forefront for comparison.

difficult to see, we use an interactive lens that localizes the color mapping underneath by extending the color range to between the min and max within the encircled region. Second, since the base layer in the backdrop is partially occluded by the aggregate heat map layer on top, we only show it with flat color. Since this base layer represents a superset of other layers, the extent can be fully observed despite occlusion. However, the heat map-based aggregation of this layer would be problematic to show due to occlusion. Thus, we provide an interactive lens that brings the encircled portion of the background layer to the forefront to be shown as a heat map. This allows one to quickly compare trends and features expressed in the forefront against the full set of data underneath in an interactive fashion, with the perceptual convenience that both are in exact alignment. Third, we over-plot lower-level details, such as trajectories and points on top of the other layers. This allows one to see the context underlying the details, but also causes occlusion. We thus use a "de-occlude" lens which hides the over-plotted details under the encircled region, allowing one to examine the higher-level features beneath. Examples using these lenses are shown in Figure 6.3.

6.2.5 Subsampling and Selection of Individual Trajectories

In some cases, *e.g.* many-particle systems or ensembles, the overall behavior of large groups of trajectories is the main object of study. However, even in such cases, scientists may want to analyze individual trajectories since; They may have direct physical meanings, and based on the behavior of one trajectory under certain conditions, similar behaviors could be hypothesized for others. As another example, fusion scientists have requested us to visualize individual particles with very large simulation weights. These particles exert disproportionately more influence than others on the overall system.

Therefore, at the lowest levels of our Focus+Context visualization, one can explore small samples of discrete trajectories. These trajectories are sampled by probing the 2D spaces using the aggregate plots for context. The selected trajectories are plotted on a top layer of the visualization, while their ids populate a list. Refinements or expansions of the manual selections can be made through intersections or unions of additional selections. Individual trajectories can be selected from the list to be highlighted in the phase-space views and studied with the recurrence plot. Finally, these selections can be saved as named subsets, used in Boolean combinations, and visualized with the different aggregated plots.

These features help support the mid-level to low-level exploratory processes that help bridge the gap between the high-level overall trends and the behaviors of individual trajectories. With this support, one can break down large groups iteratively into smaller and smaller subgroups, which can simultaneously be managed within the interface, and later brought back into the visualization. These features offer core support for the interactivity demanded by **DG4**.

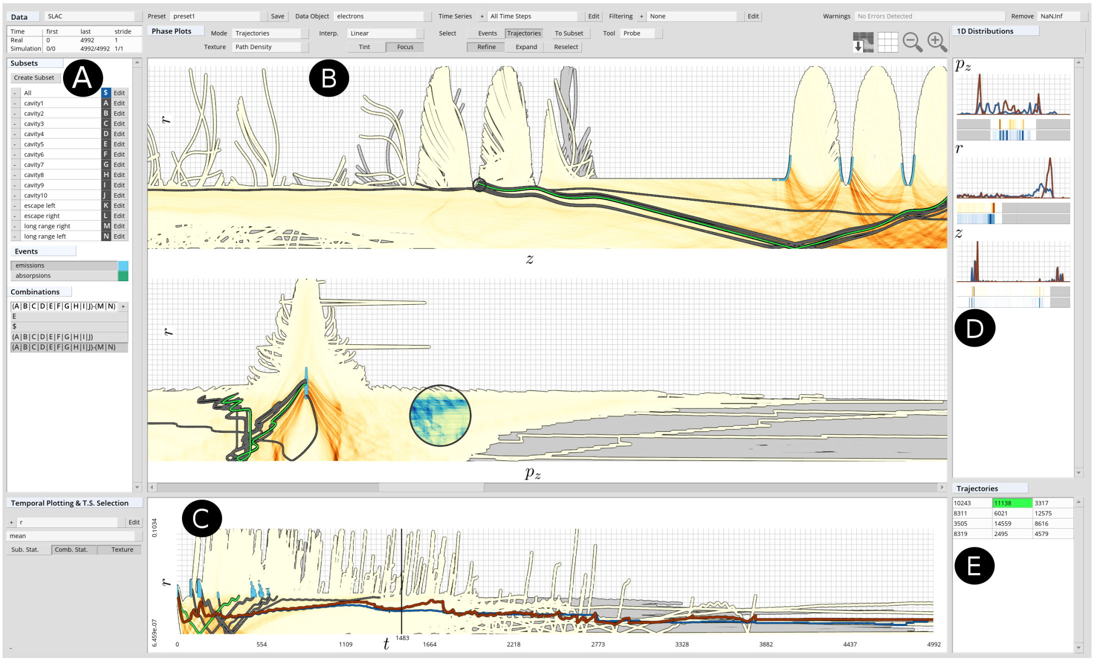
Additionally, we incorporate an interactive distance plot that is linked with the other visualizations. Distance plots, commonly referred to as unthresholded recurrence plots, are able to give a perspective into the dynamics of high-dimensional systems in a 2D view[110]. They can be useful for analyzing recurrent patterns in phase-space trajectories, as well as evolutionary phenomena such as transitions between dynamical regimes. The distance plot is defined by a symmetric matrix where the columns and rows represent time-step indexed states, $\mathbf{x_t}$, of a phase-space trajectory, and the cells (i, j) represent the pairwise distances $||\mathbf{x}_i - \mathbf{x}_j||$.

We incorporate them into the system as a linked view that compliments the phase plots highlighting the states (as points) that correspond to the mouse hovered distance plot cells. This helps one to uncover the meanings of features and transitions observed within the distance plots, as well as to give more temporal insight into the particle's motion along its trajectory.

6.2.6 Graphical Interface

The interface includes 4 main views that are always present: A) a panel on the left for managing subsets, subset combinations, and events, B) a view on the bottom that shows a selected time series plot and allows for time step selections, C) a panel on the right that shows 1D distributions of each of the active variables, and D) a view in which the linked phase plots and distance plots are visualized and interacted with. See Figure 6.4.

To support **DG6**, we design an adaptable and customizable grid-based plot layout. Each plot in the view is defined with an aspect ratio setting. There are two main options, 1) the analyst can choose how many rows and columns the plot should use, or 2) the analyst can choose only how many rows the plot should use and let the system decide the number of columns based on the value ranges of the data. This way, the analyst can define very wide plots, such as required by accelerator data, make some plots oversized relative to others, and also keep or modify the aspect ratios defined by the ranges of the axis. The system then packs the plots together into the layout. Since they may not all fit on the screen at once, the view is scrollable in the horizontal direction. Besides manually



10243	11138	3317
8311	6021	12575
3505	14559	8616
8319	2495	4579

Time	first	last	stride
Real	0	4992	1
Simulation	0/0	4992/4992	1/1

Figure 6.4. An interface for our Focus+Context visualization scheme. A) subsets, events, and Boolean combinations are managed. B) a layout for multiple linked phase-space visualizations. C) a timeline view. D) 1D density plots for each active variable. E) groups of trajectories populate a list of IDs from which to select individual trajectories. The trajectories of the listed particles are plotted in dark gray, while the selected trajectory is plotted in bright green. While not shown in this view, the distance plots are optionally computed from the selected trajectory and added into the phase-space plot layout.

scrolling, a button is used to shift the plots over by one column and snap the scroll level so that the leftmost plots line up with the view at their edges. To enlarge the plots in the view, the analyst can press buttons that either increase or decrease the number of rows in the layout, with the lower limit being the maximum number of rows required by any one plot. The time-series view at the bottom can also be resized, allowing the main plot view to take up more or less space relative to it. This allows the analyst to easily blow up the size of the time series plot as needed, and then pull it back down to enlarge the sizes of the phase plots.

This layout scheme works well to provide the customization required for different needs and is easy to interact with, satisfying **DG6**. It is an especially useful layout for plots with extreme aspect ratios such as required for accelerator data.

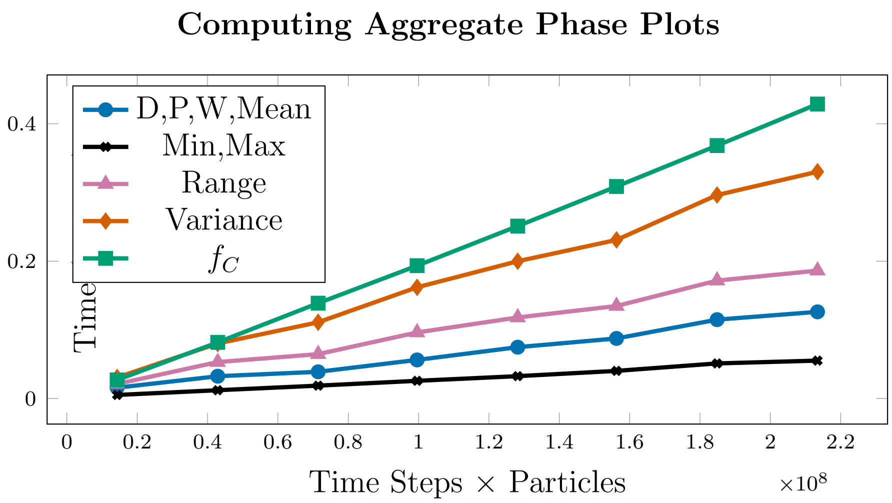


Figure 6.5. Computation times for the aggregate plots. On the horizontal axis is number of data elements, (time-steps × particles). D refers to number density, P refers to path density, and W refers to weighted density. Since each of these had very similar computation times, we averaged them into one curve. Also, the min and max times were averaged. The time to compute a Boolean combination, $f_C = (A \setminus ((B \cap C \cap D) \triangle E)) \cup F$ is shown as well.

6.3 Performance

We tested the performance of computing an example Boolean combination function, $f_C = (A \setminus ((B \cap C \cap D) \triangle E)) \cup F$, as well as each type of aggregate phase-plot. The former (f_C) is based on the Nvidia Runtime Compilation (NVRTC) Cuda API, with `-O4` optimization flags, and the latter (aggregate plots) use GLSL. The graphics card was a GeForce GTX TITAN X, and the CPU was an Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz. We measured the elapsed time over the number of data elements processed ($N_p \times N_t$), where N_p is the number of particles, and N_t is the number of time steps. To increase the number of data elements and push the limitations of our implementations, we progressively duplicated the particle trajectories. The results are shown in Figure 6.5. The computation times each follow a linear trend. We were able to maintain speeds of less than 0.5 seconds with over $213,416,346$ data points for each of them.

The aggregate plots were faster to compute than f_C, most likely because they are implemented in GLSL and utilize special graphics hardware within the GPU's rendering pipeline. Computing the variance plot takes the longest of the aggregate plots since it requires two summation passes. Likewise, the range plot requires two min/max passes.

Overall, a primary limiting factor to interactivity seems to be GPU memory limitations. This limitation depends on the number of variables that are actively used within the set of linked phase plots. Assuming the variables are stored as 32-bit floats, and that 32-bit subset bitmasks are used (supporting 30 subsets flags), then the memory use (for data storage) in bytes is $N_p \times N_t \times (N_v + 1) \times 4$, where N_p is the number of particles, N_t is the number of time steps, and N_v is the number of active variables. This memory cost limits how many data points and/or variables can be visualized interactively at one time using the GPU methods. For reference, the performance plot in Figure 6.5 show the performance up to $N_p \times N_t = 213,416,346$. The textures used to render the plots use additional memory. There are up to two textures used per plot, one for the superset and one for the subset. Each uses $w \times h \times 4$ bytes. Suppose you are using a $4k$ monitor, then you could require up to $4,000^2 \times 4 \times 2 = 128,000,000$ bytes of memory, which is of minor significance compared to the data. Our GPU has a max of 12 GB, but in practice 10 GB is a safer upper limit. Thus with $213,416,346$ data points, we could support $\lfloor 10,000,000,000/853,665,384 - 1 \rfloor = 10$ variables at one time. In Section 6.5, we discuss extending the methods to a multi-node system for increased scalability.

We also tested the performance of computing the distance plots. These plots require about $D \times \frac{N \times (N-1)}{2}$ embarrassingly parallel primitive operations, where N is the number of timesteps, and D is the dimension of the trajectory. Our implementation uses a simple OpenMP parallel loop for CPU parallelism. For $(N = 1000, D = 5)$ the plot took 0.241659 seconds to compute, and for $(N = 2000, D = 9)$ the plot took 1.73711 seconds to compute. GPU parallelism would likely increase the performance if needed.

6.4 Applications

Besides general concepts, our system was also motivated by collaborations with two groups of simulation scientists. The first is with scientists at Princeton Plasma Physics Laboratory (PPPL). Multiple discussions and email exchanges have helped guide our design, and inform our first motivating use case. The second is with scientists at Stanford Linear Accelerator National Accelerator Laboratory (SLAC), who informed our second use case.

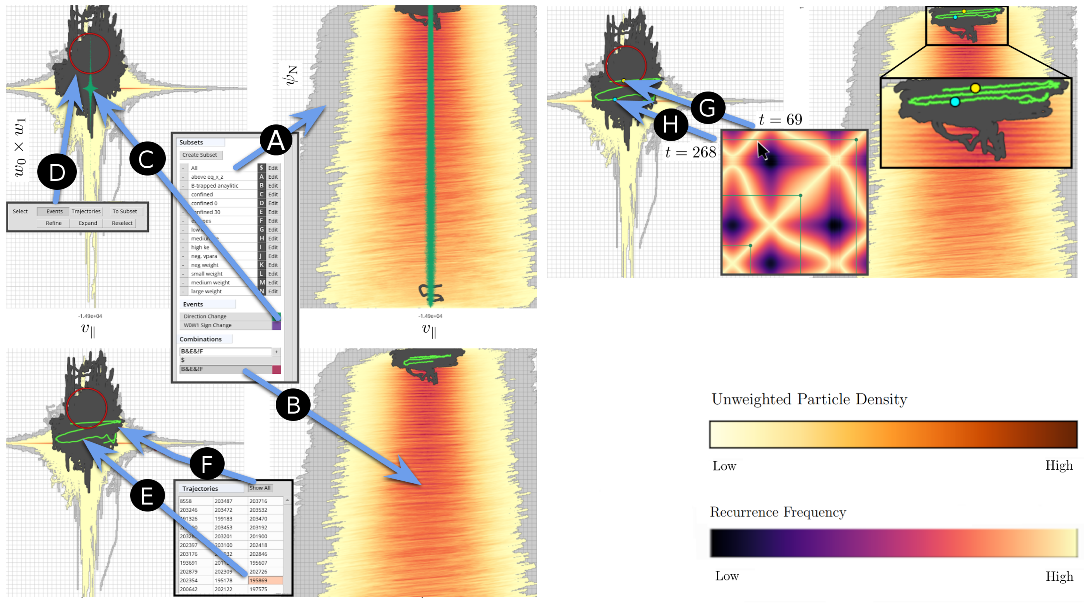
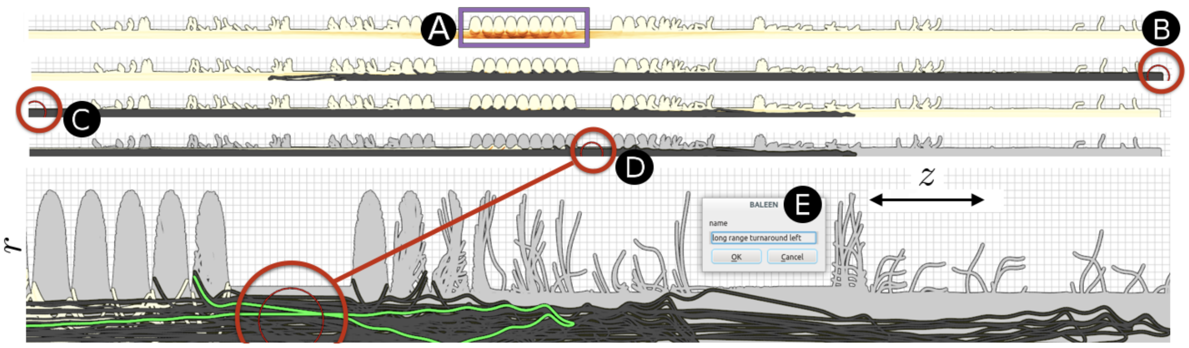


Figure 6.6. Annotated screenshot frames from a visualization session with the XGC dataset. The plots in the left column are in $v_{\|}$ vs $w0 \times w1$ space, while the plots in the right column are in $v_{\|}$ vs ψ_{N} space. The blue arrows are used to point out action applied through the interface and the associated visual elements. The figure is explained in detail in Section 6.4.1

6.4.1 XGC Fusion Simulation

The XGC data is described in Appendix A.1.1. One of the challenges it must address to enable computational feasibility, is to avoid the need to simulate the extremely large number of real particles within the device. It accomplishes this, in part, by using marker particles which each carry time-varying weights that correspond to a number of real particles. One of the accompanying challenges is that the weights can evolve away from an idea distribution [126], and undergo unwanted growth. One particular concern is the development and effects of particles with very large weights. A second concern [122] is the physics and different contributions of two kinds of trajectories, trapped and passing (Figure 5.2). Trapped particles lack the energy to leave the high field side of the poloidal plane. Instead of circling the poloidal center, they turn around and form banana-shaped trajectories. A third interest is the study of particles near the separatrix of the magnetic field. Particles that pass across this magnetic flux surface will escape and be captured by the diverter.



long range turnaround left

Figure 6.7. Trajectories of electrons (from the accelerator dataset) in r-z space (Section 6.4.2). A) the cavity in which the particles are emitted is enclosed in the purple rectangle. B) manually selecting the 'long range right' trajectories (those which reach the end of the full assembly on the right side). The selection radius is the small inner red circle, and the resulting selection of trajectories is plotted in dark gray. The bold outer red circles have been added to the figure to help make it more clear where selections were made. C) selecting the 'long range left' trajectories in the same manner. Both the 'long range right' and 'long range left' trajectories are exported to saved subsets so that they can be plotted as heatmaps and used in other Boolean combinations. D) after changing the focused Boolean combination to the 'long range left' subset, a manual selection, from these trajectories, was made, just past the right side of the cavity that the particles were emitted from. This selects the particles which escape the initial cavity in the opposite direction before turning around and becoming 'long range left' trajectories. E) a zoomed-in view before exporting this selection to a saved subset. An individual trajectory is highlighted in green.

We utilized the following 7 subsets and their combinations: ions, electrons, trapped and passing particles (time-dependent subsets), particles inside the separatrix beginning from timestep 30 (suggested to us by a collaborating physicist), and particles that do and do not escape the separatrix. The two primary spaces of the analysis include $v_{\|}$ (parallel velocity with respect to the magnetic field) vs $w0 \times w1$ (particle weight), and $v_{\|}$ vs ψ_{N}-normal (a radial measure that identifies the flux surfaces outward from the poloidal center). From these combinations and spaces, we could explore trends and patterns interactively under different combinations of conditions, for example, trapped particles, which were initially confined, which are near the separatrix, do not escape the separatrix, and that achieve very large weights.

