Visuelle Analyse großer Partikeldaten

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Tobias Rapp

aus Villingen-Schwenningen

Tag der mündlichen Prüfung:	16. April 2021
Erster Gutachter:	Prof. Dr.-Ing. Carsten Dachsbacher
Zweiter Gutachter:	Prof. Dr. Rüdiger Westermann

KURZFASSSUNG

Partikelsimulationen sind eine bewährte und weit verbreitete numerische Methode in der Forschung und Technik. Beispielsweise werden Partikelsimulationen zur Erforschung der Kraftstoffzerstäubung in Flugzeugturbinen eingesetzt. Auch die Entstehung des Universums wird durch die Simulation von dunkler Materiepartikeln untersucht. Die hierbei produzierten Datenmengen sind immens. So enthalten aktuelle Simulationen Billionen von Partikeln, die sich über die Zeit bewegen und miteinander interagieren.

Die Visualisierung bietet ein großes Potenzial zur Exploration, Validation und Analyse wissenschaftlicher Datensätze sowie der zugrundeliegenden Modelle. Allerdings liegt der Fokus meist auf strukturierten Daten mit einer regulären Topologie. Im Gegensatz hierzu bewegen sich Partikel frei durch Raum und Zeit. Diese Betrachtungsweise ist aus der Physik als das lagrange Bezugssystem bekannt. Zwar können Partikel aus dem lagrangen in ein reguläres eulersches Bezugssystem, wie beispielsweise in ein uniformes Gitter, konvertiert werden. Dies ist bei einer großen Menge an Partikeln jedoch mit einem erheblichen Aufwand verbunden. Darüber hinaus führt diese Konversion meist zu einem Verlust der Präzision bei gleichzeitig erhöhtem Speicherverbrauch. Im Rahmen dieser Dissertation werde ich neue Visualisierungstechniken erforschen, welche speziell auf der lagrangen Sichtweise basieren. Diese ermöglichen eine effiziente und effektive visuelle Analyse großer Partikeldaten.

Lagrange koheränte Strukturen

Im ersten Teil meiner Dissertation erforsche ich neue Methoden zur Analyse des zeitabhängigen Verhaltens von Partikeln. Dies erfolgt basierend auf der Betrachtung der Partikeltrajektorieren über die Zeit als Lösung eines dynamischen Systems, d.h. einer gewöhnlichen Differentialgleichung. Die Visualisierung solcher dynamischen Systeme ist ein aktuelles Forschungsthema mit vielfältigen Anwendungsgebieten, wie beispielsweise zur Visualisierung von Strömungen in einer Brennkammer oder zur Wettervorhersage. Hierbei ist die Theorie der lagrange koheränten Strukturen eine etablierte Methodik zur Identifikation einer robusten Topologie innerhalb eines begrenzten Zeitintervalls. Allerdings ist diese Charakterisierung einer Strömung aufwendig, da eine große Zahl an Trajektorien numerisch integriert werden müssen. Weiterhin gibt es mehrere Parameter, die Anwenderinnen und Anwender explorieren möchten. Im Rahmen meiner Dissertation stelle ich einen effizienten Algorithmus vor, der speziell für Partikeldaten konzipiert ist. Da die Partikel bereits im lagrangen Bezugsystem gegeben sind, ist keine teure numerische Integration erforderlich. Stattdessen kann ich bereits existierende Trajektorien interpolieren und die räumliche Deformation benachbarter Trajektorien approximieren. Mit dem von mir entwickelten GPU beschleunigten Prototypen ist es erstmals

möglich, die lagrange kohärenten Strukturen von Millionen von Partikeln interaktiv zu identifizieren.

Stochastische Strömungen

Ein weiterer Forschungsgegenstand meiner Arbeit sind Strömungen mit Unsicherheiten, welche ich als stochastische Differenzialgleichungen modelliere. Diese entstehen beispielwiese aus der mehrfachen Ausführung einer Simulation, einer Messung oder einem expliziten Fehlermodell. Zur Visualisierung dieser stochastischen Strömungen, identifiziere ich stochastische Transportbarrieren und -verstärker ähnlich zu den lagrange koheränten Strukturen. Mit diesem Ansatz wird eine teure Monte Carlo-Integration von stochastischen Differentialgleichungen vermieden.

Interaktive visuelle Analyse von Partikelströmungen

Zur Analyse von Partikeln sowie deren zeitabhängigen Verhaltens und physikalischer Attribute habe ich eine interaktive visuelle Analyseumgebung entwickelt. Dieser Ansatz ermöglicht es, Benutzerinteraktion, Visualisierung und automatische Analysen zu verbinden. Auf diese Weise können komplexe Zusammenhänge und Prozesse besser verstanden werden. Diese Methodik habe ich mit mehreren Domänenexperten aus dem Bereich der Strömungsmechanik evaluiert. Bei der Analyse der Kraftstoffzerstäubung in der Einspritzdüse einer Flugzeugturbine konnte eine veränderte Geometrie entwickelt werden, welche zu verringerten CO_2 Emissionen führt [54]. Insbesondere die Berechnung und Visualisierung der lagrange kohärenten Strukturen war hilfreich, die Entstehung und Reduktion von Wirbeln und dem dadurch beinflussten Transport des Kraftstoffes zu verstehen. Angesichts der wachsenden Datenmengen neuer Simulationen stellen große Partikeldaten jedoch weiterhin ein Problem dar. Dies betrifft nicht nur die Komplexität der Berechnungen, sondern auch die menschliche Wahrnehmung. Die effektive visuelle Analyse großer Datenmengen benötigt folglich eine geeignete Abstraktion und Datenreduktion.

Datenreduktion mittels Sampling

Im zweiten Teil meiner Dissertation erforsche ich die Reduktion von Partikeldaten zur interaktiven Visualisierung. Die erste Technik die ich vorstelle verwendet stochastisches Sampling, um einen Datensatz auf eine repräsentative Menge an Partikeln zu reduzieren. Stratifikation erlange ich durch einen gierigen Algorithmus, der die räumliche Verteilung der Partikel bzw. der Trajektorien in Raumzeit optimiert. Hierbei entwickle ich einen parallelen Algorithmus, der zur Ausführung auf einer GPU geeignet ist und damit die Optimierung großer Datenmengen ermöglicht. Diese Optimierung lässt sich auf nicht uniforme Wahrscheinlichkeiten erweitern, welche sich beispielsweise aus der lokalen Informationskomplexität einer Datendimension definieren. Abschließend präsentiere ich eine effiziente Methode zur Wahl einer geeigneten Detailstufe während der interaktiven Visualisierung.

Eine probabilistische Datenrepräsentation zur interaktiven visuellen Analyse

Zur visuellen Analyse von Milliarden von Partikeln entwerfe ich eine probabilistische Datenrepräsentation. Diese erlaubt die interaktive Exploration und Navigation, führt aber auch zu einer sehr starken Datenreduktion und einem damit einhergehenden Präzisionsverlust. Die Repräsentation basiert auf der Modellierung der Datendimensionen durch Gaussian Mixture Models. Hierbei kann ich höherdimensionale Modelle vermeiden, da für die visuelle Analyse die paarweise Kombination von Marginvalverteilungen ausreichend ist. Zur Visualisierung dieser Datenrepräsentation verwende ich direkt die Dichteverteilungen der einzelnen Gauß-Komponenten und vermeide somit aufwändiges Sampling. Insbesondere leite ich eine analytische Lösung her, um beliebige drei-dimensionale Gauß-Verteilungen perspektivisch korrekt auf ein zwei-dimensionales Bild zu projezieren. Somit können auch mehrere Millionen dieser Gauß-Komponenten interaktiv dargestellt werden.

Bildbasierte Volumenvisualisierung mittels Momenten

Zuletzt präsentiere ich eine bildbasierte Datenrepräsentation zur interaktiven Volumenvisualisierung von großen Partikeldaten. Diese bildbasierte Datenrepräsentation wird mit einer im Voraus festgelegten Kamerakonfiguration erzeugt und modelliert das Signal entlang aller Sichtstrahlen. Mittels verschiedener Transferfunktionen können die Daten daraufhin interaktiv exploriert werden. Hierbei wird das Signal in jedem Pixel in die Fourier Basis transformiert und Fourier Koeffizienten eines beschränkten Signals, sogenannte beschränkte trigonometrische Momente, werden bestimmt. Mit dem Ziel diese bildbasierte Repräsentation kompakt zu halten, bestimme ich die Anzahl der Momente adaptiv und stelle eine neue Kodierungs- und Quantisierungsstrategie vor.

ABSTRACT

Particle simulations are an established computational method used in science and engineering. For example, particle simulations are employed to investigate the injection of fuel in turbines or to study the evolution of the universe by simulating the interaction of dark matter particles. The data that is produced is enormous, with state-of-the-art simulations tracking trillions of particles that move and interact over time.

Scientific visualization plays an important role in the exploration, validation, and analysis of these datasets and their underlying computational models. A significant amount of research has been focused on structured data with a regular topology. This is in contrast to the scattered nature of particle data. In physics, the reference frame of freely moving particles is known as the Lagrangian frame. Although it is possible to convert data from the Lagrangian to a regular Eulerian frame, this requires interpolating massive amounts of particles. This is not only prohibitively expensive, but disregards the inherent nature of particles moving freely through space and time for subsequent analysis. In this thesis, we develop visualization techniques specific to the Lagrangian perspective, which allows us to find novel and efficient ways of visualizing and analyzing large amounts of particles.

Lagrangian Coherent Structures

In the first part of the thesis, we develop novel methods to visually analyze and explore time-dependent particle dynamics. To this end, we regard the trajectories of particles over time as a solution of a dynamical system, i. e. a differential equation. The visualization of dynamical systems is an active research area with various applications, for example to visualize fluid flows in a combusion engine or to predict the weather. Most notably, the theory of Lagrangian coherent structures, developed in the field of dynamical systems, has been well-established to identify and visualize a robust topology of finite-time flow behavior. This characterization of the flow is computationally demanding whilst at the same time depends on several parameters that have to be explored by domain scientists. We develop an efficient algorithm that runs interactively even for millions of particles. This is due the Lagrangian nature of our data, which allows us to avoid the expensive numerical integration. Instead, we interpolate the existing trajectories and approximate the spatial deformation of neighboring trajectories over time. With our GPU accelerated prototype, we can thus compute the Lagrangian coherent structures interactively for millions of particles.

Stochastic Flows

Furthermore, we study the dynamics of uncertain flows, stemming from multiple simulation runs, repeated measurements, or from explicit error models. To

visualize such stochastic flows, we identify stochastic transport barriers and enhancers similar to the Lagrangian coherent structures. This approach avoids expensive Monte Carlo integration of the stochastic flow and only advects the deterministic part of the flow.

Interactive Visual Analysis of Particle-based Flows

To visualize particles, their dynamics, and their associated physical attributes, we employ an interactive visual analysis approach. This combines user interaction, visualization, and automatic analysis to gain insight into the complex data and processes. With this approach, domain scientists studying the atomization of fuel in jet turbine engines were able to develop a new spray nozzle geometry with reduced emissions. Most notable, the identified coherent structures were considered very helpful in visualizing the time-dependent flow behavior. However, the large amount of particles pose a significant challenge to this interactive process. This is not only a computational issue since human perception is also quickly overloaded by displaying too much visual information at once. The interactive visual analysis thus requires suitable abstractions and data reduction to create efficient and clutter free visualizations.

Data Reduction by Sampling

The data reduction of particle data is investigated in the second part of the thesis. To drastically reduce massive amounts of particles for interactive exploration, we propose a probabilistic approach. More specifically, we present a data reduction technique based on statistical sampling to select only a subset of representative particles. Since our data is always defined in a spatiotemporal domain, we develop a sampling strategy that finds well distributed samples in space-time. This stratification stems from a greedy algorithm that optimizes the distribution of particles or particles trajectories. We further develop a parallel version of this algorithm that is suitable for GPU acceleration and thus enables the application to large datasets. In addition, we consider non-uniform probabilities based on the information in the multivariate value dimensions and select an appropriate subset of samples interactively during visual analysis.

Probabilistic Data Representation

We propose a probabilistic data representation to interactively analyze and explore massive particle datasets, containing over billions of particles. The compact data representation enables the interactive exploration and navigation, but also introduces uncertainty that has to be conveyed. Our approach is based on modeling the particle dimensions using Gaussian mixture models. We discuss the extension of several visualization techniques to directly employ this representation instead of resorting to sampling. Among others, we find an analytic solution for the perspective projection of general three-dimensional Gaussian components to an two-dimensional image.

Lastly, we present a novel image-based representation to interactively visualize large and arbitrarily structured volumetric data. This image-based representation is created from a fixed view and models the scalar densities along each viewing ray. Then, different transfer functions can be explored interactively. In detail, we transform the density in each pixel to the Fourier basis and store Fourier coefficients of a bounded signal, i.e. bounded trigonometric moments. To keep this image-based representation compact, we adaptively determine the number of moments in each pixel and present a novel coding and quantization strategy.

PUBLICATIONS

[1] T. F. Dauch, C. Ates, T. Rapp, M. C. Keller, G. Chaussonnet, J. Kaden, M. Okraschevski, R. Koch, C. Dachsbacher, and H.-J. Bauer. "Analyzing the Interaction of Vortex and Gas–Liquid Interface Dynamics in Fuel Spray Nozzles by Means of Lagrangian-Coherent Structures (2D)." In: *Energies* (2019). ISSN: 1996-1073. DOI: 10.3390/en12132552.

[2] T. F. Dauch, T. Rapp, G. Chaussonnet, S. Braun, M. C. Keller, J. Kaden, R. Koch, C. Dachsbacher, and H.-J. Bauer. "Highly Efficient Computation of Finite-Time Lyapunov Exponents (FTLE) on GPUs Based on Three-Dimensional SPH Datasets." In: *Computers & Fluids* (2018). ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2018.07.015.

[3] M. Piochowiak, T. Rapp, and C. Dachsbacher. "Stochastic Volume Rendering of Multi-Phase SPH Data." In: *Computer Graphics Forum* 40.1 (2021), pp. 97–109. DOI: 10.1111/cgf.14121.

[4] T. Rapp and C. Dachsbacher. "Visualizing Transport and Mixing in Particle-based Fluid Flows." In: *Vision, Modeling and Visualization*. 2019. ISBN: 978-3-03868-098-7. DOI: 10.2312/vmv.20191330.

[5] T. Rapp and C. Dachsbacher. "Uncertain Transport in Unsteady Flows." In: *Proceedings of IEEE Visualization*. 2020, pp. 16–20. DOI: 10.1109/VIS47514.2020.00010.

[6] T. Rapp, C. Peters, and C. Dachsbacher. "Image-based Visualization of Large Volumetric Data Using Moments." Submitted to IEEE Transactions on Visualization and Computer Graphics. 2020.

[7] T. Rapp, C. Peters, and C. Dachsbacher. "Void-and-Cluster Sampling of Large Scattered Data and Trajectories." In: *IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE Visualization 2019)* 26.1 (2020), pp. 780–789. ISSN: 077-2626. DOI: 10.1109/TVCG.2019.2934335.

[8] T. Rapp, C. Peters, and C. Dachsbacher. "Visual Analysis of Large Multivariate Scattered Data using Clustering and Probabilistic Summaries." In: *IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE Visualization 2020)* 27.2 (2021), pp. 1580–1590. ISSN: 1941-0506. DOI: 10.1109/TVCG.2020.3030379.

[9] G. Simons, S. Herholz, V. Petitjean, T. Rapp, M. Ament, H. Lensch, C. Dachsbacher, M. Eisemann, and E. Eisemann. "Applying Visual Analytics to Physically Based Rendering." In: *Computer Graphics Forum* 38.1 (2019), pp. 197–208. DOI: 10.1111/cgf.13452.

[10] M. Zeidan, T. Rapp, C. Peters, and C. Dachsbacher. "Moment-Based Opacity Optimization." In: *Eurographics Symposium on Parallel Graphics and Visualization*. 2020. ISBN: 978-3-03868-107-6. DOI: 10.2312/pgv.20201072.

Visualization gives you answers to questions
you didn't know you had.

— Ben Shneiderman

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Dr.-Ing. Carsten Dachsbacher. Carsten, joining your group in pursuit of a PhD was one of the best decisions I have made. Thank you for all of the support and advice that is reflected throughout this thesis.

Furthermore, I want to thank Dr. Christoph Peters for his advice and guidance. Christoph, thank you for making me a better scientist. This thesis would have taken a lot longer without you.

Thank you Dr. Johannes Schudeiske, Tobias Zirr, Christoph Schied, Dr. Florian Reibold, Lorenzo Tessari, Daniel Opitz, Max Piochowiak, Diana Kheil and all others of my current and former colleagues at the computer graphics group at the Karlsruhe Institute of Technology.

For going with me on a fantastic journey through computer science, I want to acknowledge, in no particular order, Daniel Hassler, Pierre Barbera, Jannik Quehl, Sebastian Krach, Michael Hauber, Michael Jakobi and everyone else that we have met on our way.

Finally, I wish to thank my parents, Günter and Christa, and my sister Stefanie for their love and support. Last but not least, I want to thank Fabienne Heinzler for supporting me throughout this thesis. Without you, all of this would not have been possible.

CONTENTS

1 INTRODUCTION 1
1.1 Motivation . . . 1
1.2 Contributions . . . 3
2 VISUALIZING PARTICLE DATA 5
2.1 Particle Data . . . 5
2.2 Visualizing Spatiotemporal Data . . . 7
2.3 Visualizing Multivariate Data . . . 13
2.4 Interactive Visual Analysis . . . 17
3 DATA REDUCTION FOR PARTICLE-BASED VISUALIZATION 19
3.1 Particle Compression . . . 19
3.2 Probabilistic Data Modeling . . . 20
3.3 Data Reduction by Sampling . . . 25
4 FLOW VISUALIZATION 27
4.1 Fluid Dynamics . . . 27
4.2 Smoothed Particle Hydrodynamics . . . 29
4.3 Flow Visualization . . . 30
4.4 Uncertain Flow Visualization . . . 35

I LAGRANGIAN FLOW VISUALIZATION
5 LAGRANGIAN COHERENT STRUCTURES 41
5.1 Finite-Time Lyapunov Exponent . . . 41
5.2 Identifying Lagrangian Coherent Structures . . . 44
5.3 Numerical Experiments . . . 46
5.4 Discussion . . . 51
6 VISUAL ANALYSIS OF TRANSPORT AND MIXING 53
6.1 Visual Analysis Framework . . . 54
6.2 Visualizing the Topology of Time-Dependent Flows . . . 55
6.3 Visualizing Mixing in Multiphase Fluid Flows . . . 55
6.4 Case Studies . . . 55
6.5 Discussion and Domain Expert Feedback . . . 58
7 UNCERTAIN TRANSPORT 59
7.1 Stochastic Flows . . . 60
7.2 Stochastic Transport Barriers and Enhancers . . . 60
7.3 Modeling Diffusion . . . 61
7.4 Visualizing Transport Uncertainty . . . 62
7.5 Numerical Experiments . . . 62
7.6 Discussion . . . 69

II DATA REDUCTION FOR VISUAL ANALYSIS
8 VOID-AND-CLUSTER SAMPLING 73
8.1 The Void-And-Cluster Technique . . . 73
8.2 Void-and-Cluster Sampling for Particle Data . . . 74
8.3 Parallel Implementation . . . 79

8.4 Local Error Measure . . . 80
8.5 Evaluation . . . 82
8.6 Future Work: Multi-Node Parallelism . . . 91
8.7 Discussion . . . 91
9 PROBABILISTIC SUMMARIES 93
9.1 Probabilistic Summaries . . . 94
9.2 Spatial Visualization . . . 97
9.3 Visual Analysis . . . 99
9.4 Evaluation . . . 103
9.5 Discussion . . . 113
10 IMAGE-BASED VOLUME VISUALIZATION USING MOMENTS 115
10.1 Using Moments to Reconstruct Bounded Densities . . . 116
10.2 Moments of Ray Densities . . . 118
10.3 Interactive Rendering . . . 119
10.4 Relations Between Moments . . . 120
10.5 Determining the Number of Moments . . . 121
10.6 Compression and Quantization . . . 122
10.7 Uncertainty Quantification . . . 125
10.8 Single Scattering . . . 126
10.9 View Projection . . . 126
10.10 Evaluation . . . 127
10.11 Discussion . . . 133
10.12 Future Work . . . 134
11 CONCLUSION 135

III APPENDIX

A APPENDIX 139
A.1 Void-and-Cluster Sampling Algorithm . . . 139
A.2 Indexing a Lower Tridiagonal Matrix . . . 139
A.3 3D Gaussian Ray Integration . . . 141

BIBLIOGRAPHY 143

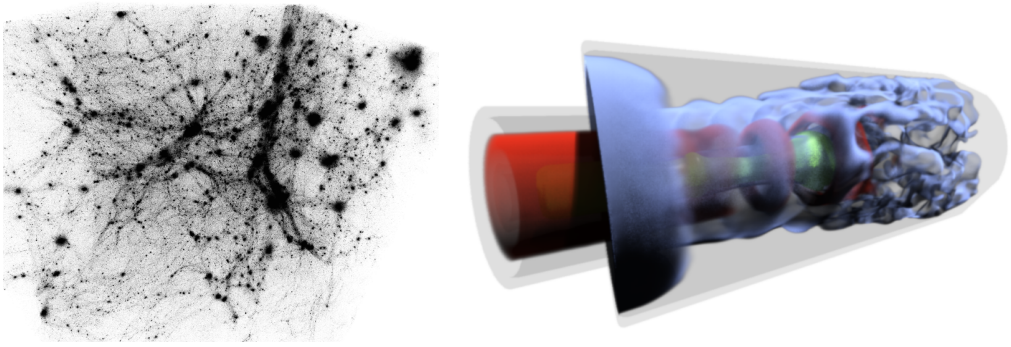
1 INTRODUCTION

We first motivate the goals and challenges addressed in this thesis by introducing several application domains and the data that they produce. We detail how visualization lets scientists and engineers explore, validate, formulate models and hypotheses, or communicate their work. Ultimately, our aim is to support scientists and engineers in gaining insight into large particle data from a Lagrangian perspective by combining visual and computational methods. Lastly, we give an overview of this thesis and outline the novel contributions.

1.1 MOTIVATION

In the context of this thesis, we define particles as scattered, or unstructured, data. They move freely through both space and time. In addition, each particle might contain one or more physical quantities, such as temperature, density, or pressure. This type of data is most prominently produced in cosmological simulations, e. g. stemming from N-body or smoothed particle hydrodynamics (SPH) simulations. For example, cosmologists study the evolution of the large-scale universe by simulating the interaction of dark matter particles, see Figure 1.1 (a). Since the introduction of SPH [88, 199] in 1977, it has been adopted by researchers and engineers to model and study more general fluid flows. In the example shown in Figure 1.1 (b), engineers simulate the injection of fuel inside a jet turbine engine to reduce CO_2 emissions. Here, fuel and gasses are represented by particles. Another source of particle data are molecular dynamics simulations that compute the physical movements of atoms and molecules. In all of these cases, the study of movement and interaction, or the dynamics, of particles is the central focus in science and engineering.

The study of flow dynamics has a long history in science, due to its manifold applications in diverse fields such as aeronautics, industrial engineering, medicine, and astrophysics. Since its inception in the 20^{th} century, the interdisciplinary science of non-linear dynamical systems has greatly contributed to the understanding of complex flow behavior. Although slight changes of the initial conditions of a system can lead to a radically different behavior [198], there are still patterns in seemingly chaotic flows. The topological analysis of flow dynamics is a promising approach for effective flow visualization. Although the topology of time-independent flows is well understood [126], the topological segmentation of time-dependent flows is still an active area of research [35]. To visualize the dynamics of time-dependent flows, the Lagrangian frame of reference is considered promising [110]. In this reference frame, the observer follows the motion of particles through space and time. Whilst most research is focused on structured data that first needs to be converted into the Lagrangian frame, it is the natural frame for particle data. Lagrangian



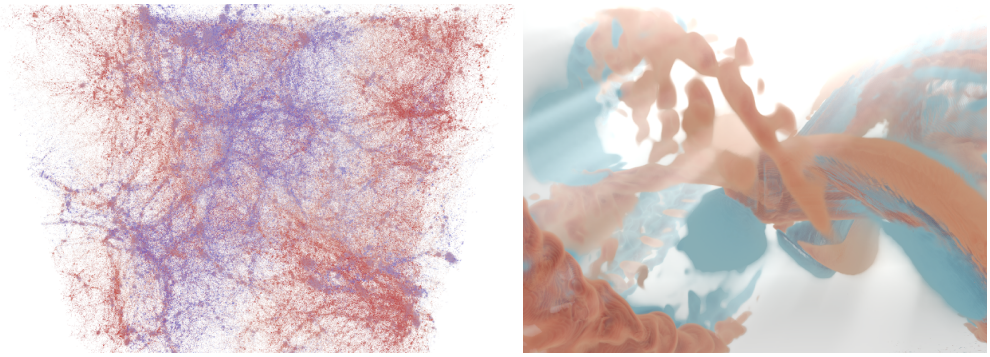
(a) The Dark Sky simulation [293] (b) Simulation of a fuel spray nozzle [46]

Figure 1.1: Visualization of large, scattered particle data in a cosmological N-body simulation of dark matter (a) and in a multiphase SPH simulation of a fuel spray nozzle (b).

methods then enable a topological segmentation of material transport and mixing processes. For example, the spread of salinity, temperature, and plankton is tightly linked to eddies in the ocean. The same analysis methods can be applied to understand the atomization of fuel in a combustion engine. We extend and apply these methods to particle data. The Lagrangian nature of particles enables more efficient analysis methods and the application to new types of flows, such as the multiphase flow shown in Figure 1.1 (b). Lastly, due to the chaotic nature of flow dynamics, i. e. the sensitive dependence on initial conditions, the study of uncertainties in the Lagrangian transport remains an important open problem.

Whilst the growth of data sizes closely resembles Moore's law [219], i. e. doubles approximately every two years, storage and bandwidth do not increase at the same rate. This discrepancy forces scientists and engineers to either capture less data, for example by reducing the temporal resolution, or to employ data reduction methods. To achieve a meaningful reduction in size, these approaches are inherently lossy, which is unavoidable due to the high entropy of continuous scientific data [181]. Moreover, the traditional approach of performing the visualization and analysis on individual workstations is no longer feasible for current data sizes. To address this limitation, we study data reduction techniques specific to particle data. We design these data reduction techniques for the visualization on a single workstation, thus enabling scientists and engineers to interactively explore large-scale particle data.

To visualize particles, their dynamics, and to understand the underlying physical processes, we propose an interactive visual analysis [324] approach with a Lagrangian perspective. This combines user interaction, visualization, and automatic analysis to explore complex data and processes. In detail, we discuss visualization techniques, domain-specific feature analysis, and efficient user interaction specific to large particle data. Here, large amount of particles pose a significant challenge for interactive exploration and analysis. However, this is not only a memory or computational issue since human perception is also quickly overloaded by displaying too much visual information at once.



(a) Halos in the Illustris simulation [225] (b) Simulation of a rotating turbine

Figure 1.2: A cosmological dataset with 2.6 billion particles is represented by 5.3 million halos (a) and can be explored interactively. A volume rendering of an SPH simulation (b) shows counteracting velocities that rotate a turbine.

The interactive visual analysis thus requires data reduction and abstractions to create efficient and clutter free visualizations.

Lastly, since particles exist in a spatiotemporal domain, an important part is the visualization of spatial particle dynamics. This includes transfer-function based exploration, i.e. how relevant parts of the data can be revealed whilst avoiding occlusion. We discuss and evaluate different approaches to render large particle data, including object-order splatting and image-based volume rendering approaches, cf. Figure 1.2. In all of these methods, we avoid visual clutter and enable the interactive visual exploration and analysis.

1.2 CONTRIBUTIONS

We first introduce fundamental concepts and prior work regarding particles (Chapter 2), data reduction (Chapter 3), and flow visualization (Chapter 4). The first part of this thesis then considers the visualization of flows represented by particles. We develop scalable algorithms to identify Lagrangian coherent structures and propose a visual analysis approach to study the underlying particle dynamics. Furthermore, we investigate uncertainties in the Lagrangian transport dynamics. In the second part, we consider the reduction of large particle data for interactive visual analysis using stochastic sampling, probabilistic data modeling, and with an image-based method. In summary, this results in the following contributions:

Efficient evaluation of Lagrangian coherent structures (Chapter 5, [55, 253]): We propose an efficient computational method to evaluate the finite-time Lyapunov exponent for large, three-dimensional particle data. From this quantity, the Lagrangian coherent structures are identified using a novel formulation. Both of these steps are performed interactively using GPU acceleration, thus avoiding long preprocessing times and enabling the interactive exploration of the parameter space.

Visual analysis of transport and mixing (Chapter 6, [253]): Based on the identified Lagrangian coherent structures, we present a visual analysis prototype to efficiently explore the transport and mixing properties of large particle data. In addition, we discuss the application to multiphase fluid flows, which are a prominent type of particle simulation methods. We discuss our approach in several case studies, one of which has already led to an improved spray nozzle geometry to be used in jet turbines [54].

Stochastic transport barriers and enhancers (Chapter 7, [254]): We investigate uncertainties in the Lagrangian transport. To this end, we employ the diffusion barrier strength to identify transport barriers and enhancers in stochastic flows. This quantity is similar to the finite-time Lyapunov exponent, but explicitly considers small-scale stochastic deviations to the flow.

Void-and-cluster sampling (Chapter 8, [256]): To reduce large scattered datasets using statistical sampling, we propose a novel sampling approach that optimizes the spatial distribution of sampled data points. This greedy optimization, based on the void-and-cluster technique [314], leads to an optimal spatial distribution of samples, supports non-uniform probabilities, and is performed in parallel using GPU acceleration. Lastly, our approach provides a continuous level-of-detail for interactive visualization.

Probabilistic summaries (Chapter 9, [257]): To explore and visually analyze extreme-scale datasets, we present a novel probabilistic representation for scattered data. These probabilistic summaries are based on Gaussian mixture-models and are applicable to multivariate particle data. Moreover, we develop novel formulations to interactively visualize the probabilistic representation without resorting to expensive sampling.

Image-based volume visualization using moments (Chapter 10, [255]): For image-based volume visualization of large particle data, we present an approach that decouples the data access and interpolation from the interactive exploration. To this end, we introduce a compact image-based representation using bounded trigonometric moments that is efficiently reconstructed during analysis.

2 VISUALIZING PARTICLE DATA

In this chapter, we formally introduce particles as scattered data in Section 2.1. In particular, we discuss the associated multivariate value range and the spatial interpolation thereof. Then, we discuss approaches for visualizing particles in space-time in Section 2.2 and the visual analysis of multivariate data in Section 2.3. These concepts are combined for the interactive visual exploration and analysis of particle data, which is introduced in Section 2.4.

2.1 PARTICLE DATA

Particle data is a form of scattered or unstructured data without a regular topology. A particle has a position $p \in \mathbb{R}^d$ in a d-dimensional spatial domain, where d is mostly two or three. Each particle references one or more dependent variables from the multivariate value range V through $v : \mathbb{R}^d \to V$. These variables are discretized physical quantities such as pressure, density, and velocity.