Figure 6.6 shows a series of screenshot frames from the visualization session. (A) shows the UI symbol for the full particle set listed along with the other subsets. The extent of the superset is shown in the light gray background. (B)shows the focused

Boolean combination that is selected in the UI, and the heat map of the focused subset. The selected subset is $B \cap E \cap \overline{F}$, where B represents particles that should be trapped according to an analytic formula (this is a time-varying subset), E represents particles that are confined before time step 30 (this time step marks the offset of a global phenomenon of interest to the collaborating scientists), and F represents particles which do not escape the separatrix. The Boolean combination is rendered using the curve density plot, with a light-yellow to red color map. (C) shows the UI has the "direction change" events (sign change of $v_{\|}$) engaged. These events are plotted as green points. (D) shows that the event-based selection is engaged. In this mode, the selection tool targets the particles (from within the focused subset) that experience the event within the selection radius (red circle). The trajectories of these particles are plotted in dark gray. (E) shows an individual particle from within the selection. It is selected from a list in the UI and is rendered over the top of the views in bright green. (F) shows the "show all" option is engaged. Otherwise, only the green trajectory would be shown over the top of the heatmap, rather than the full set of selected particles (dark gray). (G, H) show the analyst has used the mouse hover action over the unthresholded recurrence plot to make a selection of a pair of timesteps to identify along the selected (bright green) trajectory. The two time points are rendered as the blue and yellow points in each of the plots. An enlarged view of these points is shown in the upper-right plot. Within this visualization session, an observation is made that trapped particles that achieve larger weights seem to generally have less range in $v_{\|}$ and a large volatile range in $w0 \times w1$. While looking through the individual particles, and examining their patterns, the distance matrix is used interactively to highlight the positions of the particles at specific time steps, and the temporal patterns underlying their quasi-periodic dynamics.

6.4.2 ACE3P Accelerator Simulation Data

The ACE3P data is described in Appendix A.2.1. In this case, we focus on long-range trajectories of field emitted electrons. Our subsetting and analysis approach is based on a discussion with the scientists as well as an associated paper [127].

One of the main plots we use is an r-z space plot, where $r = \sqrt{x^2 + y^2}$, and z is the

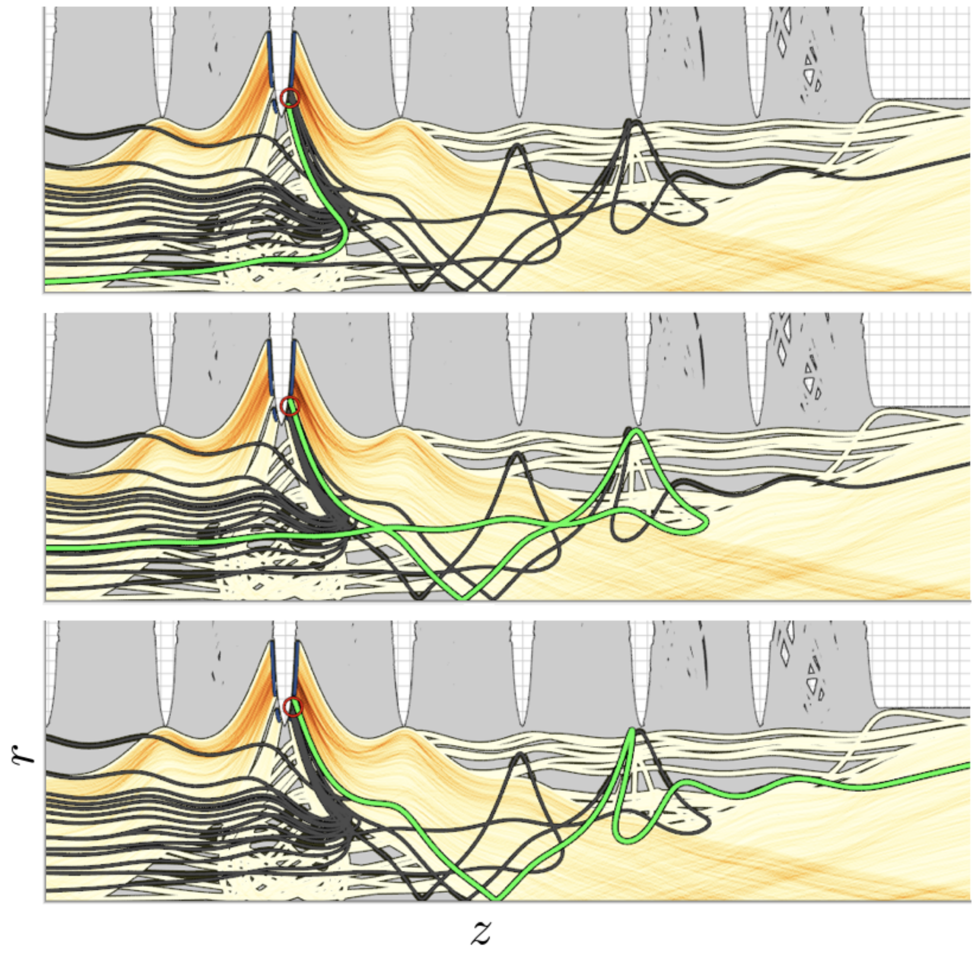


Figure 6.8. Probing groups of trajectories (dark gray) of long-range particles emitted from the 5th iris in a cryomodule cavity, and then exploring different individual trajectories around conditions from which outcomes diverge (green). The region targeted by the selection tool is within the small red circle. Each of the green trajectories in the 3 plots was emitted from nearby locations in the same iris. Particles emitted from this region appear to diverge based on small differences in their initial location. The trajectory highlighted in the top plot diverges, turning left. The trajectory highlighted in the middle plot stays right at first, but is then diverted near one of iris 8, and turns around. The trajectory in the bottom plot follows almost the same course as the one in the middle plot, except right after turning around near iris 8, it again changes direction and then stays right for the rest of the simulation.

direction along the length of the accelerator. This is a very wide aspect ratio plot that is handled well by our adaptable plot layout. As a starting point, based on the scientists' recommendations, we partition the trajectories into 10 disjoint subsets based on which iris the particles are emitted from. See Figure A.4 for a depiction of the device and its structural components. Next, we manually select trajectories from each of the ends of the module and export them as subsets. From here, we found that many full-range trajectories initially travel quite far in the opposite direction before turning around. We select those particles and export them to subsets as well. Next, we select particles that escape the cavity they are emitted from (but not necessarily the full module) and export them as subsets. These operations are shown and described in detail in Figure 6.7.

Afterward, we switch to the event-based selection mode and use the timeline plot to select based on the times of the emission events. We then export different associated groups of particles to subsets such as '1st emitted', '2nd emitted', etc. Lastly, we make selections of the highest momentum particles from a momentum space plot and export them to a subset.

Finally, we are left with a total of 20 subsets to investigate. We are able to explore combinations of them and probe for more details interactively. For example, (NOT ('cavity escape left' OR 'cavity escape right')), gives the particles which do not leave their initial cavities, and (('iris 1' OR 'iris 10') AND ('long range left' OR 'long range right')) gives trajectories, from the outer irises of the cavity, that last through the full simulation. These are only a few examples of many combinations worth investigating. Figure 6.8 shows some frames of analysis based on emission location and highlights trajectories near conditions that lead to diverging outcomes. After identifying such sensitivities to initial conditions, further exploration can be done to help understand the causal factors such as emission timing. These exploration tools help to dissect the behaviors of the particles and better understand the overall system.

6.5 Discussion

Our performance results demonstrate that the efficiency of our Boolean combination algorithm is good enough to leave memory limitations as the main bottleneck. This limitation was analyzed in Section 6.3. To increase scalability, our algorithms can be extended to work with multi-node systems. This would be straightforward since the required processing can be done independently over chunks of the different trajectories before reducing each partial result, for example using `MPI_Reduce`.

Our system supports both predefined/initial subsets, interactive creation of new subsets through direct selection over the 2D visualizations, and Boolean subset combination. In principle, the system can be used without the need for any predefined subsets, by using the direct selection tools, export to subset feature, and Boolean combination. However, more sophisticated ways to generate new subsets interactively could make the system more powerful. In many cases, subsets of interest are expected to require domain-specific algorithms (mathematical/statistical) to extract. This will likely necessitate a built-in scripting language (*e.g.* Python), custom plugin support, or similarly flexible mechanisms to enable specifying and extracting the more complex time-dependent behaviors of interest to scientists. Integrating Fast-Bit into our system is a promising direction toward supporting these future goals.

In our implementation, we use 32-bit integers as bit-masks for subset membership flags. This limits the number of subsets that can be used in the visualization session. In our case studies, we didn't find the need to use more subsets than this, however, it is a simple change to use 64-bit masks, or even larger, in order to expand the number of supported subsets.

One possible concern with our time-varying trajectory subset approach is the issue of continuity of the aggregated trajectories. Since the subsets are defined on a sub-path level, there could be gaps between them. Our position is that the possible discontinuities are critical to allow since it is assumed that this is the analyst's intention based on the time-dependent categories they have defined and selected. The system should be mathematically precise to the analyst's assumed intention. Note that besides the aggregate plots, full trajectories are shown when selected as in Section 6.2.5.

While our system handles events represented as discrete points in time and space, more complex events may represent multistage transitions or activities [128] as defined by Ozer et al. A direction for further research is how one can effectively generalize and incorporate such events into a system like ours. Additionally, there are many possibilities for other interactive lenses within this environment. For example, statistical and uncertainty-based lenses might help to bring more insight and confidence into the trends visible in the aggregate phase-space plots.

Chapter 7

Level Set Restricted Centroidal Voronoi Decomposition

This chapter is based on material from our publication, *Level Set Restricted Voronoi Tessellation for Large scale Spatial Statistical Analysis* [129]. © 2022 IEEE.

Simulation scientists continue to push computational limits for the modeling of complex spatial systems. These simulations can generate extreme-scale data with high spatial and temporal fidelity, but I/O bottlenecks and storage constraints place limitations on

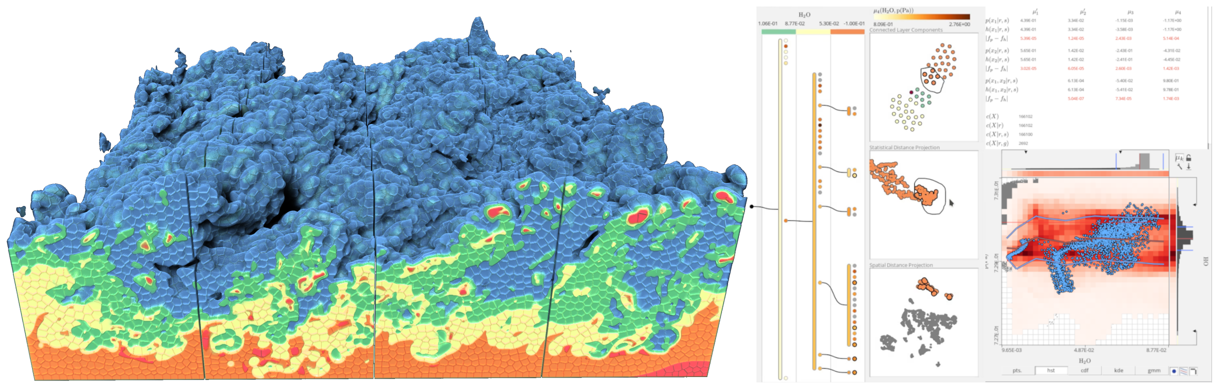


Figure 7.1. Left to right illustrates steps of our methodology. We decompose volume data by novel geodesically based discrete centroidal Voronoi tessellations restricted by sequences of level sets. Then we aggregate statics within the resulting hierarchically structured segments. Finally, the data is reorganized according the hierarchical structure, and used for interactive spatial statistical analysis of large data at a reasonable cost.

how much of this data can be saved and analyzed post hoc. This problem motivates the development of in situ data analysis and processing workflows with the goal of reducing I/O while retaining crucial information.

One way to approach this problem is spatial statistical aggregation. Such approaches usually involve segmentation or decomposition in the spatial domain, and then statistical aggregation within the resulting segments. Several methods have been developed for spatial segmentation of statistical aggregation, such as blockwise decomposition, Voronoi decomposition, and voxel clustering. These methods have demonstrated value; however, it remains a challenge to organize such a tessellation around dynamic flow features, typically defined by surfaces.

To address this problem, we segment the spatial domain dynamically using level set restricted centroidal Voronoi tessellation (CVT). The CVT may also be weighted to vary the spatial fidelity based on a spatial saliency function. Since the decomposition is aligned to the level sets, this also provides a hierarchical segmentation, from isbands (voxels between pairs of isosurfaces or contours) to connected components within isobands, to the centroidal tessellations within the connected components. The statistical aggregations may also be merged over Voronoi cells and throughout the hierarchy. For example, one may view statistics of the Voronoi cells individually, or locally merge Voronoi cells to obtain aggregations over multi-segment volumetric patches that are aligned to the surfaces of interest, or merge all Voronoi cells within a connected component or entire isoband.

For practical purposes, the implementation must be scalable and efficient. Efficient algorithms have been developed for computing CVTs; however, the focus of these methods has been on different application areas, such as re-meshing and construction of computation grids. These other applications have requirements that Voronoi must conform to, and are typically defined through precise geometry. For statistical aggregation, however, the requirements may be relaxed, since heterogeneity in region size but conflict with the goal of generating a conforming mesh in complex geometries. Meanwhile, the benefit of the discrete tessellation is that it can be done efficiently in parallel on GPUs and it fits the analysis workflow whereby voxel elements are sub-selected discretely for visualization.

Since our requirements differ from the typical use case, and because we are aiming for high performance, we implemented our own discrete method for a non-conforming level set restricted tessellation that favors volumetric and geodesic regularity.

Additionally, interacting with the data in this form is difficult without a supporting visualization system. As scientists put great preparation into planning an expensive, large-scale simulation run, it is important to also adequately prepare the configuration of the in situ workflow so that it can be relied on without concern. For example, with our approach, one must choose level sets, overall Voronoi site density, and the optional saliency function for concentrating site density. For statistical aggregation, there are several possible models for distribution functions, each with different possible parameters that should be tuned appropriately. Among models and their parameters, there are also tradeoffs affecting performance, storage costs, fidelity, and other measures of quality. Thus a working visualization system is a valuable tool to have before one attempts to apply the in situ methods. Our visualization system is able to help with this, since it can be used on existing data where the methods can be tested and the aggregates can be compared with the raw data to get an idea what information may be lost or misrepresented by the aggregates. Figure 7.1 shows the system interface.

Our primary contributions in this chapter are:

- a parallel level set restricted CVT algorithm and implementation,
- a supporting visualization system, and
- an overall approach to visualizing large scale multivariate spatial statistical information.

We apply our work to simulation data of turbulent combustion and wall-bounded turbulence, and we evaluate the performance and scalability on the Summit supercomputer at the Oak Ridge National Laboratory.

7.0.1 Motivation, Research Challenge and Novelty

Restrictions and Homogeneity. The state of the art methods work by building a conforming crust around the boundaries. But in building the crust so that it is conforming, we must concentrate more sites around sharp features, and thus we will sacrifice homogeneity of the Voronoi regions on the boundaries. But the regions on the boundaries are those of primary interest for spatial statistical analysis. This is why we relax conditions required for surface reconstruction, and instead focus on homogeneity of the region sizes. However, this creates challenges for a restricted CVT algorithm since individual regions will wrap around the flow features, and so the normal centroid is no longer valid since it may not even be within the region, and the centroidal updates could carry a Voronoi site through a boundary/restriction. Therefor, we create a novel restricted CVT algorithm based on graph geodesic distances, where the graph is constructed within the volume for each voxel to have a shortest path around the restriction boundaries to the graph geodesically nearest Voronoi sites. Besides CVTs, homogeneity guided decompositions for spatial statistics have been approach using simple linear iterative clustering (SLIC) [45], however, to our knowledge, an algorithm for restricted SLIC is not available, and since the spatial component of SLIC is based on Euclidean distance, a restricted version also could not reliably meet complex boundary restrictions while maintaining spatial homogeneity of region sizes.

Scalability and Computational Performance. Since the domain is evolving, and one application of the decomposition is for in situ processing, it is important that our algorithm is efficient, and exploits the heterogeneous parallelism of modern super computers, including distributed data parallelism, and GPGPU parallelism. Comparisons between our algorithm and others are difficult since to our knowledge, no other algorithm computes the same form of CVT as ours. However, algorithms have been proposed for standard CVTs on 2D planes and surfaces which exploit GPU parallelism [130], and in 3D parallel clipped CVTs [131]. A general data distributed parallelism for CVTs was intruduced by Starinshak et al. [132], which also handles restrictions only by clipping. However, clipping will leave randomly fragmented regions along the boundaries depending on the data.

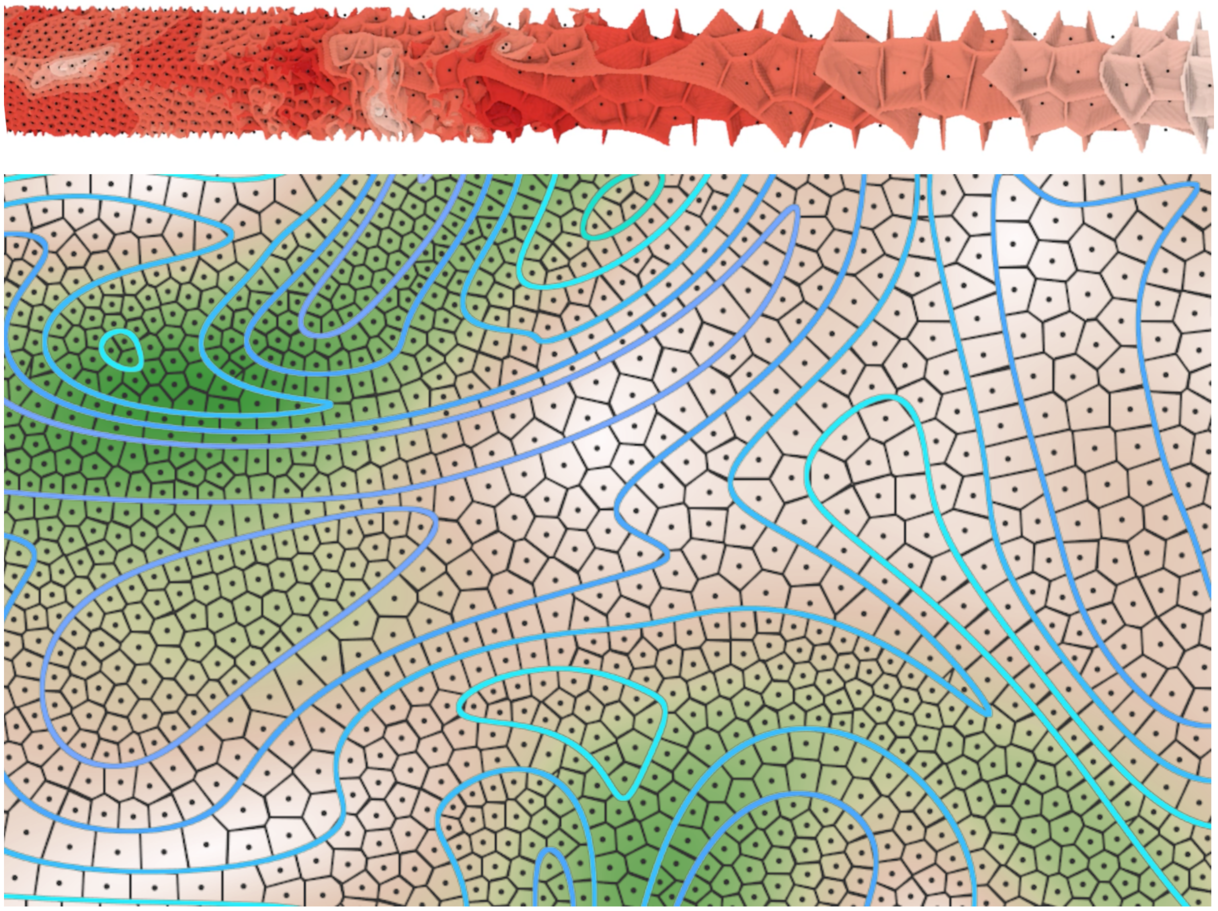


Figure 7.2. A 2D case of the weighted centroidal Voronoi tessellation, where contours of one scalar field (shades of blue) impose the restrictions, and the second field (brown to green) controls the site density.

7.1 Methods

First, we introduce a local statistical aggregation approach based on a discrete geodesically based level set restricted CVTs. It is geodesically based in a graph theoretical sense, with voxels as nodes and the geodesics being the shortest restricted paths from each voxel to its nearest Voronoi site. We then present a prototype system that utilizes the method for hierarchical spatial statistical analysis.

7.1.1 Level Set Restricted Voronoi Tessellation

Level Sets, Isobands, and Connected Components. We define a set of disjoint discrete isobands where each isobands $B_{a,b}$ is the set of voxels which have values between scalar values a, b, *i.e.*, $B_{a,b} = \{v \in V \mid a < f(v) \leq b\}$. The isobands may be comprised of

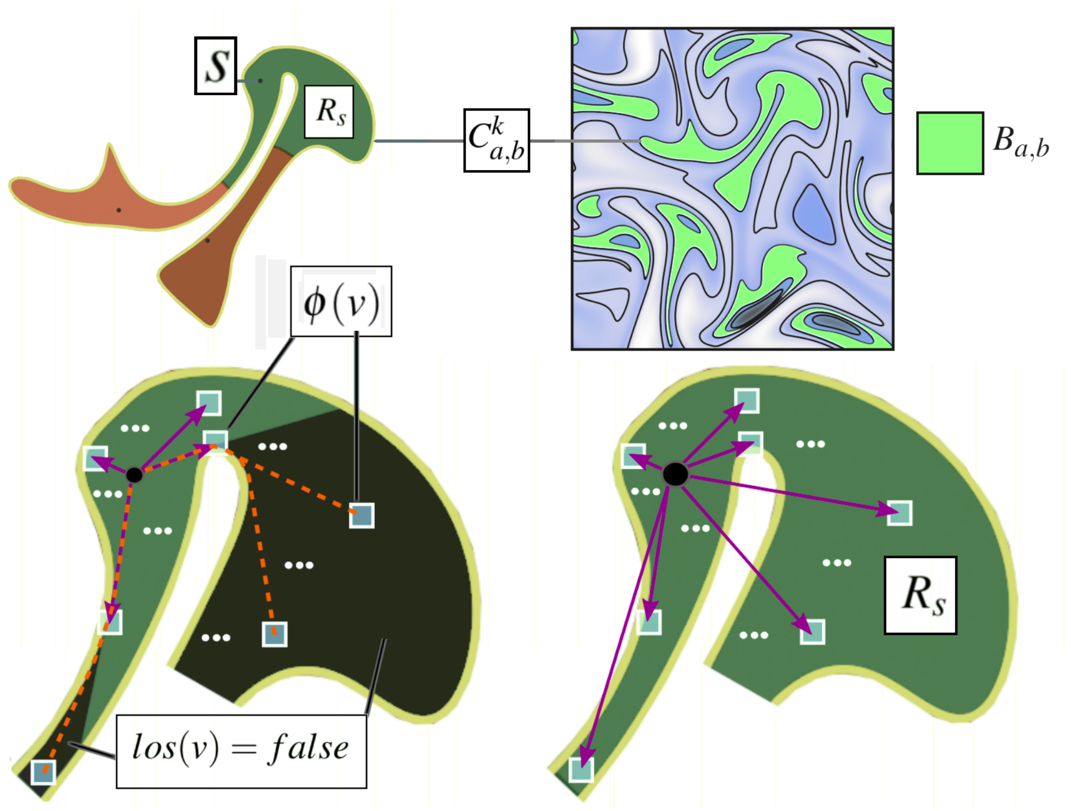


Figure 7.3. Upper right) Domain decomposed by isocontours into isobands. One isoband $B_{a,b}$ is highlighted in bright green. Upper left) One connected component of the isoband $C^k_{a,b}$ along with 3 Voronoi sites/regions. Lower right) Voronoi region R_s and illustration of standard averaging of the vectors for all voxels. Lower left) The mass of non *line-of-site* voxel v is propagated to $\phi(v)$, the LOS voxel on its geodesic path.

multiple spatially disconnected components, $C^k_{a,b}$, such that $B_{a,b} = C^0_{a,b} \cup C^1_{a,b} \cup ... \cup C^m_{a,b}$, where no voxel in $C^i_{a,b}$ is adjacent to any voxel in $C^j_{a,b}$ for any pair (i, j). Each of these components is a bounded region which is to be tessellated by a CVT.

Voronoi Sites and Regions. The Voronoi site s is defined by a point $\mathbf{x}_s$, and its Voronoi region restricted to the component C as

$$R_s = \{v \in C \mid s = \operatorname*{argmin}_{u \in S_C} gd(\mathbf{x}_v, \mathbf{x}_u)\} \tag{7.1}$$

where S_C is the set of Voronoi sites in component C, and $gd(\mathbf{x}_v, \mathbf{x}_u)$ is the graph geodesic distance from voxel v to Voronoi site u, where each voxel may be node in the graph, and no edges in the graph cross any boundaries, so that the distance $gd(\mathbf{x}_v, \mathbf{x}_u)$ is the shortest restricted path from the voxel to the Voronoi site.

Algorithm 7.1: Level Set Restricted CVT

```
1 Define LRCVT():
2     StratifiedRandomSiteDistribution()
3     do
4         RestrictedGeodesicVoronoiDecomposition()
5         RestrictedGeodesicallyWeightedUpdate()
6     while not converged()
7     GeodesicVoronoiDecomposition()
```

Centroids or Centers of Mass. A standard discrete CVT in a volume is a Voronoi tessellation where each Voronoi site point is at the centroid or center of mass of the region it encapsulates, where center of mass is defined as $\frac{1}{m(R_s)}\sum_{v\in R_s} m_v \mathbf{x}_v$, where R_s is the region of the site s, $v \in R_s$ are the encapsulated voxels, m_v is the weighting function evaluated at volume element v, $\mathbf{x}_v$ is the position vector of v, and $m(R_s)$ is the net mass for all $v \in R_s$. In the unweighted version, $(\forall v)[m_v = 1]$. However, by this definition, the center of mass of a Voronoi region could be outside the boundaries, which would make the centroidal distribution impossible, create disconnected Voronoi regions, and break Lloyd's algorithm (which repeatedly moves each site to the center of mass). To avoid escaping its component, we can move the site towards the centroid only until it runs into the boundary, but the sites can still remain stuck moving into the obstacles instead of around them. Instead we compute a weighted centroid of only *line-of-site* (LOS) voxels. The LOS voxels are those for which a strait line between them and their Voronoi site does not intersect a boundary. To account for voxels which are not LOS, we propagate their mass to the line of site nodes in their geodesic path back to their Voronoi site. This leads us to a new definition of the centroid $\mathbf{x}_c$, where instead of $\mathbf{x}_c = \frac{1}{m(R_s)}\sum_{v\in R_s} m_v \mathbf{p}_v$, we define,

$$\mathbf{x}'_c = \frac{1}{m(R_s)}\sum_{v\in R_s} m_v \mathbf{x}_{\phi(v)} \text{ s.t. } \phi(v) = v \textbf{\textit{ if }} los(v,s) \textbf{\textit{ else }} \phi(src(v)) \tag{7.2}$$

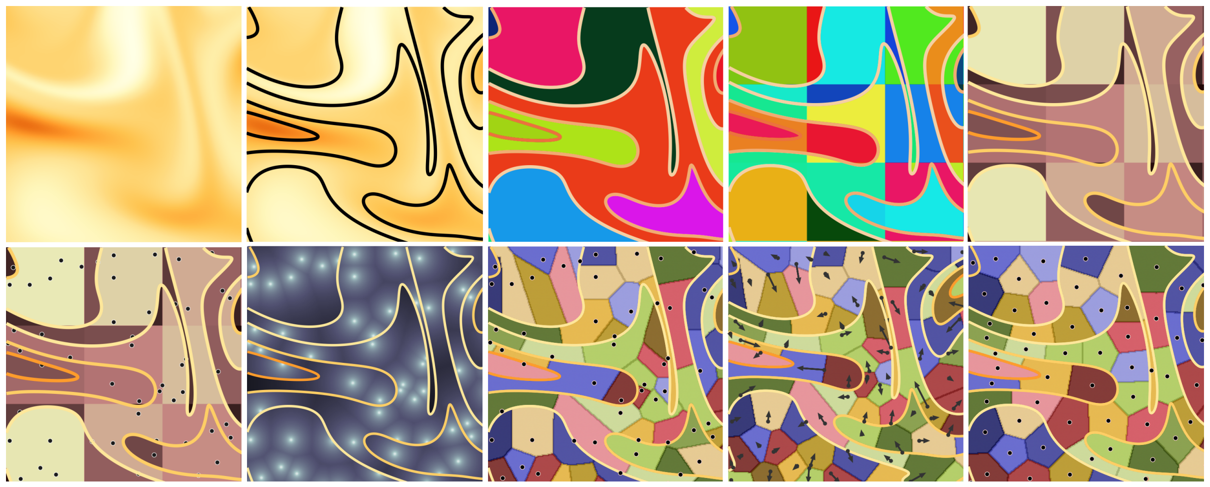


Figure 7.4. High-level illustration of our CVT algorithm. Left to right, top to bottom: The scalar field, decomposing into layers based on a set of isosurfaces/contours, connected component labelling, component/block decomposition for stratified component sampling, heat map showing the mass of each segment which determines how many Voronoi sites to place in each segment, the seed Voronoi sites, geodesic distances from each voxel to its geodesically nearest site, the initial Voronoi tesselation, a centroidal update, and lastly, the tesselation after multiple centroidal updates.

where $los(v, s) = true$ ***if*** v is line of site to s ***else*** $false$. Figure 7.2 shows weighted 2D and 3D examples of our LSRCVT.

7.1.2 Parallel Algorithm and Implementation

Our algorithm is based on Lloyd's algorithm [133], which repeatedly computes the Voronoi tessellation (VT) and then updates each Voronoi site to the centroid of its region as shown in Algorithm 7.1. The difference is we use the geodesic distances defined in Equation 7.1, and illustrated in Figure 7.3, to compute the VT, and we update the site positions based on $\mathbf{x}_c$ as defined in Equation 7.2. The full process is illustrated in Figure 7.4.

For distributed parallelism, each rank computes a LSRCVT for one block of data independently. The result is that the CVT is aligned to both the level set restrictions and the block boundaries, as can be seen in the large scale tessellation in Figure 7.1. The benefits and limitations are discussed in Section 7.5. Within each block, GPU parallelism is used to compute the CVT. The code is written in C++ using MPI for distributed parallelism, and Thrust [134] and CUDA [135] for GPU parallelism. Due to space limitations, the full implementation details are included in the supplemental material.

7.1.2.1 Stratified Random Initial Site Distribution.

The initial Voronoi sites are generated to cover the set of connected components with a target number of sites n_c in each component chosen based on its net mass and a *site-density* parameter α using the formula

$$n_c = \max\left(\left\lceil \frac{\alpha}{m(C,\gamma)} \sum_{v \in C} m_v^{\gamma} \right\rceil, 1\right). \tag{7.3}$$

Where γ is a parameter, the *attraction*, that reduces or increases the effect of the weighting, $m(C,\gamma) = \sum_{v \in C} m(v)^{\gamma}$, and $m(v)$ is the voxel's weight. We also use a block-wise spatial stratification, where each component is segmented into pieces and then the n_c sites are distributed fairly over those pieces. This is illustrated in Figure 7.4 row 1 (r1), columns 3,4, and 5, where the components (c3) are split into blocks (c4), the net mass of each component block is computed (c5), and then sites are placed based on the net mass 7.3 (r2,c1).

The process is done in parallel on the GPU. First we compute the net component masses $m(C,\gamma)$ and the net component block masses $m(b_i,\gamma)$. Next, the n_c sites are distributed by first seeding $\lfloor n_c \frac{m(b_i,\gamma)}{m(C,\gamma)} \rfloor$ sites in each block, and then placing the remaining sites to the most underrepresented blocks based on the equation,

$$ur(b_i) = 1.0 - n_{b_i} + \left\lfloor n_c \frac{m(b_i,\gamma)}{m(C,\gamma)} \right\rfloor \tag{7.4}$$

7.1.2.2 Discrete Restricted Geodesic Voronoi Tessellation.

The Voronoi tessellation, defined by Equation 7.1, is computed using a GPU parallel region growing approach emanating from the Voronoi sites. The voxels communicate with their neighbors to determine if there is a possible shorter path, and the process continues until all voxels have paths and no voxel can find a shorter one.

The paths P_v are represented recursively through a map $src(v) : V \longrightarrow V$, and can be constructed by the formula $P_v = (v, src(v), src(src(v)), ..., \phi(v), s)$, where $\phi(v)$ (defined in Equation 7.2) is the node in the path which is in *line-of-site* of s. Each voxel also has an 1-byte bit-field, $state_v$. One bit indicates $los(v, site(v))$ (whether v is *line-of-site* of its site), another indicates $active(v)$ (whether v has a current path), and another indicates $node(v)$ (whether v is a valid node).

In the initial step, the nearest volume element to each site is classified and activated. Then, repeatedly until the stopping condition is met, for each voxel v in parallel, we ask each other adjacent voxel w if it is active, and if it is, then their are two possible paths for v back to s: (1) the path (v, P_w), which is has the distance $||\mathbf{x}_v - \mathbf{x}_w||_2 + d[w]$, or (2) the path $(v, P_{src(w)})$, which has distance $||\mathbf{x}_v - \mathbf{x}_{src(w)}||_2 + d[src(w)]$. The former path (2) might intersect a boundary, so a ray must be cast from v to $src(w)$ to check for a collision. If v finds a shorter path based on the best of these possible choices, then it records the distance and new $src(v)$, but the update is delayed until a next kernel invocation to avoid race conditions. In order to determine which voxels are LOS to their Voronoi sites, in a initial stage of the above procedure, we only allow s to be active nodes so that all classified voxels are *line-of-site*. Then we activate the rest of the voxels as nodes and perform the procedure repeatedly until voxels have paths and cease to find a better path.

Figure 7.4 row 2 column 3 shows a heat map of the final geodesic distances for each voxel, and column 5 shows the associated Voronoi tessellation mapping each voxel to the geodesically nearest site. Figure 7.3, lower right, illustrates the LOS vs non-LOS voxels and pathways.

7.1.2.3 Geodesically Weighted Voronoi Site Update.

The next component of the algorithm is the updating of the site positions based on Equation 7.2. This is illustrated in Figure 7.4 row 2, column 4, and Figure 7.3. In a GPU kernel parallelized over voxels, we sum the moments $m_v\mathbf{x}_v$ of each *line-of-sight* voxel in each Voronoi region. The Cuda atomic add operator is used for the summation. The geodesic weighting comes into play as the line of sight voxels which are path nodes that connect voxels which are around corners are weighted by the amount of voxels that they are sources of.

Next, a GPU kernel parallelized over the Voronoi sites is invoked, and the resulting site's sum of moments from the previous stage is normalized by the net mass to give the vector $\mathbf{x}_c$ from Equation 7.2, which is the new target position for the $\mathbf{x}_s$. We then cast a ray from $\mathbf{x}_s$ to $\mathbf{x}_c$ and move $\mathbf{x}_s$ only as far as possible before crossing the boundary.

7.1.2.4 Data Reduction and Structured Data Layout.

Oftentimes, flow features of interest will comprise only a small portion of the full data. In this case, it makes sense to summarize and/or extract only the data within the level set restrictions of our tessellation as a way to reduce the data. Our method can optionally extract the raw data and couple it with the statistical summarization in a structured data layout before saving on disk. The data structure is hierarchically ordered by level set and connected components as shown in Figure 7.5. This ensures the components and layers are directly indexable and contiguous in storage, so one can directly load only the feature of interest for an efficient out of core approach. The statistical summaries, which are lighter weight, can be loaded first, explored, and then other data for selected features of interest can be loaded selectively.

The reduction from discarding the regions beyond the isobands can be approximated as $(n - r) \times (m + 1) + n_l + n_c$ where n is the number of volume elements in the full domain, r is the size of the subset within the levels, and m is the number of variables per volume element. n_l and n_c are the number of layers and components, respectively (for storing the offsets) which should be very small relative to r. Thus, when $(n - r)$ is large and or m is large, the additional cost of storing coordinates becomes nearly negligible. In extreme-scale combustion simulations, m can be greater than 100, in which case the coordinates can be as little as 1% of the total footprint of the reduced data.

7.1.3 Role and Choice of Parameters

Tessellation. The tessellation is both dynamic and data dependent, and also depends on a few important parameters. First, one must have the scalar field to use for the level set restrictions. For example, in our combustion application we may choose temperature isotherms, or we may use the distance function from the flame surface. The set of isovalues must then also be chosen. If the method is used in situ, one must know in advance what kinds of value ranges the simulation will produce, and what isobands will be of interest for representing the flow features. In addition, if the tessellation is weighted based on a second scalar field, then this adds another aspect to what must be known in advance to

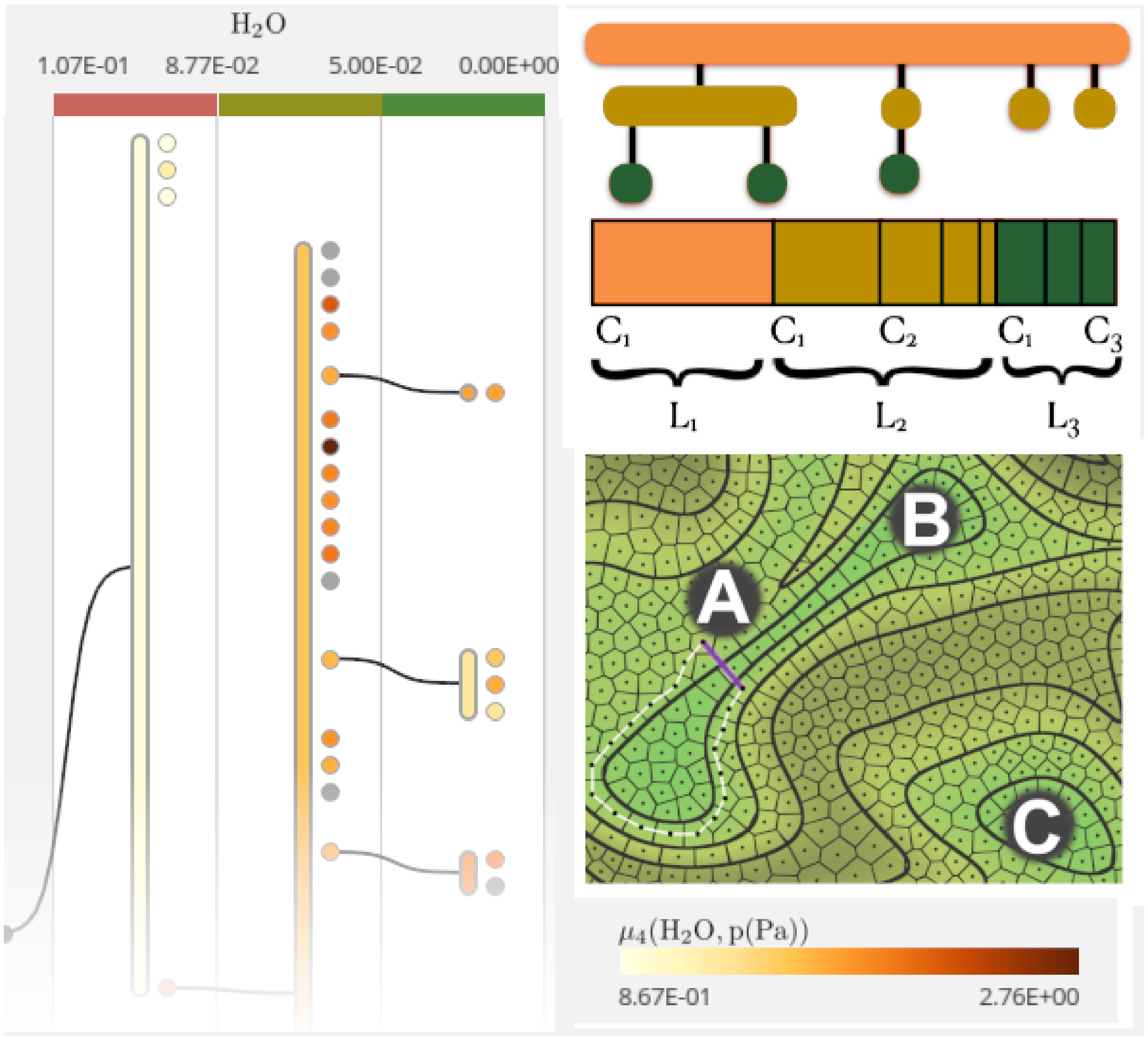


Figure 7.5. Top Right: The data layout is sorted hierarchically by layer and component. Middle right: (A) White line is a rough measure of geodesic distance based on paths along Voronoi cells following the level set restriction; blue line is Euclidean distance. (B) and (C) are two separate connected components of the same layer. Left: The tree view in our visualization system showing the iso-values defining the layers and the connected components as nodes. The grey nodes are designated as components that are too small to compare with statistical measures.

appropriately. The parameter γ controls the attraction force is equivalent to applying a non-linear re-scaling of the second scalar field used for weighting, and the effect it has depends on the data. The site density parameter α represents the choice of the number ratio of Voronoi sites to voxels. So this parameter can reasonably be decided in advanced with limited knowledge.