2.1.1 *Scattered Data Interpolation*

To reconstruct continuous variables, we employ scattered data interpolation. This is due to the large data sizes that make the creation of structured grids for interpolation, e. g. using Delaunay triangulation [56], prohibitively expensive. Scattered data interpolation methods consider all or a local subset of the unstructured data points during interpolation. One of the most used scattered data interpolation techniques is Shepard's method [286]. The interpolated function $\tilde{v}$ is thereby defined as:

$$\tilde{v}(p) = \sum_k \frac{w_k(p)}{\sum_j w_j(p)} v(p_k), \tag{2.1}$$

with the weighting function

$$w_j(p) = ||p - p_j||^{-\rho}.$$

The parameter ρ controls the shape of the interpolation function, such as its smoothness and the influence of close and far away points, see Figure 2.1.

Kernel regression, proposed by both Nadaraya[224] and Watson [323], replaces the weighting function in Equation 2.1 by an arbitrary kernel k with bandwidth h:

$$\tilde{v}(p) = \sum_k \frac{k(p - p_k, h)}{\sum_j k(p - p_j, h)} v(p_k). \tag{2.2}$$

If the kernel does not go to infinity at the origin, the regression approximates and does not interpolate. For noisy data this is generally preferable due to the

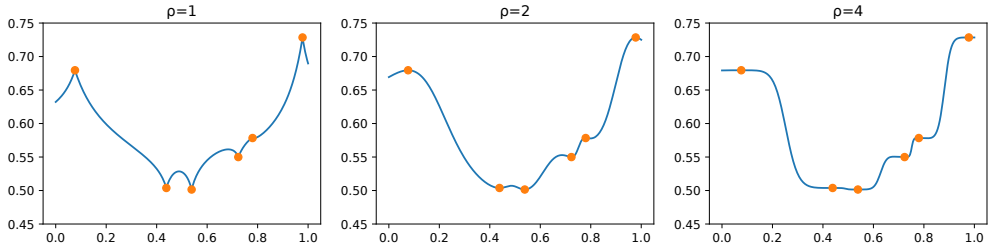


Figure 2.1: Shepard interpolation with different values of ρ.

bias-variance trade-off [22, Section 3.2]. In Section 4.2, we introduce the SPH framework, which is based on kernel regression.

All interpolation methods based on distance weighting can be limited to points in a local neighborhood. This makes the interpolation applicable to large datasets if appropriate acceleration structures, such as a k-d tree, are used. Some methods additionally compute weighting coefficients for each data point, which makes their application to large datasets more challenging. For example, moving least squares approximates a function by locally fitting polynomials. Interpolation using radial basis functions makes use of radially symmetric kernels ϕ

$$\tilde{v}(p) = \sum_k w_k \phi(||p - p_k||). \tag{2.3}$$

The coefficients w_k of each data point are hereby determined from the data by solving a linear system of equations. A linear polynomial term is often added to faithfully reproduce polynomial functions and to improve extrapolation.

2.1.2 *Particle Trajectories*

Most commonly, particles move through both space and time. In this case, the temporal domain is discretized into a sequence of time steps $T_P := (t_0, \ldots, t_{N-1})$, where $t_0 \leq \cdots \leq t_{N-1} \in \mathbb{R}$. A trajectory is only defined in a time interval $(t_j, \ldots, t_k)$, where $0 \leq j \leq k \leq N-1$. This implies that a trajectory does not necessarily exist over the whole temporal domain, for example, when particles enter or leave the spatial domain. We define a trajectory τ as an ordered sequence of points in this time interval with

$$\tau := (p_{t_j}, \ldots, p_{t_k}), \quad \text{with } 0 \leq j \leq k \leq N-1$$

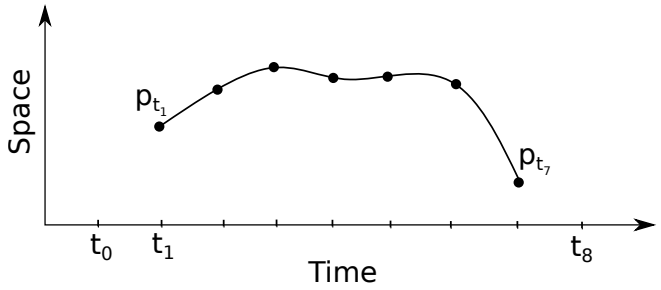
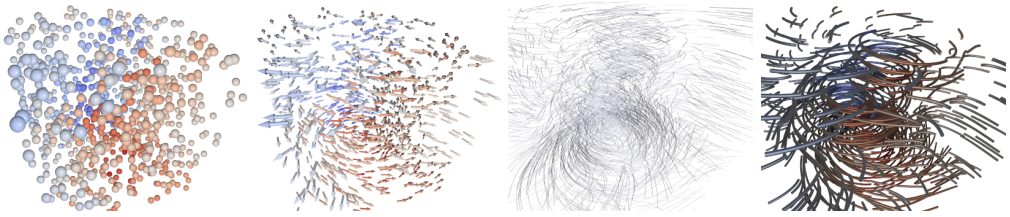


Figure 2.2: Illustration of a trajectory in one-dimensional space and time.



(a) Spheres (b) Arrows (c) Lines (d) Tubes

Figure 2.3: Different geometric glyphs to visualize particles.

and points p_{t_i} at time t_i. A trajectory is illustrated in Figure 2.2. There are several ways to reconstruct smooth curves from discrete points p_{t_i}. We employ centripetal Catmull-Rom splines [42] in this thesis, but other parametric curves can be used [34].

2.2 VISUALIZING SPATIOTEMPORAL DATA

Since particles are defined in a d-dimensional spatial domain, we directly visualize their distribution in space. To visualize the temporal component, we either use animation or visualize the particle trajectories in a fixed time interval. We use additional visual channels to convey more information about the dataset. Most commonly, we depict a single value dimension by mapping each particle to a color and transparency using a transfer function [196]. Other visual channels, such as length, area, and shape, are used when appropriate. Different approaches for rendering particles exist, specifically glyph-based (Section 2.2.1), splatting (Section 2.2.2), direct volume rendering (Section 2.2.3), image-based rendering (Section 2.2.4), and surface extraction (Section 2.2.5).

2.2.1 *Particle Glyphs*

Figure 2.3 illustrates different particle glyphs. In the following, we summarize common glyphs and their respective advantages. More complex glyphs have been developed [129, 130, 158]. For each glyph, shading and illumination is used to better convey the spatial arrangement.

SPHERES In the simplest case, we represent each particles as a sphere. Spheres are efficient to render and the surface normal is given analytically.

ARROWS To visualize the velocity of a particle, we can represent it using an arrow glyph that points in the direction of the velocity vector and is scaled by the velocity magnitude. However, the increased geometrical complexity of the glyph quickly leads to visual clutter. We recommend to show only a subset of arrow glyphs for larger particle data.

LINES To visualize flow fields, trajectories are rendered as curves. By drawing thin lines, we maximize the amount of trajectories. Line illumination [205] is employed to perform correct shading.

TUBES Instead of drawing lines, we can visualize a particle trajectory by extruding lines to tubes. This makes single trajectories more prominent and easier to distinguish, but only a small amount of trajectories can be effectively visualized at once.

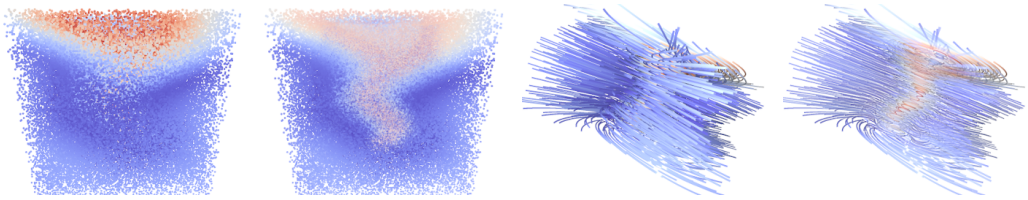
Glyphs can be rendered in image-order by casting rays or in object-order using rasterization. This decision involves different trade-offs, especially regarding transparency and scalability. In general, image-order ray casting methods scale with the image resolution if spatial acceleration structures are employed. On current GPUs, ray tracing hardware can accelerate ray casting of particles [92]. Ray casting of semi-transparent points [94, 315], metaballs [160, 302], and lines [115, 163] has been investigated extensively.

With object-order rasterization, rendering times scale with the number of glyphs. Simple glyphs with an implicit geometric representation can be rendered by rasterizing a bounding box to produce the required fragments and then ray casting the implicit geometry [262, 291]. Further improvements such as culling and deferred shading [96], optimizing the transfer of time-dependent data [93], as well as ambient occlusion [67, 97, 296] are possible. See also *MegaMol* [95] for a particle-based visualization system.

Transparency can reduce occlusion and emphasize important structures, whilst still showing the context [11, 99]. Whilst the integration of transparency in image-based ray casting is straightforward, object-order transparency is a long-standing problem in computer graphics. Specifically, pixel colors and opacities have to be composited in the correct order. By sorting the geometry from front-to-back or back-to-front, we can perform the required non-commutative compositing. Although sorting with GPU acceleration is fast, scaling remains an issue due to the algorithmic complexity of the sorting operation. Moreover, combining rendering techniques for different types of geometries is problematic in practice.

Kaehler et al. [157] use an octree to represent the particle data and to correctly composite them during traversal. To visualize large-scale cosmological data, the particles are rendered together with a volumetric grid. This combination of volumetric and particle data is also used by Schatz et al. [277] to efficiently render trillions of particles. An order-independent transparency (OIT) approach is employed to render the particle glyphs without sorting. Although it is possible to store and sort all fragments on a per-pixel basis [41], this comes at a considerable cost. Current OIT approaches thus only approximate the transparency [271]. Kern et al. [167] compare different techniques for object-order transparency and propose improvements.

In this thesis, we employ moment-based order-independent transparency (MBOIT) [223]. MBOIT renders all semi-transparent geometry twice using additive blending for both passes. In our case, the first pass renders to so-called moment buffers with a total of seven channels at 16 bits per channel. Upon completion, these moment buffers store seven Fourier coefficients of



(a) Points (b) Points (MBOO) (c) Lines (d) Lines (MBOO)

Figure 2.4: A tornado visualized using points (a) and lines (c) suffers from occlusion. Our moment-based opacity optimization [341] reveals points (b) and lines (d) forming the tornado by reducing the opacity of surrounding geometry.

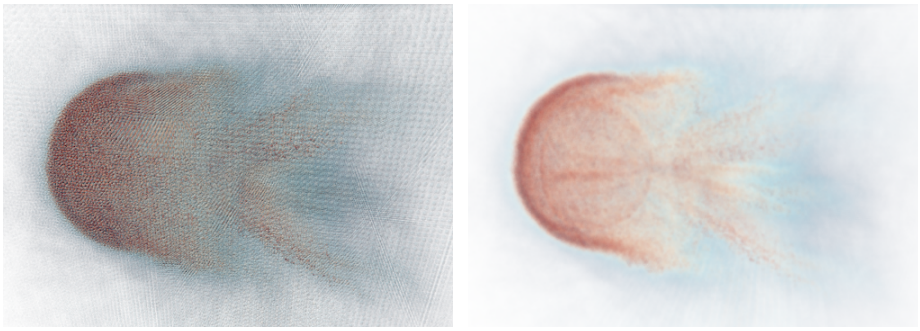
the optical depth as a function of screen-space depth. The core of MBOIT is a reconstruction based on the theory of moments that uses these Fourier coefficients to estimate the occlusion of geometry at any depth. The second pass accumulates fragment colors, multiplying them by the reconstructed transmittance values. A final full-screen pass normalizes the total brightness and composites the result with the background color.

In our work [341], we further employ moments and the MBOIT framework to store and reconstruct a feature importance per-pixel, see Figure 2.4. This is employed to optimize the opacity to emphasize important regions of the data and to reduce visual clutter.

2.2.2 *Splatting*

Object-order splatting of semi-transparent kernels [328, 329] is used to efficiently reconstruct a continuous volume from unstructured data. On modern GPUs, both rasterization as well as ray tracing hardware [173] can be used to splat semi-transparent particles. In both cases, kernel functions of particles are projected and composited to the image plane. Compared to image-based volume rendering using ray marching, splatting is limited with respect to quality and flexibility. Specifically, splatting classifies and shades before projection and the use of different illumination models or transfer functions is difficult if not impossible. Furthermore, splatting is based on the assumption that kernels do not overlap, which leads to temporal flickering otherwise. In general, aliasing is an issue for splatting that should be addressed [221, 301, 346]. However, rendering scattered data using splatting is fast and well suited for GPU acceleration. In comparison to glyph-based visualization, splatting is especially useful to render large datasets with a transfer function that maps large regions as semi-transparent, see Figure 2.5.

Splatting has been studied extensively in the past. For large data sizes, out-of-core processing and a level-of-detail mechanism is required. Hopf et al. [136] construct a hierarchical data structure using principal component analysis (PCA) clustering. During rendering, the data structure is traversed and each cluster in the hierarchy is either splatted or recursively traversed based on a maximum screen error measure. In their following work [137], the authors additionally address rendering of spatiotemporal scattered data. To interpolate



(a) Sphere glyphs (b) Particle splatting

Figure 2.5: Rendering sphere glyphs (a) and splatting (b) of a large SPH dataset (Section 6.4.2) using a transfer function.

particle positions in the temporal domain, cubic splines are employed and the control points are stored in the hierarchy. For rendering particle traces in large flow simulations, Ellsworth et al. [71] store the particles on disk in Morton order, also known as a z-curve. This enables efficient retrieval of a 3D box of particles. Fraedrich et al. [77] employ an octree as a multi-resolution representation of large cosmological data. Neophytou and Mueller [226] splat time-varying data encoded in a 4D lattice. Note that all of these approaches incorporate some form of compression, which is discussed in Section 3.1.

Splatting has also been used to efficiently render large amounts of radial basis functions (RBFs). To efficiently render unstructured data, Co et al. [50] create a hierarchy using PCA and fit RBFs using a least-squares approximation. The work from Jang et al. [151] is applied to large, rectilinear gridded volume data and similarly uses PCA for clustering. Weiler et al. [326] discuss the application to multivariate gridded data and feature detection in the radial basis. Jang et al. [149] propose elliptical basis functions to better model the data, which requires rendering anisotropic Gaussian kernels. To relax the limitation of rotational invariant and symmetric kernels, Zwicker et al. [346] discuss splatting of elliptical Gaussians by approximating the footprint after perspective projection. Moreover, Neophytou et al. [227] efficiently slice elliptical three-dimensional kernels on the GPU, whilst Juba et al. [156] perform GPU-based ray casting. Hong et al. [135] discuss the modeling of anisotropic RBFs specifically for unstructured data, for which weighted least squares and a Delaunay triangulation is employed. In general, the modeling of RBFs does not scale to large particle data and is mostly applied to structured and unstructured meshes. State of the art methods for rendering structured and unstructured data are usually performed in image-order using ray casting [220] due to the inherent limitations of both splatting and RBF fitting.

2.2.3 *Image-Order Volume Rendering*

Direct volume rendering evaluates a physically-based model of light transport [105, 209]. Although volume rendering has been performed in object-order in the past, current methods perform image-order ray marching of the volumet-

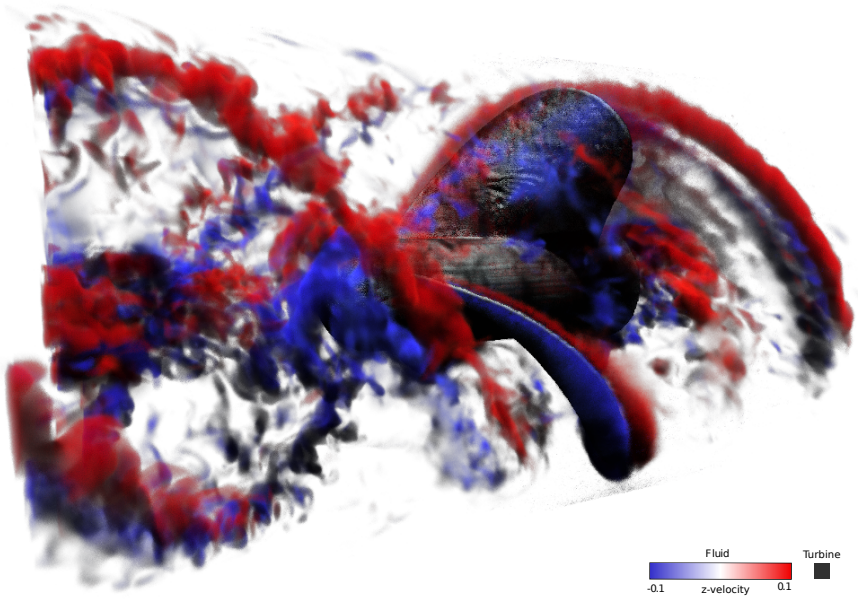


Figure 2.6: Stochastic volume rendering of SPH data with single-scattering illumination [246]. The counteracting velocities around the turbine blades force the turbine to rotate.

ric data. To reconstruct a continuous scalar field from the particles, scattered data interpolation is performed, cf. Section 2.1.1. The unstructured particle data can be resampled to a regular grid to enable efficient sampling of the scalar field. For large data, compression [13] and out-of-core rendering techniques [19] are consequently required. For example, Reichl et al. [261] resample cosmological data to a sparse octree and perform wavelet-based compression in a time-consuming preprocess. Subsequent volume rendering is then able to create high-quality visualizations in a reasonable amount of time using GPU decompression.

Several approaches specific to particle data have been proposed to speed up the expensive resampling and to reduce storage requirements. In detail, Jang et al. [150] avoid resampling and directly ray cast particle data on the GPU using binary space partitioning. To speed up the computation, small kernel radii are used for interpolation while generating the hierarchical data structures. Orthmann et al. [231] enhance octrees with topological caches for fast neighborhood lookup coupled with a level-of-detail mechanism. A level-of-detail hierarchy is also created by Fraedrich et al. [76]. Based on the current view frustum, the appropriate levels in the hierarchy are resampled to a perspective grid that is updated in each frame. Hochstetter et al. [133] similarly perform volume rendering of particles in a sparse perspective grid. The grid is traversed in bundles of rays that are adaptively sampled, bounded by a user-controlled error in screen-space. To render radial basis functions, Knoll et al. [174] employ ray bundles to traverse a bounding volume hierarchy on a CPU cluster.

Reichl et al. [260] convert the particle data to a binary voxel representation encoded in a sparse octree to render photorealistic fluid simulations. In

contrast, Zirr and Dachsbacher [344] perform on-the-fly voxelization using a perspective grid. Note that a binary voxel representation limits these approaches to homogeneous fluids. The methods are well suited for photorealistic rendering of fluids. In our work [246], we propose a direct rendering of SPH data, see Figure 2.6. To accelerate the evaluation of the interpolation kernels during ray marching, we only consider a stochastically sampled subset of particles. Moreover, we explicitly reconstruct and shade the interfaces between different types of particles.

To summarize, most of these approaches combine hierarchical space partitioning and level-of-detail for fast, interactive ray marching. The costly resampling is thereby either performed in a preprocess or evaluated on-the-fly.

2.2.4 *In Situ and Image-Based Visualization*

To visualize massive datasets produced by state-of-the-art supercomputers, in situ visualization [49] is becoming increasingly popular. Thereby, the simulation is tightly coupled with the visualization pipeline to produce intermediate results whilst the simulation is still being run. This has the advantage that the data can be directly processed on the supercomputer. However, exploring or interacting with the data is more difficult.

Ahrens et al. [5] present the *cinema* framework for in situ visualization of extreme scale data by collecting and organizing a large database of images taken with different parameters. The authors point out that storing a massive amount of images, e.g. in the order of 10^6, still leads to a reduced amount of data compared to state-of-the-art simulations, which are in the order of $\geq 10^{15}$. Lukasczyk et al. [201] additionally generate depth images in situ to determine a minimal set of images that best approximates the data. This reduces the number of images in the *cinema* database and enables the approximation of views that were not stored. Lukasczyk et al. [200] further present a deferred rendering framework using geometry buffers that are generated in situ. Lastly, Woodring et al. [334] extend the *cinema* framework for data besides images and provide a relational data model to enable querying.

To enable a change of the transfer function or the lighting configuration, storing color images is not sufficient for volumetric data. Tikhonova et al. [306–308] propose explorable images that store image slices to allow modification of the transfer function by solely relying on image-space operations. Ye et al. [337] employ explorable images to render pathtubes of flows in situ.

Image-based rendering approaches are an alternative approach to perform surface [47] or volume rendering [222] from a small, view-dependent proxy image. These approaches are especially beneficial for scattered or unstructured data since the generation of the proxy image is costly, but the rendering step is independent from the data representation. For this reason, Shareef et al. [285] perform volume rendering on unstructured grids. This representation is based on layered depth images [283], where each pixel in the proxy image contains a list of depth-ordered samples. Volumetric depth images [78] partition rays in image-space into segments consisting of a composited color and opacity, as well as a depth range. Volumetric depth images have been extended to

space-time [75], by exploiting both inter-ray and inter-frame coherence. Wang et al. [319] partition each ray into segments, but subdivide adaptively based on the Shannon entropy of ray densities. In each segment, the density distribution is approximated with a histogram, which is storage intensive and introduces aliasing. Their work has been extended for time-varying data [318]. This image-based representation is used to interpolate between the raw data at discrete time steps.

2.2.5 *Surface Reconstruction*

Surface extraction from particle data has been extensively studied. Most methods are based on the marching cubes algorithm [197], but use different scalar fields [2, 230, 294, 343]. The resulting surfaces usually suffer from bumpiness due to the irregular distribution of particles [7, 339]. Moreover, the required preprocessing often forms a bottleneck in the visualization pipeline due to large computational time and memory requirements.

As shown by our work [246], surface reconstruction can be performed during volume rendering, see Figure 2.6. The costly volumetric interpolation is thereby performed on-the-fly, as discussed in Section 2.2.3. Although an isosurface is just a transfer function with a Dirac impulse at the isovalue, it requires special rendering techniques in practice. Hadwiger et al. [107] use the secant method for finding isosurfaces. Their approach has been extended by Knoll et al. [172] who propose peak finding as an alternative to pre-integrated transfer functions. Igouchkine [142] propose a multi-material volume renderer that supports physically-based surface models at the interfaces between different volume components.

2.3 VISUALIZING MULTIVARIATE DATA

The visualization of higher-dimensional data is traditionally being studied in the field of information visualization [193, 195]. In this section, we focus on multivariate scientific data [38, 165] and the application to large particle data. The aim is the visual analysis of correlations, trends, and outliers between the different variables. This type of exploratory data analysis, as first introduced by Tukey [311], helps in forming and validating hypotheses to ultimately create new models of the data.

2.3.1 *Common Visualizations*

We recapitulate several visualization techniques that are employed throughout this thesis. A scatter plot displays two (or three) variables in a Cartesian coordinate system, see Figure 2.7 (left). A scatter plot is most commonly used to visualize correlations between two variables. For higher-dimensional data, a scatter plot matrix [44] displays all scatter plots of pairwise combinations of variables in a matrix layout. Since particle data usually contains not more than ten dimensions, the quadratic scaling of the visualization with regards to the

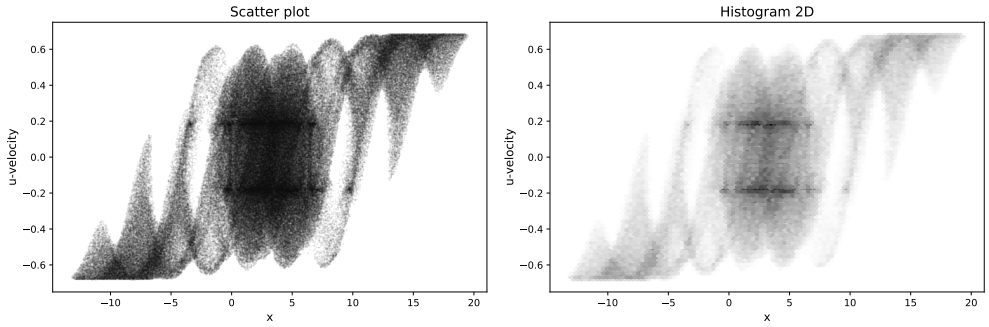


Figure 2.7: Scatter plot and (hexagonal) histogram of position and velocity in x-direction.

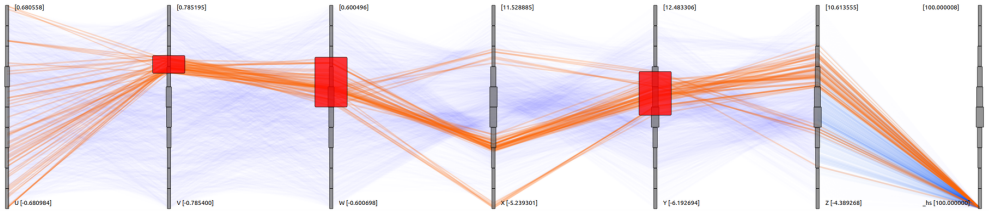


Figure 2.8: Parallel coordinate plot. The data dimensions are placed parallel to each other, with each line representing a single particle. We have selected several value ranges (red boxes). The corresponding particles are highlighted in orange.

dimensionality is generally acceptable. Furthermore, scagnostics [80] can be used to identify interesting scatter plots.

In contrast to a scatter plot, a two-dimensional histogram subdivides the domain into rectangular (or hexagonal) bins and computes the frequency of values, see Figure 2.7 (right). The histogram thus scales better with growing data sizes, but suffers from aliasing and deemphasizes outliers.

A parallel coordinate plot [125], first introduced by Inselberg [145], shows all data dimensions at once by placing them parallel to each other, see Figure 2.8. For each dimension an axis is created and for every particle a polyline is drawn over all axes based on the particles' values. To enable interaction, every axis can be brushed to select particles. The polylines of selected particles are then highlighted in the parallel coordinate plot. Figure 2.8 additionally depicts a 1D histogram on each axis to better convey the distribution in each dimension.

A parallel coordinate plot allows correlating each pair of neighboring dimensions. To this end, the axes can be reordered to compare different dimensions. Determining the best arrangement, for example with respect to some similarity measure, is an NP-hard problem [8]. Since particle data is typically limited to few dimensions, this is less of an issue for our application.

Star coordinate plots [161, 162] arrange the coordinates circular, see Figure 2.9. A higher dimensional data point $v \in V$ is then projected to a two-dimensional point by summing the unit vectors of each coordinate multiplied by v. Similar to parallel coordinate plots, rearranging the axes interactively is crucial. Additionally, the user can modify the orientation and length of the axes to alter the projection. Since this defines a general affine projection,

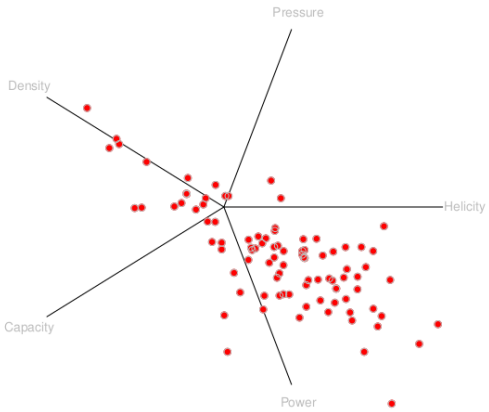


Figure 2.9: A star coordinate plot arranges the coordinates on a circle with the origin at the center.

strong distortions are possible. To address this, Lehmann and Theisel [185] extend star coordinates with an orthographic constraint to better preserve the structure of the dataset in the projection. Star coordinates are used extensively in the *SmoothViz* system [215] to visualize particle data.

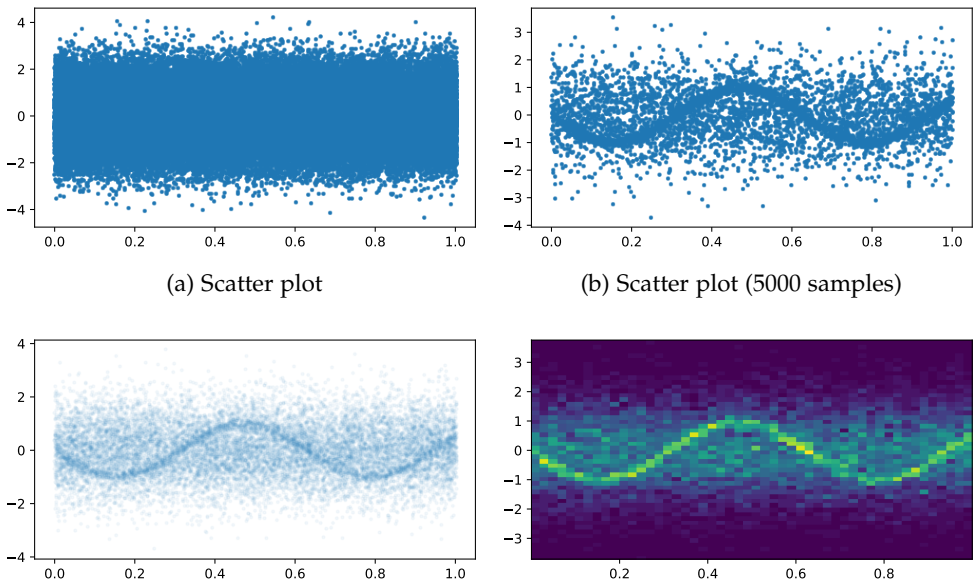
2.3.2 *Multiple Coordinated Views*

Most, if not all, visualizations do not scale well with respect to dimensionality. The visualization of multivariate data thus requires interactive visualizations [310]. Multiple, coordinated views [264] are commonly employed to enable interaction for data exploration. Different variables are hereby explored and analyzed in linked views of the data. Each of these views, including scatter plots, function graphs, histograms, or parallel coordinates, support interaction by brushing. The linked views thereby highlight the brushed data values. Additionally, logical combinations of brushes support the creation of increasingly complex selections. This is often combined with query-driven visualization [297], where queries can be formulated in SQL [43] or a custom feature definition language [116].

This interaction scheme, referred to as brushing and linking [16, 206], enables the intuitive visual exploration and analysis of higher-dimensional data. There are many variations, for example angular brushing for parallel coordinate plots [118]. Note that the brushing operation must not be binary. Instead, a fractional degree of interest $d \in [0, 1]$ can be assigned to each data point. Such a continuous degree of interest can be obtained by smooth brushing [59]. Lastly, focus and context [117] emphasizes brushed data elements, whilst the surrounding context is still visible.

2.3.3 *Clutter Reduction for Large Data*

Visualizing large amounts of particle data leads to overdraw and visual clutter, especially for glyph-based visualizations such as scatter and parallel coordinate



(a) Scatter plot

(b) Scatter plot (5000 samples)

(c) Scatter plot with transparency

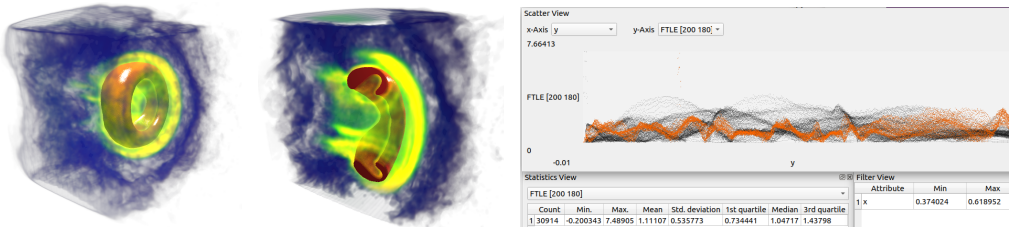
(d) 2D Histogram

Figure 2.10: For large datasets, glyph-based plots suffer from visual clutter and occlusion (a). By sampling the data, we reveal previously hidden correlations (b). Plotting semi-transparent points (c) is a simple form of density estimation. In (d), a 2D histogram is used for estimating the density.

plots. Since visual displays are small relative to common data sizes, clutter reduction is required to create efficient visualizations [70].