Aggregation. When aggregating data for spatial statistical summarization, information is lost if the raw data is not also output. Additionally, statistical aggregation is usually not a competitive form of pure compression, so if the primary goal is a reduced form of data to reconstruct the original data from, then this is probably not the best approach. Instead, we consider the statistical summarization as an explorable set of analysis results, which is useful for summarizing and analyzing certain aspects of the data, especially the dynamic spatial multivariate correlation structure. We assume existing methods for statistical summarization of the segments in the tessellation will be used (*e.g.*, histograms, Gaussian's, Gaussian mixtures, etc.). It also depends on the method and application what the value of the summarization will be, how much space it will take, and how much of the raw data if any can be replaced with the summary. Our view is that the summaries have multiple opportunities for added value. First, they are lightweight and, when coupled appropriately with a visualization system, can support scalable interactive visualization of a larger domain than would otherwise be possible from using the raw data directly. Second, even if the summary is not used to replace the raw data, it represents an analysis result that would be useful anyways in addition to the raw data. And since simulations are already commonly unable to output full raw data at each time step due to I/O and storage limitations, outputting the more lightweight summaries at higher temporal fidelity would add value to the simulation result without reducing any information at only a nominal extra cost. The lighter weight summaries can be possible to stream through a network for in situ monitoring. Finally, considering the hierarchically structured data layout that our tessellation provides, the summaries corresponding to flow features can be extracted out of core, and if coupled with raw data, if it exists, can also be a basis for drilling down into the full details from the raw data.

7.2 Interactive Visualization System

To showcase our feature-based approach, we designed and implemented a prototype visualization system. The system represents one promising way that the dynamic spatial decomposition approach could be leveraged. The system demonstrates the qualitative value of the approach, and we hope it can inspire more adaptations and ideas for similar work.

The visualization workflow follows the natural hierarchical pattern for sub-selection that is implicit in spatial decomposition. As the user drills down into details, based on layers, connected components, Voronoi cells, smaller subsets of volume elements are brought into play for spatial statistical analysis. The layers, components, Voronoi sites, and volume elements are interacted on and visualized in 3 associated sets of views as shown in Figure. 7.1. Since the system supports interactive multivariate statistical analysis, data from many fields may need to be active simultaneously. The hierarchical sub-selection becomes useful in this case, not only as a means to narrow down the features of interest, but also to improve performance since sub-selections at each level will significantly reduce the amount of active data.

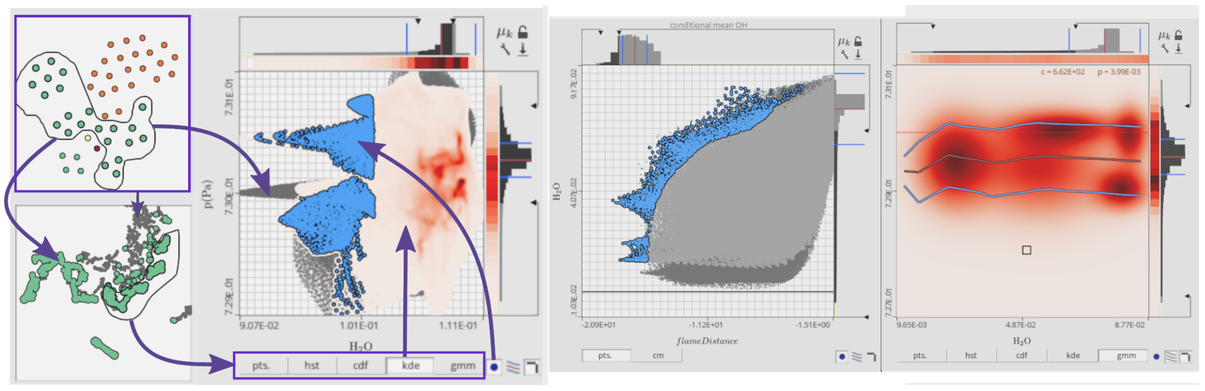


Figure 7.6. Upper left) t-SNE projection of components with lasso selection. Lower left) projection of Voronoi regions within selected components with second level lasso selection. Middle Left) KDE plot with voxel sub-selection (blue), and sub-marginals displayed. Middle right) scatter plot option is selected. Right) GMM with 1D conditional plot on top.

7.2.1 Motivation and Design Goals

This system in its first iteration was designed based on discussions between visualization researchers and combustion scientists. Both the visualization researchers and scientists are co-authors of this paper. Since, to our knowledge, no other systems exist that could explore the same data in sufficiently comparable way, and due to the limited length and scope of the paper, we are not able to do a thorough presentation of the system and robust evaluation at this stage. Instead we view this system as a starting point, and proof of concept for how to utilize the tessellation result, which may be built on, optimized, and evaluated in depth in future work. We provide below the set of visualization and analysis goals used to inform our design. In Section 7.5 we further lay out our vision for improvement and evaluation of the system.

Design Goals:

1. Be able to use the system on both raw and aggregated data for testing and comparison.

2. Be able to link and visualize the multivariate statistical plots and the features in the spatial domain.

3. Support the following common statistical plots: 1/2D probability density functions (PDF's), cumulative distribution functions (CDF's), 1/2D conditional plots, and scatter plots.

4. Be able to compare different candidate models for the probability distribution functions.

5. Be able to inspect the higher order joint statistical moments and how well the PDF models preserve them.

6. Be able to drill down into the details with a focus on the hierarchical structure of the data that is decomposed around flow features.

7.2.2 Isobands and Connected Components

The first set of views that the user will interact with are for selecting isobands and their connected components. A tree-based visualization is used to show the nested hierarchy as depicted in Figure. 7.5, and another linked view represents the components based on a 2D t-distributed stochastic neighbor embedding (t-SNE) projection, as can be seen in Figure. 7.6 (Left). The input to the projection is a featurization of the components, for example based on the bin values of histograms, parameters of parametric statistical models, or statistical moments. t-SNE will try to group the points with similar statistical features together. The points can be selected by the user through a lasso selection tool in the projection or from the tree view. The lasso selection operation has several modes besides new selection ($\circ$), based on set combination operations with the existing selection ($\cap, \cup$, or $\setminus$).

In the projection, the points are color coded by isoband, but in the tree view, that information is implicitly encoded by level of the node. Thus, for tree nodes, color encodes other information. For example, in Figure. 7.5 color is used to encode $\mu_4(\mathrm{H_2O}, p)$, which is the cokurtosis of hydrogen mixture fraction and pressure. Since very small components have too few voxels to get quality statistics, we color encode them in gray instead. This can also be helpful information since those small components may be particularly meaningful (*e.g.*, in combustion they may indicate the incipient development of an ignition event).

7.2.3 Voronoi Cell Views

The Voronoi regions are also projected into t-SNE projection views. Figure. 7.6 (lower left) shows such a projection. After selecting a set of components, the Voronoi cells in those components are highlighted in the Voronoi region projections and become selectable using the same lasso tool with the same set operator modes.

In addition to t-SNE projection based on statistical features, we found that a projection based on spatial similarity is a useful option for one or more of the linked Voronoi region projections, since it can group points based on spatial proximity. In Figure 7.1 (right) the bottom projection is based on such a 2D spatial projection. For this projection, we use a pre-computed distance/similarity matrix based on a custom pairwise distance measure between Voronoi sites. Since the tessellation is restricted by level sets, it makes sense to utilize the geodesic/path distance obeying the restrictions, as illustrated in Figure. 7.5 (A). The blue line shows the Euclidean distance, while the white line shows geodesic distance. An example where this useful in capturing the condition where regions around a folded surfaces (*e.g.*, flame) are geodesically distant but yet close together in Euclidean distance. In combustion, for example, this may preclude or represent interesting interactions of the flame as the Euclidean distance approaches the laminar flame thickness while the geodesic distance remains larger. One example distance function that can be used is,

$$d(s_i, s_j) = \left(\frac{||\mathbf{x}(s_i) - \mathbf{x}(s_j)||_2}{gd(s_i, s_j)} \right) \;\; \boldsymbol{if} \quad P_{i,j} \text{ exists} \;\; \boldsymbol{else} \; c \tag{7.5}$$

where $P_{i,j}$ is the graph geodesic of adjacent Voronoi sites, gd is the path's distance, $c \geq 1$ is a parameter where a higher value results in more separation of sites outside of the same component. To find $P_{i,j}$, we compute the connectivity of the Voronoi sites based on region adjacency, and then Johnson's all-pairs shortest path algorithm (which internally uses Dijkstra's algorithm) is used. Since the neighborhood is also restricted to within connected components the computation is parallelized by letting different threads operate on different components. The Boost graph library's implementation is used, which has time complexity $O(|V| \cdot |E| \cdot log(|V|))$, where V is the set of nodes, and E is the set of edges. An alternative to Johnson's/Dijkstra's is to use the Floyd-Warshall all pairs shortest path algorithm which has time complexity $O(|V|^3)$. Since our graph is only locally connected it is sparse enough that $O(|V| \cdot |E| \cdot log(|V|))$ is better than $O(|V|^3)$.

7.2.4 Interactive Joint Plots

Once a selection of Voronoi regions is made, statistical plots are generated, either from their pre-aggregated statistics, or from the raw data loaded out of core based on the aforementioned structured data layout.

Based on the experience of the domain scientists, the following statistical plots are chosen: joint PDF's and their marginals, joint CDFs, conditional plots (which estimate one variable conditioned on 1 or 2 others), and scatter plots. Since each of these plots share the same axis with the same scaling, we can switch between them, or combine layers on a single interactive generalized joint plot.

The joint plot includes information from both the background layer which is generated from the set of selected components, and foreground layer which is generated from the set of Voronoi regions within the components. The background layer in the joint plot is a static gray scatter plot, while the foreground layer is plotted over the background and depends on the selected plot mode. There are also marginals for both the background layer which are shown as gray histograms drawn as bar charts, and marginals of the foreground data as heatmaps beneath the bar charts as seen in Figure 7.6. The user can zoom and pan by scrolling or dragging the mouse when within either the joint plot (to zoom/pan both axis), or over one of the marginals (to zoom/pan only one axis). Highlighting on the background marginals, and arrows are used as a form of mini-map to depict the zoom level.

The joint plot of the foreground layer can be either scatter plot, histogram, CDF, kernel density estimation (KDE), or Gaussian mixture models (GMM's). The KDE plots are useful for capturing high-level detail and smoothness in the potentially complex nonlinear distributions. Histograms and GMM's are models which are commonly used for in situ statistical summarization. By using our system on existing data, before employing in situ, the choices of models and their parameters can be tested and the user can get an idea what information might be lost or misrepresented using the candidate methods. As the user interacts with the data, the PDF's may be interactively recomputed as the data inputs and the data range changes. Since the KDE and GMM plots are often not able to be recomputed immediately at interactive rates, the plot will automatically switch from KDE or GMM to a histogram, for example, while zooming/panning is occurring.

To evaluate the PDF's, the system estimates how well they preserve joint moments, which are used in the analysis of turbulent flows [136, 137]. The moments (*i.e.*, mean, covariance, coskewness, and cokurtosis) estimated from the PDF's are compared against the moments computed from the raw data. Each plot has a button (in the upper right corner, labeled μ_k) that can be pressed to open/close an attached table that is autogenerated as a LaTeX document as shown in Figure. 7.1. This functionality can also aid featurization of high-order moments-based anomaly detection algorithms to [138, 139].

If voxel data is available, the voxels deriving the foreground can be selected using the lasso tool and aforementioned set combination operators. The selections are linked across views and plotted as blue points. This layer can be toggled on/off per-plot to manage occlusion. These selections are also linked in the 3D view where their bounding surfaces are rendered. Conditional plots can be toggled on/off as well, and are rendered on the top most layer as a red curve estimating mean y as a function of x, with blue lines above and below the curve at ± 1 standard deviation. 2D conditional plots, showing estimated mean z as a function of (x, y), rendered as a heatmap, is also supported. For these plots, the forground layer can also be toggled to show a scatter plot of (x, y) as shown in Figure 7.6 (second from right).

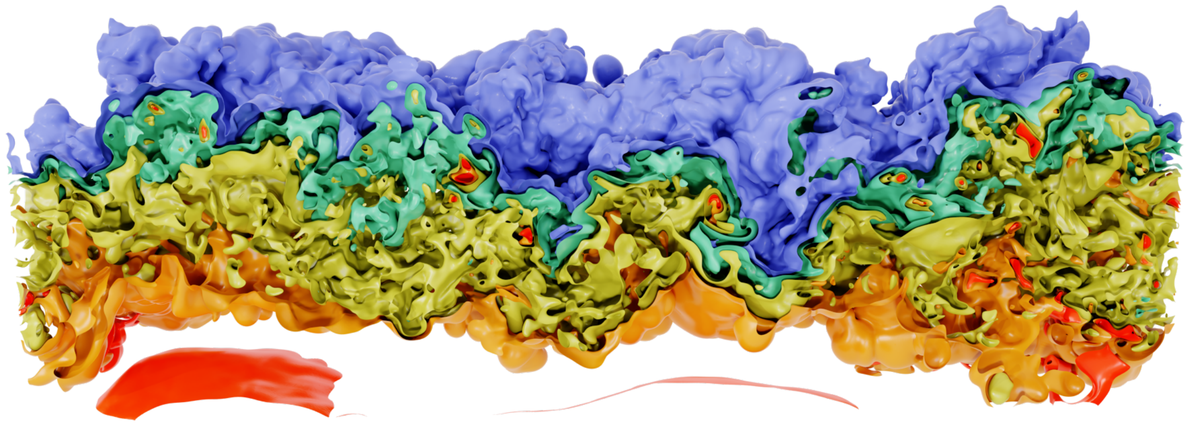


Figure 7.7. Isotherms in the premixed hydrogen/ammonia/nitrogen flame.

7.3 Performance

The data used to obtain these results is the hydrogen/ammonia/nitrogen flame data described in Section 7.4. Five isobands of temperature were used. The bands are highly contorted as can be seen in Figure 7.7. This makes the data a good stress test for the algorithm as the Voronoi sites need to find their way around complex obstacles. At each iteration of Lloyd's algorithm, we measure the distance moved, d_s, for each site point, s. And we also compute the mean, $\bar{d}_s$, which is used in the formulation of our stopping conditions.

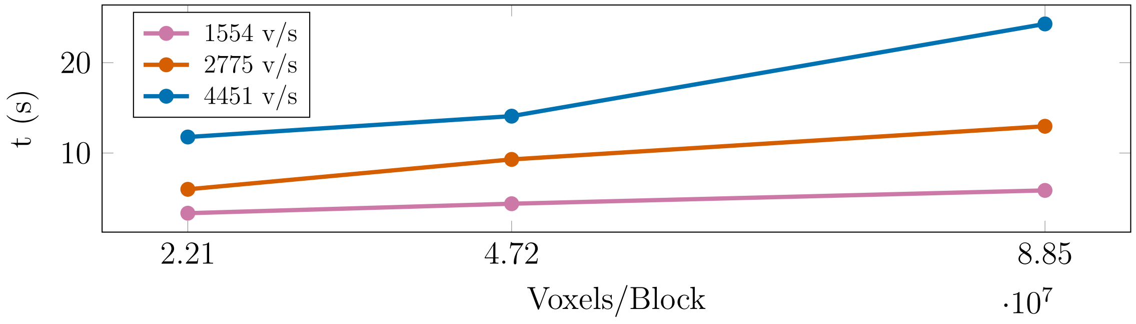


Figure 7.8. Computation time per-node to reach a mean d_s of less than 0.5 (unit voxel length). Each line represents a different block size for distributed computation. Since the computation is independent for each block of data in a distributed setting, the scalability can be assumed to depend primarily on the number of GPU nodes available.

First, we evaluate the computation time with respect to site density and number of voxels on a node; shown in Figure 7.8. The performance drops as the number of voxels on a GPU node increases. The primary bottleneck is the Voronoi classification described in Section 7.1.2.2. This computation is parallelized on the GPU using one thread per voxel without communication. Thus we can expect the GPU parallelism to scale well with the number of GPU threads. However, the performance decreases approximately linearly with respect to site density, as lower site density means longer paths traced out per voxel/thread.

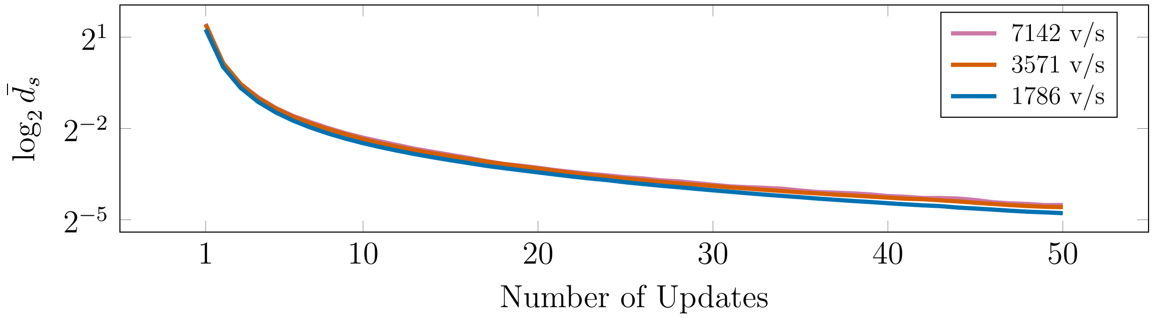


Figure 7.9. The reduction in $\log_2 \bar{d}_s$ (log base 2 of mean d_s) over the number of geodesically weighted centroidal update steps, shown for 3 different site densities. The result is similar in each case. The mean continues to improve slightly over time with diminishing returns on the benefit for the cost.

The second result, in Figure 7.9, shows how quickly the tessellation quality improves in terms of the number of steps in Algorithm 1. Diminishing returns for the computation cost are seen after only a handful iterations. Since the computation is independent over each node/block of the block decomposition domain, we were able to use a single user machine to perform these first two tests; an Arch-Linux machine with an Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz with 64 GB of main memory, and an NVIDIA GeForce GTX TITAN X GPU with 12 GB of memory. The number of Cuda threads per-block was set at 128, which gave us the best performance. The Cuda code was compiled with compiler flags, 'nvcc -std=c++11 -O3 -w -m 64 -rdc=false -ftz=true -use_fast_math -Xptxas -O4, -Xcompiler -O4 -arch=sm_52'.