One approach to address visual clutter is the use of sampling [58], to stochastically reduce the number of glyphs that are drawn, see Figure 2.10 (a) and (b). Bertini and Santucci [18] provide a formal model to measure the overlap in a given area. To reduce overdraw, different sampling strategies are automatically derived from this model. Reinhardt et al. [263] use stochastic sampling to improve performance and reduce visual clutter for the visual debugging of smoothed particle hydrodynamics (SPH) simulations. Ellis et al. [68, 69] present a sampling lens that restricts the use of sampling to a movable region, whilst the user is still able to view the context. Sampling for data reduction is further discussed in Section 3.3.

Instead of drawing discrete glyphs, methods based on density estimation, such as histograms, reconstruct and visualize the density of data values. For scatter plots, a simple form of density estimation is to draw individual points semi-transparently using alpha blending, see Figure 2.10 (c). Histograms and hexagonal binning are often employed to convey frequency information, see for example Figure 2.10 (d), but can lead to aliasing due to their discrete nature. The concept of histograms has also been extended to parallel coordinate space by drawing a quadrilateral for each bin [9, 229]. Blaas et al. [25] discuss the application to large datasets. Splatterplots [210] group dense data points and visualize them as contours, but explicitly sample representative outliers. In general, outliers are often lost in density-based visualizations and require special treatment.



(a) FTLE (b) Brushed particles (c) Brushing in the linked views

Figure 2.11: Interactive visual analysis exemplified on an air bubble moving through water (Section 6.4.2). The FTLE of all particles is shown in (a) using a transfer function in combination with the extracted phase interface between air and water particles. In (c), we use a combination of brushes to select a subset of particles according to statistics and a scatter plot of the FLTE. The corresponding particles are shown in (b).

Although kernel density estimation would allow for an improved reconstruction of a continuous density, it is computationally too expensive for large particle data. In the field of scientific visualization, continuous scatter plots have been introduced [10] to construct density plots by considering the topology and interpolation of data samples in their spatial domain. Continuous scatter plots have been extended to parallel coordinate space [124]. To improve computational efficiency, Heinrich et al. [123] progressively sample the spatial domain with Gaussians kernels to construct continuous scatter and parallel coordinate plots. Their approach has been extended to arbitrary projections of the value range and applied to SPH data [214].

2.4 INTERACTIVE VISUAL ANALYSIS

In this section, we review the interactive visual exploration and analysis of scientific data according to Weber and Hauser [324] and discuss the application to large particle data. In detail, a tight coupling of computation, user interaction, and visualization, provides new insights into the data and enables the generation and validation of hypotheses. This approach, closely related to visual analytics [166, 304], combines the advantages of computational and interactive, human-guided data analysis.

Foremost is the combination of the spatiotemporal views and the multivariate value views, as discussed in Section 2.2 and Section 2.3. All views are thereby connected using brushing and linking and employ a focus and context visualization to emphasize the brushed values. This enables the seamless transition between an overview and a detailed view on the data, according to the mantra of Shneiderman [289]: “Overview first, zoom and filter, then details-on-demand”.

Lastly, the analysis is supported by computing derived variables, e. g. statistics or domain-specific features. Weber and Hauser [324] identify several levels:

1. Show and brush,

2. Relational analysis: combination of brushed and complex queries,
3. Complex analysis: computation of derived variables from statistical and machine learning techniques,
4. Proprietary analysis: identification and extraction of domain-specific features.

The application of this method must be tailored to a specific domain. As an example, we show an SPH simulation of an air bubble moving through water in Figure 2.11, cf. Section 6.4.2 and Piochowiak et al. [246]. To visualize the movement of particles, a domain-specific feature, the FTLE (Section 5.1), has been computed and is visualized using a transfer function. In addition, the phase interface between the air and water particles is shown in orange. Using statistics and a scatter plot of the FTLE, we perform a combination of brushes to select a subset of particles in (c). The corresponding particles are consequently shown in the linked spatial view (b). This example thus shows the combination of brushes and queries, the spatial and other linked views, and the use of derived variables and statistics.

3

DATA REDUCTION FOR PARTICLE-BASED VISUALIZATION

As we have seen in the previous chapter, interactive exploration and analysis are essential to gain insight into large and complex datasets. However, data sizes are growing rapidly due to advancements in high-performance computing or increasingly accurate measurement devices. These growing data sizes are challenging for the interactive visualization and analysis. Moreover, storage and bandwidth capacities do not increase at the same rate. Data reduction is thus a necessary means to reduce storage requirements and to enable subsequent data analysis.

Due to the stochastic nature of continuous scientific data, lossless or near-lossless compression is generally unable to significantly reduce the data sizes [188]. Moreover, we are not just interested in compression to reduce storage requirements, but also for the interactive visualization and exploration of large data. For rendering structured volumes, compression [13] and multi-resolution [19] approaches exist. A popular family of lossy compression methods applies transformations, such as the Fourier or discrete cosine transforms [338], custom transforms [190], or transforms based on tensor decomposition [12], followed by quantization and encoding of the coefficients to achieve compression. However, these transform-based methods are only applicable to uniformly structured data. The compression of particle data is more difficult since the positions of data points are irregular and not implicitly defined. Particle compression thus results in a significantly reduced compression rate [342]. We review several compression methods that are applicable to particle data (Section 3.1).

To achieve a meaningful reduction in size, probabilistic representations can be derived from the original data. These representations are much smaller in size and are specifically tailored for interactive visualization and analysis tasks, but lead to a bigger loss in accuracy. Specifically, we discuss probabilistic data models (Section 3.2) and statistical sampling (Section 3.3) to compactly represent large particle datasets.

3.1 PARTICLE COMPRESSION

Truly lossless compression schemes reconstruct exactly the same data, bit by bit. Dictionary coding, usually a variant of LZ77 [345], match strings and replace them by references to a dictionary. Entropy coding exploits non-uniform probabilities of data values. The data size is reduced by storing variable-length codewords for each data value, with a length proportional to its probability of occurrence. The most common entropy encoding techniques are Huffman [138] and arithmetic [333] coding.

Lossy compression does not necessarily match the original data, thus achieving a larger reduction in size. The simplest form of lossy compression, scalar quantization, discretizes real numbers in a known interval to a small number of bins. When the data distribution is non-uniform, the binning scheme can be adjusted to reflect the distribution. Scalar quantization has been used extensively to quantize the relative positions of particles in a hierarchy [71, 77, 136, 137].

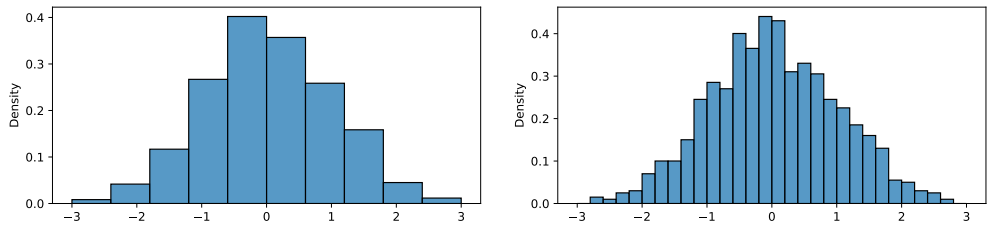
Vector quantization groups vectors with similar values together. Each group is then represented by its centroid and the mapping scheme is stored in a codebook. The process of finding a good codebook is thus the most important part of this method and makes the encoding phase computationally expensive. In contrast, the decoding is fast since the codebook is simply used as a look-up table. Fraedrich et al. [77] employ vector quantization to compress the multivariate attributes of particles.

Predictive coding [146] can be used for both lossless and lossy compression. For particle data, we can compress the positions and values along a particle trajectory $\tau = (p_{t_0}, \ldots, p_{t_{N-1}})$. The method predicts a value from previous values and only stores the difference to the actual value, i. e. the residual. These residuals are more amenable to compression since they exhibit lower entropy. Linear prediction uses a linear combination of previous values to predict future ones. While zeroth order prediction determines that the next position is the same as the previous one, first order prediction assumes that the particle travels in a straight line. Ellsworth et al. [71] report similar compression rates for zeroth and first order prediction of particle positions, whilst second order prediction performed significantly worse. Han et al. [114] test both zeroth, first, and second order prediction and select the prediction with the best fit for each value along a trajectory.

More general methods for the compression of floating point data exist. For example, *Fpzip* [191] uses the Lorenz predictor [141] and arithmetic coding of the residuals for lossless compression of floating point data. However, the Lorenz predictor assumes uniformly structured data. ISABELA [181] uses B-splines as predictors and additionally pre-conditions the data points by sorting them before prediction. Although a sorted array can be accurately modeled using a B-spline, the sorting must be reversed again. More lossless and lossy compressors for general floating point data exist [40, 336], which typically chain multiple components together, including pre-conditioners, prediction, and lossless compression coders. Since these methods are developed for compute clusters and supercomputers, they focus on parallel execution and high throughput.

3.2 PROBABILISTIC DATA MODELING

Probabilistic data models can be used to represent large scientific datasets. The data is thereby first partitioned. A compact, probabilistic model is then created for the value distribution in each partition, thus trading data size for accuracy.



(a) 10 bins (b) 30 bins

Figure 3.1: Two histograms of the same distribution with differing amounts of bins.

3.2.1 *Probabilistic Models for Data Reduction*

A value distribution might be represented by parametric or non-parametric probabilistic models. Non-parametric models, such as histograms, explicitly store frequencies of the probability density function. Parametric models, such as Gaussian models, estimate a small amount of parameters that define a continuous density function. We briefly review probabilistic models that are commonly employed for data reduction.

Histograms

Histograms, credited to Pearson [240], discretize a distribution into a number of bins and store the frequency or density of values in each bin. Two histograms are shown in Figure 3.1. Although histograms are a non-parametric model, they still depend on the number of bins. Whilst few bins reduces noise, a larger number increases the accuracy. There are several rules to determine the amount of bins from the data, such as Sturges rule [298], but they often assume normally distributed data.

Gaussian Models

A univariate Gaussian model $\mathcal{N}$ is completely defined by mean μ and standard deviation σ. The density is given by:

$$\rho_{\mathcal{N}}(x) := \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{3.1}$$

This is a special case of the multivariate Gaussian model defined by mean and covariance matrix Σ:

$$\rho_{\mathcal{N}}(x) := \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{(x-\mu)^\intercal \Sigma^{-1}(x-\mu)}{2}\right). \tag{3.2}$$

Scientific data is often normally distributed, i. e. can be modeled by such a Gaussian. This is due to the central limit theorem, which asserts that when independent random variables are added, their sum tends toward a Gaussian distribution in the limit. This holds even if the original random variables are not normally distributed. Of course, this does not imply that all data is normally distributed, but it can be a good approximation. Figure 3.2 shows a

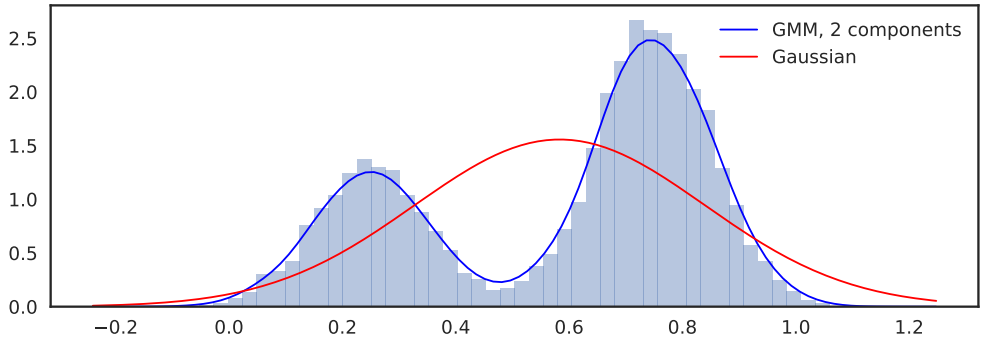


Figure 3.2: To the distribution shown by the histogram, we have fitted a Gaussian (red) and a mixture of two Gaussians (blue).

Gaussian fitted to a distribution that is apparently not normally distributed. In this case a Gaussian is not a good model of the data distribution.

To test if a dataset is indeed normally distributed, graphical methods as well as statistical tests of normality exist. For univariate Gaussians, the Shapiro-Wilk test [284] is generally preferred due to its statistical power [259]. It returns a likelihood (or p-value), where values between $[0.05, 0.1]$ are often used as a threshold to decide if the data is normally distributed. If the p-value is above this threshold, it is reasonable to model the corresponding distribution using a Gaussian. Normality tests for multivariate distributions exist, but selecting one is more difficult [127, 212].

Gaussian Mixture Models

Gaussian mixture models (GMMs) are a parametric model that represent arbitrary distributions as a mixture of Gaussians, see Figure 3.2. More formally, the density of a GMM $\mathcal{G}$ is defined as a weighted combination of K Gaussian distributions:

$$\rho_{\mathcal{G}}(x) := \sum_{k=1}^{K} w_k \rho_{\mathcal{N}}(x), \text{ where } \sum_{k=1}^{K} w_k = 1. \tag{3.3}$$

In theory, any function can be accurately represented using a GMM, with a potentially infinite amount of components. In practice, the number of components is kept relatively low. Moreover, selecting the best number of components is an open problem that is closely related to model selection in the field of machine learning. Too many components will lead to overfitting and large storage requirements, whilst not enough components lead to a poor fit of the data. Bayesian model comparison [22, Section 3.4] is commonly used to compare different models. This implies that all possible GMMs are computed beforehand. Then, the best model is selected. The Bayesian information criterion (BIC) [280] and the Akaike information criterion (AIC) [6] are commonly employed to compare GMMs. We make use of the BIC, which rewards a high likelihood over the training data and penalizes by the number of components.

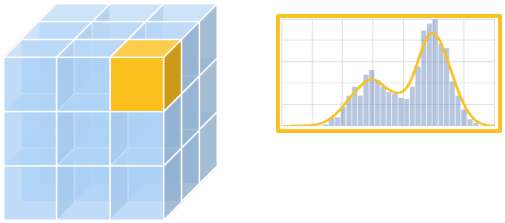


Figure 3.3: To represent volumetric data using probabilistic data models, the volume is partitioned into blocks and the value distribution in each block is then modeled separately.

It is defined using the number of free parameters $p(\mathcal{G})$ in a GMM $\mathcal{G}$ and the number of samples n as

$$\mathrm{BIC}(\mathcal{G}, n) := p(\mathcal{G}) \log(n) - 2 \log(\mathcal{L}(\mathcal{G})), \tag{3.4}$$

where $\mathcal{L}(\mathcal{G})$ denotes the likelihood of $\mathcal{G}$. For a d-dimensional GMM, the number of free parameters is given by

$$p(\mathcal{G}) := K \left(\frac{d(d+1)}{2} + d \right) + d - 1, \tag{3.5}$$

i. e. equal to the number of entries in the symmetric covariance matrix, the mean, and the weights.

Given a number of components, the parameters of a GMM are estimated using the expectation maximization (EM) procedure [57]. This iterative method seeks maximum likelihood estimates of the model parameters. It alternates between an expectation step, which evaluates the log-likelihood of the input samples using the current parameters, and a maximization step, which computes the parameters by maximizing the expected log-likelihood found in the expectation step.

3.2.2 *Probabilistic Models of Volumetric Data*

To compactly represent volumetric data, Thompson et al. [305] first partition a volume into larger blocks of voxels, see Figure 3.3. Then, histograms are used to represent the value distribution in each block. The degree of data reduction is thus determined from the size of a block and the size of the histogram. Dutta et al. [62, 63] use Gaussian mixture models to represent the value distribution. Later on the authors [64] improve the data model and thus the reconstruction by partitioning the data into irregular, homogeneous regions. This increases spatial coherency and thus minimizes variance in the partitions.

A fixed number of Gaussian components is used for these mixture models. However, the value distribution is tested for normality and might be modeled by a single Gaussian. Since only the value distribution is modeled inside a block, the spatial correlations are left out. Wang et al. [317] use GMMs to represent the spatial distribution in each data block. This improves the quality of the reconstruction, but would require storing 4D distributions. To avoid modeling 4D GMMs, the value range is discretized and a 3D GMM is modeled

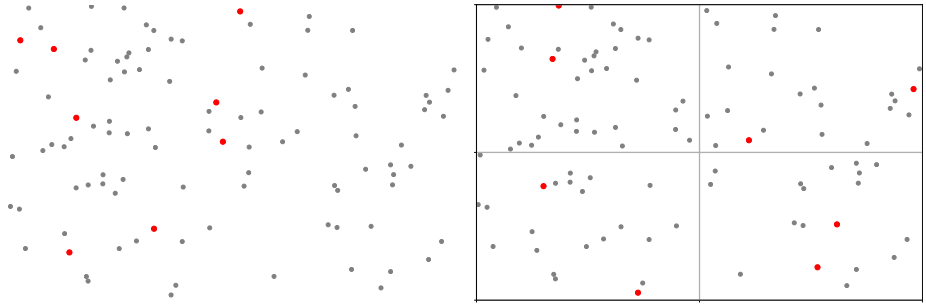
in each bin. To reduce storage size, the number of GMM components is chosen adaptively by brute force computation of GMMs and selection based on the BIC. In the context of multi-resolution volume rendering, Sicat et al. [290] directly model 4D distributions of the spatial and data domain using GMMs. Their approach utilizes a multi-resolution hierarchy to model and simplify the GMMs.

For parameter studies in cosmological simulations, Wang et al. [320] store GMMs as compact prior knowledge to reconstruct high-resolution datasets from multiple simulations runs. Li et al. [187] reduce unstructured cosmological simulation data in situ by subdividing space using a k-d tree and estimating particle density using a GMM in each leaf node. The number of GMM components is determined from the data, but is applied to all GMMs for a dataset. To this end, the Akaike information criterion (AIC) is employed and extended to account for all parameters and likelihoods in the dataset. Afterwards, GMMs with a low likelihood are refined by further subdividing space and then using multiple GMMs. Lastly, during the analysis stage particles are sampled from the GMMs using a Monte Carlo approach.

Hazarika et al. [119] propose a copula-based uncertainty modeling approach to represent a multivariate distribution using different types of univariate distributions, including GMMs, separately from their interrelation. To summarize large-scale multivariate volumetric data, a copula-based analysis framework has been introduced [121]. This approach is the first to address the modeling of multivariate data, but the Gaussian copula function limits the correlations between dimensions to a single Gaussian. Moreover, the visualization of this representation requires sampling, which hinders its applicability to interactive visual analysis, especially for rendering scattered data.

Although probabilistic data models achieve a significant reduction in data size, they are difficult to extend to high-dimensional data due to the curse of dimensionality. This refers to the observation that when the dimensionality increases, the volume of space and thus also the required amount of data grows exponentially. In high-dimensional space, everything thus appears to be sparse and dissimilar.

Most probabilistic data models have been developed for uniformly structured data. While scattered data can be visualized by first reconstructing a structured representation [76, 261], this approach has its own drawbacks and is not an option for all analysis techniques. As shown throughout this thesis, particle-based visualizations benefit from specific visualization and analysis techniques. However, sampling a probabilistic data model to create particle data leads to large data sizes and is performance intensive [187]. Still, many feature-based and query-driven analysis and visualization tasks rely on computing local statistics [90, 203] or are inherently distribution-based [45, 62, 63, 322]. In this case, probabilistic data models are a suitable choice for compact data representation.



(a) Random sampling (b) Stratified sampling

Figure 3.4: Whilst random sampling (a) chooses samples (red) by random, stratified sampling (b) ensures a more uniform spatial distribution of the samples by sampling each strata separately.

3.3 DATA REDUCTION BY SAMPLING

Statistical sampling of data is gaining popularity in the field of scientific visualization [58]. Sampling is useful for both data reduction and to create clutter free visualizations. For data reduction, a representative subset of the original dataset is selected and used for subsequent visualizations and general data analysis.

The sampling strategy directly controls the quality of the representation. Although simple random sampling selects an unbiased subset, stratification generally increases the precision of subsequent estimates [51], see Figure 3.4. For particle data, we typically want to stratify at least in the spatial domain and possibly in the temporal and value domains.

Woodring et al. [335] describe a simulation-time stratified sampling strategy for large-scale particle data. Here the sampling is performed in situ, i.e. whilst the simulation is being executed, and only the sampled data is stored. For stratification, the authors construct a k-d tree from the data that is also used as a level-of-detail representation. As shown by Kumar et al. [178], data management and especially the order of read and write operations is an important factor when storing and accessing particle data. Reordering the particles can not only improve I/O performance, but can be leveraged as a level-of-detail mechanism.

Su et al. [299] stratify both in the spatial and the value domain. Sampling is performed on the server, when a client queries the data. Bitmap indices are employed to support efficient subsetting. Wei et al. [325] extend their approach with an information-guided sampling strategy and recovery technique. During sampling, they measure the information per stratum by computing the entropy of a value distribution. The amount of samples that are drawn is then proportional to the entropy. Biswas et al. [24] similarly employ an information-guided sampling strategy in an in situ setting, but use a global histogram for the entropy computation. In their follow-up work, Biswas et al. [23] present a generic data-driven importance-based sampling algorithm using both local and global properties.

These approaches are limited to univariate data since they consider only a single variable for information-guided sampling. Dutta et al. [61] present a sampling strategy that explicitly considers multivariate data. The authors assign importance to each data point based on the pointwise mutual information and multi-variable generalizations thereof. In contrast, Hazarika et al.[120] partition the spatial domain, transform each partition using PCA, and then perform sampling. Since only a subset of principal components is kept, the dimensionality of samples is reduced. At the same time, correlations between variables are preserved for subsequent multivariate data analysis.

A desirable property of point distributions is the blue noise characteristic [313], which leads to large mutual distances between points without noticeable regularity artifacts. This implies an optimal stratification in the spatial domain. Balzer et al. [14] compute capacity-constrained Voronoi diagrams (CCVD) to optimize the blue noise property of point distributions, which allows adapting the point distributions to a given density function. To find representative particles, Frey et al. [79] propose loose capacity-constrained Voronoi diagrams (LCCVD) that relax the capacity-constraints of the CCVD method and are computed on the GPU. All methods based on capacity-constrained Voronoi diagrams can be used to sample scattered data, but are computationally demanding. Bridson [29] presents a Poisson disk sampling technique to generate blue noise samples in arbitrary dimensions by enforcing a minimal and maximal distance between nearest neighbors. However, the technique is designed to produce entirely new sample sets, not to reduce an existing sample set.

4 FLOW VISUALIZATION

In this chapter, we first introduce fluid dynamics in Section 4.1. In this thesis, we focus on time-dependent Lagrangian flows given by the motion of particles in space and time. This Lagrangian frame of reference on the data offers a different perspective compared to the more common visualization of structured velocity fields. We introduce smoothed particle hydrodynamics (SPH), a prominent particle-based fluid flow simulation method, in Section 4.2. The state of the art of flow visualization is recapitulated in Section 4.3. Lastly, we discuss uncertain flows in Section 4.4.

4.1 FLUID DYNAMICS

Fluid dynamics are studied in physics, engineering, and are an important topic in the field of scientific visualization. Applications range from aerodynamics, meteorology, and medical visualization to the formation of galaxies and cosmic nebulae. Already in 1509, Leonardo da Vinci systematically studied turbulent flows using hand drawn visualizations, see Figure 4.1. However, research on fluid dynamics is still ongoing. Especially turbulency still remains as one of the greatest scientific challenges to date [65].

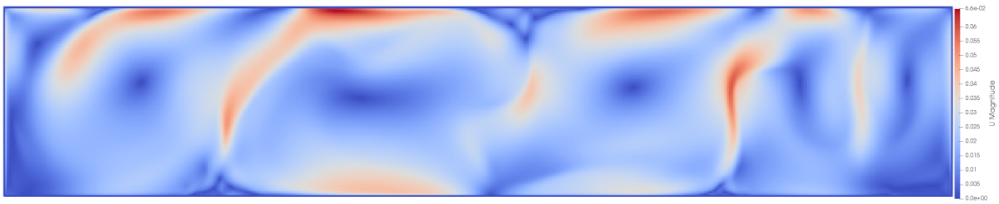
There are different sources of flow data, both from physical and numerical experiments. The former can be obtained, for example, using particle image velocimetry (PIV). Thereby, seeding particles are inserted and tracked in the fluid under investigation and their velocity is measured with a camera and a laser. In contrast, numerical experiments are performed in the field of computational fluid dynamics (CFD). The Navier-Stokes equations of fluid motion are thereby solved by discretizing the input domain. An example of a numerical simulation is shown in Figure 4.2, where water is heated from below leading to a Rayleigh-Bénard convection [87].

CFD simulations are most commonly performed in the Eulerian frame of reference. The velocity of the flow is thereby measured at fixed locations in space and time, cf. Figure 4.3 (a). In contrast, we use a Lagrangian frame of reference by following the motion of particles through space and time, cf. Figure 4.3 (b). To change the frame of reference from the Lagrangian to the Eulerian frame, particle values can be interpolated at fixed locations in space and time. Inversely, tracer particles can be integrated in the velocity field by solving an ordinary differential equation:

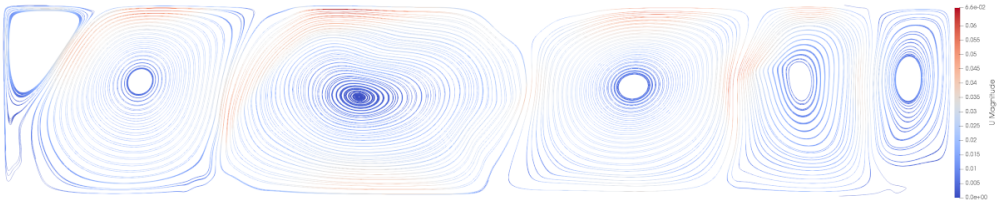
$$\frac{\mathrm{d}L(t)}{\mathrm{d}t} = v(L(t), t), \qquad L(0) = p_0, \tag{4.1}$$

with velocity field v, integral line L parameterized by time t, and the initial value p_0. This equation can be solved by numerical integration. The velocity field of the Rayleigh-Bénard convection and the corresponding integrated

Figure 4.1: Visualization of a turbulence wake as water streams past an obstacle, drawn by Leonardo da Vinci [207].

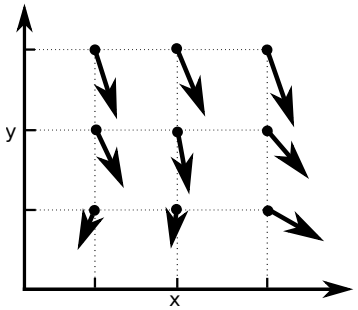


(a) Velocity magnitude

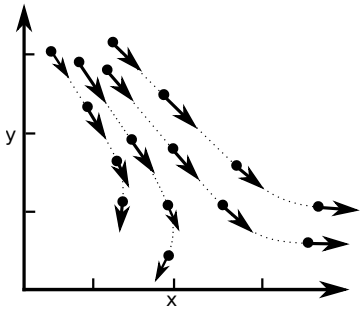


(b) Streamlines

Figure 4.2: Velocity magnitude (a) and integrated streamlines (b) from a Rayleigh-Bénard convection. The domain is filled with water, while a plate on the bottom is heated. This leads to the formation of several convection cells.

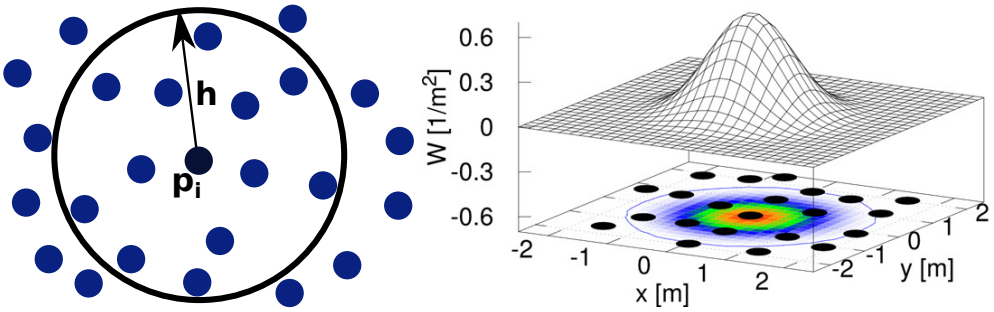


(a) Eulerian frame of reference



(b) Lagrangian frame of reference

Figure 4.3: In the Eulerian frame (a), we observe the fluid at fixed positions in space and time. In the Lagrangian frame (b), we follow the motion of fluid particles as they move through space and time.



(a) Neighborhood $\mathcal{N}(p_i)$ and smoothing length h

(b) Particle neighborhood and kernel function [46]

Figure 4.4: Illustration of the particle neighborhood and the relation to the smoothing kernel.

streamlines, computed using an embedded Runge-Kutta [180] solver, are shown in Figure 4.2.

4.2 SMOOTHED PARTICLE HYDRODYNAMICS

The smoothed particle hydrodynamics (SPH) method originates from astrophysics [88, 199], but is becoming increasingly popular in different disciplines of science and engineering, for example in computer graphics [144] or in computational fluid dynamics. The SPH method is especially well-suited for free-surface [218] and multiphase flows [175]. In contrast to the majority of current grid-based CFD approaches, it is a Lagrangian and mesh-free method based on a discretization of the computational domain using moving particles. The method is derived by first representing a function f as a convolution:

$$f(p) = \int f(p')\delta(p - p')\ \mathrm{d}p',$$

for position $p \in \mathbb{R}^d$ and with the Dirac function δ. The Dirac function is then approximated by a smoothing function or kernel $W(p - p', h)$, parametrized by the smoothing length h, which leads to:

$$f(p) \approxeq \int f(p')W(p - p', h)\ \mathrm{d}p'.$$

Finally, the integral is approximated by a finite sum resulting in the quadrature:

$$f(p_i) \approxeq \sum_j f(p_j)W(p_i - p_j, h)V_j, \tag{4.2}$$

where we have added the volume V_j of particle j. If the kernel W is compact, the approximation depends only on the neighboring particles $p_j \in \mathcal{N}(p_i)$. The size of the neighborhood is defined by the smoothing length h, as illustrated in Figure 4.4.

Spatial derivatives are approximated by differentiating the kernel function:

$$\nabla f(p_i) \approxeq \sum_j f(p_j)\nabla W(p_j - p_i, h)V_j. \tag{4.3}$$

Although it is possible to compute second derivatives the same way, this is discouraged. Instead, we use the following approximation [31]:

$$\nabla^2 f(p_i) \approxeq \sum_j (f(p_j) - f(p_i)) \frac{(p_j - p_i)^\intercal \nabla W(p_j - p_i, h)}{||p_j - p_i||} V_j. \tag{4.4}$$

The SPH method is most commonly employed to solve the Navier-Stokes equations of fluid motion. This involves the repeated computation of attracting and repelling forces and the advection of particles over time. In this thesis, we focus on the analysis and visualization based on the introduced approximations.