Nodes	8	16	32	48	60
GPUs	48	96	192	288	360
time(s)	26.939	12.996	5.618	3.990	3.091

Table 7.1. Parallel Performance results for Ammonia Data of Size $3456 \times 1280 \times 2560$. The computation was performed till mean d_s reached 0.25 voxel lengths.

Last, we performed a full distributed parallel test for the combustion data on the Summit supercomputer. The volume has a size of $3456 \times 1280 \times 2560$ voxels. The site density was 2700 voxels per Voronoi site. We ran the system until it reached the point where mean d_s dropped below 0.25 unit voxel widths. The results are shown in Table 1. Each of the nodes on Summit has 6 NVIDIA V100 Tesla accelerators, 512 GB of DDR4 memory, and two IBM POWER9 CPUs. We ran our code with 1 GPU and 1 CPU per rank, and 6 ranks per node. We testing the performance using 48, 96, 192, 288, and 360 GPUs. Since Summit has 6 GPUs per-node, we mapped this to 8, 16, 32, 48, and 60 nodes respectively. The run command for example was 'jsrun -n48 -a1 -c4 -g1 ./t8' for the case of 48 GPUs. The CUDA code was compiled with NVCC, with command-line arguments: `-std=c++11 -O3 -w -m 64 -rdc=true -Xptxas -O3`. The CUDA version was 11.4.2. The C++ program that links the GPU code was compiled with GCC 10.2.0 (the GNU Compiler Collection) with arguments: `-std=c++11 -O3 -fpic -fopenmp`. Since each node took a different amount of time to complete, we report the worst case, which represents the overall time for the entire data to process. The CUDA kernels used a block size of 128.

7.4 Applications

We tested our algorithms on two types of flow data, turbulent combustion and wall bounded channel flow. The flame surface in the turbulent combustion data is more connected, but still highly complex, while the channel flow data has many small vortical structures. The combustion data features strong coupling between the flame wrinkling and the underlying turbulent strain, where the flame response occurs in a composition phase space (dependent locally on species concentrations and temperature which control the burning rate and are modulated by the turbulent strain), while turbulent strain evolves in physical space. Hence, there is an inherent need to interactively toggle between physical and composition space to drill down and understand causality. Depictions of the two datasets and tessellations made using them are shown in Figure 7.7 and Figure 7.10, respectively.

7.4.1 Hydrogen/Ammonia/Nitrogen Turbulent Flame

We apply the nested hierarchical approach presented here to turbulent combustion data from Direct Numerical Simulation (DNS) of a premixed hydrogen/ammonia/nitrogen flame [140]. The DNS study aims to gain an understanding of the combustion behavior of hydrogen/ammonia/nitrogen as a carbon-free replacement for conventional fuel (*i.e.*, natural gas). The simulation was run on 900 nodes of the Oak Ridge National Laboratory OLCF Summit cluster requiring approximately 110k node-hours. The aggregate amount of data produced comprises 400 checkpoint files, where each checkpoint file is 2TB. The data is described further in the Appendix A.3.2.

Due to the nature of turbulent combustion and the need to resolve all flow and chemical scales, turbulent combustion DNS data is characterized by a large separation of scales, with portions of the data often not relevant for statistical analysis or over-resolved on the numerical grid that is designed to capture the smallest scales (*e.g.*, regions of hot products feature small gradients/large flow and chemical features). The Voronoi tessellation effectively reduces the amount of data to be processed for the purpose of statistical analysis and allows for efficient and fast data exploration. While conventional statistical analysis (*e.g.*, (joint) statistical distributions, statistical moments, correlations between species and temperature and between reaction rates and strain rate) is important in turbulent combustion, the analysis of spatially and temporally intermittent features is often of interest (*e.g.*, flame-flame interactions, reactant pocket formation enriched with minor species that are chemically crucial, and formation of localized flame extinction holes). The hierarchical selection of connected components together with distance-based metrics, and with the linked views between spatial and statistical features, allows scientists to quickly identify and select specific flame features and directly extract necessary statistics to understand causality between the 'turbulence-chemistry' interactions and to steer further downstream analysis, and adaptive labeling of 'feature' data for searchable training and validation for machine learning.

The mass fraction of H_2O, normalized by the values obtained in burned and unburned gas of a laminar, unperturbed flame at the same conditions as the turbulent flame, acts as a so-called progress variable, which monotonically increases from 0 to 1 across the flame brush. This progress variable is usually used to identify specific zones of a premixed flame and specific values can be selected to obtain iso-surfaces that identify the various zones in the flame front (*i.e.* preheat zone, reaction zone, oxidation zone in a premixed flame). Here, we choose a specific value that represents the reaction zone, *i.e.*, the location where the heat released by the flame is at its peak. We refer to the iso-surface corresponding to this value as the flame surface. Of particular interest is the analysis of the value of other quantities on the flame surface. For example, a large value of the OH radical species represents a region that is burning particularly strongly and in the present dataset, this corresponds to a flame element that experiences preferential diffusion effects (*i.e.*, hydrogen reactant diffusing faster than heat), where the burning is amplified locally by a large influx of fuel. In other regions of the flame, OH is low, which represents flame elements that are locally extinguishing, *i.e.*, strain and stretch-induced by turbulence disrupt the reactions to a large extent. Distance-based metrics also provide a conditioning variable in physical space indicating the response of different zones in the flame, upstream and downstream of the flame isosurface, to the strain rate. Distance function iso-contours represent the locus of all points that are equidistant from the flame surface.

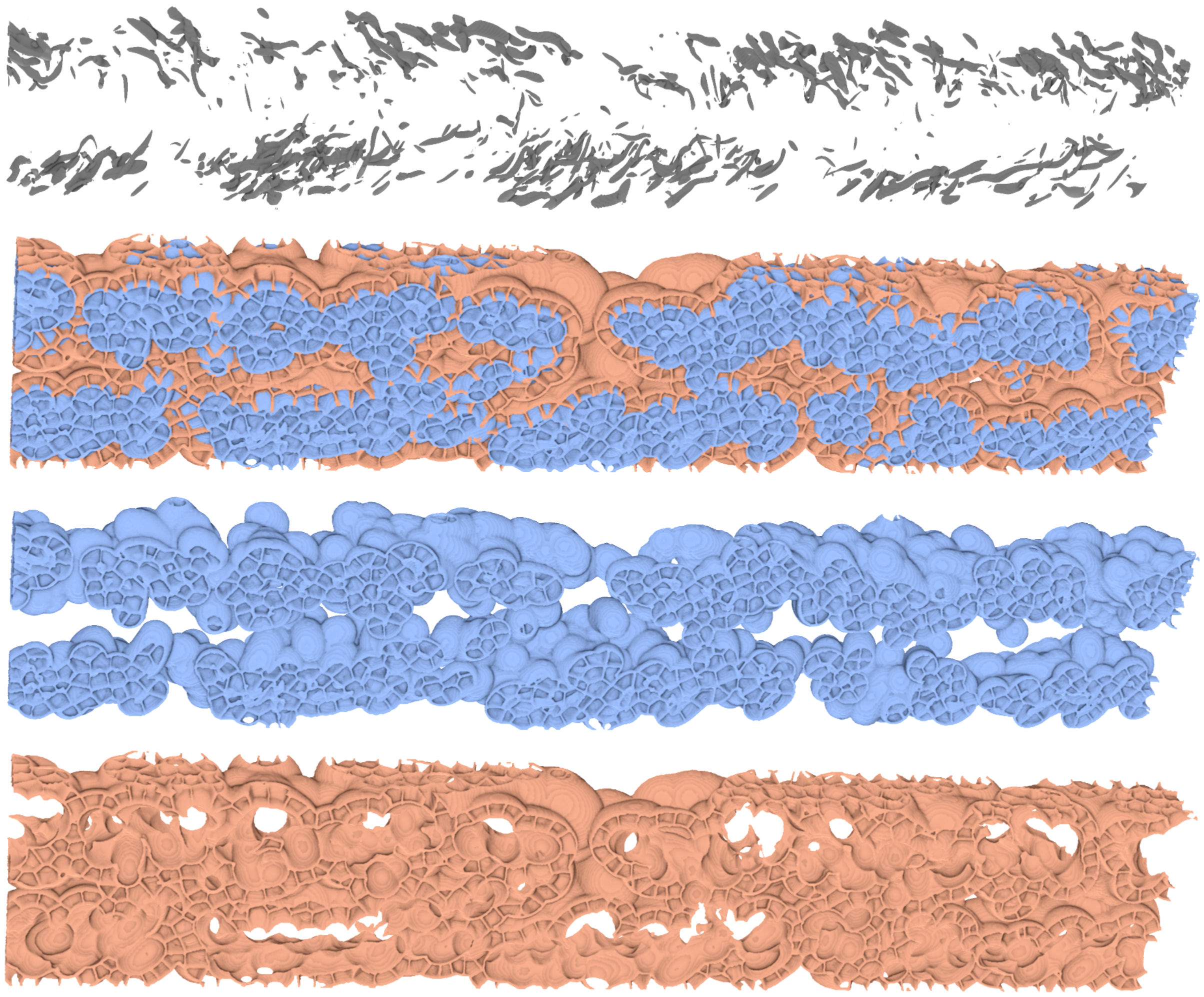


Figure 7.10. Channel flow decomposition using the λ_2 field to encapsulate vortical structures, and the distance function to define iso-levels. The dark grey surface represents the vortical structures.

7.4.2 Turbulent Channel Flow at High Reynolds Number

Here we examine is the incompressible turbulent channel flows at a high Reynolds number, *Re* (Figure 7.10). High *Re* wall-bounded turbulence has great importance in many engineering applications with moving objects at high speed. However, the fundamental dynamics of wall-bounded flows are not yet fully understood. The turbulent channel flow is one of the simplest canonical flows to study the high *Re* wall-bounded turbulence. Still, the DNS of high *Re* turbulent channel flows is quite expensive due to the scale separation similar to the turbulent combustion as explained above. The simulation performed with 33K nodes in Mira (Bluegene/Q, ALCF) used 242 billion DOFs, and it remains the largest DNS simulation of wall-bounded turbulence [141, 142, 143]. The data is described further in the Appendix A.4.1.

In this study, we use the λ_2 field, which is calculated from a subset of the aforementioned DNS. The interaction of coherent structures in turbulent flows is a long-standing problem. Naturally, the methods to identify vortex structures are well developed. λ_2 is one of the widely used vortex identification methods. It uses the eigenvalues of a 3×3 matrix constructed with a velocity gradient tensor[144]. Even though the vortex identification method is mature itself, using it to study the dynamics of high Re wall-bounded turbulence is still a challenging task because the identified vortex structures are too numerous to analyze readily from the raw data[145]. Note that other types of high Re turbulence also encounter the same flow complexity. The dynamic nested hierarchical statistical decomposition approach can efficiently identify the vortical structures and analyze their interactions with other state variables, such as velocity and pressure fields.

7.5 Discussion

Block Based Distributed Parallelism. For large-scale domains, a distributed approach is needed. There are two possibilities; (1) perform the CVT globally over the whole domain, which will be computationally intensive due to the communication requirements across nodes, or (2) compute the CVT separately on large blocks of the domain on different ranks. We have only implemented the second option. The result is that the boundaries of the blocks will act as further restrictions so that the Voronoi sites will be aligned to those boundaries in addition to the isobands. Such alignments are already implicit at the boundaries of the full domain, and the alignment to the block boundaries does not break the level set based hierarchy, or alignment of the CVT to the level set surfaces. Additionally, it offers one advantage over a global CVT; whole blocks can be loaded without clipping the tessellation. This is practical because large datasets are often analyzed post-hoc through block-wise sub-selection due to the large scale. It also makes a more efficient implementation possible and is trivial to scale up.

CVT Algorithm and Optimality. In general, it is difficult to prove guarantees for CVTs [146]. Liu et al. provide theory on CVT convergence [147]. Despite theoretical challenges, CVT algorithms adaptively minimize energy and are argued to have practical expectations of convergence. For our algorithm it depends on the definition of $\mathbf{x}_c$, the new target site position in the update step for Lloyd's algorithm. As site density increases and/or the restriction complexity decreases, $\mathbf{x}_c$ approaches the Euclidean centroid. While the tessellation will evolve from less optimal to more optimal distributions, we are not able to prove theoretical guarantees. A more optimal definition is also possible, but it is unclear what that should be in order to balance efficiency and quality. Our implementation can be extended or improved in the future by redefining $\mathbf{x}_c$ and modifying the update step. As an alternative to Lloyd's algorithm, one can also try an approach based on limited memory BFGS method for large scale optimization (L-BFGS) [148]. Lastly, the performance depends strongly on the choice of the stopping condition. As our tessellation must conform to complex boundaries, the spatial homogeneity of the tessellation cells is generally imperfect. This means that the value gained from extensive refinement through iterations of Lloyd's algorithm may not be worthwhile considering the performance trade-off. The tessellation usually reaches a decent state with only a few iterations. Thus, if time is a constraint, *i.e.*, if the algorithm must complete in time to prevent the delay of a simulation (when used in situ), then one could define a small minimum number of iterations, and then run as many iterations of Lloyd's algorithm as possible within the allotted time beyond the minimum.

Need to Choose Parameters A Priori and Loss of Information. As discussed in Section 7.1.3, there is a significant limitation in the need to specify parameters before hand, and the choice of parameters depends on dynamic data. This becomes a bigger concern when one is less clear what kind of data they can expect the simulation to produce. When one has some existing data which they expect to have similar characteristics to the data the simulation will produce, then they can apply the methods on the raw data and use our prototype system to explore the results while being able to compare what information may be lost from the raw data given a parameterization of the methods. As future work, one may explore robust dynamic methods to adapt the statistical aggregation/modelling approach as the data evolves based on the uncertainty or error. For example, when a model is no longer able to represent the data sufficiently, one can trigger more output of the raw data, or employ a more expensive yet superior algorithm or model.

Applicability and Extensibility. Our implementation works on Cartesian grids and rectilinear grids. For rectilinear grids, one can perform the tessellation as if it were a Cartesian grid in which case the tessellation will be stretched to match the grid. In some cases, this approach is sensible because it results in more homogeneity in region sample sizes (volume elements per region), which may yield greater statistical consistency. However, one can also use the stretching function for the tessellation. The algorithm can be extended to work in most cases where a discrete binning of volume elements is acceptable.

Impact of the Visualization Approach While visualization tools for 3D rendering and surface analysis of the large scale data have matured over the years, there is still a lack of scalable tools for exploring large scale multivariate data and spatial statistical correlation. Our work contributes new ideas and software to help push the state of the art in this area, and can be expanded on by future research.

Evaluation and Refinement of Visualization Design. The visualization system and overall approach has clear motivation and has been applied to help combustion scientists better understand their data. The design is still in an early stage and represents only one for approach visualizing spatial statistics using the hierarchical segmentation. We have determined several ways it can be improve. First, a direct selection of connected components in the 3D view would be useful. This could be supported either based on an explorable image with depth buffers [149], through 3D ray intersection testing. Second, we would like to region growing of the selections based on spatial or statistical proximity. Lastly, the visualization system needs better support for temporal visualization. The statistical summaries of components can be efficiently visualized through time utilizing a dynamic tracking graph [55] and the structured data layout from Section 7.1.2.4.

Once the visualization system has matured, a more robustly evaluation can be done. Evaluation of methods designed for use by qualified scientists for exploratory visualization and analysis can be challenging, because few are qualified to participate in studies, and appropriate tasks are difficult to define for visualizations when the result of the visualization is qualitative insight. If a user study is overly simplified or narrow in scope, it can give a false impression of the usefulness and potential of the design [150]. Feedback from the participating scientists on the value of the methods and tools is included in Section 7.4. However, the qualified scientists we have worked with are co-authors and helped design the system around their interests, so there is a level of potential bias that may be expected in their direct feedback, and the potential use cases for the methodology spans domains we are not experts in. Our vision for the future evaluation and refinement of the system involves iterative refinement based on lessons learned as we apply it in our domain [151], as well as through as much feedback as can be gained by other users outside our domain.

Chapter 8

Conclusion

The dissertation concludes with some of my views about the challenges in collaborative scientific visualization research, remarks on the overall impact of our work, some of the overall limitations, and opinions about promising directions for future work.

8.1 Collaborative Visualization Research with Domain Scientists

Some challenges I faced as a visualization researcher collaborating with domain scientists include: (1) The visualization researcher is not the leading expert in the scientific field of application. One may readily become acquainted with the application domain but would not be expected to attain the level of expertise necessary to perform research independently. Thus one must rely intimately on the collaborating scientists to help guide the research. (2) The visualization researcher needs to do novel research while many of the domain scientist's immediate and intermediate needs are often solvable without novel visualization methodology. Furthermore, the novel directions of research that are useful for the salient applications to the domain scientists may be highly customized and narrow in application. Thus one needs to be determined to find novel solutions and broaden the work's scope and impact. (3) Research papers where the methods are designed and motivated based on specific domain science needs are difficult for members of the visualization community to review. Relying on the collaborating domain scientists to evaluate and give testimony as to the value of the work presents a conflict of interest since they are often co-authors or invested in the research themselves. However, they may be the only qualified persons who are readily available to evaluate the work. Furthermore, visualization is highly qualitative in nature, and its insights are difficult to measure. (4) The datasets are large and difficult to acquire. This leads to difficulties in testing and comparing work against the previous works and exasperates the already difficult evaluation challenges.

Despite the challenges, I believe there is a wealth of rich and novel visualization problems yet to be discovered through domain science-driven collaboration. And through time, the common factors between the visualization needs of scientists in different niche areas will be better understood, and the value of the past research will thus increase with age.

8.2 Overall Impact

The problem of visualizing large-scale, multivariate, spatially, and temporally distributed simulation data is broad. The data will have such characteristics in a wide range of scientific domains where physical systems are modeled. In each case, specific requirements are necessary, and goals constrain the approach. This makes it difficult to declare what is precisely the state-of-the-art when it comes to the visualization methodology or approach. When it comes to distribution-driven visualization methods, this is especially the case since the methodology can be constrained by the statistical quality of the data and assumptions, as well as the end goals, which can vary from monitoring in search of anomalies [43] or errors, exploratory visualization to pinpoint known or undiscovered patterns [152], validation or monitoring of simulation progress, filtering [151], fast but lossy data compression [47], or some combination of any of these.

Within the scope of my research, we focused not on developing new statistical methods but on spatial decomposition and layout, visualizations, visualization system design, interactions, and efficiency and scalability. Before our work, there was some work on spatial decomposition in combination with statistical summarization, but there was still a lack of work on utilizing the spatially decomposed statistics for interactive multivariate visualization. In both the tokamak fusion and combustion cases, achieving the goals conveyed to us by the domain scientists require novel solutions. The solutions in each case are multifaceted.

8.3 Limitations

One of the main limitations of this work is the limited forms of supported data types. These are not necessarily fundamental limitations, but future work may be necessary to determine the suitability of our approach in different cases, and one may need to modify, redesign, or replace some of our algorithms. Overall, our driving applications represent limited classes of 3D simulation data and have yet to be tested widely on the potential application domains. Because of this limitation, we can only guess at the benefits of our approaches to the domains we have yet to consider. For example, our methods might be useful for applications to the visualization of cosmology data, space plasma data, or blood flow data, to name a few. Still, it is unclear what changes need to be made for effective application and what common factors can effectively generalize.