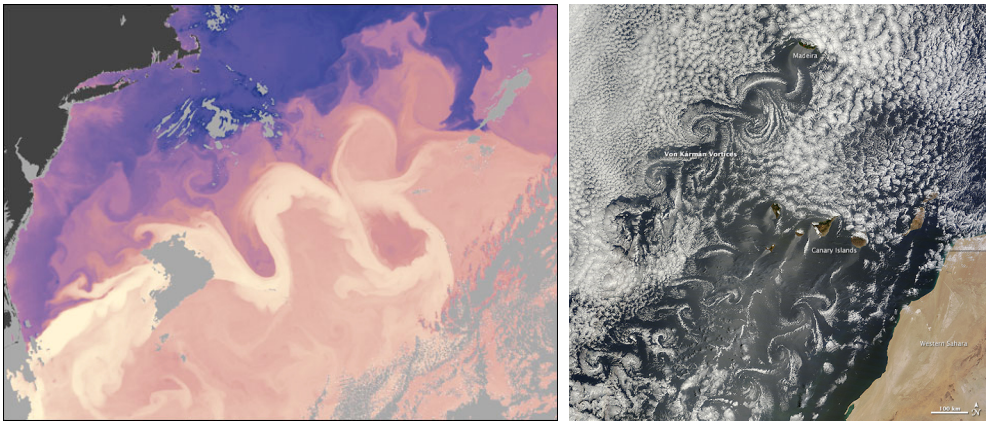
4.3 FLOW VISUALIZATION

The visualization of fluid flows is a mature, but active research topic. Existing methods can be categorized by the dimensionality of the flow in both space and time. In the spatial domain, we typically differentiate between two- and three-dimensional datasets. Flows defined on a two-dimensional manifold embedded in 3D are often referred to as 2.5D, but are untypical for particle data. In the temporal domain, we differentiate between stationary (steady) flows and time-dependent (unsteady) flows. The visualization of steady flows is considerably simplified since integrating over time does not change the behavior of the flow. This enables a compact classification of the vector field topology [184], which cannot be extended to unsteady flows. Since particle-based flows are inherently time-dependent, we will focus only on the unsteady case (Section 4.3.1). Lagrangian coherent structures are a well established topological segmentation of an unsteady flow in a finite-time interval (Section 4.3.2). Geometric approaches visualize the flow behavior by placing and integrating geometric primitives (Section 4.3.3). Lastly, there are several methods specific to particle-based flows (Section 4.3.4).

4.3.1 *Unsteady Vector Field Topology*

Topological approaches segment the flow domain into regions of different flow behavior, thus providing a compact description. For steady flows, critical points, separatrices, and closed orbits are found. Based on these features, a small set of integral lines is able to compactly represent a two-dimensional flow [126]. Although three-dimensional flows contain separating stream surfaces, the visualization becomes cluttered and difficult to understand, requiring more suitable abstractions [327]. In general, this topological segmentation is based on the flow behavior at the temporal limits of $\pm\infty$. This limit does not exist for unsteady data that are restricted to a finite-time interval, making such asymptotic assumptions problematic.

In addition, vector field topology is always relative to a reference frame [241]. This implies that the movement of an observer influences the resulting characterization of flow features. A desired property of features is that they are objective, i. e. independent of the reference frame. Hadwiger et al. [106] propose a global optimization of the reference frame to extract an observer invariant



(a) Transport of warm water in the Gulf stream [179] (b) Vortex shedding around the canary islands [179]

Figure 4.5: The transport of warm water in the Gulf stream current is revealed by the sea surface temperature (a). A satellite image of the canary islands reveals several von Kármán vortex street in the movement of clouds (b). Images are courtesy of NASA.

velocity field. Their work and the definition of objectivity has been extended to curved spaces in 2D [258]. Kim and Günther [168] employ convolutional neural networks to extract an observer independent reference frame. Baeza Rojo and Günther [265] split unsteady vector fields in two components: a vector field in which the flow becomes steady, and the remaining ambient flow that describes the motion of the topological features. A local optimization is performed to find a locally observer independent reference frame. These approaches thus enable the application of steady vector field topology to the visualization of unsteady flows.

There are more approaches to unsteady flow topology [35, 247, 321] based on feature extraction [100, 249] or space-time [82, 164, 269, 312]. Most approaches are defined in the Lagrangian frame since they characterize the behavior of time-dependent integral lines or surfaces. Lagrangian methods are generally objective, i. e. do not depend on the observer. The most prominent approach is the theory of Lagrangian coherent structures.

4.3.2 *Lagrangian Coherent Structures*

Although the trajectories of particles are seemingly chaotic, patterns can be observed even in complex flows, see Figure 4.5. These patterns are governed by Lagrangian coherent structures (LCS), a robust topological skeleton of Lagrangian particle dynamics. They are the most repelling, attracting, and shearing material surfaces in a flow. The detection of these structures from experimental or numerical flows promises a simplified understanding of the overall flow topology and the global transport and mixing behavior. Although the existence of these structures can be observed, their mathematical description is still being investigated for unsteady flows, see Shadden [281] or Haller [110] for a recent overview.

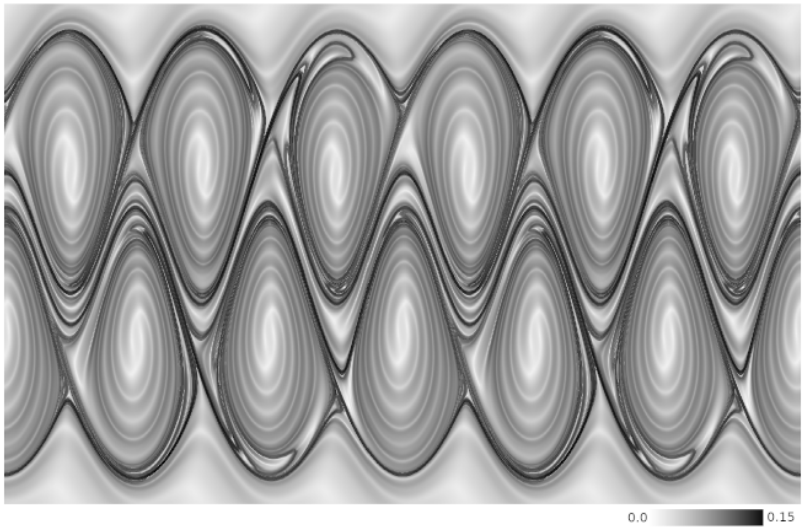


Figure 4.6: Forward finite-time Lyapunov exponent of the analytic Bickley jet [21] flow. A high FTLE indicates strong separation in the flow. Here, height ridges indicate Lagrangian coherent structures.

Finite-Time Lyapunov Exponent

In order to identify LCS, the finite-time Lyapunov exponent (FTLE) has been proposed in the seminal work by Haller [113]. This scalar measure describes the rate of separation (or attraction) with respect to infinitesimally close particles over a finite-time interval, see Figure 4.6. More specifically, let the flow map $\phi_{t_0}^{t_1}(p)$ be the map from a position p of a particle at time t_0 to its position at time t_1. The flow map can be computed by numerical integration using Equation 4.1. With the spatial gradient $\nabla\phi_{t_0}^{t_1}(p)$, the right Cauchy-Green strain tensor is given as

$$\mathbf{C}(p, t_0, t_1) := \nabla\phi_{t_0}^{t_1}(p)^\intercal \nabla\phi_{t_0}^{t_1}(p). \tag{4.5}$$

Using the largest eigenvalue $\lambda_{\max}$ of $\mathbf{C}$, the FTLE is defined as:

$$\sigma(p, t_0, t_1) := \frac{1}{|t_1 - t_0|} \ln\left(\sqrt{\lambda_{\max}\left(\mathbf{C}(p, t_0, t_1)\right)}\right). \tag{4.6}$$

The FTLE thus describes the average exponential stretching of an infinitesimally close particle neighborhood at time t_0 when the flow is integrated to t_1. The FTLE that is computed by integrating forward in time from t_0 to t_1 consequently measures the rate of separation. By integrating backward in time, the rate of attraction is computed instead, which is referred to as the backward FTLE.

Computational Efficiency and Approximation

There has been a lot of research to reduce the computational effort of the FTLE [15, 33, 52, 148, 268]. Most of the effort in the computation of the FTLE stems from the dense integration of tracer particles that is required to approximate the spatial gradient $\nabla\phi_{t_0}^{t_1}$. For Lagrangian flows, this integration

can be replaced by computing the derivative from existing pathlines. More specifically, Agranovsky et al. [4] integrate a sparse set of particles and employ moving least squares to compute the derivatives necessary for the computation of FTLE. Shi et al. [288] solve a least square fitting problem to compute the FTLE for particle data. Sun et al. [300] derive a SPH formulation for the FTLE computation, where the derivatives are obtained by deriving the smoothing kernels. However, the authors note that the computation is still expensive and not suited for large, three-dimensional datasets. In Chapter 5, we extend their approach to large, three-dimensional particle datasets.

Extracting Lagrangian Coherent Structures

Lagrangian coherent structures are material surfaces that remain coherent over time, although different definitions of coherency exist [104]. Here, we limit the discussion to the extraction of LCS based on the FTLE and the strain tensor. Haller [110] reviews further approaches to identify LCS.

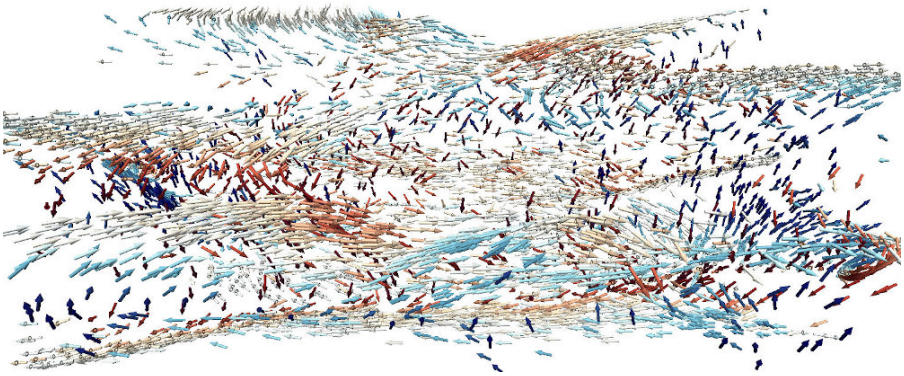
Locally maximizing surfaces in at least one dimension of the FTLE field, referred to as height ridges, are commonly used to define Lagrangian coherent structures [108, 282]. For example, the visible ridges in Figure 4.6 indicate LCS. The extraction of LCS as height ridges of the FTLE can be performed with the Marching Ridges algorithm [83]. Sadlo and Peikert [266, 267] discuss the application of Marching Ridges and describe filter operations that are used to avoid the amplification of noise. Garth et al. [86] propose an efficient approximation of the FTLE field, but prefer a direct visualization instead of ridge extraction. The authors discuss the challenge of ridge extraction from intrinsically noisy FTLE fields. Kindlmann et al. [169] propose a domain-specific language to separate the definition of complicated mathematical features from the numerical computation. Ridge extraction is one example that the authors investigate.

In several examples, Haller [109] shows that observable LCS are not necessarily FTLE ridges and vice versa. Instead, Haller proposes the definition of weak LCS. This definition is not only based on the FTLE and its derivatives, but directly on the strain tensor $\mathbb{C}$. Schindler et al. [278] propose a similar ridge concept based on the strain tensor, which they call *C-Ridges*.

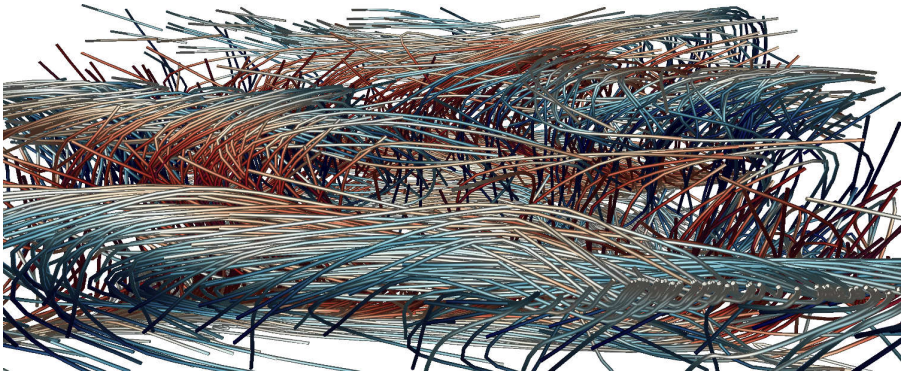
4.3.3 *Geometry-based Flow Visualization*

To visualize flow behavior, geometric primitives such as vector glyphs, integral lines and surfaces can be employed [211]. Aside from simple vector glyphs, such as the arrow glyphs in Figure 4.7 (a), these geometric primitives are based on the numerical integration of the vector field, as given by Equation 4.1. For Lagrangian flows, we can directly encode the trajectories of particles or create new ones by interpolation [3, 34]. The placement of glyphs and integral lines is a longstanding research topic [273].

Integral lines can be represented as lines, tubes, or ribbons that additionally encode the rotational component. Figure 4.7 (b) shows streamtubes in a three-dimensional Rayleigh-Bénard convection. For three-dimensional datasets,



(a) Arrow glyphs



(b) Streamtubes

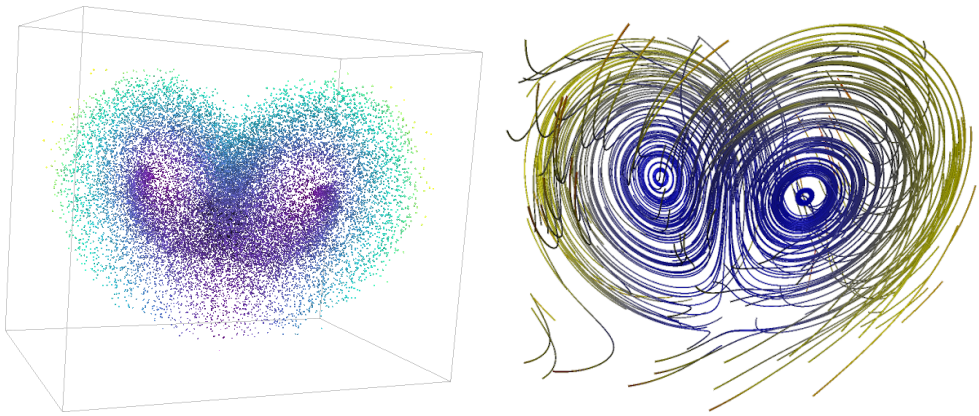
Figure 4.7: Visualization of a three-dimensional Rayleigh-Bénard convection using glyphs (a) and integral lines extruded to streamtubes (b).

illumination has been shown to greatly improve the perception of shape and depth [205]. Surface-based techniques [66] are also commonly employed to visualize three-dimensional flows, but are not directly applicable to particle data.

4.3.4 *Particle-Based Flow Visualization*

Several particle tracing systems have been proposed, where particles are advected to visualize unsteady flows, see Figure 4.8. The first particle tracing system has been presented by Lane [183]. Bruckschen et al. [32] precompute a large number of particles and store them on disk using a space-filling curve. The data is then streamed on-demand to render them in real-time as spherical glyphs. Their system has been extended to advect particles on compute clusters and to scale the out-of-core visualization to up to 293 billion particles on contemporary hardware [71].

Krüger et al. [177] present a particle tracing system for steady flows that interactively integrates particles in a vector field using GPU acceleration and displays them using different glyphs, streamlines, and streamribbons. Additionally, derived variables, such as divergence or the λ_2 criterion [152], are computed and encoded with color. Bürger et al. [37] trace particles in unsteady flows and employ focus and context visualization. Based on this



(a) Sphere glyphs (b) Streamtubes

Figure 4.8: Interactive particle tracing is used to visualize the Lorenz system [198] using animated spherical glyphs (a) and streamtubes (b).

system, Bürger et al. [36] propose an importance measure to reveal important structures of flows. Similar to their work, we use derived variables and focus and context visualization to emphasize important flow structures.

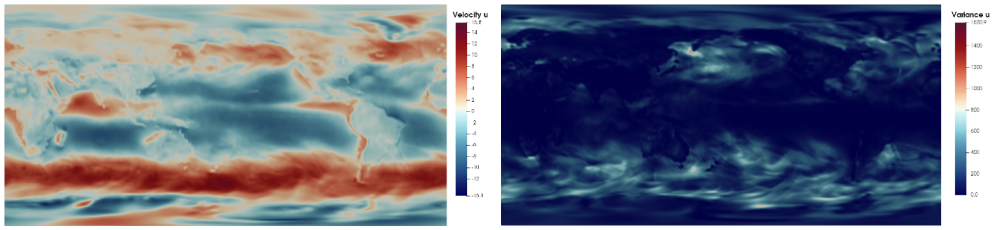
To visualize multivariate particle data, Jones et al. [155] employ transparent glyphs and pathlines. The authors emphasize the advantages of using multiple coordinated views, such as a parallel coordinate plot together with a spatial visualization, for visually analyzing particle data. To visually debug SPH simulations, Reinhardt et al. [263] similarly combine a spatial 3D view with additional views, such as scatter and parallel coordinate plots. Molchanov et al. present the SmoothViz [215] system to visualize SPH data. These studies emphasize the use of interaction and the combination of spatial and abstract views of the particle data, as discussed in Chapter 2.

To visually analyze particle trajectories over time, Salzbrunn et al. [272] propose pathline predicates, which filter a set of pathlines according to properties that are of interest to the user. Shi et al. [287] analyze unsteady 3D flow fields by computing multiple properties of selected pathlines. The resulting multivariate data is analyzed with a set of linked views, including brushing and a focus and context visualization. Lež et al. [186] further extend this analysis of pathlines.

4.4 UNCERTAIN FLOW VISUALIZATION

Uncertainty visualization has been an active research topic in the field of visualization for more than two decades [154]. Several surveys [27, 30, 238, 251] motivate uncertainty visualization, introduce its challenges, and the different sources of uncertainty. Here, we focus on the visualization of uncertain flows.

As an example, Figure 4.9 shows the longitudinal velocity of a climate simulation from the European Centre for Medium Range Weather Forecasts (ECMWF). Since climate and weather simulation models are known to be chaotic [198], ten ensemble runs with perturbed initial conditions have been performed in this example. The survey by Wang et al. [316] studies ensem-



(a) Mean longitudinal velocity (b) Variance of longitudinal velocity

Figure 4.9: Mean and variance of longitudional velocity estimated from an ECMWF climate simulation [128].

ble data in more detail. Uncertainty might also stem from an explicit error model or it might be introduced in any step of the visualization pipeline [53]. For uncertain flows, Bhatia et al. [20] analyze and quantify the uncertainty introduced by streamline integration. Chen et al. [48] and Hummel et al. [139] model the error introduced by integrating pathlines in unsteady flows.

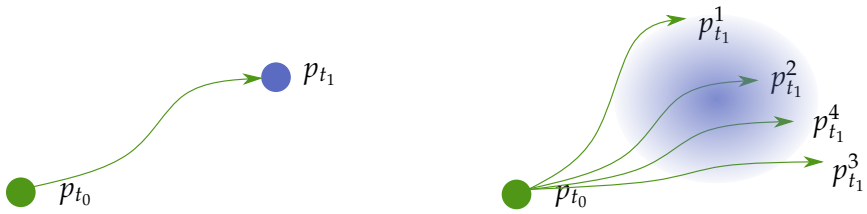
Depending on the source of uncertainties, different representations of an uncertain flow are possible. An uncertain flow can be represented by storing individual ensemble members or by modeling parametric or non-parametric probability distributions. Due to the central limit theorem, Gaussian flow fields are often employed. We refer to the discussion in Section 3.2 since modeling data distributions for data reduction is closely related to modeling uncertain data.

To visualize uncertain flows, glyph [131] and texture-based methods [28, 232] have been proposed. These approaches consider only the local uncertainty, but not the global uncertainty of the flow.

For uncertain steady flows, Otto et al. [233–235] introduce uncertain vector field topology. Here, critical points, closed orbits, and streamlines are generalized, leading to an uncertain topological segmentation. A Monte Carlo approach is employed to stochastically integrate particles. Moreover, Otto and Theisel [236] study the extraction and visualization of vortices in uncertain vector fields. Petz et al. [245] use Monte Carlo estimation in Gaussian flow fields to extract probabilistic local features. In contrast, Pöthkow et al. [250] investigate non-parametric models of uncertainty, e. g. to compute probabilities for the occurrence of features.

Uncertain Transport

In uncertain flows, particles are advected stochastically. The flow map ϕ thus describes a distribution of positions where particles might be advected, see Figure 4.10. Schneider et al. [279] estimate this stochastic flow map using a Monte Carlo approach. The authors then estimate the variance in the stochastic flow map, which defines the finite-time variance analysis (FTVA), a FTLE-like metric. Hummel et al. [140] discuss the comparative visual analysis of Lagrangian transport in CFD ensembles based on the FTVA. Hollister et al. [134] cluster ensemble members to investigate similarities in the transport.



(a) Flow map (b) Stochastic flow map

Figure 4.10: A deterministic flow map (a) maps a particle p_{t_0} to a position p_{t_1}. A stochastic flow map (b) describes a probability distribution of possible end positions.

Guo et al. [103] quantify variations in pathlines integrated from different ensemble members.

Guo et al. [101] propose two extensions of the FTLE: either by estimating the expectation of the strain tensor and then computing a single FTLE value (FTLE-D), or by estimating a distribution of FTLEs (D-FTLE). Both approaches depend on Monte Carlo estimation of the stochastic flow [102]. In Chapter 7, we introduce and discuss a quantity that does not require expensive Monte Carlo estimation and is built upon a more solid theoretical foundation.

Part I

LAGRANGIAN FLOW VISUALIZATION

5 LAGRANGIAN COHERENT STRUCTURES

In this chapter, we discuss the efficient identification of Lagrangian coherent structures (LCS) from particle data. The finite-time Lyapunov exponent (FTLE), introduced in Section 4.3.2, is commonly employed to visualize and identify the LCS. To compute the FTLE, the spatial gradient of the flow map must be approximated, which requires a costly, dense numerical integration. In contrast, we propose an efficient algorithm to compute the finite-time Lyapunov exponent for particle data (Section 5.1). We then employ the FTLE to efficiently and robustly identify Lagrangian coherent structures (Section 5.2). This fast characterization of the finite-time flow dynamics enables the interactive exploration of particle datasets, but also to explore the parameter space. We validate our approach in several numerical experiments (Section 5.3).

5.1 FINITE-TIME LYAPUNOV EXPONENT

We first review a computational SPH scheme for the evaluation of the FTLE. Afterwards, we present a scalable algorithm that uses GPU acceleration. We can thus compute the FTLE interactively even for large, three-dimensional data.

5.1.1 *FTLE in the SPH Framework*

Since our data already consists of particle trajectories, the numerical integration of tracer particles to compute the flow map is not necessary. Instead, we employ the regular SPH quadrature for interpolating quantities and their spatial derivatives. In detail, we build upon the method from Sun et al. [300] to evaluate the finite-time Lyapunov exponent.

The computation of the FTLE in the time interval $[t_0, t_1]$ depends only on the particle positions at time t_0 and t_1. Intuitively, the FTLE then measures the deformation of the particle neighborhood over the time interval. In the following, we denote the position of a generic neighboring particle j in time t as $p_{t,j}$. Similarly, the volume of particle j in time t is referred to as $V_{t,j}$.

With the following formula from Sun et al. [300], the spatial derivative of the flow map is computed as follows:

$$\nabla \phi_{t_0}^{t_1}(p_i) \approxeq \sum_j (p_{t_1,i} - p_{t_1,j}) \otimes \mathbb{L}(p_{t_0,i}) \nabla W(p_{t_0,i} - p_{t_0,j}, h) V_{t_0,j}, \tag{5.1}$$

$$\mathbb{L}(p_{t_0,i}) \approxeq \left[\sum_j (p_{t_1,i} - p_{t_1,j}) \otimes \nabla W(p_{t_0,i} - p_{t_0,j}, h) V_{t_0,j} \right]^{-1}. \tag{5.2}$$

The symbol $\otimes$ denotes the dyadic product. The tensor $\mathbb{L}$ is used for correcting the kernel depending on the particle arrangement [26]. Once $\nabla \phi_{t_0}^{t_1}$ is known,

the FTLE is computed using Equation 4.6. Note that the backward FTLE can be determined by simply reversing the time interval.

5.1.2 *Computational Complexity*

When computing fields of the FTLE on grid-based data, a transformation from the Eulerian to the Lagrangian frame has to be performed. Since SPH datasets are inherently Lagrangian, this computational step is not needed. However, SPH simulations typically contain a large amount of particle data and an inefficient implementation will not scale well with the data size. In order to realize an efficient and scalable implementation, the data structures and algorithms have to be carefully selected. The whole procedure of the FTLE computation used in this thesis is illustrated in Fig. 5.1. Although the procedure shares some commonalities with traditional SPH simulation codes, it still addresses a unique problem: neighboring particles at time step t_0 have to be efficiently queried and located at any other time step t_1.

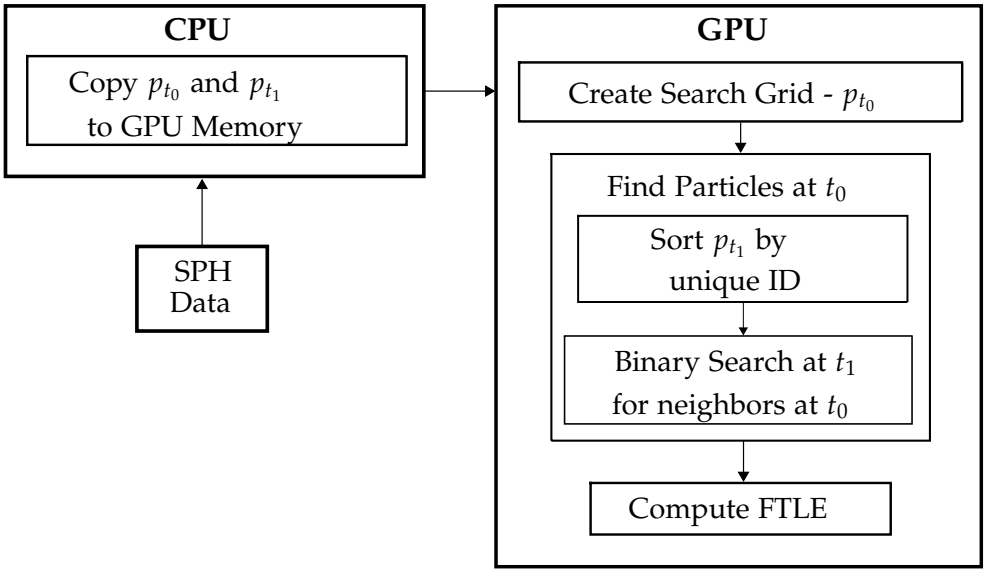


Figure 5.1: Illustration of the steps to compute the FTLE.

To efficiently employ the SPH weighting kernels, all neighbors contained in the support domain of each particle in step t_0 need to be identified as indicated on the left hand side of Fig. 5.2. This is similar to the neighborhood search performed during a SPH simulation. However, it has to be performed only once for every particle for the first time step. Similar to search grids used for SPH simulations (see e.g. Ihmsen et al [144] for an overview) the computational domain is represented by a uniform grid, where each particle is assigned to one cell. Since the cell size is equal to the kernel support, only the direct neighbors have to be queried during the neighborhood search. To create the grid in parallel, the particles are sorted according to their cell index and a unique key is assigned to each cell [159, 252]. For each grid cell only a reference to the first particle in the sorted list is stored. To improve the spatial

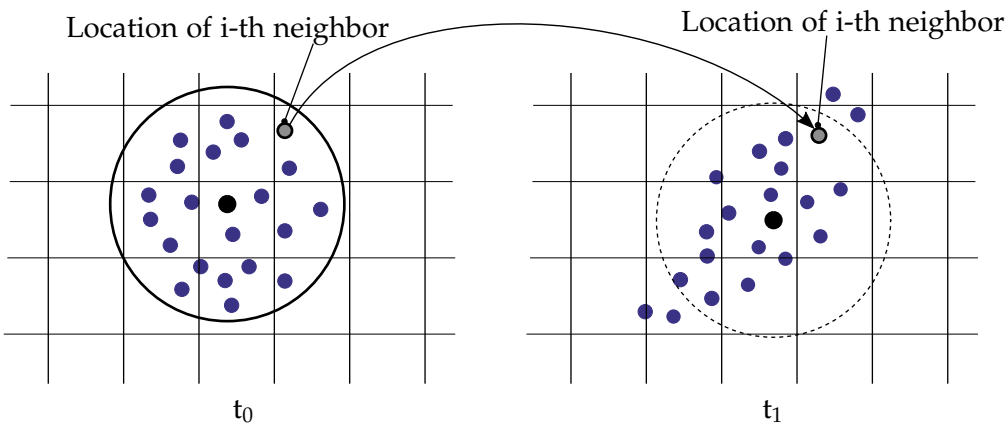


Figure 5.2: Configuration of particles and search grid at time t_0 and t_1.

locality of neighboring cells, the cell index is computed using a space-filling Z-curve similar to the methods presented by Goswami et al. [91] and Ihmsen et al. [143]. A requirement for this approach is that every particle is uniquely identified at each time step, because for every particle in step t_0, the same particle has to be located for step t_1 as illustrated in Fig. 5.2.

If the particles are not sorted, the whole set of particles in time t_1 has to be searched for the corresponding particle for every particle in t_0. Moreover, if particles are entering or leaving the domain, there can be particles in step t_0 that do not exist anymore in t_1 and vice versa. Hence, as a pre-process, in time step t_0 an index is assigned to the matching particle in t_1. By first sorting the particles in time step t_1 according to its identifier, this mapping can be efficiently computed by a binary search.

5.1.3 *GPU Acceleration*

CUDA [228] is a programming model that enables the use of GPUs for general purpose computations. A GPU is a highly parallel multi-core processor that can significantly outperform a CPU on tasks that can be executed in parallel. In both workstations and computing clusters, GPUs are employed as accelerators to which suitable workloads can be offloaded. Since most GPUs can only address their dedicated memory, memory allocation and transfer of data has to be managed by means of CUDA. All computations are defined in CUDA kernels that are run by millions of threads in parallel.

The computation of the FTLE is inherently parallel since the value of the FTLE of one particle does not rely on the one of others. Thus, a CUDA kernel can be used for every particle at the time t_0 to perform all computations in parallel. During the computation, the previously created uniform grid and the particle mapping from t_0 to t_1 are used.

The uniform grid is created on the GPU using a kernel to assign a cell index to each particle in step t_0. Afterwards, the particles are sorted according to this cell index and their positions are written to the grid cells in another kernel. With the list of sorted particles and the grid cells, containing references to the sorted particles, all neighbors of a given particle can efficiently be identified.

To find matching particles on the GPU, the particles in t_1 are sorted by their unique identifier. Subsequently, a kernel is executed for every particle in t_0 to perform the binary search in parallel.

5.2 IDENTIFYING LAGRANGIAN COHERENT STRUCTURES

As previously discussed, height ridges can be defined as Lagrangian coherent structures [108, 282]. Additionally, we consider the concept of weak LCS, as introduced by Haller [109].

5.2.1 *Definitions*

To detect LCS as $(d-1)$-dimensional height ridges within the d-dimensional FTLE field σ at position p, we compute the gradient $\nabla\sigma(p)$ and Hessian $H_\sigma(p)$, the smallest eigenvalue $e_{\min}$ and corresponding eigenvector $v_{\min}$ of $H_\sigma(p)$. The following two criteria must hold:

$$c_1^h(p) := \nabla\sigma(p) \cdot v_{\min} = 0, \tag{5.3}$$

$$c_2^h(p) := e_{\min} \leq 0. \tag{5.4}$$

Note that we relax the inequality in the second criterion to include thick ridges, similar to Farazmand and Haller [73].

We also use the more accurate definition of weak LCS, but reformulate it to fit into two criteria. To this end, we require the strain tensor $\mathbb{C}(p, t_0, t_1)$ that is used to compute the FTLE. Specifically, we compute eigenvalues λ_i of $\mathbb{C}(p, t_0, t_1)$, with $\lambda_1 \leq \cdots \leq \lambda_i \leq \cdots \leq \lambda_d$, and the corresponding eigenvectors ξ_i. Since the major eigenvector ξ_d of $\mathbb{C}(p, t_0, t_1)$ is equal to the direction of maximal stretching, it must be orthogonal to the LCS:

$$c_1^w(p) := \langle \nabla\lambda_d, \xi_d \rangle = 0. \tag{5.5}$$

For the second criterion, we reformulate Equation 5.4 since LCS must occur at a local maximum in direction ξ_d. An additional condition ensures that the normal repulsion rate is larger than the tangential stretch, typically caused by shearing in the flow. To summarize, we compute the second criterion as:

$$c_2^w(p) := \langle \xi_d, \nabla^2\lambda_d\xi_d \rangle \leq 0 \wedge \lambda_{d-1} \neq \lambda_d \wedge \lambda_d > 1. \tag{5.6}$$

With these criteria, we fulfill three of the conditions necessary for a weak LCS (cf. Haller [109], theorem 7). Note that the height ridge definition only fulfills two of those conditions, but does not depend on the strain tensor. We can make use of both definitions and thus refer to the criteria simply as c_1 and c_2.

5.2.2 *Grid Extraction*

We shortly recapitulate the extraction of LCS for uniformly structured data using Marching Ridges. Here, the locations where the first criteria c_1 are met is found using Marching Cubes [197], i.e. computed at the grid vertices and

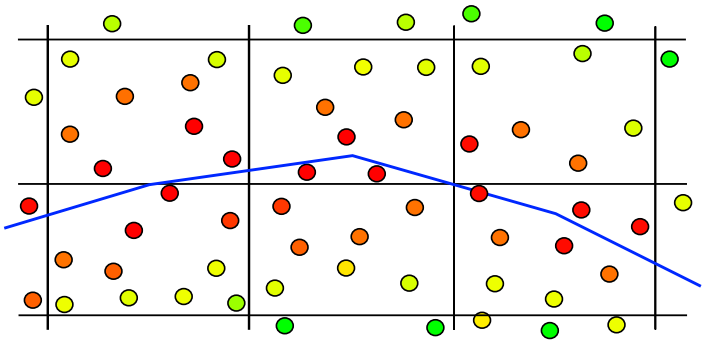


Figure 5.3: Illustration of the per-particle LCS distance (increasing from red to green over yellow), and the corresponding LCS created by marching ridges (blue).

approximated using linear interpolation inside of each grid cell. Note that the orientation of eigenvectors in a cell must be made consistent, which can be performed using principal component analysis. Afterwards, a triangle is created only if the second criterion c_2 is met for the corresponding edges.

5.2.3 *LCS from Particle Data*

Performing Marching Ridges on particle data would require resampling the FTLE to a grid and computing the derivatives afterwards. This step is computationally inefficient and is prone to amplify noise in the derivatives. Instead, we compute the derivatives and evaluate the LCS criteria for each particle using its local neighborhood. Similar to Marching Ridges, we compute zero crossings of the first criteria from a particle with respect to its neighboring particles; however, we cannot triangulate the zero crossings. Instead, we compute the minimal distance to any LCS in its neighborhood for each particle, cf. Figure 5.3. In the following, we discuss this procedure in detail.

Given a particle at position $p \in \mathbb{R}^d$, we compute the distance from p to the nearest LCS in its neighborhood $\mathcal{N}(p)$, which contains all particles inside radius r around p. First, we make the orientation of eigenvectors for the particle and its neighbors consistent using PCA. Afterwards, we check each neighboring particle $p_j \in \mathcal{N}(x)$ if $c_1(p_j)$ has a different sign than $c_1(p)$. In this case, the position of the zero crossing $\mathrm{LCS}(p, p_j)$ between the two particles can be determined using linear interpolation. Finally, if the zero crossing exists, and both particles additionally fulfill criterion c_2, we obtain the distance to p as:

$$d_{\mathrm{LCS}}(p, p_j) := \begin{cases} \infty & \text{if } \neg \left(c_2(p) \wedge c_2(p_j)\right), \\ \infty & \text{if } \operatorname{sign}(c_1(p)) = \operatorname{sign}(c_1(p_j)), \\ ||\,\mathrm{LCS}(p, p_j) - p|| & \text{otherwise.} \end{cases} \tag{5.7}$$

We define the LCS distance for p as the minimal distance to the LCS towards all neighboring particles:

$$d(p) := \min_{p_j \in \mathcal{N}(p)} d_{\mathrm{LCS}}(p, p_j). \tag{5.8}$$

(a) FTLE (b) Height ridge LCS (c) Weak LCS

Figure 5.4: The forward FTLE of the double gyre computed from the integrated particle trajectories is shown in (a). The LCS distance mapped to color is shown in (b) using the height ridge criteria and in (c) using the weak LCS criteria.

This function defines an unsigned distance field. Even though we can compute points on the LCS using linear interpolation, an extraction of a surface is in general not possible since the distances are unsigned. Instead, we store for each particle the distance to the closest LCS. These per-particle distances can be added to the dataset and used for further analysis and visualization. Lastly, we normalize the distance using the radius r of the particle neighborhood:

$$\mathcal{L}(x) := \frac{d(p)}{r}. \tag{5.9}$$

5.2.4 *GPU Acceleration*

We perform all computations on the particles in parallel on the GPU by splitting them in two steps: first, we compute the spatial derivatives and evaluate the LCS criteria for every particle using its local neighborhood. In the second step, we compute the LCS distance for each particle. For a particle at position p, given either vector v_{min} to detect height ridges or ξ_d for the weak LCS definition, we determine the mean vector and the covariance matrix from the local neighborhood $\mathcal{N}(p)$. Then we extract the major eigenvector of the covariance matrix, according to which we orient the vectors of all particles in the neighborhood. Lastly, we evaluate the zero crossings and compute the distance from p to the closest LCS.

5.3 NUMERICAL EXPERIMENTS

In this section, we validate the computation of the FTLE and the extraction of LCS in several experiments. The numerical setup and validation of the flow around a cylinder in 2D (cf. Section 5.3.2) and a surface-mounted cylinder in 3D (cf. Section 5.3.3) are described in more detail in our publication [55]. As kernel function, we use a quintic spline [192].

5.3.1 *Double Gyre*

The double gyre is a two-dimensional periodic unsteady vector field that describes two counter rotating gyres. It is commonly used for the validation of FTLE and LCS. We use the following definition:

$$v(x,y,t) = \begin{pmatrix} -\pi A \sin(f(x,t)\pi)\cos(y\pi) \\ \pi A \cos(f(x,t)\pi)\sin(y\pi)\frac{\partial}{\partial x}f(x,t) \end{pmatrix}, \tag{5.10}$$

where

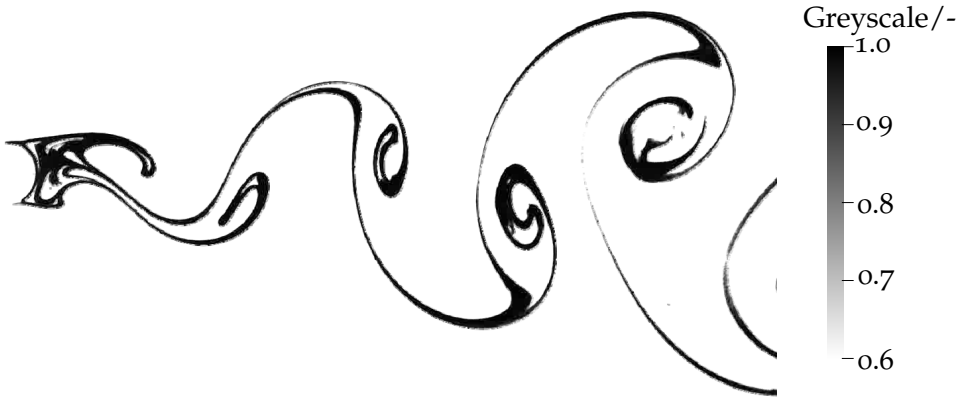
$$\begin{aligned} f(x,t) &= a(t)x^2 + b(t)x \\ a(t) &= \epsilon \sin(\omega t) \\ b(t) &= 1 - 2\epsilon \sin(\omega t). \end{aligned}$$

We set $A = 0.1$, $\omega = 2\pi/10$, and $\epsilon = 0.1$. We convert this analytic vector field into a particle-based representation by integrating particles using a fourth order Runge-Kutta scheme and sampling the trajectories at fixed time steps of 0.1. The particles are uniformly seeded in the domain $[0,2] \times [0,1]$, but jittering is applied to reduce aliasing artifacts. In total, this leads to 2048×1024 particles. Then, we compute the forward FTLE over the time interval $[0,20]$. We apply an FTLE threshold of 0.08 to compute the LCS distance.

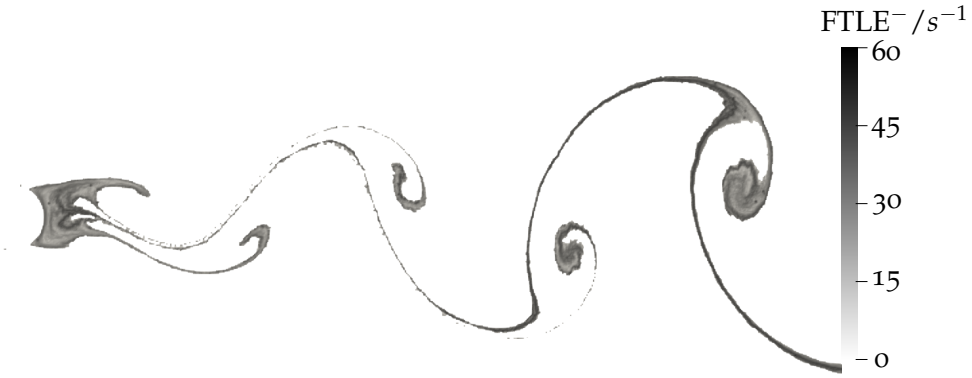
Figure 5.4 shows FTLE and LCS distance by applying a linear color map. The FTLE shows the presence of the two counter rotating gyres. The LCS distance visualizations in (b) and (c) depict the expected coherent structures, cf. Farazmand and Haller [73]. The ridges can also be visually inferred from the FTLE field. As expected, the height ridge and weak LCS criteria identify the same coherent structures, but the height ridge LCS are thicker.

5.3.2 *Flow Around a Circular Cylinder in 2D*

The flow around circular cylinders is relevant to a variety of technical applications, as discussed by Zdravkovich [340]. In our numerical experiment, particles enter a rectangular domain on the left side and exit on the right, where each time step contains approximately 5 million particles. The circular cylinder in the middle of the domain causes vortex shedding downstream of the cylinder. The resulting von Kármán vortex street is revealed by the (attracting) Lagrangian coherent structures. In Figure 5.5, we compare an experimental result from Taneda [303] in (a) with the backward FTLE (b) and the weak LCS distance (c). Compared to the experimental results, the FTLE and LCS reveal qualitatively similar structures. Although the LCS distance identifies the transport barriers in the flow, it leads to a reduced accuracy compared to the FTLE. From experiments with a higher resolution we have confirmed that this loss of accuracy is due to the resolution. This result is not surprising considering the use of second order derivatives in the computation of the LCS distance.



(a) Experimental result



(b) Backward FTLE

(c) LCS distance

Figure 5.5: Comparison of the experimental visualization result from Taneda [303] (a), backward FTLE (b), and LCS distance (c).

5.3.3 *Surface-Mounted Cylinder in 3D*

This flow consists of a surface-mounted, circular cylinder with a free end. The cylinder is placed on the bottom of a box-shaped channel. The particles move through this channel, entering on one and leaving on the other side, leading to approximately 46 million particles per time step. Figure 5.6 shows a sketch of the different vortices occurring in such a flow. The volume rendering of the FTLE field shown in Figure 5.7 identifies similar structures in our numerical experiment. Most notable is the horseshoe vortex that forms in front of the cylinder. On top of the cylinder, the two tip vortices lead to an indented wake downstream of the cylinder.

The LCS distance also indicates the existence of these structures. Moreover, we can use the LCS distance to select particle close to coherent structures and visualize, for example, arrow glyphs and pathlines, see Figure 5.8. In contrast to the FTLE, this visualization shows the presence of the arch vortex directly behind the cylinder in flow direction.

5.3.4 *Performance and Scalability*

In order to evaluate the computational efficiency of our method and implementation, the time for the computation of the backward FTLE is measured. The datasets used for the assessment are the 2D and 3D cylinder flows in different resolutions.

The runtime required by the GPU implementation is compared to the runtime required by a multi-threaded shared-memory CPU implementation. The GPU and the CPU implementation are based on the same algorithms in order to enable a fair comparison. All measurements are performed on a Nvidia GTX 1080Ti with 11GB of dedicated memory and an Intel Core i7-6000 featuring four physical cores with additional hyper-threading. The measurement results are shown in Table 5.1 for 2D and 3D cases at different spatial resolutions. The previously outlined steps of the method are included: creation of the uniform grid, mapping of the time steps, and computation of the FTLE. Note that the execution times of the individual steps of the method do not sum up to the total time since there is additional time required for memory allocation and copying to the GPU. The execution time of the GPU implementation is at least one order of magnitude faster than the CPU implementation. Per GB memory on the GPU around 27 Mio. of particles can be handled as follows from the peak memory allocation. This value is approximately constant for all domain sizes. However, on our and newer hardware we use virtual addressing and automatic page migration to automatically migrate non-resident memory to the GPU. We were thus able to apply our implementation to data sets containing over 300 million particles per time-step.

In Figure 5.9 (a) the runtime of the different implementations is presented for the 2D case in order to demonstrate scalability. The GPU implementation is not only significantly faster, but also scales better with respect to the number of particles. Therefore one can conclude that the GPU implementation is suitable for the aspired use case. It is expected that the scalability persists also for even

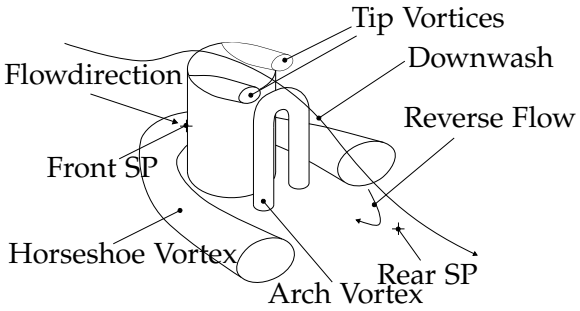


Figure 5.6: Vortices occuring at the surface-mounted cylinder according to Pattenden et al. [239].

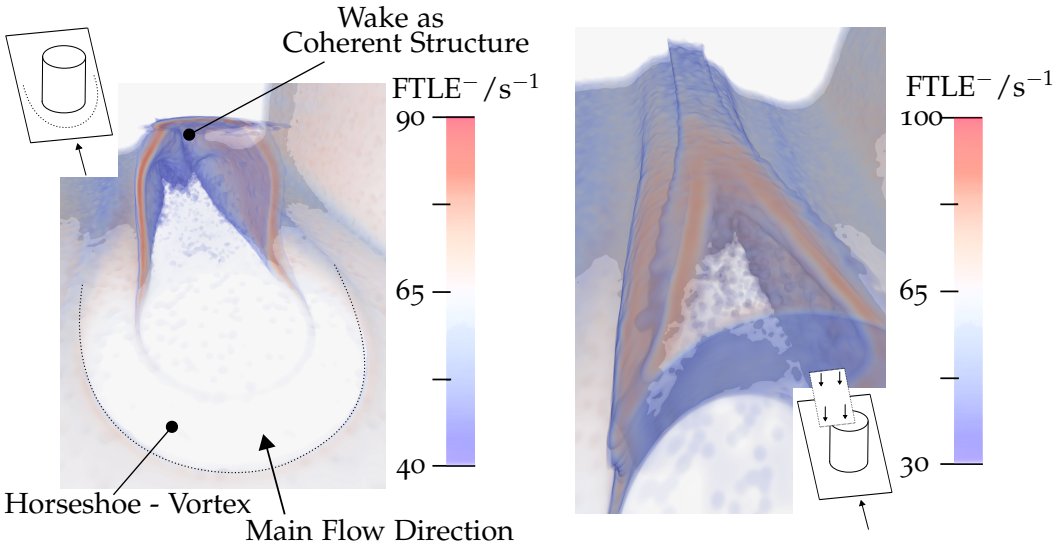


Figure 5.7: The backward FTLE of the surface-mounted cylinder in 3D depicts the expected coherent structures, including the prominent horseshoe vortex that forms in front of the cylinder and the tip vortices that indent the wake behind the cylinder.

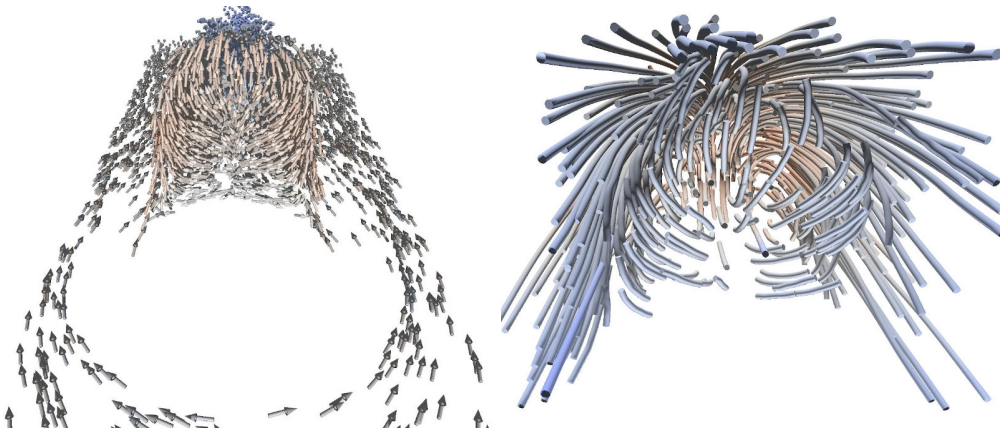
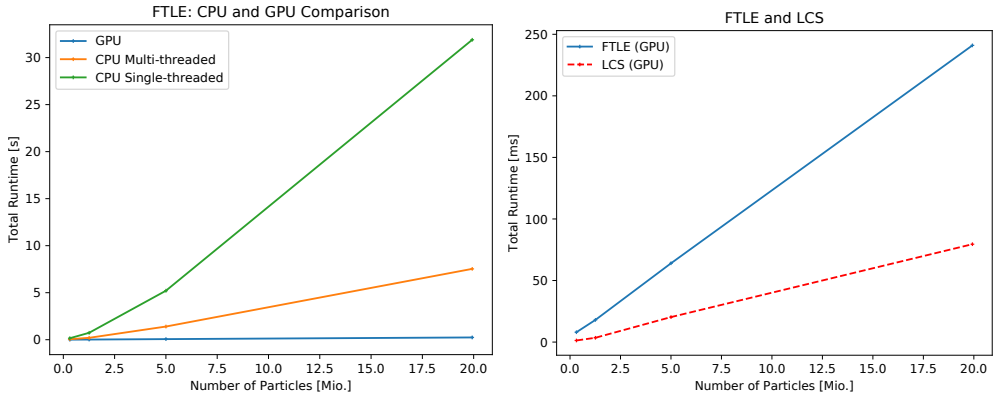


Figure 5.8: Arrow glyphs and pathlines created from particles close to the attracting coherent structures give a different perspective on the flow and reveal the arch vortex behind the cylinder.

Table 5.1: Run-time measurements of the FTLE computation.

			Execution time (GPU)				
Dim.	Particles t_0	Memory	Grid	Mapping	FTLE	**Total**	Total CPU
2D	317,093	11.7 MB	2.9 ms	1.2 ms	0.3 ms	8 ms	54 ms
2D	1,257,697	46.5 MB	4.3 ms	1.9 ms	3.0 ms	18 ms	202 ms
2D	5,001,649	185.5 MB	6.7 ms	5.1 ms	21.8 ms	64 ms	1,401 ms
2D	19,929,578	704.7 MB	20.7 ms	18.5 ms	93 ms	241 ms	7,534 ms
3D	17,043,633	622.5 MB	18 ms	16 ms	68 ms	195 ms	6,445 ms
3D	46,610,372	1702.9 MB	48 ms	43 ms	233 ms	585 ms	23,438*ms*
3D	58,019,422	2088.2 MB	56 ms	53 ms	181 ms	604 ms	40,420 ms

larger data sets. Even though there is no inherent restriction to the size of the data for the GPU method, the implementation is simplified if all of the data fits into the GPU memory at the same time, which is the case for all the simulations considered in this thesis. Figure 5.9 (b) compares the GPU runtime of the FTLE computation and the LCS identification. Computing first and second-order spatial derivatives to identify the LCS is fast and is similarly well-suited for GPU acceleration.



(a) FTLE (b) LCS

Figure 5.9: Run-time measurements on the flow around a circular cylinder in 2D. In (a), we compare the runtime of the FTLE computation on the CPU and GPU. The GPU runtimes of the FTLE and LCS are illustrated in (b).

5.4 DISCUSSION

In this chapter, we introduce a novel computational method to efficiently compute the FTLE and LCS from particle data. Although our method is based on the SPH framework to perform interpolation, it is not inherently limited to SPH data. In fact, the interpolation method could be easily replaced with a different method such as moving least squares [4].

Although the computation on particle data does not depend on numerical integration, irregular data requires fast neighborhood queries and efficient

data management. In contrast to well established numerical methods for flow integration, this currently requires custom software. However, our method enables an extremely fast computation of the FTLE and LCS. Since flow visualization requires the evaluation of several parameters, such as the time interval, the computational efficiency is important for interaction. In practice, loading the data from disk is by far the biggest bottleneck of our method, emphasizing the need for data reduction specific to particle data.

6 VISUAL ANALYSIS OF TRANSPORT AND MIXING

In this chapter, we discuss the visual analysis of particle-based flows using the finite-time Lyapunov exponent and the distance to the nearest Lagrangian coherent structure, as introduced in the previous chapter. We discuss the integration of these Lagrangian flow features into a visual analysis framework. This enables domain scientists to explore the correlation of these Lagrangian flow features to the multivariate particle data and thus to gain insight into complex flow behavior.

For an effective particle-based flow visualization, the amount of visual occlusion has to be reduced to make important parts of the flow visible. This is crucial for particle data where datasets contain millions of particles, most of which are not of immediate interest to the user. We employ the FTLE, LCS distance, and the multivariate particle data to select particles of interest in multiple linked views on the data, as discussed in Chapter 2. For example, Figure 6.1 depicts a parallel coordinate plot, where a low LCS distance has been brushed. Consequently, particles near a LCS are selected and we can immediately see how these particles are correlated to and distributed in other dimensions. We discuss our visual analysis framework in Section 6.1. We extend this concept to particle trajectories in Section 6.2.

In Section 6.3, we specifically consider the application to multiphase fluid flows, which are a prominent application of particle-based methods. Multiphase fluid flows contain two or more distinct fluid phases, for example a liquid and a gas phase. We apply our approach to visualize the transport and mixing of the different flow phases. Since the Lagrangian coherent structures act as transport barriers that are minimally diffusive, they are closely related to the interface between two phases, e.g. liquid and gas particles, and their mixing properties [208]. By correlating the phase interface with the LCS, we find an effective visualization of the mixing behavior in multiphase flows.

The potential of combining visual analysis and feature extraction has already been demonstrated in previous work [39, 186, 287]. Our findings indicate that the visual analysis based on Lagrangian coherent structures is an effective approach to study the transport behavior in particle flows. In collaboration with

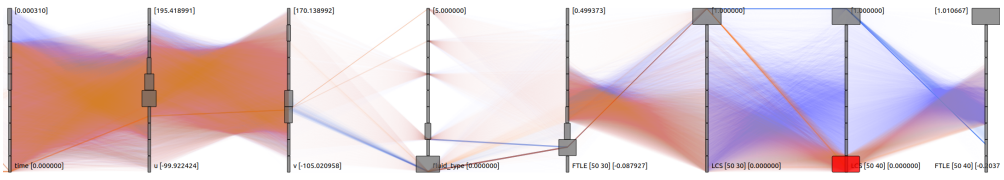


Figure 6.1: The parallel coordinate plot shows all variables of each particle at once. Here, particles close to a LCS are brushed (red) on the corresponding axis. The brushed lines, highlighted in orange, allows correlating particles near a LCS to the other dimensions, for example to the FTLE (on the right of the brushed axis) or to LCS over a different time interval (on the left).

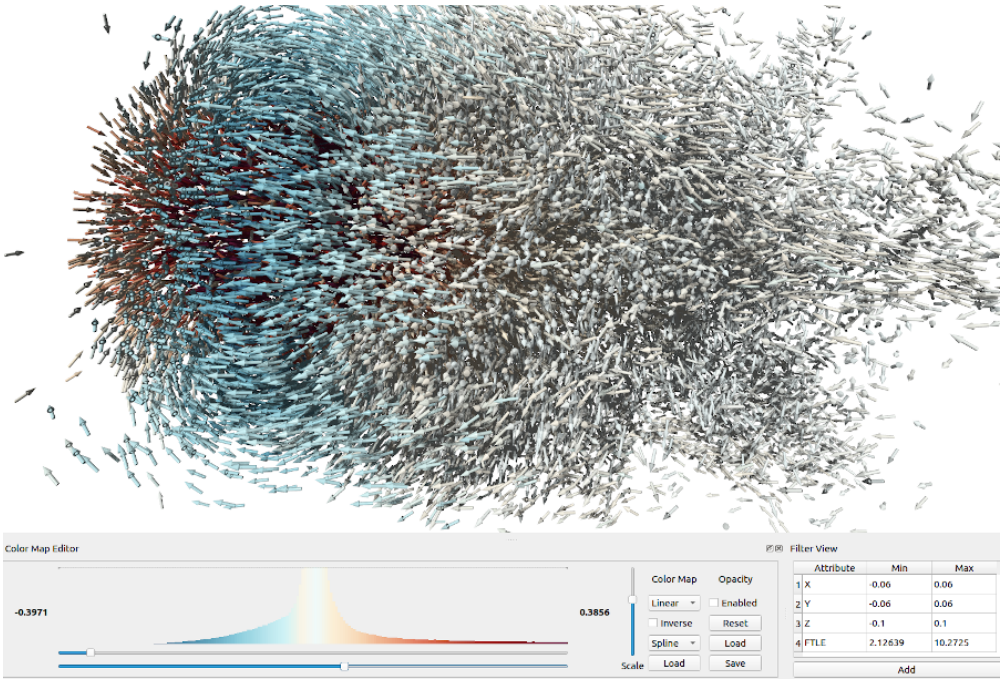


Figure 6.2: The spatial view (top), transfer function editor (bottom left) and a view of the currently active brushes, i. e. range queries performed on the data (bottom right).

domain experts, the proposed methodology has been successfully employed to discover and validate a modified spray nozzle geometry in a jet engine that leads to reduced emissions [54]. We present this and another case study in Section 6.4.

6.1 VISUAL ANALYSIS FRAMEWORK

We now introduce our visual analysis approach to support scientists and engineers in gaining insight into large particle-based flows. Most prominent is the spatial view that displays particles or their trajectories using glyphs (cf. Section 2.2.1) or splatting (cf. Section 2.2.2). A transfer function is employed to map a selected variable to color and opacity. Both of these views are depicted in Figure 6.2. To visualize the multivariate value domain and derived variables, we use multiple coordinated views. These are all linked so that brushing in one view is reflected in all of the others. To add or refine brushes, we also offer a view to directly enter or modify the queries, see Figure 6.2. Brushed values are highlighted differently in each view, for example using color in Figure 6.1. The spatial view shows only brushed particles, which is necessary to reduce clutter and occlusion. Especially for three-dimensional datasets, we found that drastically reducing the number of visible particles is necessary to make regions and features of interest visible. Derived variables and features, coupled with interaction, efficiently support this exploration.

6.2 VISUALIZING THE TOPOLOGY OF TIME-DEPENDENT FLOWS

We now discuss the visualization of unsteady flows by selecting a sparse set of particles and their trajectories that best represent the flow topology. For particle-based flows, trajectories are obtained by interpolating between the positions of particles over time. Note that these are not necessarily equivalent to massless tracer particles integrated in a velocity field. In particular, these particles are less affected by the exponential error growth near LCS. This is due to the particle-based simulations that apply attracting and repelling forces to the particles that enforce a uniform discretization of the domain. Moreover, these trajectories are well-suited to visualize particle-based fluid flows since they depict the actual movement of particles and do not depend on numerical integration. By brushing and linking, particles and trajectories can be selected intuitively using any of the views. Using the proposed per-particle LCS distance, trajectories near LCS are thus easily selected and visualized.

6.3 VISUALIZING MIXING IN MULTIPHASE FLUID FLOWS

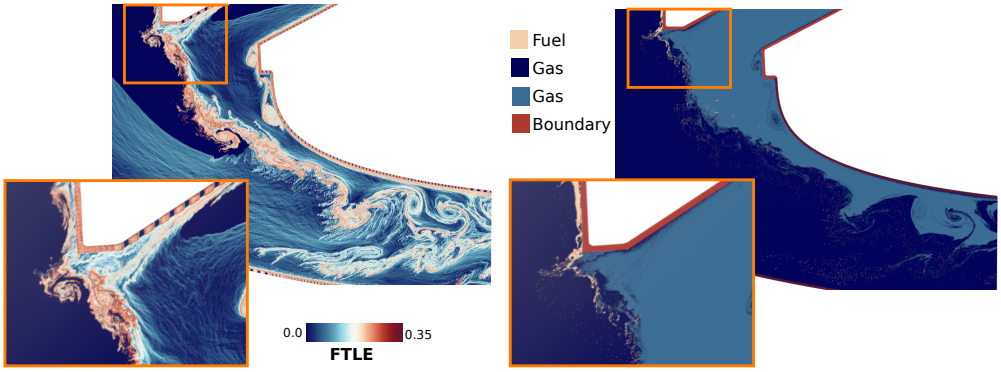
Multiphase fluid simulations contain particles of different types. By color mapping each particle according to its type, the distinct phases and their interface become visible, cf. Figure 6.3. The visualization strongly indicates the presence of vortices and separatrices, i.e. separating lines with little cross flux, that are strongly related to the mixing behavior. To gain insight into the dynamics of the phase interface and the mixing regions over time, we identify the transport barriers in the form of LCS. As shown in Figure 6.3, we can display LCS together with the phase interface by emphasizing particles near a LCS. Although the FTLE also indicates the transport behavior, cf. Figure 6.3 (a), the sparse representation of the coherent structures significantly reduces visual clutter. Since the LCS are minimally diffusive, little to no mixing should occur across the boundaries. Accordingly, the separatrices coincide with the LCS, whilst vortices near phase interfaces indicate possible mixing regions. The LCS further show the behavior of the flow surrounding the interface, which influences the evolution of the interface over time.

6.4 CASE STUDIES

To detail the visual analysis in two case studies, we make use of a two- and a three-dimensional SPH dataset.