Another limitation is the lack of testing for in situ processing and visualization. Our approaches, specifically in Chapters 4 and 7, are geared towards in situ use. In Chapter 4, histograms were generated in situ, but we have not supported live in situ monitoring. In Chapter 4, we could not easily test the methods for in situ processing and monitoring because the simulation is too expensive to re-run for testing purposes with our limited HPC allocations. In general, large-scale simulations of turbulent combustion, or tokamak fusion, are costly. Smaller versions of the simulations may be used for testing in situ methods, but the smaller simulations may not produce similar results to the larger-scale simulations. We thus may need to test on existing data, if available, or make apriori assumptions based on expert knowledge to prepare the in situ workflows for a future big simulation run. Furthermore, the methods need to be trustworthy enough to bring into the dependency chain of a highly expensive simulation when mistakes can be costly.

8.4 Directions for Future Work

In Chapters 3 through 7, we discuss ideas for future work specific to the individual topics. Here we focus more broadly on what can be done to advance the state-of-the-art in practice.

First, some obvious directions for future work are to seek to overcome the limitations mentioned above. For example, the extension of our methods to work on different kinds of data, as well as extensions to better support different applications, will help increase the impact of the work. Furthermore, research is needed to determine how to better evaluate these methods, which are currently assessed primarily logically and qualitatively through expert feedback. I think that one problem we face is that distribution-driven visualization, or statistical summarization, are very general topics with different use cases. A taxonomy of these different use cases pertaining to visualization and an in-depth review of the statistical characteristics and modeling requirements for data from different scientific domains would be a nice contribution.

Another direction for future work that I suggest is the development of automated statistical summarization workflows to be used in situ. Since processing performed in situ may not be able to be redone later on, it is imperative that the processing succeeds the first time. When done in situ, the statistical model and any parameters need to be either data-driven or decided apriori. In some cases, lightweight parametric models might be sufficient, except in special circumstances that cannot be determined in advance. In such cases, fallback models could be leveraged. Moreover, based on saliency measures, one could trigger more expensive processing, or the output of more data (in time or space), in different regions. I envision a framework supported by a declarative domain-specific language, *e.g.* Diva [153], for specifying the assumptions, rules, and priority or saliency measures and how the in situ workflow should adapt based on the data and hardware constraints to optimize the value of the output while minimizing the costs in resources.

Next, future work is needed to figure out how to incorporate different forms of data into the same integrated visualization system. For example, how can we best visualize collections of histograms with different extents or bin edges, when some distribution functions are parametric and others are non-parametric, or when certain forms of data are only available for some time steps or spatial regions?

Modern scientific simulation datasets, like those from the domains described in this dissertation, are already so large that they are cumbersome to analyze and manage in storage. But in the big-picture, this data is small compared to the amount of information nature produces about the same phenomena. I believe that we will always be seeking to simulate at a larger scale. As compute capabilities necessary to process and store the resulting data increase, our ability to effectively visualize the data will still be limited by our inability to sift through it, perceive the information, and interpret it. Yet this is a valuable process since inspecting the data may be crucial for discovering the unexpected.

Appendices

Appendix A

Datasets Used in This Dissertation

Most of the data used in this dissertation are large and managed by external organizations and thus can only be obtained through channels outside my control. Inquiries in these cases should be directed to the corresponding authors of the cited papers.

A.1 Particle-in-Cell Simulation Data

The fusion simulation datasets were provided to us by scientists at Princeton Plasma Physics Laboratory (PPPL). The accelerator simulation data was provided by scientists at the Stanford Linear Accelerator (SLAC).

A.1.1 XGC ITER Fusion Simulation Data

XGC [37] is a tokamak fusion simulation code designed to study edge physics of magnetically confined plasmas. The fusion datasets come from large-scale simulations developed by Princeton Plasma Physics Laboratory research teams. This dataset represents a simulation of the International Tokamak Experimental Reactor (ITER) [154]. The dataset we were provided contains both field and particle data, as well as constants and formulas for deriving variables of interest from the raw data. A rendering of the device and its coordinate systems are shown in Figure A.1. A deeper understanding of the code and research supported by the XGC1 ITER simulation is provided by Chang Et al. [155].

Mesh and Fields The simulation includes both a magnetic field and an electric field data stored on a mesh grid. The spatial coordinates are positions in cylindrical coordi-

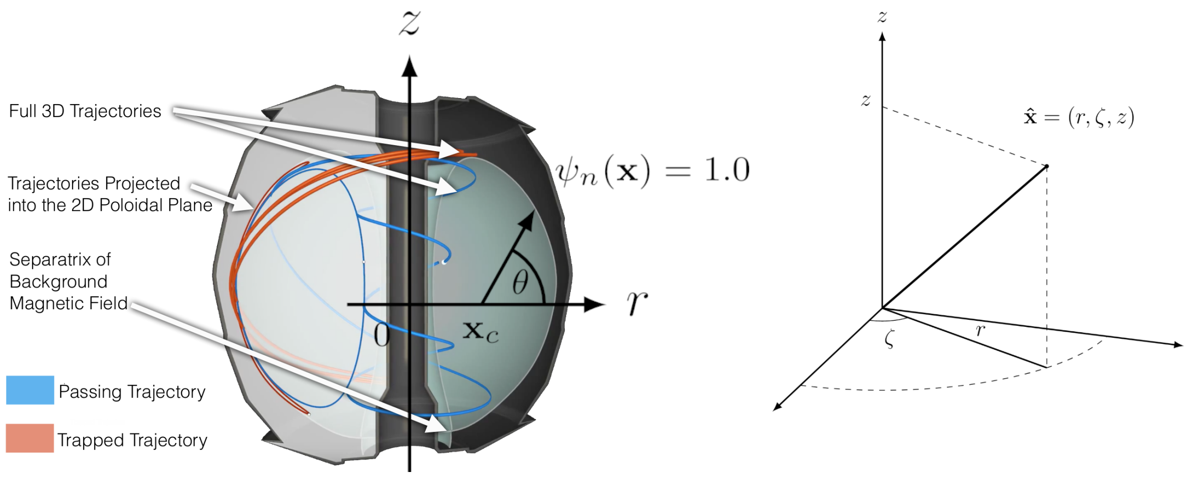

Full 3D Trajectories
Trajectories Projected into the 2D Poloidal Plane
Separatrix of Background Magnetic Field
Passing Trajectory
Trapped Trajectory
z
r
θ
0
$\mathbf{x}_c$
$\psi_n(\mathbf{x}) = 1.0$
$\hat{\mathbf{x}} = (r, \zeta, z)$
ζ

Figure A.1. An illustration of the XGC ITER Simulation domain. On the right, the cylindrical coordinate system, with positions (r, ζ, z), on the left, the tokamak device is cut in half. On the left side of the device, the face is the 2D poloidal plane mesh. Two types of particle trajectories, passing and trapped, are rendered in both the 3D space and projected onto the poloidal plane. On the right side of the device, the separatrix of the magnetic field is rendered. Overlaid in black is the 2D (r, z) axis, an illustration of the center of the poloidal plane $\mathbf{x}_c$, and the poloidal angle θ. The separatrix is the isosurface of the magnetic field where it crosses over itself. ψ_n, the normalized magnetic radius, is equal to 0 at the center of the poloidal plane and equal to 1.0 at the separatrix.

nates, (r, ζ, z), and the mesh grid consists of a series of identical 2D meshes representing the poloidal plane (r, z), organized toroidally around the device (at different toroidal angles, ζ).

The magnetic field $\mathbf{B}$ is modeled as a static (non time-varying) field, while the electric field is time-varying. In this work, the electric field data, however, is not used.

Particles The particle we were provided from this simulation consists of a sample of simulation particles, which includes 286,000 electron particles and 286,000 ion particles, each for 282 time steps. The number of particles simulated is much larger. However, only a small sample had been available to us. Each particle has positions (r, z, ζ), variables $\rho_{\|}$ and μ, which are the parallel Larmor radius and the magnetic moment, respectively, and weights w_0 and w_1. The velocities, $v_{\|}$ and $v_{\perp}$, in magnetic coordinates (parallel and perpendicular to the magnetic field), as well as energy, are derived from the raw data.

Particle Weights Each particle also has a pair of weights, w_0, w_1, where $w = w_0 \times w_1$ is equal to the number of real particles that the super particle represents. Each super particle must be weighted by w to get its net contribution to the distribution. w is time-varying, and since the particle density is represented through perturbations from a background distribution, w can also be negative.

Electrons For the electrons, we derived the following variables:

$$v_\perp = \sqrt{\frac{2\mu\|\mathbf{B}\|}{m_e}}$$

,

where $\mathbf{B}$ is the magnetic field at (r, ζ, z) and $m_e = \frac{m_i}{m_r}$ (the mass of the electrons in the simulation), and $m_i = 2 \times 1.660539040e - 27$ (the mass of the ions in the simulation, which for the ITER simulation case are pairs of hydrogen atoms). m_r is a simulation parameter, which for the ITER simulation is 1000.

$$v_\| = \frac{-e\rho_\|\|\mathbf{B}\|}{m_e}$$

, and

$$E = \mu\|\mathbf{B}\| + 0.5m_e\left(\frac{-e\|\mathbf{B}\|\rho_\|}{m_e}\right)^2$$

Ions for the ions, we derived the following variables:

$$v_\perp = \sqrt{\frac{2\mu\|\mathbf{B}\|}{m_i}}$$

,

$$v_\| = \frac{e\rho_\|\|\mathbf{B}\|}{m_i}$$

, and

$$E_k = 0.5m_i(v_\|^2 + v_\perp^2)$$

Analysis Interests In speaking to scientists at PPPL, we learned that the analysis interests for tokamak simulation data are highly varied and dependent on the specializations of the researcher. Through our collaborations with several members of the PPPL theory group, we focused on helping to support a subset of those analysis interests. In each case, the scientists were primarily interested in the visualization of the particles' toroidally integrated statistics and the positions projected onto the 2D poloidal plane (r, z) by discarding ζ as depicted in Figure A.1.

A.1.2 GTS Fusion Simulation Data

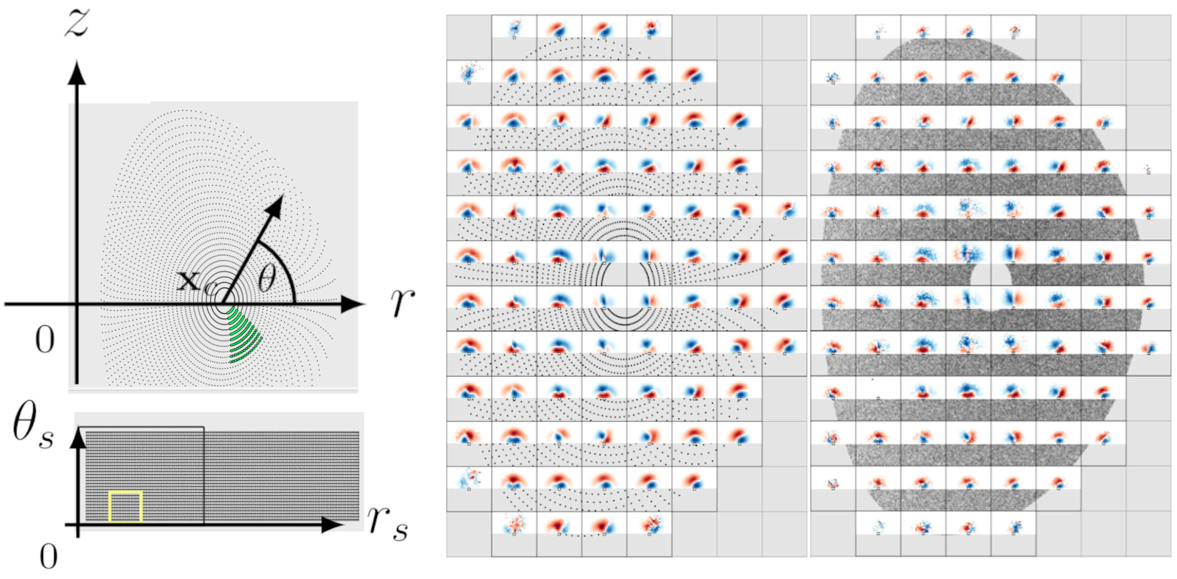


Figure A.2. Rendering of the GTS data. On the upper left, the grid points of the probes where local statistics were gathered from. Below are the same grid points projected onto (r_s, θ_s), which are variables included in the data set that give radial positions and poloidal angles. The function mapping between (r, z) to (r_s, θ_s) had applied a transformation, twisting the grid to concentrate points where the scientists wanted more of them. The green points show a linked selection of the grid points between the lower left and upper left projections. On the right, the histograms associated with the grid points are aggregated based on the overlaid Cartesian grid, and the same is done using the raw particle sub-sample for comparison.

GTS [36] is a gyrokinetic simulation that focuses on studying microturbulence in fusion devices. This simulation dataset contains ~ 500 million particles which were used to generate 3840 weighted histograms, with each histogram summarizing a local physical region based on a coarse grid. This processing was done by our collaborators at PPPL, and the aggregation grid and raw histogram data (one histogram per-grid point) were provided to us. Each histogram has a resolution of 33 by 17 bins. As with the XGC data, the histograms represent perturbations from a background phase-space particle distribution. Each grid point has an associated phase volume where the particles were sampled from that is used for normalization. One frame of data was provided along with a sub-sample of ~ 15 million particles and their weights for comparison.

In addition, a time series of the spatially organized histograms was provided, which includes about 2,000 time steps and several versions of the velocity histograms for each time step, including histograms representing particle density, and histograms where the particles were weighted by momentum, momentum flux, energy, energy flux, particle flux, and bootstrap current.

As with the XGC data, the histograms were computed in the 2D poloidal plane. The same spatial coordinate system is also used, with (r, ζ, z). However, the grid points in this data also each include variables named `radial_position` and `poloidal_angle`, which we denote as (r_s, θ_s). These are mapped from (r, z) based on a transformation that has twisted the grid to concentrate more of the points in salient spatial locations. Each of the grid points also includes a constant representing the static magnetic field strength, B. Renderings illustrating the coordinate systems and data are shown in Figure A.2.

A.2 Alcator C-MOD Data

Alcator C-Mod was a tokamak device developed by MIT and PPPL, which was used as a test platform for ITER. We obtained a small set of simulated Alcator C-Mod particles, which we used as a proof of concept for our visualization approach leveraging interactive grids of spatially organized histograms in Chapter 3.

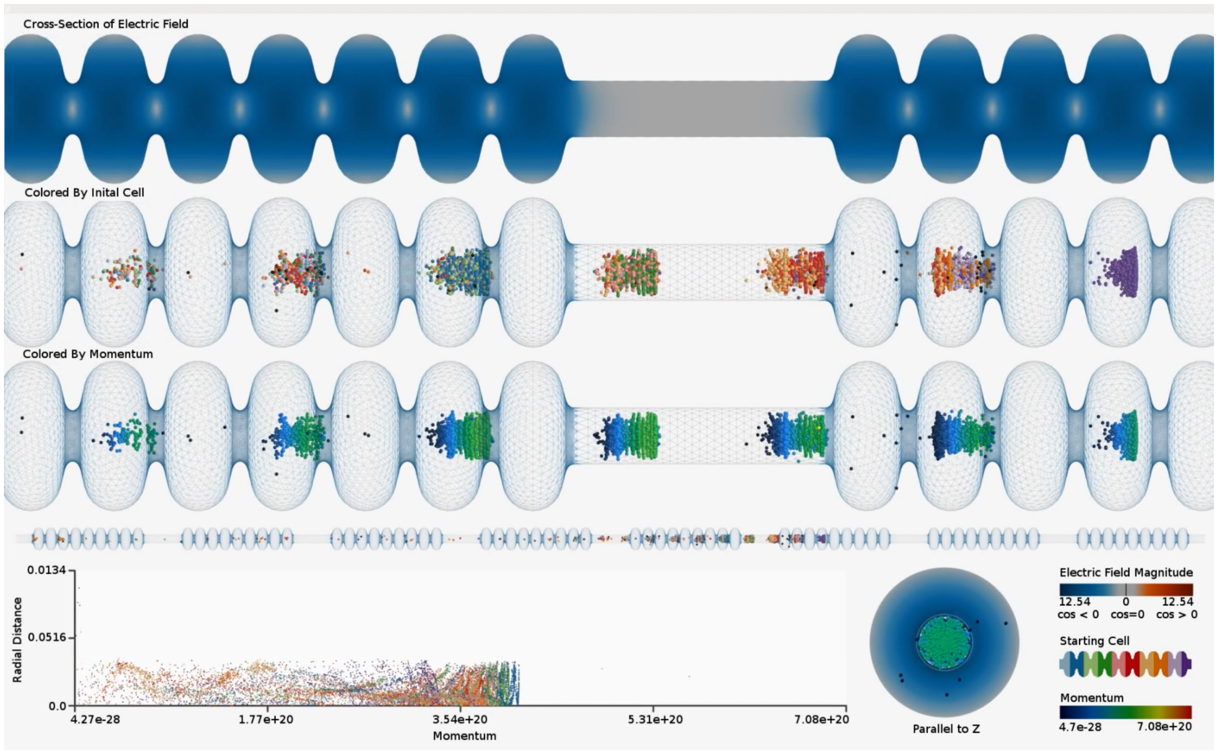


Figure A.3. A rendering of the ACE3P cryomodule dataset. The top is a cross-section of the magnetic field, which oscillates between positive and negative to push/pull the particles through the accelerator. The middle views show the particles colored by their initial position and by their momentum from top to bottom, respectively. The bottom view shows the same particles in momentum vs. radius from the longitudinal axis of the device.

A.2.1 ACE3P Accelerator Simulation Data

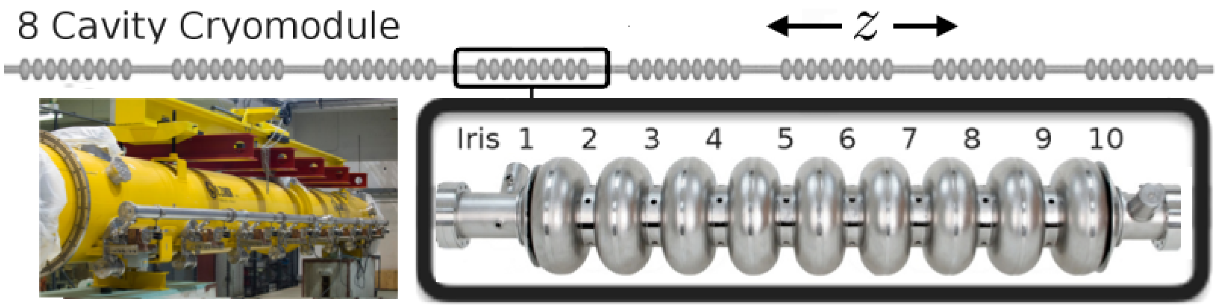


Figure A.4. Top) an ILC 8-Cavity Cryomodule particle accelerator. Bottom Right) a 9-cell cavity (image courtesy of the Linear Collider Collaboration). The irises are the regions between the cells. Bottom left) a full cryomodule assembly (image courtesy of the Linear Collider Collaboration).

ACE3P is a set of simulation codes that model electrodynamics in particle accelerators. The simulated accelerator is made up of a series of cryomodules, which themselves are made up of a series of resonating cavities driven by RF antennas. Electromagnetic field fluctuations accelerate the charged particles contained within. The data we used consists of particles from a simulation for studying dark current and field emissions [127] within ILC cryomodule particle accelerators. The cryomodules are depicted in Figures A.4, and A.3.