6.4.1 *Fuel Spray*

This two-dimensional dataset from a simulation of a fuel spray contains about 12.6 million particles per time step. As shown in Figure 6.3 (b), fuel particles (yellow) are injected on the top left and mix with two distinct gas phases (light and dark blue). The atomization of the fuel into a spray of fine particles in the



(a) Backward finite-time Lyapunov exponent (b) Multiphase visualization

Figure 6.3: The backward finite-time Lyapunov exponent (a) and a visualization of the different fluid phases (b) in the spray nozzle dataset.

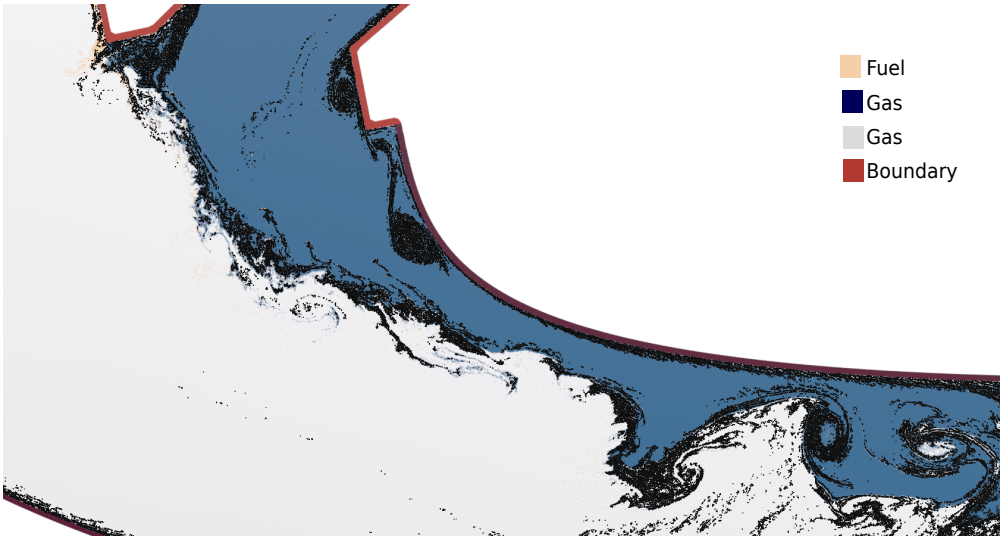
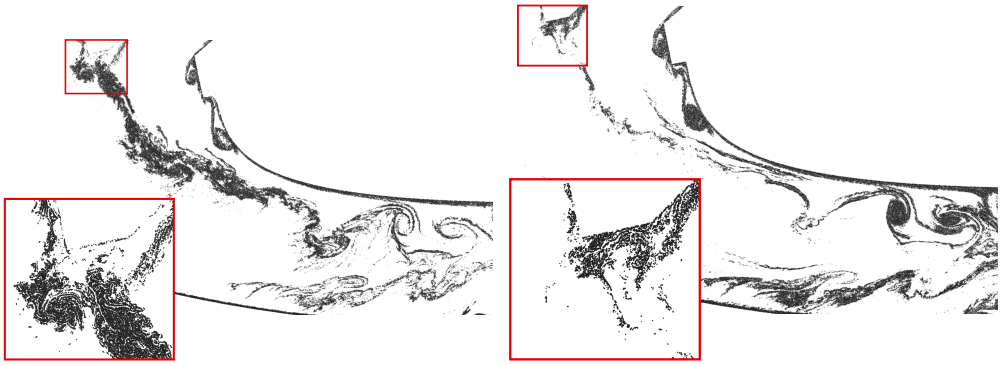


Figure 6.4: We visualize time-dependent transport and mixing in a multiphase fluid simulation of a fuel spray nozzle. The different fluid types are illustrated together with particles close to Lagrangian coherent structures (black).

surrounding gas phase is investigated by domain scientists to determine the quality and characteristics of the spray.

Figure 6.3 (a) shows the backward FTLE. The LCS together with the fluid phases are depicted in Figure 6.4. The LCS capture most of the ridges in the FTLE field and give a clear indication of the global transport and mixing behavior. On smaller scales, the LCS seem to lose some accuracy compared to the FTLE, indicating a lack of resolution.

Since the computation of the FTLE and LCS is fast, it is possible to quickly explore different time intervals. As shown in Figure 6.5 (a) and (b), a large interval smooths out short-lived structures, whilst a small time interval emphasizes these, but might fail to detect longer-lived structures. We further correlate the LCS over different time intervals, for example using the parallel coordinate plot in Figure 6.1, to determine which structures exist in one or both time scales. From the LCS in different time scales shown in Figure 6.5



(a) Small time interval (b) Large time interval

Figure 6.5: Comparing the coherent structures in a small (a) and large (b) time interval.

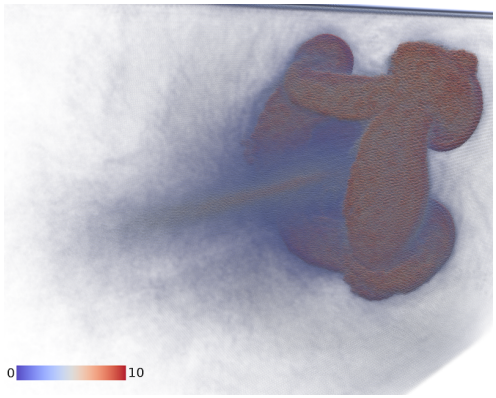
(a) and (b) and the parallel coordinate plot, we gather that the vortical region (red) is short lived. The fluid phase visualization in Figure 6.3 (b) indicates strong mixing in this region caused by this vortex. However, the longer lived transport barriers in the middle of the complete flow show little cross flux and thus completely separate the phases. On the right, the transport barrier itself starts to swirl. Even though it still shows little cross flux, the phases start to mix on a greater scale.

6.4.2 *Air Bubble in Water*

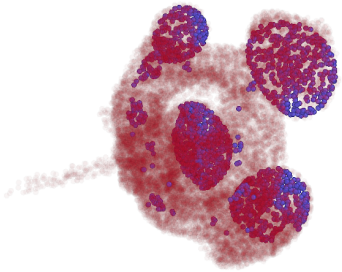
The Bubble dataset is a laminar two-phase flow of an air bubble moving through water. The dataset can be reproduced with the GPUSPH [85] simulation code. The domain is 6 times the size of the spherical bubble and is discretized using 4.3 million particles in each of the 50 discrete time steps in the interval $[0, 0.5]$.

Figure 6.6 shows the backward FTLE (a) and particles close to the corresponding height ridge LCS (b), which we color according to the fluid type. The LCS correspond to the ridges of the FTLE, but without the small-scale disturbances visible in the FTLE. The attracting coherent structures convey the transport of the air bubble that moves from left to right and starts to split. By visualizing both LCS and the fluid phases, the splitting of the air bubble in smaller bubbles becomes clearly visible.

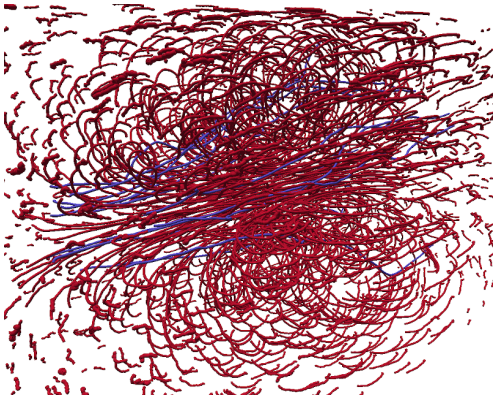
In (c), we visualize particles trajectories, chosen by random from the whole dataset and colored according to their phase. Although the different fluid phases become visible, the visualization suffers from significant amount of clutter due to a large number of short and relatively uninteresting trajectories. By creating trajectories from particles near LCS, these trajectories are effectively filtered out in (d). The movement and splitting of the air bubble is better visible. In total, this reveals the major flow features with respect to the air particles and how they mix with the surrounding fluid.



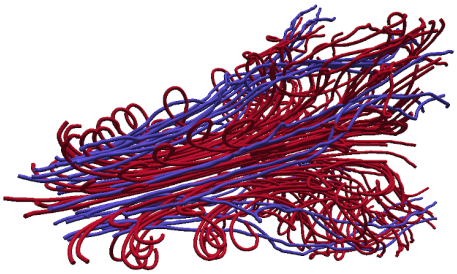
(a) Backward FTLE



(b) Particles near LCS



(c) Trajectories



(d) Trajectories near LCS

Figure 6.6: Visualization of the Bubble dataset using the backward FTLE (a), particles near LCS (b), particle trajectories (c), and trajectories created from particles near LCS (d).

6.5 DISCUSSION AND DOMAIN EXPERT FEEDBACK

In this chapter, we present a visual analysis framework for the visualization of transport and mixing behavior in particle-based flows. Domain experts have evaluated the approach and were especially impressed by the concept of brushing in the coordinated views. Interactive exploration was not possible for their data beforehand and was thus quickly integrated in their workflow. Whilst our analysis framework was mostly oriented towards qualitative analysis and exploration, the need for more detailed quantitative analysis tasks was identified. For example, support for more sophisticated queries, plots, and custom derived variables was requested.

Lastly, the big particle data sizes proved challenging for the interactive visualization and analysis. This is in part due to the required visualization and data management techniques, but also due to more practical concerns. For example, copying the data from a compute cluster to a workstation is time and storage intensive, if possible at all. Moreover, the largest data sets the domain experts had available were simply too big for interactive exploration on current workstations. This again emphasizes the need for data reduction closely integrated with the visual analysis.

7

UNCERTAIN TRANSPORT

Although most experiments and simulations produce deterministic data, uncertainty exists in all measured or simulated flows. This uncertainty might be estimated from repeated simulation runs or measurements, it might be introduced by data processing and reduction, or it can be explicitly modeled. Studying uncertainty is especially relevant in unsteady flows, where small variations in the initial conditions can cause dramatic changes to the flow [198]. In this chapter, we investigate uncertainties in the Lagrangian transport, i.e. the advection of a material by the flow. Since our approach is applicable to all flow data, we do not limit it specifically to particle data.

As discussed previously, the Lagrangian coherent structures identify a topological skeleton of the flow dynamics in a finite-time interval. LCS are material surfaces that remain coherent over time, although different definitions of coherency exist. Recent work has extended the definitions of coherent structures to uncertain flows. The probabilistic [101] or averaged [279] transport is estimated using a Monte Carlo approach, i.e. by stochastically advecting a large amount of particles. While the LCS are theoretically well established, this is, to our knowledge, not the case for probabilistic extensions.

Based on recent work from Haller, Karrasch, and Kogelbauer [111, 112], we employ the diffusion barrier strength (DBS) to identify transport barriers and enhancers to stochastic flows. These are material surfaces that show minimal or maximal stochastic cross flux. By assuming only small stochastic deviations, Monte Carlo integration is avoided and only the deterministic part of the flow has to be advected. To this end, we first define uncertain unsteady flows as stochastic differential equations that consist of an advective component and an added stochastic component modeled as a Gaussian. The central limit theorem makes this assumption reasonable in practice. Moreover, we discuss how to model uncertainty information in this stochastic differential equation, e.g. due to data reduction, to model small-scale deviations, or to model aggregated ensemble members. To complement the visualization of stochastic transport barriers and enhancers, which is based on the assumption of small-scale deviations, we propose a novel visualization of the scale of uncertainties encountered during advection.

In this chapter, we first define uncertain flows as stochastic differential equations in Section 7.1. In Section 7.2, we introduce the theory of stochastic transport barriers and enhancers, which leads to the definition of the diffusion barrier strength. We describe how to model stochastic flows from Gaussian flow fields in Section 7.3. Lastly, we visualize uncertain transport in real-world datasets. In several experiments, we investigate the relationship between the stochastic transport barriers and enhancers, Lagrangian coherent structures, and its probabilistic extensions in Section 7.5.

7.1 STOCHASTIC FLOWS

To visualize uncertainty in the transport in unsteady flows, we first define a stochastic flow as a deterministic flow with small stochastic deviations. More formally, we model an uncertain flow by a stochastic differential equation (SDE), i.e. we extend the ordinary differential equation from Equation 4.1 with a stochastic component

$$\mathrm{d}x(t) = \underbrace{v(x(t),t)\,\mathrm{d}t}_{\text{deterministic}} + \underbrace{\sqrt{s}B(x(t),t)\,\mathrm{d}W(t)}_{\text{stochastic}}. \tag{7.1}$$

Here, $W(t)$ is an d-dimensional Wiener process with disturbance $\sqrt{s}B(x(t),t)$. The Wiener process W consists of independent standard Gaussian distributions at every time t. The notation $\mathrm{d}W(t)$ represents a random variable that is distributed with respect to a standard, multivariate Gaussian. The disturbance, which controls the scaling and anisotropy, is separated into a scaling parameter $s > 0$ and a scale-independent matrix $B \in \mathbb{R}^{d\times d}$. In the following, we will assume only small deviations, i.e. s is small.

NUMERICAL INTEGRATION In general, SDEs can be solved by numerical integration, for example using the Euler-Marayuma or the Runge-Kutta methods for SDEs [170]. These Markov chain Monte Carlo strategies involve sampling of the stochastic component. The numerical integration is thus significantly more involved compared to deterministic flows since it requires a large amount of stochastically integrated particles. At the same time, it is non-trivial to decide how many particles should be integrated. For these reasons, we want to avoid the numerical integration of stochastic flows.

7.2 STOCHASTIC TRANSPORT BARRIERS AND ENHANCERS

In this section, we introduce stochastic transport barriers and enhancers. We give an intuitive introduction and refer to the work of Haller, Karrasch, and Kogelbauer for the formal derivation [111, 112]. Transport barriers are inhibitors of the spread of substances in a flow, whilst transport enhancers maximize such diffusion or mixing processes. Remarkably, these barriers and enhancers do not depend on the actual value of the diffusivities, i.e. the scaling parameter s. They are also well-defined for deterministic flows when we consider the case of $s \to 0$. In this case, they present an alternative to the Lagrangian coherent structures, but do not depend on any specific definition of coherency. The diffusion barrier strength (DBS) visualizes the barriers and enhancers, which can be defined as ridges of the DBS, similar to the LCS that can be defined as ridges of the FTLE.

The DBS is computed from a deterministic flow v and a diffusion component that describes the amount and anisotropy of diffusion at each point in space and time. First, we introduce the tensor T from the gradient of the flow map $\nabla\phi$ and the diffusion $D \in \mathbb{R}^{d\times d}$ as

$$T(x,t_0,t) := \left[\nabla\phi(x,t_0,t)\right]^{-1} D(x,t) \left[\nabla\phi(x,t_0,t)\right]^{-T}. \tag{7.2}$$

If the diffusion is isotropic, i.e. $D \equiv I$, then

$$T(x, t_0, t) = \mathbb{C}(x, t_0, t)^{-1}.$$

However, we have to incorporate the diffusion $D(x, t)$ at every time in the interval $[t_0, t_1]$, in contrast to the FTLE that only considers the deformation at the end of the time interval. Therefore, the time-averaged, diffusivity weighted right Cauchy-Green strain tensor $\bar{\mathbb{C}}$ is computed as

$$\bar{\mathbb{C}}(x_0, t_0, t_1) := \frac{1}{|t_1 - t_0|} \int_{t_0}^{t_1} \det(D(x, t)) T(x, t_0, t)^{-1} \, \mathrm{d}t, \tag{7.3}$$

where x is the position during integration at time t, i.e. $x = \phi(x_0, t_0, t)$. Since we require only the inverse of T, we compute

$$T(x, t_0, t)^{-1} = [\phi(x, t_0, t)]^\intercal \, D(x, t)^{-1} \, [\nabla \phi(x, t_0, t)] \tag{7.4}$$

instead of Equation 7.2. Lastly, Haller et al. [111] define the DBS as the trace of $\bar{\mathbb{C}}$. Since this quantity is exponential, we take the logarithm for visualization:

$$\mathrm{DBS}(x_0, t_0, t_1) := \log\left(\mathrm{tr}(\bar{\mathbb{C}}(x_0, t_0, t_1))\right). \tag{7.5}$$

Although the integral in Equation 7.3 might seem daunting at first, we are already performing this integration when computing the flow map ϕ. Thus, to compute the DBS, we integrate the deterministic flow v and at each step evaluate T^{-1} to accumulate the diffusivity weighted and time-averaged strain tensor $\bar{\mathbb{C}}$.

7.3 MODELING DIFFUSION

To compute the DBS, we require a scale-independent diffusion component D. For completely deterministic flows, we set the diffusion to the identity matrix, i.e. $D = I$. For stochastic flows, with a scale-independent disturbance B (cf. Equation 7.1), the diffusion is defined as

$$D := \frac{1}{2} B B^\intercal. \tag{7.6}$$

For uncertain and unsteady flows modeled by Gaussians, we now discuss how to obtain B. The scale-independent disturbance represents the anisotropy and the scaling relative to other regions of the flow. Given a Gaussian with covariance $C(x, t)$, we want to separate it into a global scaling parameter s and a disturbance $B(x, t)$.

Since the disturbance should be, on average, centered around the identity matrix I, we standardize all covariance matrices. That is, given the set of all covariance matrices $\mathcal{C}$, we subtract the mean of all variances, i.e. the diagonal elements of each covariance matrix $C \in \mathcal{C}$. Then, we divide out the maximal standard deviation over all dimensions:

$$B(x, t) = \frac{C(x, t) - I\mu_C}{\sigma_C^{\max}}, \tag{7.7}$$

where μ_C is the mean of all variances:

$$\mu_C := \begin{bmatrix} \mathbb{E}[\mathcal{C}_{0,0}] \\ \vdots \\ \mathbb{E}[\mathcal{C}_{n-1,n-1}] \end{bmatrix} \tag{7.8}$$

and $\sigma_C^{\max}$ is the maximum of the standard deviation of all variances in $\mathcal{C}$:

$$\sigma_C^{\max} := \max\left(\sqrt{\mathbb{E}[\mathcal{C}_{i,i} - \mu_{C_{i,i}}]}\right), \quad \text{where } i = 0 \dots n-1. \tag{7.9}$$

Although a different scaling than $\sigma_C^{\max}$ could be used since it is canceled out in Equation 7.3, our definition increases the numerical stability.

7.4 VISUALIZING TRANSPORT UNCERTAINTY

By design, the diffusion barrier strength ignores the absolute scale of stochasticity, i.e. the amount of uncertainty of the transport. However, this quantity is still relevant, especially if the amount of stochastic deviations varies strongly in the flow. To this end, we propose a visualization that complements the DBS by directly conveying the scale of stochastic deviations.

Although it is possible to directly visualize the time-dependent variance of a Gaussian flow field, we are interested in the uncertainty of the transport, which is inherently defined in a Lagrangian frame. We propose to measure the uncertainty encountered during the integration of a tracer particle. In other words, this visualizes the transport of uncertainty in the flow. Moreover, this enables us to integrate only the deterministic part of the stochastic flow and avoid stochastic numerical integration.

First, we discuss how to measure the uncertainty of a Gaussian flow with covariance $C(x,t)$ at a single point in time and space. Since we are not interested in the variance along individual dimensions, we employ the generalized variance [330, 331] defined as $|\det(C(x,t))|$. Intuitively, this measures the multidimensional scatter of a Gaussian. To enable comparisons in different dimensions, we standardize this quantity by taking the d-th root in d-dimensional space. Lastly, we average this measure over time during the material transport:

$$\sigma_T(x_0, t_0, t_1) := \frac{1}{|t_1 - t_0|} \int_{t_0}^{t_1} |\det(C(x,t))|^{\frac{1}{d}} \, \mathrm{d}t, \tag{7.10}$$

where $x = \phi(x_0, t_0, t)$.

7.5 NUMERICAL EXPERIMENTS

In this section, we study the uncertain transport in a synthetic and three real-world datasets.

7.5.1 *Double Gyre*

This two-dimensional synthetic and time-dependent vector field describes two counter-rotating gyres and has been introduced in Section 5.3.1. In the following, we study the time interval $[0, 10]$ and integrate forward in time.

The forward FTLE of this flow is shown in Figure 7.1 (a) and the forward DBS is shown in (b). Now, we add different amounts of isotropic Gaussian noise. The transport uncertainty is thus constant everywhere and the DBS is not affected by changing the absolute scale of noise. However, for increasing amounts of noise the assumption of small deviations is no longer valid. The behavior of the uncertain flow is visualized by stochastic integration of a large amount of particles and mapping their density in (c), (g), and (k). By integrating backwards in time, this illustrates the separatrices in the flow. For small deviations, this visualization aligns with the DBS. With increasing amounts of noise, the visualization deviates more. The FTLE-D, obtained by averaging the right Cauchy-Green strain tensor, and the probabilistic D-FTLE from Guo et al. [101] similarly illustrate the influence of adding more noise.

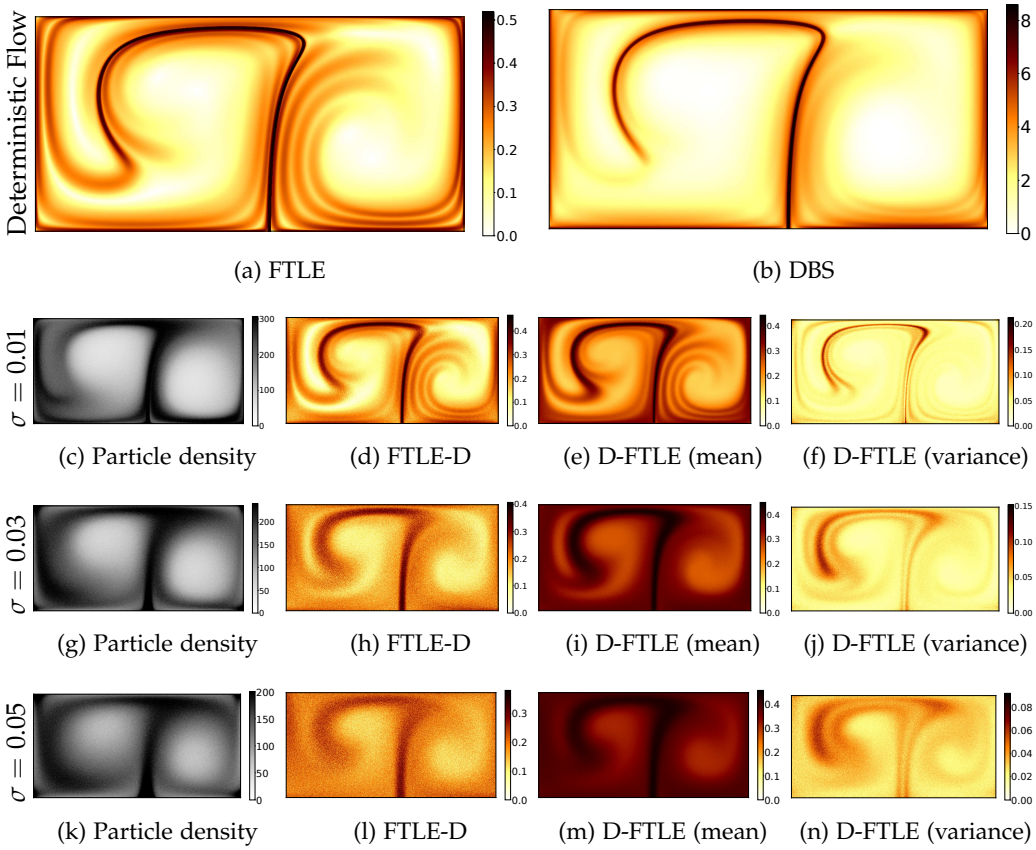
We now reduce the dataset to a discrete grid of size $[256 \times 128 \times 10]$ and estimate a Gaussian error model. The resulting variance and covariance averaged over time are visualized in Figure 7.2. The amount of variance (d, e) and covariance (f) varies periodically. In contrast, the transport uncertainty shown in the Figure 7.3 (c) directly visualizes the impact of the uncertainty on the advected tracer particles.

In Figure 7.3 (a), we have stochastically advected a large amount of randomly distributed particles to visualize separating manifolds in the flow. The DBS shown in (b) clearly corresponds to these structures. The uncertainty of the transport is visualized in (c) and indicates a high uncertainty in the midst of both gyres. The DBS is low in these regions. Note that the FTLE of the mean flow in Figure 7.1 (a) does not consider the stochastic component of the flow. The FTLE indicates the presence of several smaller features around the two gyres that are not depicted in the density visualization in (a) or the DBS (b) and are located in regions of high uncertainty (c).

The FTLE-D and the mean D-FTLE from Guo et al. [101] shown in (d) and (e) indicate a larger amount of structures. The center of the left gyre even shows additional structures that are not present in the FTLE or the advected particles (a). The transport uncertainty indicates a high uncertainty in this area (c). However, the variance of the D-FTLE (f) is only high near the central barrier of the flow. At the same time, the presence of this barrier is far from uncertain. A high variance in the D-FTLE thus does not necessarily imply uncertainty of the transport barriers.

7.5.2 *Red Sea*

This dataset from the SciVis contest 2020 is an ensemble simulation of the circulation dynamics in the Red Sea [274]. Eddies in the ocean play a major role in the transport of energy and particles. Uncertainty is estimated from 50 ensemble members created from perturbed initial conditions. Here, we



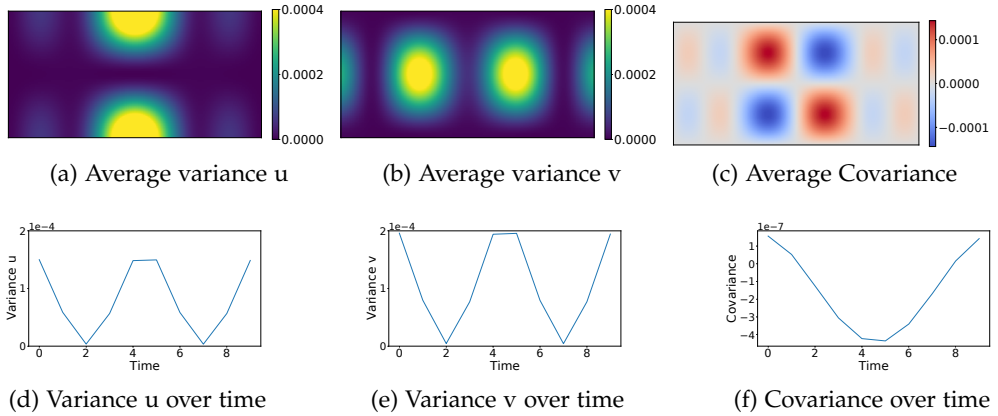
(a) FTLE (b) DBS

(c) Particle density (d) FTLE-D (e) D-FTLE (mean) (f) D-FTLE (variance)

(g) Particle density (h) FTLE-D (i) D-FTLE (mean) (j) D-FTLE (variance)

(k) Particle density (l) FTLE-D (m) D-FTLE (mean) (n) D-FTLE (variance)

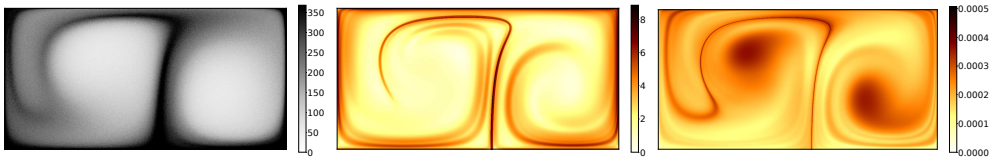
Figure 7.1: The double gyre dataset with different amounts of isotropic Gaussian noise. In (a) and (b) the FTLE and DBS of the deterministic flow are shown. Note that the DBS is not affected by changing the amount of isotropic noise since it assumes only small deviations.



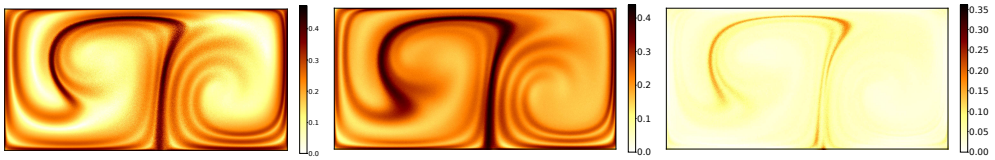
(a) Average variance u (b) Average variance v (c) Average Covariance

(d) Variance u over time (e) Variance v over time (f) Covariance over time

Figure 7.2: Gaussian error model of the double gyre reduced to a resolution of $[256 \times 128 \times 10]$.



(a) Density of advected particles (b) DBS (c) Transport uncertainty



(d) FTLE-D (e) D-FTLE (mean) (f) D-FTLE (variance)

Figure 7.3: The Double Gyre dataset with uncertainty estimated during data reduction to a grid of size $[256 \times 128 \times 10]$.

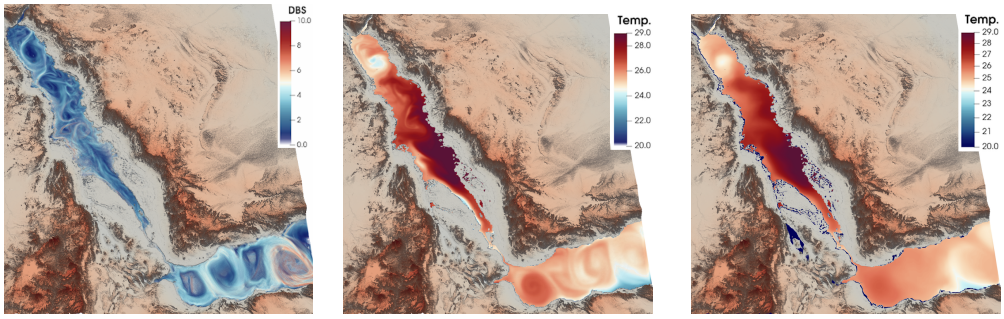
estimate the mean and covariance from the individual members and analyze the resulting uncertain flow. Since the depth of the dataset is irregularly spaced, we have resampled it to a grid of size $500 \times 500 \times 150$ with 60 time steps. To aid the understanding of the dataset, we have added topography and bathymetry [84] to the visualizations in Figure 7.4.

Figure 7.4 (a) shows the backward DBS over a time interval of 182 hours that indicates enhancers to stochastic transport. In (b) and (c) we visualize the diffusion of the temperature over time near the surface. Note that this diffusion corresponds to the enhancers indicated by the DBS. The forward DBS of the Red Sea dataset is shown in Figure 7.4 (d), integrated over the same 182 hours. Additionally, we visualize the mean concentration of salinity in the upper layers of the ocean at the beginning (d) and the end of the time interval (e). The diffusion of salinity visibly aligns with the most significant transport barriers identified by the forward DBS.

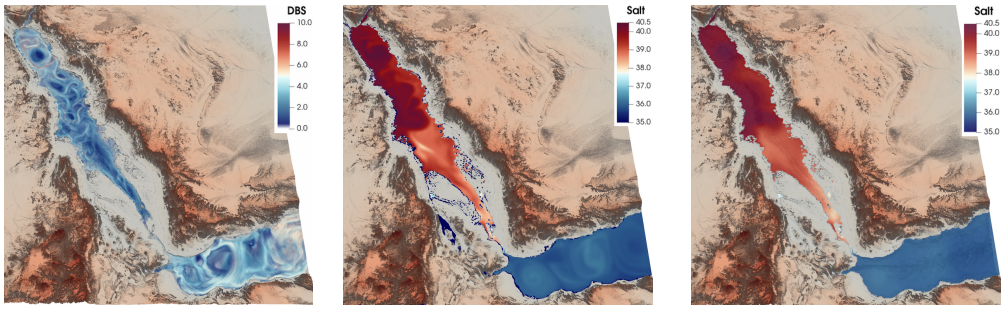
The DBS visualizes the transport of the aggregated stochastic flow, but does not consider individual ensemble members and assumes only small-scale deviations. Although this is a limitation, visualizing unsteady flows with large uncertainties is generally difficult if not infeasible due to non-linear flow dynamics. Lastly, a prior clustering of the ensemble members could be performed if the Gaussian assumption of the ensemble members is considered problematic, which was not the case for the Red Sea dataset.

Figure 7.5 (b), (c), (e), and (f) illustrate the time averaged variance of velocity, salinity, and temperature. Variance in w-direction, i.e. along the depth axis, is not shown here since it is close to zero. For most quantities, the variance is highest in the gulf of Aden, the lower right part of the dataset. Nonetheless, the DBS depicts strong transport barriers and enhancers in this region that correspond to the diffusion of salinity and temperature.

The forward FTLE of the mean flow is shown in Figure 7.5 (a). Although the FTLE contains more noise than the DBS in Figure 7.4 (d), it indicates similar structures in the Red Sea. In the higher variance region in the Gulf of Aden, the differences are more pronounced. Since the FTLE only uses the mean

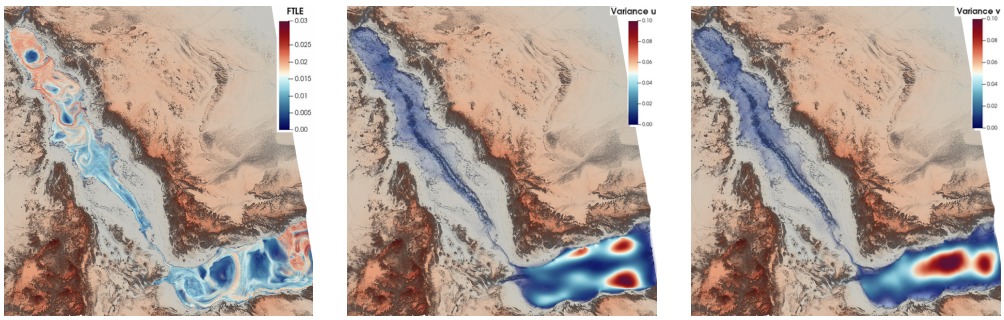


(a) Backward DBS (b) Temperature t_0 (c) Temperature t_1

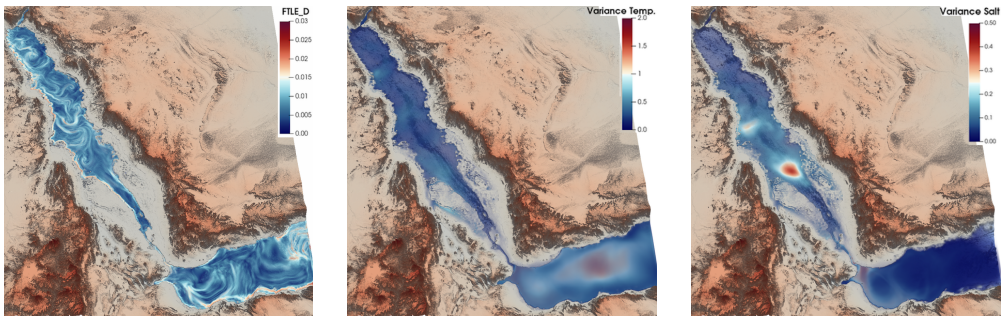


(d) Forward DBS (e) Salinity t_0 (f) Salinity t_1

Figure 7.4: The DBS in (a) and (d) visualize enhancers and barriers to stochastic transport. The mean temperature distribution in t_0 (b) and t_1 (c) visualizes the diffusion of temperature. The salinity concentration at time t_0 and t_1 is shown in (e) and (f).



(a) Forward FTLE (b) Variance u (c) Variance v



(d) Forward FTLE-D (e) Variance of temperature (f) Variance of salinity

Figure 7.5: The forward FTLE of the mean flow is shown in (a) and the FTLE-D of the stochastic flow is visualized in (d). The time-averaged variance in the Red Sea dataset estimated from the ensemble members is shown in (b), (c), (e), and (f).

flow, the results do not take the uncertainty into account. In comparison, the FTLE-D in Figure 7.5 (d) is computed from the stochastic flow. The FTLE-D is not as noisy as the FTLE, but requires a significant amount of computational effort to achieve this. In the gulf of Adan, the FTLE-D shows a large amount of finer features. The visible structures in the temperature and salinity are barely, if at all, present.

7.5.3 *Heated Cylinder*

This dataset consists of a 2D time-dependent flow generated by a heated cylinder using the Boussinesq approximation. The dataset is due to Günther et al. [98] and was simulated using the Gerris flow solver [248]. It shows a turbulent plume that contains several small vortices rotating around each other. The dataset is stored in a regular grid of resolution $[150 \times 450 \times 2001]$.

The backward FTLE of the time interval $[10, 0]$ is shown in Figure 7.6 (d). We have reduced the temporal resolution of the data set to 100 time steps and estimated the mean and covariance matrix in each grid cell. We visualize attractors in the uncertain flow in (a) by integrating particles forward in time and visualizing their density. The DBS of the stochastic flow is shown in (b) and the transport uncertainty in (c). The mean and variance of the D-FTLE is shown in (e) and (f). The FTLE-D is not shown here, but corresponds closely to the D-FTLE.

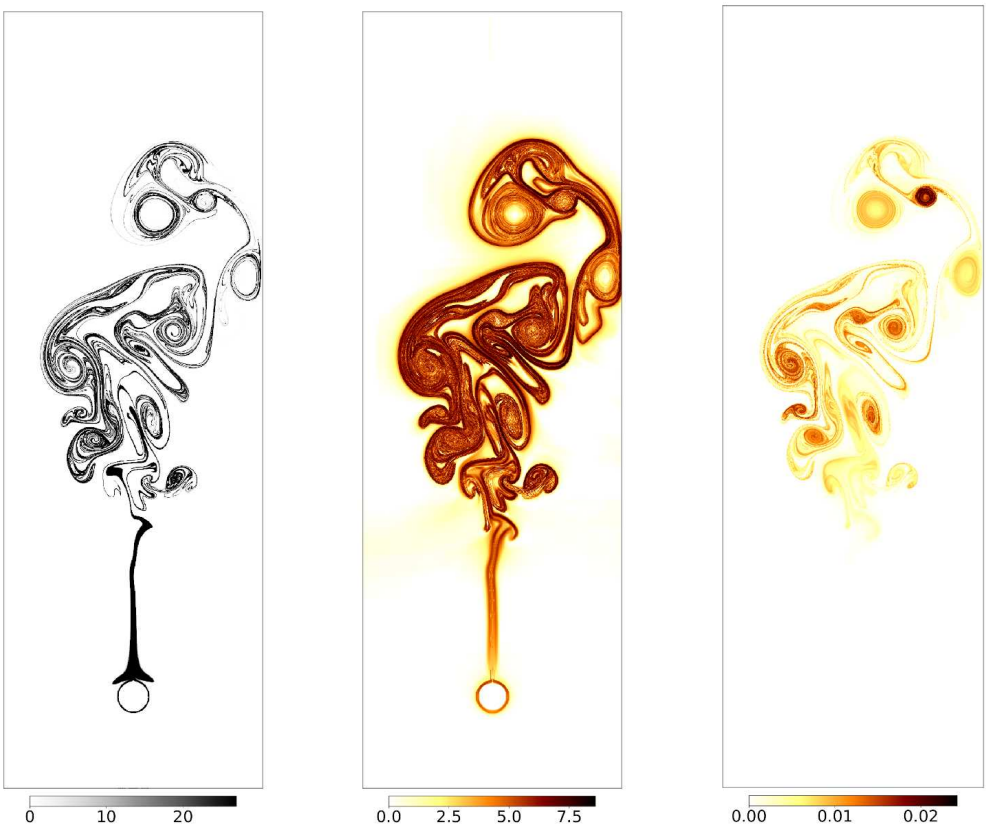
The uncertain flows indicates a changed flow behavior compared to the FTLE in (d) since some vortices have been stretched or rotated. Since the DBS incorporates the relative scaling and anisotropy, these changes are taken into account without requiring stochastic numerical integration. Moreover, the DBS tends to smooth out the center of vortical regions, in correspondence to the density visualization shown in (a) and the transport uncertainty in (c), which is not the case for the FTLE-D.

7.5.4 *Flow Around Corners*

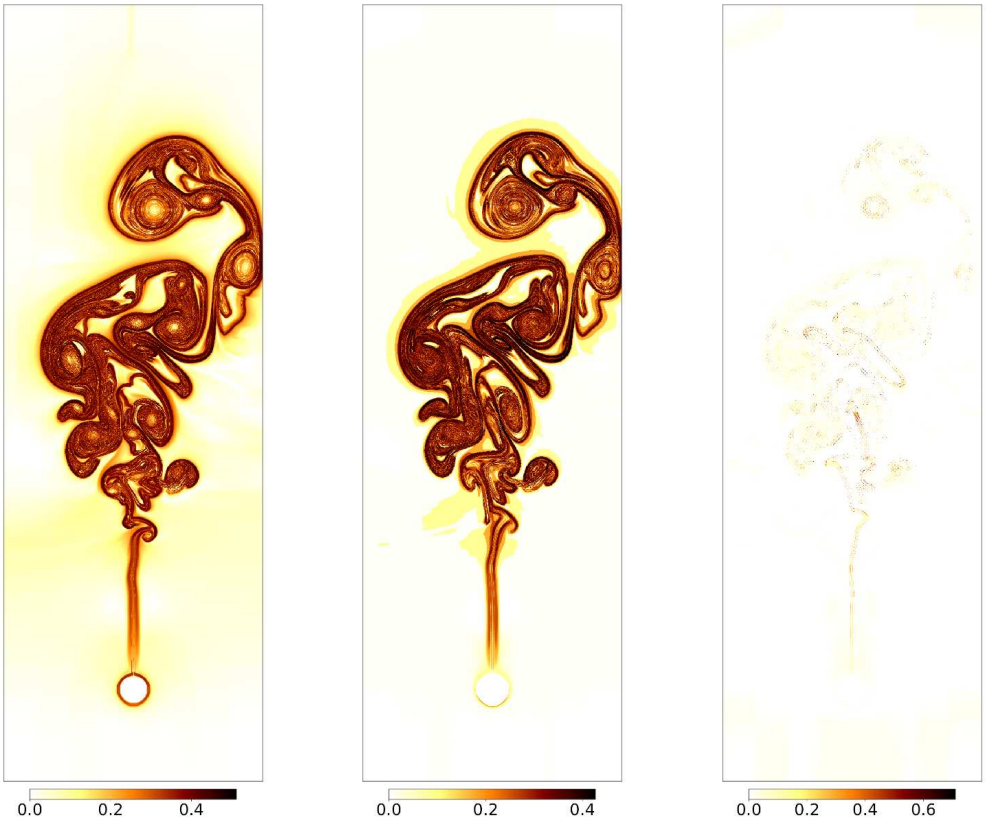
This flow around two cylinders and corners has been simulated using the Gerris flow solver [248] and is due to Baeza Rojo and Günter [265]. Here, we visualize the deterministic flow without explicitly constructing an error model, i.e. we set $D = I$ and assume $s \to 0$. Figure 7.7 shows the backward FTLE (a) and the DBS (b) in the time interval $[10, 5]$. The quantities are similar, but not identical. For example, the DBS smooths out fine-scale features inside vortical regions.

7.5.5 *Performance*

All of our evaluations were performed using GPU acceleration on an NVIDIA Quadro RTX 8000 with CUDA. To integrate deterministic flows, a fourth-order Runge-Kutta scheme is used. Stochastic flows are integrated using the Euler-Maruyama method with a constant number of 100 Monte Carlo runs.

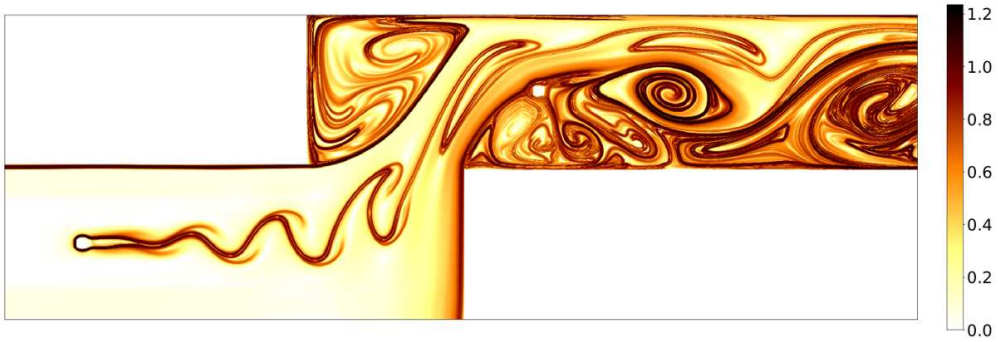


(a) Particle density (b) DBS (c) Transport uncertainty

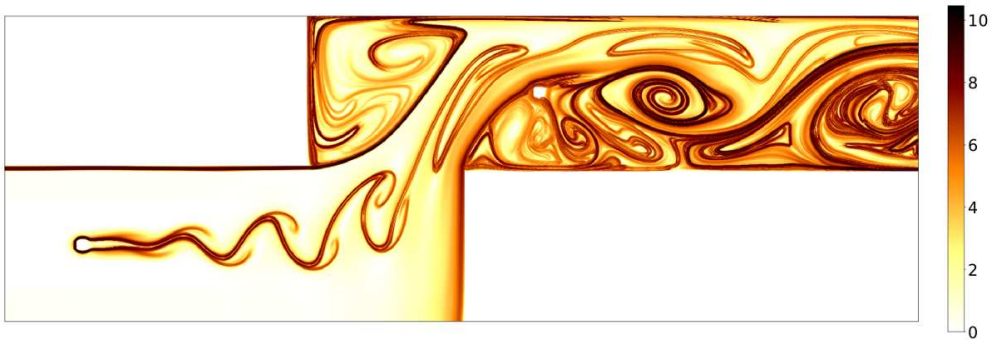


(d) Deterministic FTLE (e) D-FTLE (mean) (f) D-FTLE (variance)

Figure 7.6: The heated cylinder dataset with uncertainty estimated during data reduction to a grid of size $[150 \times 450 \times 100]$.



(a) FTLE



(b) DBS

Figure 7.7: The backward FTLE (a) and DBS (b) of the deterministic flow around corners.

Table 7.1: Performance measurements of our datasets.

Dataset	Resolution	FTLE	DBS	FTLE-D
Double Gyre	1024×512	4.1ms	7.4ms	2173ms
Red Sea	$1500^2 \times 150$	2739ms	5193ms	2,129,030ms
Heated Cylinder	600×1800	258.7ms	680.0ms	143,243ms
Flow Around Corners	2250×750	110.8ms	456.2ms	220,309ms

Performance measurements for different datasets are shown in Table 7.1. Computing the DBS requires evaluating Equation 7.4 during each integration step. Computing the DBS thus takes two to four times longer than the FTLE. In comparison, methods that depend on stochastic integration increase the runtimes by several orders of magnitude.

7.6 DISCUSSION

Our results show that the DBS is significantly faster to compute than probabilistic extensions of the FTLE since no stochastic integration is performed. In our experiments, the DBS closely aligns with the density of stochastically advected particles, whilst many features from the FTLE and its probabilistic extensions are not visible. In fact, probabilistic extensions of the FTLE show

features of possible realizations of an uncertain flow. In contrast, the DBS indicates features that exist in an inherently stochastic flow. Although both approaches have merit, it makes the probabilistic D-FTLE hard to interpret. At first glance, the variance of the D-FTLE might suggest uncertainties of the transport barriers and enhancers, however, this is not the case. Indeed, none of the approaches convey the actual amount of uncertainty encountered during integration. Our visualization of the transport uncertainties efficiently illustrates this uncertainty in the Lagrangian frame, thus providing additional insights.

Lastly, note that all of the results in this chapter are applicable to particle data. In fact, we could extend our method from Chapter 5 to compute the DBS, but this would require loading all time steps in the interval $[t_0, t_1]$, leading to a more costly computation.

Part II

DATA REDUCTION FOR VISUAL ANALYSIS

8

VOID-AND-CLUSTER SAMPLING

In this chapter, we investigate the use of statistical sampling to reduce large data sets to a representative subset. As discussed in Section 3.3, sampling scales to higher dimensional data and is well-suited for scattered data. Although simple random sampling gives decent results, recent work improves upon this using stratified [299, 335] and information-guided sampling [24, 325]. These results emphasize the significance of stratification in the spatial domain and adaptive sampling guided by the value domain.

We propose a sampling strategy for scattered data (Section 8.2) generalizing the void-and-cluster technique from Ulichney [314] (Section 8.1) that stratifies optimally in the spatial domain. Specifically, we find samples that are well distributed with respect to the blue noise property, which implies large mutual distances between samples without causing regularity artifacts. Additionally, we discuss how to adapt the sampling strategy to the value dimensions by better sampling regions of value distributions with high entropy. Moreover, the sampling technique implicitly defines an ordering on the samples that enables progressive data loading and continuous level-of-detail during visualization and analysis. Our proposed algorithm is fast, scalable, and well-suited for GPU acceleration (Section 8.3). Therefore, it is applicable in situ, i.e. while a simulation is running, but also as a traditional post-processing step.

Furthermore, we extend our sampling technique to time-dependent particle data. Instead of considering each time step independently, we sample particle trajectories. We find representative trajectories that evenly cover the spatiotemporal domain based on an efficient iterative extension of the void-and-cluster technique.

Lastly, we introduce an error measure to quantify how well a set of samples represents the data with respect to both the spatial and the value domain (Section 8.4). In particular, we derive a continuous error measure that quantifies the difference in the value distributions for every point in the dataset. This error measure integrates well into our sampling technique, where we use it to determine when a sufficient number of samples has been taken. We evaluate the quality of our proposed sampling technique on different synthetic and real-word datasets using this error measure and other derived quantities, such as the quality of scattered data interpolation (Section 8.5). Finally, we investigate the performance and scalability of our proposed sampling technique and compare it to related approaches.

8.1 THE VOID-AND-CLUSTER TECHNIQUE

Ulichney [314] introduces the void-and-cluster sampling technique in the context of halftoning and dithering. The technique ranks all pixels in a rastered image, thus producing a dithering mask. If we think of all pixels that are

already ranked as white and mark the others black, applying a Gaussian filter yields an image that indicates the local density of ranked pixels. The tightest cluster is brightest, the largest void is darkest. The void-and-cluster technique goes through three phases to fill large voids and reduce tight clusters greedily. The order of greedy additions implies an order on the sample set such that each prefix has good blue noise characteristics. In the end, all pixels are ranked and the resulting dithering patterns have blue noise characteristics. In the next sections, we discuss the extension of this technique to scattered data, augment it with non-uniform sampling densities, and propose a parallel implementation.

8.2 VOID-AND-CLUSTER SAMPLING FOR PARTICLE DATA

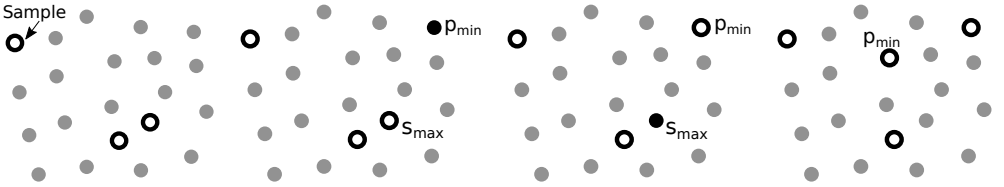
Ulichney's algorithm is restricted to uniform grids and produces samples with a uniform density. In this section, we generalize the approach to scattered data with a non-uniform spatial distribution. To preserve the spatial density, we compute a density estimate on the whole dataset. Our generalized void-and-cluster sampling works on scattered data and enforces the given density (Section 8.2.1). Like the original algorithm, it orders all sample points to enable level of detail and progressive data loading (Section 8.2.2). The algorithm is efficient because each iteration only requires local updates with compact kernels (Section 8.2.3). Our parallel implementation in Section 8.3 further exploits this locality. Supporting arbitrary sampling densities lets us emphasize regions of high entropy in the value domain (Section 8.2.4). Finally, we extend our technique to the sampling of time-dependent trajectories (Section 8.2.5).

8.2.1 *Void-and-Cluster Sampling*

Assume we have a dataset with points $P \subset \mathbb{R}^d$ in a d-dimensional spatial domain. Each sample is mapped to a value in the possibly multivariate value range V through $v : P \to V$. Among these points, we want to pick a representative subset $S \subset P$. Therefore, we optimize the placement of samples in the spatial domain by estimating the density of selected samples $\lambda_S : P \to \mathbb{R}^+$ for each point $p \in P$. A high sample density indicates a large number of nearby samples, whilst a low density indicates few. We want to place samples such that dense regions (clusters) and empty regions (voids) are avoided. Or, in other words, reduce the maximum of the sample density λ_S and increase its minimum.

This does not work for spatially non-uniformly distributed data points since we have to account for the original distribution in the spatial domain. Even for uniformly distributed points the border region of the spatial domain is less densely populated. We account for the spatial distribution of the points by first computing a point density ρ_P for each $p \in P$:

$$\rho_P(p) := \sum_{p_i \in P} k(\|p - p_i\|), \tag{8.1}$$



(a) Initial random sampling (b) Find $p_{\min}$ and $s_{\max}$ (c) Exchange $p_{\min}$ and $s_{\max}$ (d) Find and add the next $p_{\min}$

Figure 8.1: Overview of the void-and-cluster sampling technique for scattered data. After initial random sampling (a), the samples are optimized by finding (b) and exchanging (c) the largest void $p_{\min}$ with the tightest cluster $s_{\max}$ until $p_{\min} = s_{\max}$. We then iteratively find and add (d) the largest void $p_{\min}$ until we have enough samples.

using a kernel function k. Given a subset of samples $S \subset P$, the sample density at $p \in P$ is then defined as:

$$\lambda_S(p) := \frac{\sum_{s \in S} k(\|p - s\|)}{\rho_P(p)}. \tag{8.2}$$

We will now describe a strategy to find the optimal set of samples in the spatial domain with respect to the sample density λ_S, by extending the void-and-cluster algorithm. This is an iterative and greedy algorithm that at each step finds a locally optimal distribution of samples. An overview of our modified void-and-cluster sampling technique is depicted in Figure 8.1.

Initially, we take a fixed number of random samples. Although the point density ρ_P stays constant, we have to update λ_S when we change the set of samples. The sample density is computed incrementally when a sample s is added (or removed), by adding to (or subtracting from) the density $\lambda_S(p)$ for all points.

We then optimize these initial samples by removing the tightest cluster

$$s_{\max} = \arg\max_{s \in S} \lambda_S(s) \in S, \tag{8.3}$$

i.e. the sample with largest λ_S. Then, we add the largest void

$$p_{\min} = \arg\min_{p \in P \setminus S} \lambda_S(p) \in P \setminus S, \tag{8.4}$$

i.e. the point with the lowest λ_S that is not a sample yet. Since we add and remove a sample, we have to update the sample densities accordingly. The optimization stops once the tightest cluster that we remove then becomes the largest void, that is $s_{\max} = p_{\min}$.

After construction of the optimal initial sampling, we iteratively find and add the largest void to the set of samples until we have reached the desired amount of samples. We provide detailed pseudocode of the entire algorithm in the appendix, see Section A.1.

8.2.2 *Ordering of Samples*

A positive side-effect of the greedy approach is that the void-and-cluster strategy implicitly defines an ordering of the samples. With respect to this ordering, any prefix of the sample set S still has good blue noise characteristics. Ulichney [314] denotes it as the rank $r : P \to \mathbb{N}$, where $r(p) = \infty$ for all $p \in P \setminus S$. To compute this ordering, we assign and increment the rank when adding a sample during the initial random sampling or the void filling steps. For a sample s_i that is added as the i-th sample, we set $r(s_i) = i$. During the void-and-cluster optimization, when we exchange the tightest cluster $s_{\max}$ with the largest void $p_{\min}$, we have to swap the rank accordingly, i.e. we set $r(p_{\min}) = r(s_{\max})$ and $r(s_{\max}) = \infty$.

We re-order (or index) the samples according to this mapping. We can use this ordering for continuous level-of-detail and for progressive data loading during the subsequent visualization and analysis.

8.2.3 *Compact Kernels*

If the kernel k is compact, i.e. has a finite extent, only a local neighborhood has to be considered when updating the densities of samples and points. For compact kernels, the optimization is thus defined locally. In our experiments, we found that the choice of kernel does influence the distribution of samples and the quality of the blue noise. Nonetheless, we always achieved good results as long as the kernel size h_P was in a reasonable order of magnitude with respect to the spatial domain. If we take a fraction of all samples $|S| < |P|$, we have to increase the kernel size h used for sampling accordingly:

$$h := h_P \sqrt[d]{\frac{|P|}{|S|}}, \tag{8.5}$$

using the spatial dimension d. Although we did experiment with a Gaussian kernel, we use a cubic spline [217] in the remainder of this work since it yields similar results, but is compact. Lastly, we denote points in the support of kernel k at point $p \in P$ as its neighborhood $\mathcal{N}(p) \subset P$.

8.2.4 *Adaptive Sampling*

So far we have taken all samples with equal probability and proportional to the spatial density. Now, we discuss the use of non-uniform probabilities to better capture complicated behavior in the value dimensions.

In general, we would like to take samples $S \subset P$ according to the probability mass function $\phi : P \to [0, 1]$. To sample a representative subset, we must re-weight all samples $s \in S$ proportionally to the reciprocal $\phi^{-1}(s)$. With our void-and-cluster approach, we implement this adaptation by using a modified density:

$$\tilde{\rho}_P(p) := \rho_P(p)\phi(p). \tag{8.6}$$

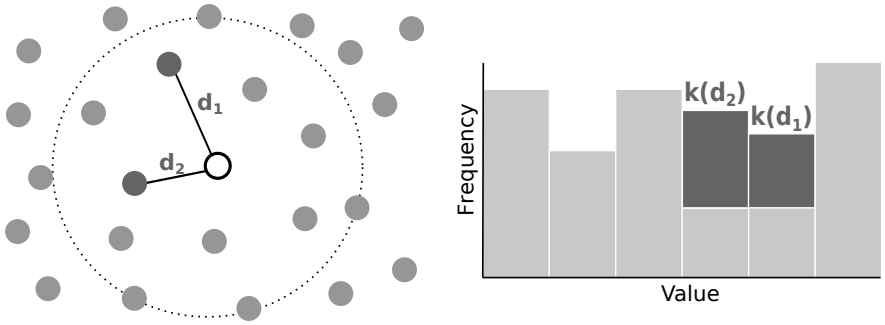


Figure 8.2: We compute the entropy of a point in its local neighborhood from a histogram of the value distribution, weighted by the radially symmetric kernel k.

Thus, any normalized importance measure, for example a computed feature or derived variable, can be used to guide the placement of samples.

ENTROPY SAMPLING Similar to recently proposed sampling techniques [24, 325], we place more samples in regions with a high entropy, i.e. value distributions of high complexity.

For each point $p \in P$ we compute the entropy using its local neighborhood $\mathcal{N}(p)$, see Figure 8.2. Specifically, we create a histogram of the value distribution of all points in the neighborhood. We use the global value range for the computation of the histogram to ensure that the entropy is consistent everywhere. To obtain a continuous entropy in the spatial domain, we weight the contribution of each neighbor with respect to its distance to p using the kernel k. From the weighted and normalized histogram h_p of size N_{bins}, we compute the entropy:

$$\mathbb{H}(p) := -\sum_{i=0}^{N_{\text{bins}}-1} h_p(i) \log_2 h_p(i). \tag{8.7}$$

Similar to Wei et al. [325], we derive a sampling probability that is independent of the size of the histogram as

$$\phi_H(p) := \frac{2^{\mathbb{H}(p)}}{N_{\text{bins}}} \tag{8.8}$$

and then derive a correctly normalized probability mass function as

$$\phi(p) = \frac{\phi_H(p)}{\sum_{p_i \in P} \phi_H(p_i)}. \tag{8.9}$$

For multivariate data, we have to construct a single probability from multiple value dimensions. Hence, we compute the entropy individually in each dimension and use the maximal entropy at each point. Intuitively, we consider a data point relevant if at least one dimension shows high entropy. Dependent on the application, we could also select a subset of the value dimensions to guide the entropy sampling.

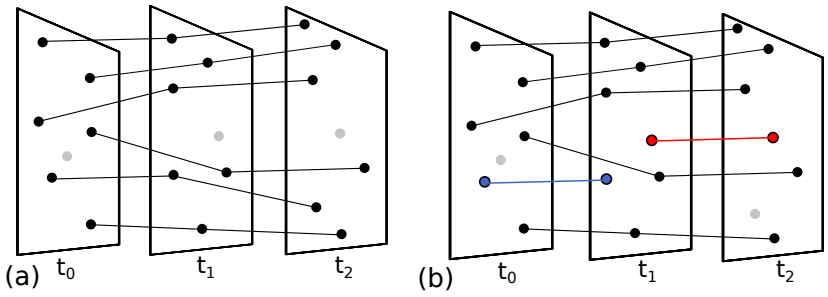


Figure 8.3: In (a), we sample trajectories that bundle and separate over time. We optimize the distribution of sampled trajectories in (b), by stopping (blue) and starting (red) trajectories in t_1.

8.2.5 *Trajectory Sampling*

In addition to sampling a single time step, we extend our sampling technique to time-dependent data. Specifically, we consider trajectories of scattered data points in discrete time steps $t_0, ..., t_{N-1} \in \mathbb{R}$. A trajectory is then defined as a sequence of points over time $\tau := (p_{t_j}, \ldots, p_{t_k})$ with $0 \leq j \leq k \leq N-1$ and points $p_{t_i} \in P_{t_i}$ at time t_i, cf. Section 2.1.2. We now discuss how to sample a subset T from the set of all trajectories.

To avoid an optimization of trajectories over all time steps, we sample iteratively. In the first time step, we employ our void-and-cluster sampling strategy to sample a subset $S_{t_0} \subset P_{t_0}$ that defines an initial set of trajectories T. In the next time step, a number of trajectories could end, i.e. no longer exist in the following steps. We first compute the point density ρ_P and the sample density λ_S from the trajectories T that still exist in the current time step. For n ending trajectories, we then add the trajectories from the n largest voids to T and thus start new trajectories from this time step. To start a trajectory τ in time step t_i means that we create a new trajectory $\tau_s := (p_{t_i}, \ldots, p_{t_k})$.

Hlawatsch et al. [131] observed that longer trajectories have greater accuracy than a series of shorter trajectories. However, longer trajectories may bundle together or move away and create regions with little coverage, see Figure 8.3. Thus, we forcefully stop up to a user-defined amount of trajectories ϵ_T in each time step t_i. To stop a trajectory τ in time step t_i, we take the prefix $\tau_e := (p_{t_j}, \ldots, p_{t_i})$ instead of τ. The parameter ϵ_T depends on the dataset and the specific application. In general, it should be inversely proportional to the amount of trajectories ending. In datasets where all trajectories exist in all time steps, ϵ_T should be high. To select which trajectories to stop and which to start in time step t_i, we perform the void-and-cluster optimization. That is, we exchange the tightest cluster with the largest void up to ϵ_T times or until the sample distribution is optimal, i.e. the tightest cluster is equal to the largest void, and start or stop the corresponding trajectories. Note that longer trajectories may again be obtained by interpolation from shorter ones [3].