The data we used in this dissertation work includes the fluctuating electric field, which oscillates between positive and negative, and particle data. The electric field is represented by a tetrahedral grid. The particle data represents field-emitted electrons that pass through the cavity, and gain high momentum as they are accelerated by the cavities. The particles each have their own start and end time steps as they are emitted at different times and absorbed at different times. Because the particles reach relativistic speeds, their momentum and energy grow larger and larger with a small increase in velocity. The more momentum the field-emitted particles gain, the more dangerous they become. The analysis interests are described in detail by Li Et al. [127].

This data provided us with a good test case for our software from Chapter 6. A challenge to accelerator visualization has been due to their lengths and a lack of support for the visualization of domains with large aspect ratios. It is also used in a case study in Chapter 4.

A.3 S3D Combustion Simulation Data

Data from three S3D simulations are used in this work.

A.3.1 2D Autoignition

The first is a 2D auto-ignition simulation which has been previously used for the study of statistical anomaly detection and its use for detecting autoignition events [156]. The dataset is used in Chapter 7 for illustration of our Voronoi decomposition algorithm.

A.3.1.1 Lifted Ethylene Jet Flame

This dataset is from a petascale combustion simulation developed by scientists at Sandia National Labs [102]. The particular simulation run depicts a 3D highly turbulent lifted ethylene jet flame, which can lead to a better understanding of phenomena such as autoignition in engines. We were provided particle data from this data, which is used in our work in Chapters 3 and 4.

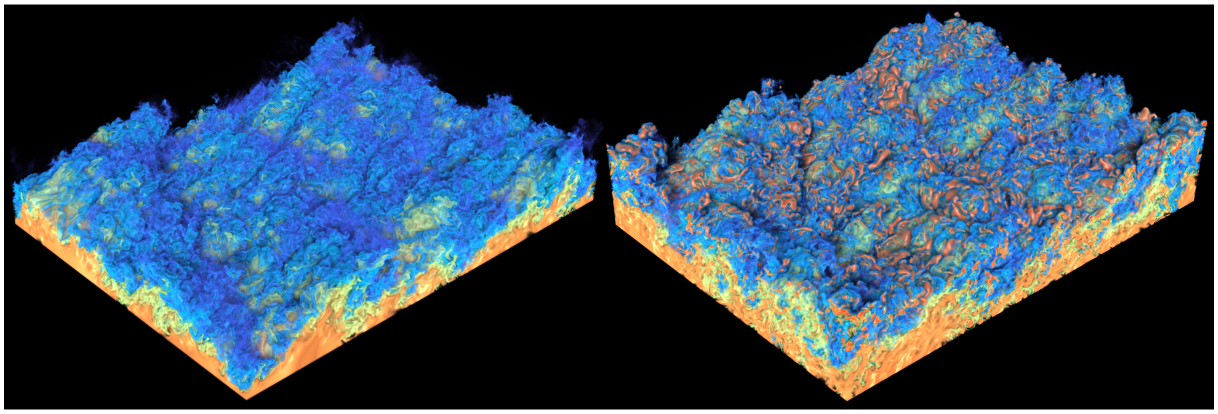


Figure A.5. Renderings of the Hydrogen/Ammonia/Nitrogen data. On the left is the 1 atm case, and on the right is the 10 atm case. The orange structures in the 10 atm case are the cellular flame elements that destabilize the flame.

A.3.2 Hydrogen/Ammonia/Nitrogen Fuel Simulation Data

This data is from a Direct Numerical Simulation (DNS) of a hydrogen/ammonia/nitrogen flame [140]. The DNS study aims to better our understanding of the combustion behavior of hydrogen/ammonia/nitrogen as a carbon-free replacement for conventional fuel (*i.e.*, natural gas). The simulation was run on 900 nodes of the Oak Ridge National Laboratory OLCF Summit cluster requiring approximately 110k node hours. The aggregate amount of data produced comprises 400 checkpoint files, where each checkpoint file is 2TB. Two simulations were produced, each with different pressure parameters. A rendering of the temperature field for the data is shown in Figure A.5.

A.4 Miscellaneous Data

Besides combustion data and plasma data, we also used a channel flow data set and a marine life data set. The marine data is the only data we used that is not simulation data.

A.4.1 Channel Flow Data

The channel flow data represents a simulation of incompressible turbulent channel flows at a high Reynolds number, *Re*. High *Re* wall-bounded turbulence is important in many engineering applications with moving objects at high speeds [141, 142, 143]. This data was used in Chapter 7.

A.4.2 Marine Life Data

The last data we test comes from the Tagging of Pelagic Predators (TOPP) [103] dataset. This is a geospatial movement dataset that tracks the motion of marine life throughout the Pacific Ocean. Datasets like these help us to understand animal behavior in terms of migratory or feeding patterns. This data was used in work from Chapter 3. The dataset is open source and can be accessed at `https://oceanview.pfeg.noaa.gov/erddap/tabledap/gtoppAT.html`.

Appendix B

Software and Media Produced by This Work

The software produced by this work includes three interactive visualization systems: Baleen, Cirrina, and Marrus.

B.1 Baleen

The visualization software described in Chapter 4, which leverages spatially organized histograms, has been named *Baleen*, after the bristled teeth that whales use to filter plankton out of the ocean. This visualization system has two operational modes, one that works with scalar data and another which works with pre-generated grids of spatially organized histograms.

Code Repository	`https://gitlab.com/tneuroth/baleen`
Project Page	`https://tneuroth.gitlab.io/baleen`

B.2 Cirrina

The visualization methods and systems described in Chapters 6 and 5 are incorporated into a software package called *Cirrina*, after the suborder of octopuses, in keeping with the marine theme. This software supports the visualization of distributions of multivariate trajectories.

Code Repository `https://gitlab.com/tneuroth/cirrina`

B.3 Marrus

Marrus is a software package that includes both the tessellation implementation from Chapter 7, and the visualization system based on it to explore multivariate volume data. It is named *Marrus*, after the genus of siphonophores, again in keeping with the marine life theme.

Code Repository `https://gitlab.com/tneuroth/marrus`

B.4 External Media

Videos we've produced illustrating datasets, as well as demoing the visualization systems, are listed:

- A video showing the ammonia/hydrogen/nitrogen flame data:
  `https://tneuroth.gitlab.io/combustion-highlight`
- A video depicting trapped and passing particles in the ITER dataset that highlights a discovery made by our collaborators at PPPL:
  `https://tneuroth.gitlab.io/xgc-highlight`
- A video heat loss to the diverter in the XGC data:
  `https://tneuroth.gitlab.io/xgc-heat-loss-highlight`
- A video illustrating the ACE3P simulation:
  `https://tneuroth.gitlab.io/ace3p-highlight`
- A video demo of the Baleen system that is presented in Chapter 4:
  `https://tneuroth.gitlab.io/baleen-demo`

- A video demo of the Cirrina system that is presented in Chapter 6:
  `https://tneuroth.gitlab.io/cirrina-demo`
- A video demo of the system from Chapter 7:
  `https://tneuroth.gitlab.io/lsrvt-highlight`

References

[1] R. V. Jensen, "Classical chaos," *American Scientist*, vol. 75, no. 2, pp. 168–181, 1987.

[2] M. Lesieur, O. Métais, P. Comte *et al.*, *Large-eddy simulations of turbulence.* Cambridge university press, 2005.

[3] G. V. Trunk, "A problem of dimensionality: A simple example," *IEEE Transactions on pattern analysis and machine intelligence*, no. 3, pp. 306–307, 1979.

[4] D. Donoho and J. Tanner, "Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1906, pp. 4273–4293, 2009.

[5] I. M. Johnstone and D. M. Titterington, "Statistical challenges of high-dimensional data," pp. 4237–4253, 2009.

[6] R. H. Kraichnan, "Kolmogorov's hypotheses and eulerian turbulence theory," *The Physics of Fluids*, vol. 7, no. 11, pp. 1723–1734, 1964. [Online]. Available: https://aip.scitation.org/doi/abs/10.1063/1.2746572

[7] D. Drikakis, M. Frank, and G. Tabor, "Multiscale computational fluid dynamics," *Energies*, vol. 12, no. 17, p. 3272, 2019.

[8] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors *et al.*, "Multiphysics simulations: Challenges and opportunities," *The International Journal of High Performance Computing Applications*, vol. 27, no. 1, pp. 4–83, 2013.

[9] D. H. Bailey, "Extra high speed matrix multiplication on the cray-2," *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 3, pp. 603–607, 1988.

[10] D. Monroe, "Fugaku takes the lead," *Communications of the ACM*, vol. 64, no. 1, pp. 16–18, 2020.

[11] D. B. Kothe, "Science prospects and benefits with exascale computing," *Oak Ridge National Laboratory, Tech. Rep. ORNL/TM-2007/232*, 2007.

[12] C. North, "Toward measuring visualization insight," *IEEE Computer Graphics and Applications*, vol. 26, no. 3, pp. 6–9, 2006.

[13] M. Chen, L. Floridi, and R. Borgo, "What is visualization really for?" in *The Philosophy of Information Quality.* Springer, 2014, pp. 75–93.

[14] J. van Wijk, "The value of visualization," in *VIS 05. IEEE Visualization, 2005.*, 2005, pp. 79–86.

[15] G. Richer, A. Pister, M. Abdelaal, J.-D. Fekete, M. Sedlmair, and D. Weiskopf, "Scalability in visualization," *arXiv preprint arXiv:2210.06562*, 2022.

[16] H. Hauser, "Generalizing focus+ context visualization," in *Scientific visualization: The visual extraction of knowledge from data.* Springer, 2006, pp. 305–327.

[17] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *The craft of information visualization.* Elsevier, 2003, pp. 364–371.

[18] B. E. Rogowitz, L. A. Treinish, and S. Bryson, "How not to lie with visualization," *Computers in physics*, vol. 10, no. 3, pp. 268–273, 1996.

[19] R. M. Shifflin, G. T. Gardnerff, and D. H. Allmeyer, "On the degree of attention and capacity limitations in visual processing," *Perception & Psychophysics*, vol. 14, no. 2, pp. 231–236, 1973.

[20] M. H. W. Rosli and A. Cabrera, "Gestalt principles in multimodal data representation," *IEEE computer graphics and applications*, vol. 35, no. 2, pp. 80–87, 2015.

[21] B. Yost and C. North, "The perceptual scalability of visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 837–844, 2006.

[22] R. Veras and C. Collins, "Discriminability tests for visualization effectiveness and scalability," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 749–758, 2019.

[23] A. E. Kaufman and K. Mueller, "Overview of volume rendering." *The visualization handbook*, vol. 7, pp. 127–174, 2005.

[24] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," *Computers & Graphics*, vol. 30, no. 5, pp. 854–879, 2006.

[25] N. Elmqvist and P. Tsigas, "A taxonomy of 3d occlusion management for visualization," *IEEE transactions on visualization and computer graphics*, vol. 14, no. 5, pp. 1095–1109, 2008.

[26] P. Ljung, J. Krüger, E. Groller, M. Hadwiger, C. D. Hansen, and A. Ynnerman, "State of the art in transfer functions for direct volume rendering," in *Computer Graphics Forum*, vol. 35, no. 3. Wiley Online Library, 2016, pp. 669–691.

[27] U. D. Bordoloi and H.-W. Shen, "View selection for volume rendering," in *VIS 05. IEEE Visualization, 2005.* IEEE, 2005, pp. 487–494.

[28] S. Bruckner and M. E. Groller, "Exploded views for volume data," *IEEE transactions on visualization and computer graphics*, vol. 12, no. 5, pp. 1077–1084, 2006.

[29] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post, "Topology-based flow visualization, the state of the art," *Topology-based methods in visualization*, pp. 1–19, 2007.

[30] J. Kniss, G. Kindlmann, and C. Hansen, "Multidimensional transfer functions for interactive volume rendering," *IEEE Transactions on visualization and computer graphics*, vol. 8, no. 3, pp. 270–285, 2002.

[31] J. Ahrens, B. Geveci, and C. Law, "Paraview: An end-user tool for large data visualization," *The visualization handbook*, vol. 717, 2005.

[32] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. C. Miller, C. Harrison, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rubel, M. Durant, J. M. Favre, and P. Navratil, "VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data," October 2012. [Online]. Available: https://visit.llnl.gov

[33] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[34] S. Li, N. Marsaglia, C. Garth, J. Woodring, J. Clyne, and H. Childs, "Data reduction techniques for simulation, visualization and data analysis," *Computer Graphics Forum*, vol. 37, no. 6, pp. 422–447, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13336

[35] J. M. Dawson, "Particle simulation of plasmas," *Reviews of modern physics*, vol. 55, no. 2, p. 403, 1983.

[36] W. X. Wang, Z. Lin, W. M. Tang, W. W. Lee, S. Ethier, J. L. V. Lewandowski, G. Rewoldt, T. S. Hahm, and J. Manickam, "Gyro-kinetic simulation of global turbulent transport properties in tokamak experiments," *Phys. Plasmas*, vol. 13, no. 19, p. 092505, Oct. 2006.

[37] M. Adams, S.-H. Ku, P. Worley, E. D'Azevedo, J. Cummings, and C.-S. Chang, "Scaling to 150k cores: Recent algorithm and performance engineering developments enabling xgc1 to run at scale," *Journal of Physics: Conference Series.*, vol. 180, no. 1, 2009.

[38] J. H. Chen, A. Choudhary, B. De Supinski, M. DeVries, E. R. Hawkes, S. Klasky, W.-K. Liao, K.-L. Ma, J. Mellor-Crummey, N. Podhorszki *et al.*, "Terascale direct numerical simulations of turbulent combustion using s3d," *Computational Science & Discovery*, vol. 2, no. 1, p. 015001, 2009.

[39] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization.* John Wiley & Sons, 2015.

[40] G. R. Terrell and D. W. Scott, "Variable kernel density estimation," *The Annals of Statistics*, pp. 1236–1265, 1992.

[41] A. Chaudhuri, T. Y. Lee, B. Zhou, C. Wang, T. Xu, H. W. Shen, T. Peterka, and Y. J. Chiang, "Scalable computation of distributions from large scale data sets," in *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, Oct 2012, pp. 113–120.

[42] K. Lu and H. W. Shen, "A compact multivariate histogram representation for query-driven visualization," in *2015 IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV)*, Oct 2015, pp. 49–56.

[43] S. Dutta, C. M. Chen, G. Heinlein, H. W. Shen, and J. P. Chen, "In situ distribution guided analysis and visualization of transonic jet engine simulations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 811–820, Jan 2017.

[44] Y. C. Ye, T. Neuroth, F. Sauer, K. Ma, G. Borghesi, A. Konduri, H. Kolla, and J. Chen, "In situ generated probability distribution functions for interactive post hoc visualization and analysis," in *2016 IEEE 6th Symposium on Large Data Analysis and Visualization (LDAV)*, 2016, pp. 65–74.

[45] S. Dutta, J. Woodring, H.-W. Shen, J.-P. Chen, and J. Ahrens, "Homogeneity guided probabilistic data summaries for analysis and visualization of large-scale data sets," in *2017 IEEE Pacific Visualization*, April 2017.

[46] S. Hazarika, A. Biswas, and H. Shen, "Uncertainty visualization using copula-based analysis in mixed distribution models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 934–943, 2018.

[47] K.-C. Wang, K. Lu, T.-H. Wei, N. Shareef, and H.-W. Shen, "Statistical visualization and analysis of large data using a value-based spatial distribution," in *2017 IEEE Pacific Visualization*, April 2017.

[48] D. Thompson, J. Levine, J. Bennett, P.-T. Bremer, A. Gyulassy, V. Pascucci, and P. Pebay, "Analysis of large-scale scalar data using hixels," in *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, Oct 2011, pp. 23–30.

[49] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth, "A survey of topology-based methods in visualization," in *Computer Graphics Forum*, vol. 35, no. 3. Wiley Online Library, 2016, pp. 643–667.

[50] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas, "Adaptively sampled particle fluids," *ACM Transactions on Graphics*, vol. 26, no. 99, p. 48, Jul. 2007.

[51] J. Woodring, J. Ahrens, J. Figg, J. Wendelberger, S. Habib, and K. Heitmann, "In-situ Sampling of a Large-Scale Particle Simulation for Interactive Visualization and Analysis," *Computer Graphics Forum*, vol. 30, no. 3, pp. 1151–1160, Jun. 2011.

[52] R. Ando, N. Thurey, and R. Tsuruno, "Preserving fluid sheets with adaptively sampled anisotropic particles," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1202–1214, Aug 2012.

[53] Y. Zhang, B. Solenthaler, and R. Pajarola, "Adaptive sampling and rendering of fluids on the gpu." in *Volume Graphics*, 2008, pp. 137–146.

[54] P.-T. Bremer, G. H. Weber, J. Tierny, V. Pascucci, M. S. Day, and J. B. Bell, "A topological framework for the interactive exploration of large scale turbulent combustion," in *2009 Fifth IEEE international conference on e-science.* IEEE, 2009, pp. 247–254.

[55] J. Lukasczyk, C. Garth, G. H. Weber, T. Biedert, R. Maciejewski, and H. Leitte, "Dynamic nested tracking graphs," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 249–258, 2019.

[56] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, and M. Petersen, "An image-based approach to extreme scale in situ visualization and analysis," in *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014, pp. 424–434.

[57] J. Jankowai and I. Hotz, "Feature level-sets: Generalizing iso-surfaces to multi-variate data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 2, pp. 1308–1319, Feb 2020.

[58] Y. Lu and H. H. Zhou, "Statistical and computational guarantees of lloyd's algorithm and its variants," *arXiv preprint arXiv:1612.02099*, 2016.

[59] N. Amenta, M. Bern, and M. Kamvysselis, "A new voronoi-based surface reconstruction algorithm," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998, pp. 415–421.

[60] N. Amenta and M. Bern, "Surface reconstruction by voronoi filtering," *Discrete & Computational Geometry*, vol. 22, no. 4, pp. 481–504, 1999.

[61] N. Amenta, S. Choi, and R. K. Kolluri, "The power crust," in *Proceedings of the sixth ACM symposium on Solid modeling and applications*, 2001, pp. 249–266.

[62] Q. Du and M. Gunzburger, "Grid generation and optimization based on centroidal voronoi tessellations," *Applied mathematics and computation*, vol. 133, no. 2-3, pp. 591–607, 2002.

[63] A. Abdelkader, C. L. Bajaj, M. S. Ebeida, A. H. Mahmoud, S. A. Mitchell, J. D. Owens, and A. A. Rushdi, "Vorocrust: Voronoi meshing without clipping," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 3, pp. 1–16, 2020.

[64] R. M. Kirby, H. Marmanis, and D. H. Laidlaw, “Visualizing multivalued data from 2d incompressible flows using concepts from painting,” in *Proceedings of the Conference on Visualization ’99: Celebrating Ten Years*, ser. VIS ’99. Los Alamitos, CA, USA: IEEE Computer Society Press, 1999, pp. 333–340. [Online]. Available: http://dl.acm.org/citation.cfm?id=319351.319429

[65] H. Obermaier and K. Joy, “Derived metric tensors for flow surface visualization,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 12, pp. 2149–2158, Dec 2012.

[66] J. Helman and L. Hesselink, “Representation and display of vector field topology in fluid flow data sets,” *Computer*, vol. 22, no. 8, pp. 27–36, aug 1989.

[67] C. Tominski, S. Gladisch, U. Kister, R. Dachselt, and H. Schumann, “A Survey on Interactive Lenses in Visualization,” in *EuroVis - STARs*, R. Borgo, R. Maciejewski, and I. Viola, Eds. The Eurographics Association, 2014.

[68] C. Wittenbrink, A. Pang, and S. Lodha, “Glyphs for visualizing uncertainty in vector fields,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 2, no. 3, pp. 266–279, Sep 1996.

[69] M. Hlawatsch, P. Leube, W. Nowak, and D. Weiskopf, “Flow radar glyphs - static visualization of unsteady flow with uncertainty,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 12, pp. 1949–1958, Dec 2011.

[70] N. Ihaddadene and C. Djeraba, “Real-time crowd motion analysis,” in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, Dec 2008, pp. 1–4.

[71] A. Romanoni, M. Matteucci, and D. G. Sorrenti, “Background subtraction by combining temporal and spatio-temporal histograms in the presence of camera movement,” *Machine Vision and Applications*, vol. 25, no. 6, pp. 1573–1584, dec 2014.

[72] N. Dalal, B. Triggs, and C. Schmidi, “Human detection using oriented histograms of flow and appearance,” *Computer Vision - ECCV 2006*, vol. 3952, pp. 428–441, dec 2006.

[73] C. Jung, L. Hennemann, and S. Raupp Musse, “Event detection using trajectory clustering and 4-d histograms,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 11, pp. 1565–1575, Nov 2008.

[74] T. Salzbrunn, C. Garth, G. Scheuermann, and J. Meyer, “Pathline predicates and unsteady flow structures,” *The Visual Computer*, vol. 24, no. 12, pp. 1039–1051, 2008.

[75] K. Shi, H. Theisel, H. Hauser, T. Weinkauf, K. Matkovic, H.-C. Hege, and H.-P. Seidel, “Path line attributes-an information visualization approach to analyzing the

dynamic behavior of 3d time-dependent flow fields," in *Topology-Based Methods in Visualization II*. Springer, 2009, pp. 75–88.

[76] S. Breß, M. Heimel, N. Siegmund, L. Bellatreche, and G. Saake, "Gpu-accelerated database systems: Survey and open challenges," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XV*. Springer, 2014, pp. 1–35.

[77] K. Wu, S. Ahern, E. W. Bethel, J. Chen, H. Childs, E. Cormier-Michel, C. Geddes, J. Gu, H. Hagen, B. Hamann *et al.*, "Fastbit: interactively searching massive data," in *Journal of Physics: Conference Series*, vol. 180, no. 1. IOP Publishing, 2009, p. 012053.

[78] P. Muigg, J. Kehrer, S. Oeltze, H. Piringer, H. Doleisch, B. Preim, and H. Hauser, "A Four-level Focus+Context Approach to Interactive Visual Analysis of Temporal Features in Large Scientific Data." *Comput. Graph. Forum*, vol. 27, no. 3, pp. 775–782, 2008.

[79] H. Doleisch, M. Gasser, and H. Hauser, "Interactive feature specification for focus+ context visualization of complex simulation data," in *VisSym*, vol. 3, 2003, pp. 239–248.

[80] C. Tominski, S. Gladisch, U. Kister, R. Dachselt, and H. Schumann, "Interactive lenses for visualization: An extended survey," in *Computer Graphics Forum*, vol. 36, no. 6. Wiley Online Library, 2017, pp. 173–200.

[81] P. Constantin and C. Foias, *Navier-stokes equations*. University of Chicago Press, 2020.

[82] M. C. Hao, U. Dayal, R. K. Sharma, D. A. Keim, and H. Janetzko, "Visual analytics of large multidimensional data using variable binned scatter plots," in *Visualization and Data Analysis 2010*, vol. 7530. SPIE, 2010, pp. 57–67.

[83] A. Buja, D. Cook, and D. F. Swayne, "Interactive high-dimensional data visualization," *Journal of computational and graphical statistics*, vol. 5, no. 1, pp. 78–99, 1996.

[84] C. Jones, K.-L. Ma, A. Sanderson, and L. R. Myers Jr, "Visual interrogation of gyrokinetic particle simulations," in *Journal of Physics: Conference Series*, vol. 78, no. 1. IOP Publishing, 2007, p. 012033.

[85] A. Leonard, "Energy cascade in large-eddy simulations of turbulent fluid flows," in *Advances in geophysics*. Elsevier, 1975, vol. 18, pp. 237–248.

[86] U. Frisch and A. N. Kolmogorov, *Turbulence: the legacy of AN Kolmogorov*. Cambridge university press, 1995.

[87] J. C. Bennett, V. Krishnamoorthy, S. Liu, R. W. Grout, E. R. Hawkes, J. H. Chen, J. Shepherd, V. Pascucci, and P. T. Bremer, "Feature-based statistical analysis of combustion simulation data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 1822–1831, Dec 2011.

[88] T. Leung, N. Swaminathan, and P. Davidson, "Geometry and interaction of structures in homogeneous isotropic turbulence," *Journal of Fluid Mechanics*, vol. 710, pp. 453–481, 2012.

[89] N. A. K. Doan, N. Swaminathan, and N. Chakraborty, "Multiscale analysis of turbulence-flame interaction in premixed flames," *Proceedings of the Combustion Institute*, vol. 36, no. 2, pp. 1929–1935, 2017.

[90] X. Liu, M. Mishra, M. Skote, and C.-W. Fu, "On visualizing continuous turbulence scales," in *Computer Graphics Forum*, vol. 38, no. 1. Wiley Online Library, 2019, pp. 300–315.

[91] I. Bermejo-Moreno and D. Pullin, "On the non-local geometry of turbulence," *Journal of Fluid Mechanics*, vol. 603, pp. 101–135, 2008.

[92] A. Sanderson, G. Chen, X. Tricoche, D. Pugmire, S. Kruger, and J. Breslau, "Analysis of recurrent patterns in toroidal magnetic fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1431–1440, Nov 2010.

[93] X. Tricoche, C. Garth, and A. Sanderson, "Visualization of topological structures in area-preserving maps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 1765–1774, Dec 2011.

[94] D. Crawford, K.-L. Ma, M.-Y. Huang, S. Klasky, and S. Ethier, "Visualizing gyrokinetic simulations," in *IEEE Visualization 2004*, 2004, pp. 59–66.

[95] C. Jones, K.-L. Ma, S. Ethier, and W.-L. Lee, "An integrated exploration approach to visualizing multivariate particle data," *Computing in Science'I&' Engineering*, vol. 10, no. 4, pp. 20–29, 2008.

[96] J. Kress, D. Pugmire, S. Klasky, and H. Childs, "Visualization and analysis requirements for in situ processing for a large-scale fusion simulation code," in *Proceedings of the 2nd Workshop on In Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization*. IEEE Press, 2016, pp. 45–50.

[97] F. Sauer, Y. Zhang, W. Wang, S. Ethier, and K.-L. Ma, "Visualization techniques for studying large-scale flow fields from fusion simulations," *Computing in Science & Engineering*, vol. 18, no. 2, pp. 68–77, 2016.

[98] T. Neuroth, F. Sauer, W. Wang, S. Ethier, and K.-L. Ma, "Scalable visualization of discrete velocity decompositions using spatially organized histograms," in *2015 IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, 2015, pp. 65–72.

[99] T. Neuroth, F. Sauer, W. Wang, S. Ethier, C. Chang, and K. Ma, "Scalable visualization of time-varying multi-parameter distributions using spatially organized histograms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 12, pp. 2599–2612, Dec 2017.

[100] T. Neuroth and K.-L. Ma, "Interactive spatiotemporal visualization of phase space particle trajectories using distance plots," in *2019 IEEE Pacific Visualization Symposium (PacificVis)*, 2019, pp. 232–236.

[101] T. Neuroth, F. Sauer, and K. Ma, "An interactive visualization system for large sets of phase space trajectories," *Computer Graphics Forum*, vol. 38, no. 3, pp. 297–309, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13690

[102] C. S. Yoo, E. Richardson, R. Sankaran, and J. Chen, "A dns study on the stabilization mechanism of a turbulent lifted ethylene jet flame in highly-heated coflow," *Proceedings of the Combustion Institute*, vol. 33, no. 1, pp. 1619–1627, Oct. 2011.

[103] B. A. Block, I. D. Jonsen, S. J. Jorgensen, A. J. Winship, S. A. Shaffer, S. J. Bograd, E. L. Hazen, D. G. Foley, G. A. Breed, A.-L. Harrison, J. E. Ganong, A. Swithenbank, M. Castleton, H. Dewar, B. R. Mate, G. L. Shillinger, K. M. Schaefer, S. R. Benson, M. J. Weise, R. W. Henry, and D. P. Costa, "Tracking apex marine predator movements in a dynamic ocean," *Nature*, vol. 475, pp. 86–90, Jul. 2011.

[104] T. Neuroth, F. Sauer, W. Wang, S. Ethier, C.-S. Chang, and K.-L. Ma, "Scalable visualization of time-varying multi-parameter distributions using spatially organized histograms," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 12, pp. 2599–2612, 2017.

[105] NVIDIA, "Kepler GK110 whitepaper." [Online]. Available: http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf

[106] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *ACM siggraph computer graphics*, vol. 21, no. 4. ACM, 1987, pp. 163–169.

[107] A. Partow, "C++ mathematical expression toolkit library (exprtk)," 2015. [Online]. Available: https://github.com/ArashPartow/exprtk

[108] O. Kononenko, L. Ge, K. Ko, Z. Li, C.-K. Ng, and L. Xiao, "Progress on the multiphysics capabilities of the parallel electromagnetic ace3p simulation suite," in *Applied Computational Electromagnetics (ACES), International Review of Progress in.* IEEE, 2015, pp. 1–2.

[109] T. Neuroth, F. Sauer, W. Wang, S. Ethier, and K. L. Ma, "Scalable visualization of discrete velocity decompositions using spatially organized histograms," in *Large*

Data Analysis and Visualization (LDAV), 2015 IEEE 5th Symposium on, Oct 2015, pp. 65–72.

[110] N. Marwan, “A historical review of recurrence plots,” *The European Physical Journal-Special Topics*, vol. 164, no. 1, pp. 3–12, 2008.

[111] D. Angus, A. Smith, and J. Wiles, “Conceptual recurrence plots: Revealing patterns in human discourse,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 6, pp. 988–997, June 2012.

[112] Ç. Demiralp, J. Cirimele, J. Heer, and S. K. Card, “The verp explorer: a tool for exploring eye movements of visual-cognitive tasks using recurrence plots,” in *Workshop on Eye Tracking and Visualization.* Springer, 2015, pp. 41–55.

[113] S. Ku, C. Chang, M. Adams, J. Cummings, F. Hinton, D. Keyes, S. Klasky, W. Lee, Z. Lin, S. Parker *et al.*, “Gyrokinetic particle simulation of neoclassical transport in the pedestal/scrape-off region of a tokamak plasma,” in *Journal of Physics: Conference Series*, vol. 46, no. 1. IOP Publishing, 2006, p. 87.

[114] J. Kehrer and H. Hauser, “Visualization and visual analysis of multifaceted scientific data: A survey,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 3, pp. 495–513, March 2013.

[115] S. L. Seyler, A. Kumar, M. F. Thorpe, and O. Beckstein, “Path similarity analysis: a method for quantifying macromolecular pathways,” *PLoS computational biology*, vol. 11, no. 10, p. e1004568, 2015.

[116] T. W. Liao, “Clustering of time series data—a survey,” *Pattern recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[117] T. D. Pham, “Fuzzy recurrence plots,” *EPL (Europhysics Letters)*, vol. 116, no. 5, p. 50008, 2016.

[118] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle, “Recurrence plots of dynamical systems,” *EPL (Europhysics Letters)*, vol. 4, no. 9, p. 973, 1987.

[119] N. Marwan, M. C. Romano, M. Thiel, and J. Kurths, “Recurrence plots for the analysis of complex systems,” *Physics reports*, vol. 438, no. 5, pp. 237–329, 2007.

[120] A. P. Schultz, Y. Zou, N. Marwan, and M. T. Turvey, “Local minima-based recurrence plots for continuous dynamical systems,” *International Journal of Bifurcation and Chaos*, vol. 21, no. 04, pp. 1065–1075, 2011.

[121] N. Marwan, “How to avoid potential pitfalls in recurrence plot based data analysis,” *International Journal of Bifurcation and Chaos*, vol. 21, no. 04, pp. 1003–1017, 2011.

[122] R. Hager and C. Chang, “Gyrokinetic neoclassical study of the bootstrap current in the tokamak edge pedestal with fully non-linear coulomb collisions,” *Physics of Plasmas*, vol. 23, no. 4, p. 042503, 2016.

[123] O. D. Lampe and H. Hauser, “Curve density estimates,” in *Computer Graphics Forum*, vol. 30, no. 3. Wiley Online Library, 2011, pp. 633–642.

[124] R. Scheepens, N. Willems, H. van de Wetering, and J. J. van Wijk, “Interactive visualization of multivariate trajectory data with density maps,” in *2011 IEEE Pacific Visualization Symposium*, March 2011, pp. 147–154.

[125] G. Nicolis, *Foundations of complex systems: Nonlinear dynamic, statistical physics, information and prediction.* World Scientific Publishing Company, 2007.

[126] D. Faghihi, V. Carey, C. Michoski, R. Hager, S. Janhunen, C.-S. Chang, and R. Moser, “Moment preserving constrained resampling with applications to particle-in-cell methods,” *arXiv preprint arXiv:1702.05198*, 2017.

[127] K. Liu, Y. Li, A. Palczewski, and R. Geng, “Research on field emission and dark current in ilc cavities,” Thomas Jefferson National Accelerator Facility, Newport News, VA (United States), Tech. Rep., 2013.

[128] S. Ozer, D. Silver, K. Bemis, and P. Martin, “Activity detection in scientific visualization,” *IEEE transactions on visualization and computer graphics*, vol. 20, no. 3, pp. 377–390, 2014.

[129] T. Neuroth, M. Rieth, K. Aditya, M. Lee, J. H. Chen, and K.-L. Ma, “Level set restricted voronoi tessellation for large scale spatial statistical analysis,” *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–11, 2022.

[130] G. Rong, Y. Liu, W. Wang, X. Yin, D. Gu, and X. Guo, “Gpu-assisted computation of centroidal voronoi tessellation,” *IEEE transactions on visualization and computer graphics*, vol. 17, no. 3, pp. 345–356, 2010.

[131] D.-M. Yan, W. Wang, B. Lévy, and Y. Liu, “Efficient computation of 3d clipped voronoi diagram,” in *International Conference on Geometric Modeling and Processing.* Springer, 2010, pp. 269–282.

[132] D. P. Starinshak, J. Owen, and J. Johnson, “A new parallel algorithm for constructing voronoi tessellations from distributed input data,” *Computer physics communications*, vol. 185, no. 12, pp. 3204–3214, 2014.

[133] Q. Du, V. Faber, and M. Gunzburger, “Centroidal voronoi tessellations: Applications and algorithms,” *SIAM review*, vol. 41, no. 4, pp. 637–676, 1999.

[134] N. Bell and J. Hoberock, “Thrust: A productivity-oriented library for cuda,” in *GPU computing gems Jade edition.* Elsevier, 2012, pp. 359–371.

[135] J. Nickolls, I. Buck, M. Garland, and K. Skadron, “Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for?” *Queue*, vol. 6, no. 2, pp. 40–53, 2008.

[136] K. R. Sreenivasan and R. Antonia, "The phenomenology of small-scale turbulence," *Annual review of fluid mechanics*, vol. 29, no. 1, pp. 435–472, 1997.

[137] J. C. Bennett, V. Krishnamoorthy, S. Liu, R. W. Grout, E. R. Hawkes, J. H. Chen, J. Shepherd, V. Pascucci, and P.-T. Bremer, "Feature-based statistical analysis of combustion simulation data," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 12, pp. 1822–1831, 2011.

[138] K. Aditya, H. Kolla, W. P. Kegelmeyer, T. M. Shead, J. Ling, and W. L. Davis, "Anomaly detection in scientific data using joint statistical moments," *Journal of Computational Physics*, vol. 387, pp. 522–538, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002199911930172X

[139] H. Kolla, K. Aditya, and J. H. Chen, *Higher Order Tensors for DNS Data Analysis and Compression*. Cham: Springer International Publishing, 2020, pp. 109–134. [Online]. Available: https://doi.org/10.1007/978-3-030-44718-2_6

[140] S. Wiseman, M. Rieth, A. Gruber, J. R. Dawson, and J. H. Chen, "A comparison of the blow-out behavior of turbulent premixed ammonia/hydrogen/nitrogen-air and methane–air flames," *Proceedings of the Combustion Institute*, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1540748920301887

[141] M. Lee, N. Malaya, and R. D. Moser, "Petascale direct numerical simulation of turbulent channel flow on up to 786k cores," in *SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2013, pp. 1–11.

[142] M. Lee, R. Ulerich, N. Malaya, and R. D. Moser, "Experiences from leadership computing in simulations of turbulent fluid flows," *Computing in Science and Engineering*, vol. 16, pp. 24–31, 2014.

[143] M. Lee and R. D. Moser, "Direct numerical simulation of turbulent channel flow up to reτ = 5200," *Journal of Fluid Mechanics*, vol. 774, pp. 395–415, 7 2015. [Online]. Available: http://www.journals.cambridge.org/abstract_S0022112015002682

[144] J. Jeong and F. Hussain, "On the identification of a vortex," *Journal of fluid mechanics*, vol. 285, pp. 69–94, 1995. [Online]. Available: http://journals.cambridge.org/abstract_S0022112095000462

[145] A. Lozano-Durán and J. Jiménez, "Time-resolved evolution of coherent structures in turbulent channels: characterization of eddies and cascades," *Journal of Fluid Mechanics*, vol. 759, pp. 432–471, 10 2014. [Online]. Available: http://www.journals.cambridge.org/abstract_S0022112014005758

[146] Q. Du, M. Emelianenko, and L. Ju, "Convergence of the lloyd algorithm for computing centroidal voronoi tessellations," *SIAM journal on numerical analysis*, vol. 44, no. 1, pp. 102–119, 2006.

[147] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang, "On centroidal voronoi tessellation—energy smoothness and fast computation," *ACM Transactions on Graphics (ToG)*, vol. 28, no. 4, pp. 1–17, 2009.

[148] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

[149] A. Tikhonova, C. D. Correa, and K.-L. Ma, "Explorable images for visualizing volume data." *PacificVis*, vol. 10, pp. 177–184, 2010.

[150] S. Greenberg and B. Buxton, "Usability evaluation considered harmful (some of the time)," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2008, pp. 111–120.

[151] Y. Ye, F. Sauer, K.-L. Ma, K. Aditya, and J. Chen, "A user-centered design study in scientific visualization targeting domain experts," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 6, pp. 2192–2203, 2020.

[152] T. Wei, C. Chen, J. Woodring, H. Zhang, and H. Shen, "Efficient distribution-based feature search in multi-field datasets," in *2017 IEEE Pacific Visualization*, April 2017.

[153] Q. Wu, T. Neuroth, O. Igouchkine, K. Aditya, J. H. Chen, and K.-L. Ma, "Diva: A declarative and reactive language for in situ visualization," in *2020 IEEE 10th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, 2020, pp. 1–11.

[154] R. Aymar, P. Barabaschi, and Y. Shimomura, "The iter design," *Plasma physics and controlled fusion*, vol. 44, no. 5, p. 519, 2002.

[155] C. S. Chang, S. Ku, A. Loarte, V. Parail, F. Koechl, M. Romanelli, R. Maingi, J.-W. Ahn, T. Gray, J. Hughes *et al.*, "Gyrokinetic projection of the divertor heat-flux width from present tokamaks to iter," *Nuclear Fusion*, vol. 57, no. 11, p. 116023, 2017.

[156] K. Aditya, H. Kolla, W. P. Kegelmeyer, T. M. Shead, J. Ling, and W. L. Davis, "Anomaly detection in scientific data using joint statistical moments," *Journal of Computational Physics*, vol. 387, pp. 522–538, Jun. 2019.