8.3 PARALLEL IMPLEMENTATION

In this section, we discuss the parallel implementation of the sampling technique, specifically the computation of the point and sample densities, and the parallelization of the void filling step.

8.3.1 *Computing the Densities*

One of the most computationally demanding parts of the algorithm is creating the density ρ_P and updating λ_S. Each point p has to scatter its density, weighted by the distance and kernel function, to all neighboring points $\mathcal{N}(p)$. This is an embarrassingly parallel task and is especially well-suited for GPU acceleration. If the kernel function is compact, data structures, such as a k-d tree or a regular grid, should be employed to efficiently retrieve the neighborhood of a point or sample.

In our implementation, we use a uniform grid of cell size h. To find the neighborhood $\mathcal{N}(p)$ of p, we thus have to query 3^d cells around p. To speed-up the neighborhood search, we layout all cells in memory using a space-filling Z-curve to optimize memory access to neighboring cells.

8.3.2 *Parallel Void Filling*

The void filling step seems to enforce a sequential bottleneck: in each step, we find the sample $p \in P \setminus S$ with the smallest sample density $\lambda_S(p)$. Then we add p to the sample set S and increase sample densities in the neighborhood before we search the smallest sample density again. To overcome this sequential dependency, we store each added sample p alongside the sample density $\lambda_S(p)$ that it has when it is added. Since we only ever add to the densities and pick the minimum in each step, these densities grow monotonically. If we can guarantee that they are computed correctly for each added sample, sorting by the densities guarantees that we rank all selected samples correctly.

To provide this guarantee, we must never add a sample too early. All samples in a neighborhood $p_n \in \mathcal{N}(p)$ with a rank $r(p_n) < r(p)$ must have contributed to the sample density $\lambda_S(p)$ before we add $p \in P \setminus S$ and store $\lambda_S(p)$. We can be certain that this is the case if p has the smallest sample density in its neighborhood, i.e.

$$\lambda_S(p) \leq \min_{p_n \in \mathcal{N}(p)} \lambda_S(p_n). \tag{8.10}$$

By selecting the sample $p \in P \setminus S$ that minimizes $\lambda_S(p)$ globally, we will never select another sample in $\mathcal{N}(p)$ before p. Hence, we know that the rank $r(p)$ is also minimal within the neighborhood $\mathcal{N}(p)$.

This principle enables our parallel implementation. We first sort all $p \in P \setminus S$ in ascending order by their density $\lambda_S(p)$ and take the first n points

$p_0, p_1, \ldots, p_{n-1}$ in each iteration. Then we compute an adjacency matrix in parallel using the kernel size h:

$$A := \begin{pmatrix} 0 & 0 & \ldots & 0 \\ \|p_1 - p_0\| - h & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \|p_{n-1} - p_0\| - h & \ldots & \|p_{n-1} - p_{n-2}\| - h & 0 \end{pmatrix}.$$

A negative entry in the i-th row and j-th column with $i > j$ indicates that p_i is in the neighborhood of p_j. Since $\lambda_S(p_i) \geq \lambda_S(p_j)$ due to the sorting, this means that p_i might not satisfy Equation (8.10) and it is flagged accordingly. Once the process is complete, all points that have not been flagged satisfy Equation (8.10). They are added to the sample set and their densities are stored. Note that the matrix A is never stored. We only need the flags.

Although we can parallelize this computation, the workload is unevenly distributed. The i-th row of A has i non-zero entries. Therefore, we index the non-zero entries of A with a single linear index $k \in \{0, \ldots, \frac{n(n-1)}{2}\}$. In Section A.2, we show that row and column indices can be computed from this flat index through

$$i = \left\lfloor \frac{1}{2} + \sqrt{\frac{1}{4} - 2k} \right\rfloor, \qquad j = k - \frac{(i-1)i}{2}. \tag{8.11}$$

For large k, the square root has to be evaluated in double-precision to avoid rounding errors.

Thanks to the sorting by density, the procedure described above guarantees a correct relative rank of selected samples. However, samples may be missing if we just terminate after a particular iteration. For a complete result, we perform additional iterations. If the largest density of a sample added in the last proper iteration was $\lambda_{\max}$, we continue iterating until the smallest sample density in $P \setminus S$ is greater than $\lambda_{\max}$. At this point, we can be certain that we have not missed a sample that should have been added up until the last proper iteration. This way, we guarantee that we take the same samples as the sequential algorithm.

8.4 LOCAL ERROR MEASURE

In this section, we discuss an error measure to quantify how well a set of samples represents a dataset. We propose a measure that takes not only the spatial domain into account, but also how well the value domain is represented in each region of the dataset. To this end, we first discuss how such a local error can be defined, before we discuss how to compare value distributions. Lastly, we discuss an error guided sampling strategy that relies on an efficient iterative error estimation to sample just below a given error threshold, instead of drawing a fixed amount of samples.

8.4.1 *Locality and Continuity*

We derive a local error measure that compares the value distribution $V_S \subset V$ of the sampled dataset with the value distribution V of the original dataset. Specifically, we propose to compare value distributions in the local neighborhood $\mathcal{N}(p)$ for each corresponding $p \in P$. This method implicitly accounts for non-uniformly distributed data points. Additionally, we weight the contribution of each $p_i \in \mathcal{N}(p)$ to the value distribution by its distance $k(\|p - p_i\|)$ so that the error varies smoothly over the spatial domain.

8.4.2 *Wasserstein Distance*

To measure the difference between the original value distribution given by values $V = \{X_0, ..., X_{n-1}\}$ and a sampled subset $V_S \subset V$, we use the corresponding cumulative distribution functions (CDFs) F_V and F_S. The CDF at a point $p \in P$ is estimated as

$$F_V(p,t) = \frac{1}{\sum_{i=0}^{n-1} k(\|p - p_i\|)} \sum_{i=0}^{n-1} \begin{cases} k(\|p - p_i\|) & \text{if } X_i \leq t, \\ 0 & \text{otherwise.} \end{cases} \tag{8.12}$$

In practice, this implies that we need to sort the X_i before accumulating them. Since the samples are a subset $V_S \subset V$, it is sufficient to sort the values V to estimate both CDFs. Alternatively, we estimate the CDFs based on a histogram of V and V_S, which introduces a discretization, but is more efficient to evaluate.

To measure the distance, we found the Wasserstein distance [237], or earth movers distance, to be a good choice. In the one-dimensional case, it is defined as the L1-norm between the two CDFs:

$$\mathrm{W}(p, F_V, F_S) := \int_{-\infty}^{\infty} |F_V(p,x) - F_S(p,x)| \, \mathrm{d}x. \tag{8.13}$$

In contrast, we found the Kolmogorov-Smirnov distance, defined as the infinity norm between the CDFs, to be unsuited since it is not robust to small shifts in the value dimension.

Note that this definition of the Wasserstein distance is only valid for one-dimensional value distributions. Thus, we compute a separate error for each value dimension. We can further deduce the error across dimensions, e.g. by taking the mean or maximum. In the following we will use the maximum; however, this is an application and data specific decision.

8.4.3 *Error Guided Sampling*

During void-and-cluster sampling, we efficiently keep track of the error distribution, for example to stop sampling if the average error falls below a given threshold. In detail, we compute the error for all samples after the initial void-and-cluster optimization. When adding a sample $p_{\min}$, we compute the error of $p_{\min}$ and additionally update the error for all neighbors $\mathcal{N}(p_{\min})$ since these have changed as well.

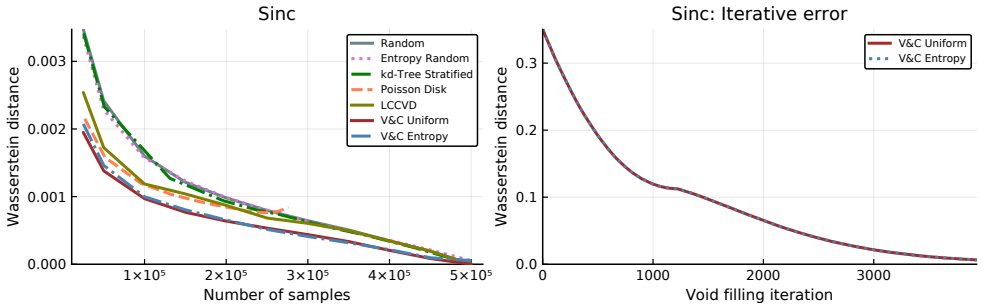


Figure 8.4: Left: Comparison of different sampling strategies using our proposed error measure. Right: Error measured during sampling.

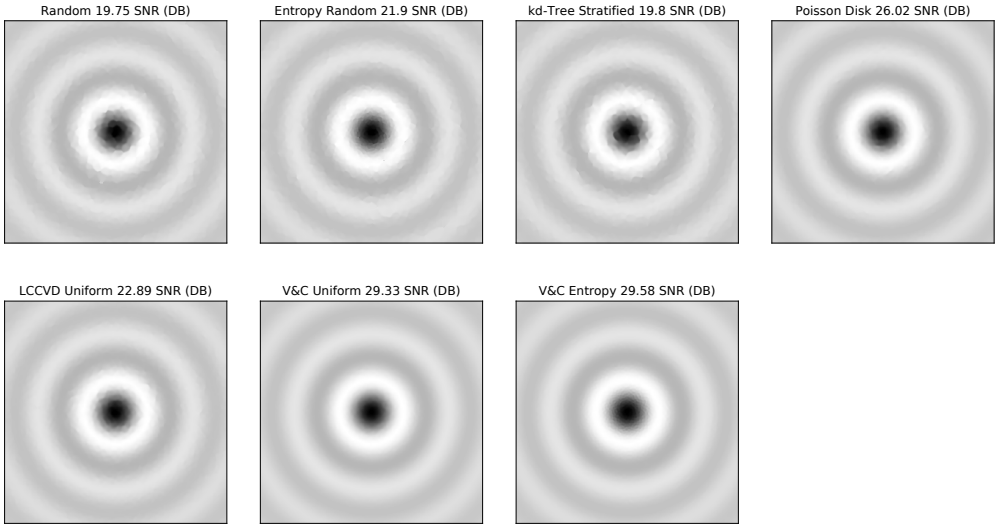


Figure 8.5: Reconstruction of the sinc dataset using scattered data interpolation after taking 5000 samples with different strategies.

To describe the distribution of errors during sampling, we found the average error to be a robust statistic that is efficient to compute. In contrast, the maximal error does not decrease smoothly with respect to the number of samples and is not robust against outliers, e.g. stemming from small, but complex value regions.

8.5 EVALUATION

In this section, we evaluate our sampling technique using four real-world datasets and the synthetic sinc signal.

8.5.1 *Synthetic Data: Sinc*

We have created the sinc dataset by randomly placing 500,000 points in the domain $[-5,5]^2$ and by evaluating for each point $p \in [-5,5]^2$ the function

$$\operatorname{sinc}(\|p\|) = \frac{\sin(\pi\|p\|)}{\pi\|p\|}. \tag{8.14}$$

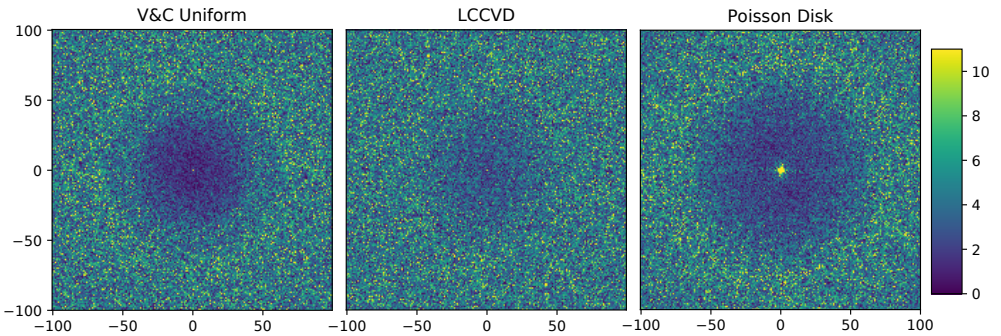


Figure 8.6: Fourier transform of the sinc dataset after taking 5000 samples using our uniform void-and-cluster method, loose capacity constrained Voronoi diagrams (LCCVD), and Poisson disk sampling.

Figure 8.4 (left) compares different sampling strategies and shows the mean Wasserstein distance. We employ simple random sampling and random sampling with non-uniform probabilities based on the entropy. Moreover, we compare to stratified sampling utilizing a k-d tree based on a median split, similar to Woodring et al. [335]. Lastly, we employ Poisson disk sampling [29] and loose capacity constrained Voronoi diagrams (LCCVD, [79]). For an increasing number of samples, the error from most strategies converges to zero, which implies that these strategies sample a representative subset. However, Bridson's Poisson disk sampling [29] lacks explicit control over the sample count, which is instead steered by the enforced minimal and maximal distance. The technique is unable to surpass a certain sample count for this dataset. Our proposed void-and-cluster sampling strategies perform consistently better for all sample counts. The entropy-based strategies perform similar to their uniform counter parts.

In Figure 8.4 (right), the mean error has been computed iteratively during sampling until the error was less than $\epsilon = 0.0065$, which led to a sampling percentage of 34.1 %. The error first falls rapidly then converges asymptotically to zero. Initially, we sample 5000 and iteratively add the remaining samples. In each void filling step, we take only 32 samples in parallel to still keep the error up to date. In comparison, we take up to 12,288 voids in parallel if we do not perform error guided sampling.

We use scattered data interpolation to interpolate the sampled data values to a grid of size 1024^2, see Figure 8.5. Our void-and-cluster strategies show a major improvement compared to the other sampling strategies. The signal-to-noise (SNR) ratios shown in the logarithmic decibel scale support this assessment. Note that the LCCVD and Poisson disk sampling strategies also achieve good results, but are still worse than our proposed methods. For reconstruction, the entropy-based sampling strategies perform slightly better compared to the uniform approaches. For all sampling strategies, the quality of the reconstruction agrees with the error measure.

Lastly, Figure 8.6 shows the spectrum of the sinc dataset reduced to a subset of 1 % with our uniform void-and-cluster strategy, LCCVD, and with Poisson disk sampling. The Fourier transform shows the blue noise property for these methods. Low frequencies are substantially weaker and the spectrum is

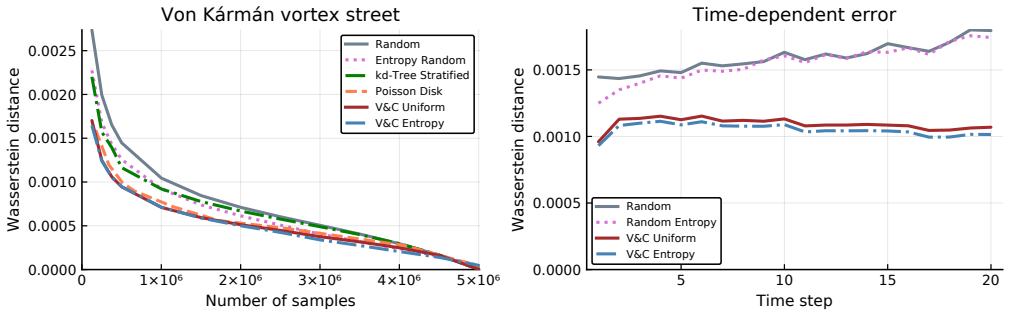


Figure 8.7: Comparison of the local error measure after sampling a single time step of the von Kármán vortex street (left) and over all time steps (right).

isotropic. However, LCCVD has more energy in low frequencies than our void-and-cluster strategy. Poisson disk sampling has less energy in low frequencies, but contains a noticeable spike near zero. Note that the random and stratified sampling strategies do not have this property, which suggests that the blue noise property is desirable for scattered data interpolation. Indeed, the error of kernel estimation has been shown to depend on the disorder of particles [216].

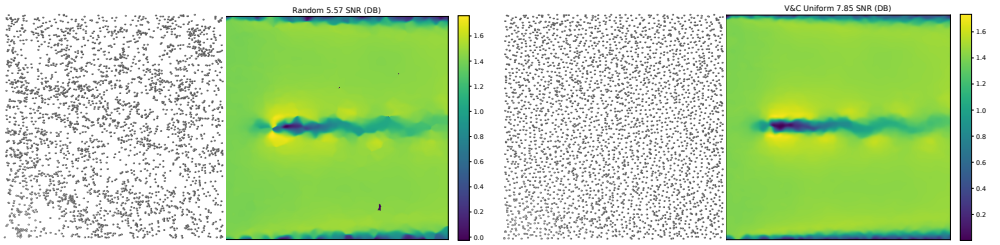
8.5.2 *Von Kármán Vortex Street*

The von Kármán vortex street is a time-dependent SPH dataset. It contains about 5 million particles in each time step. Since the particles enter the domain on the left side and exit on the right, the amount of particles per step changes. A circular boundary in the mid of the domain causes a repeating pattern of swirling vortices, the vortex street.

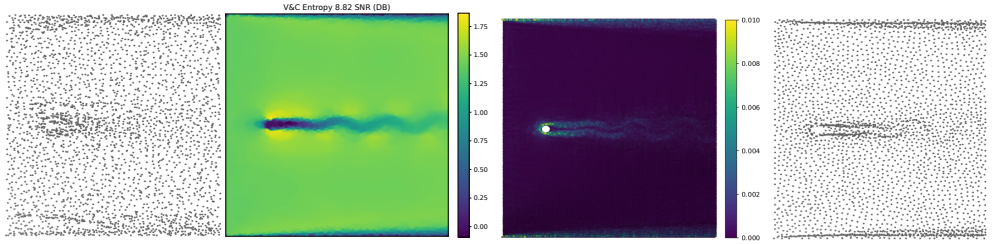
We compare the error measured from the different techniques for sampling the first time step in Figure 8.7 (left). Since the original dataset already contains well distributed samples, as a result of the SPH simulation, the Poisson disk sampling can correctly sample the dataset even for large sample counts. Still, the void-and-cluster techniques consistently lead to the lowest error. The decreased error of stratified k-d tree sampling and entropy random sampling compared to naive random sampling implies that both stratification and entropy-based sampling are beneficial for this dataset.

We sample 10 % of the trajectories in the discrete time interval $[0, 20]$. Since particles frequently enter and exit the domain, we do not explicitly stop trajectories to sample new ones. In Figure 8.7 (right) we plot the error over time for the different sampling strategies. The void-and-cluster strategy is not only consistently better, but also stays nearly constant over time. In contrast, we observe an increase of the error over time for the random sampling techniques. Lastly, the entropy-based sampling strategies show a noticeable improvement for this dataset because they are able to focus more samples on the difficult vortex and boundary regions.

A comparison between random sampling, uniform, and entropy void-and-cluster sampling after 10 time steps is shown in Figure 8.8. Although all sampling strategies deteriorate slightly over time, the samples are still well distributed for the uniform void-and-cluster sampling approach. We recon-



(a) Random sampling in t_{10} (b) Void-and-cluster uniform sampling in t_{10}



(c) Void-and-cluster entropy sampling in t_{10} (d) Entropy sampling error in t_{10} (e) Entropy sampling distribution in t_1

Figure 8.8: The von Kármán vortex street after ten time steps using random (a), uniform (b), and entropy (c) void-and-cluster trajectory sampling. The corresponding u-velocity fields are shown, which have been created using scattered data interpolation. Our error measure is shown in (d). The entropy void-and-cluster sample distribution in the first time step is shown in (e).

struct the u-velocity field using scattered data interpolation. The results are considerably better for the void-and-cluster approaches. Moreover, the entropy sampling strategy leads to a better reconstruction compared to the uniform void-and-cluster technique. Our error measure of the entropy strategy is shown in (d). Although the entropy strategy already places most samples near the vortex street, the lower boundary, and the upper boundary, the error is still highest in these regions.

We illustrate the sample distribution for the entropy sampling technique in the first time step in Figure 8.8 (e). More samples are placed behind the circular boundary, where vortex shedding occurs, and near the bottom and top of the domain. In these regions, the velocity differs considerably. The sampling distribution in the tenth time step has deteriorated considerably for the entropy strategy, but still leads to better results. The entropy strategy thus seems to require shorter trajectories to accurately place samples with respect to the entropy.

8.5.3 *Surface-Mounted Cylinder*

This dataset stems from a 3D SPH simulation that simulates the flow around a surface-mounted cylinder (cf. Section 5.3.3). In detail, particles move through a wall-bounded box where an empty cylinder is placed on the bottom. The dataset contains about 46 million particles in each time step, each of which has a position, velocity, and pressure and either belongs to the static domain

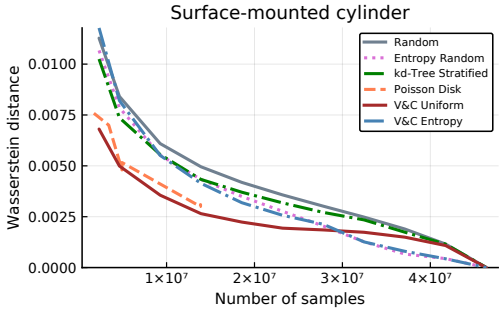
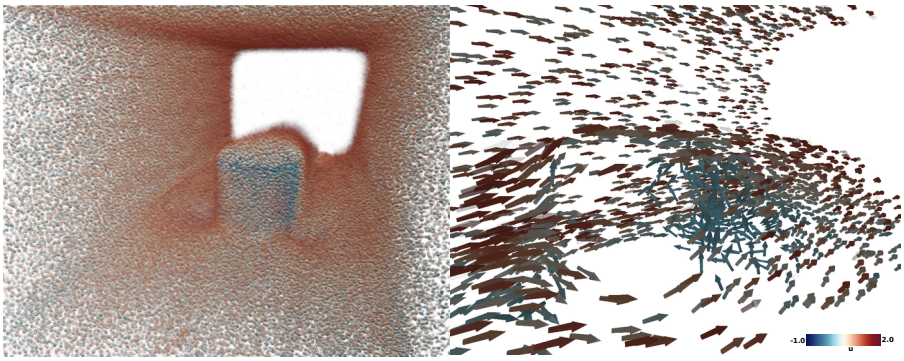


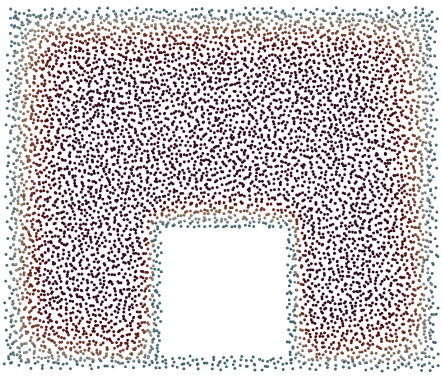
Figure 8.9: Comparison of the local error measure of the surface-mounted cylinder.

boundary or the simulated fluid. In Figure 8.9, we compare the error of the different sampling techniques. Most notably, the entropy-based techniques show a larger error for smaller sample counts.

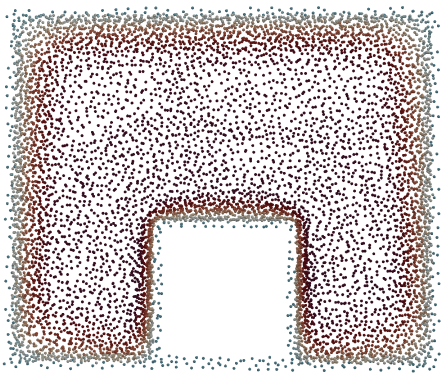
We sample 1 % of the dataset and visualize the particles as sphere and arrow glyphs in Figure 8.10 (a). We map u-velocity to color, i.e. velocity in



(a) We visualize the particles using sphere and arrow glyphs with our level-of-detail representation

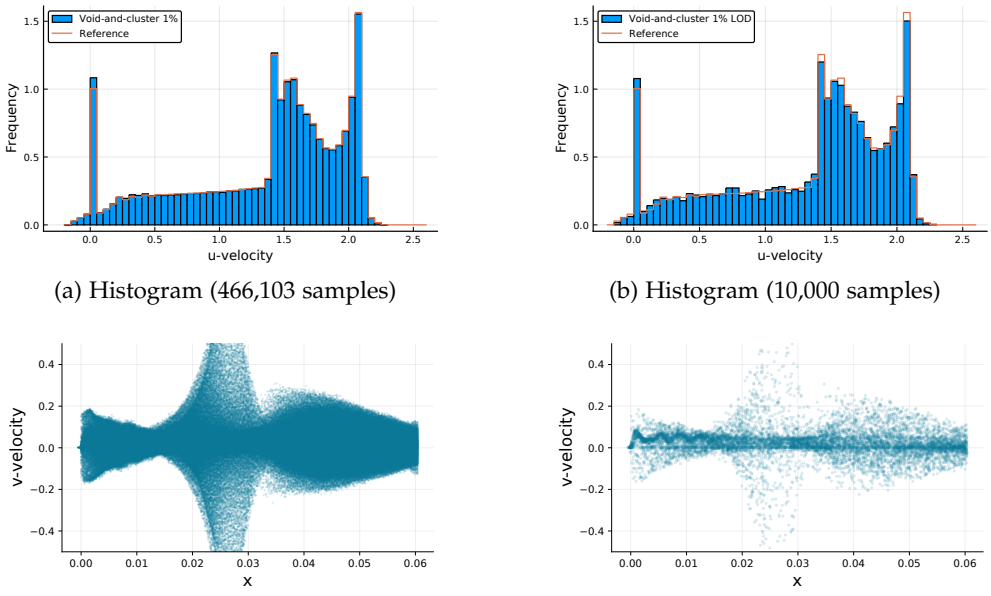


(b) Uniform sampling



(c) Entropy sampling

Figure 8.10: The surface-mounted cylinder, after sampling 466,103 particles, is shown in (a). We use the continuous level-of-detail, in addition to a transfer function, to further reduce the amount of particles. Slices of the dataset using the uniform (b) and entropy (c) sampling illustrate the difference between the sampling strategies. The entropy strategy samples the less interesting region above the empty cylinder less densely.



(a) Histogram (466,103 samples)

(b) Histogram (10,000 samples)

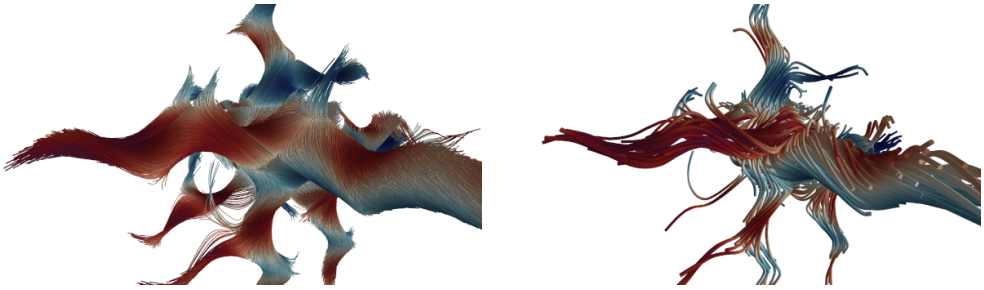
(c) Scatter plot (466,103 samples)

(d) Scatter plot (10,000 samples)

Figure 8.11: We create a histogram (a) and a scatter plot (c) of the surface-mounted cylinder, after sampling 466,103 particles using the void-and-cluster entropy strategy. With our level-of-detail, we select a subset of 10,000 particles and create a histogram (b) and scatter plot (d).

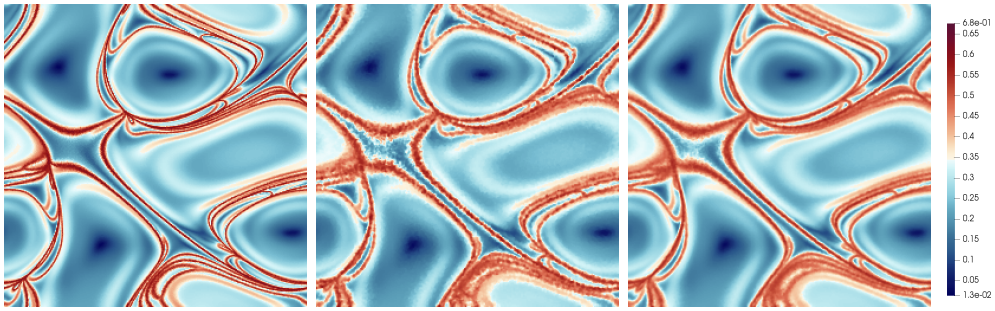
the principal flow direction. Since the sampled subset still contains a large amount of particles, we make use of the continuous level-of-detail in addition to a transfer function, which maps fast particles to transparent, to further reduce the amount of visual clutter. The vortex shedding in the wake of the cylinder thus becomes visible. Furthermore, we illustrate the difference between uniform and entropy void-and-cluster sampling in Figure 8.10 (b) and (c). The entropy strategy samples the regions near the wall and close to the cylinder more densely due to fluctuating velocities, but also due to the interface between fluid and boundary particles that leads to a high entropy. In contrast, the regions above and next to the cylinder contain a large amount of particles that move unobstructed through the domain.

In Figure 8.11, we create a histogram of u-velocity (a) and a scatter plot of x and v-velocity (c) of the dataset sampled with the entropy void-and-cluster technique. We use our level-of-detail mechanism to create a subset of 10,000 particles and compute similar plots in (b) and (d). The histograms in (a) and (b) are similar, even though we reduce the amount of samples considerably. In the scatter plot (d), the amount of clutter is significantly reduced. The periodic changes in v-velocity, caused by the swirling vortices in the wake of the cylinder, then become visible. Lastly, the ordering of samples allows us to optimize loading times and latency. In particular, when opening a new file or time step, we initially load only a small subset and asynchronously continue to load more samples to reduce the latency. Especially for larger datasets, we found working with a small subset of the data to be preferable due to the fast and less cluttered visualizations.



(a) 50,000 pathlines (b) 500 pathlines

Figure 8.12: We have sampled pathlines of the ABC flow with our technique, which implicitly defines a continuous level-of-detail, to render a greater (a) and smaller subset (b) of the trajectories.



(a) Reference (b) Random (c) Void-and-cluster

Figure 8.13: Slices of the finite-time Lyapunov exponent (FTLE) from the ABC flow. The reference, computed on all trajectories, is shown in (a). We have computed the FTLE after sampling 10 % of the trajectories by random sampling (b) and using the uniform void-and-cluster (c) technique.

8.5.4 *The ABC Flow*

The Arnold-Beltrami-Childress (ABC) flow is a three-dimensional, steady velocity field:

$$\begin{aligned}\dot{x} &= A\sin z + C\cos y\\ \dot{y} &= B\sin x + A\cos z\\ \dot{z} &= C\sin y + B\cos x.\end{aligned}$$

We set $A = \sqrt{3}$, $B = \sqrt{2}$, $C = 1$. We represent the flow in the Lagrangian basis with 134,217,728 trajectories that start in the spatial domain $[0, 2\pi]^3$ and are integrated using a 4th-order Runge-Kutta scheme over the time interval $[0, 10]$. We then sample 10 % of the trajectories, without stopping and starting new trajectories.

In Figure 8.12, 50,000 and 500 trajectories are shown as illuminated pathtubes using the continuous level-of-detail that is implicitly given by the rank of our sampling strategy. Since we reorder the samples by their rank, we only have to load the first samples for visualization and can load additional samples progressively.

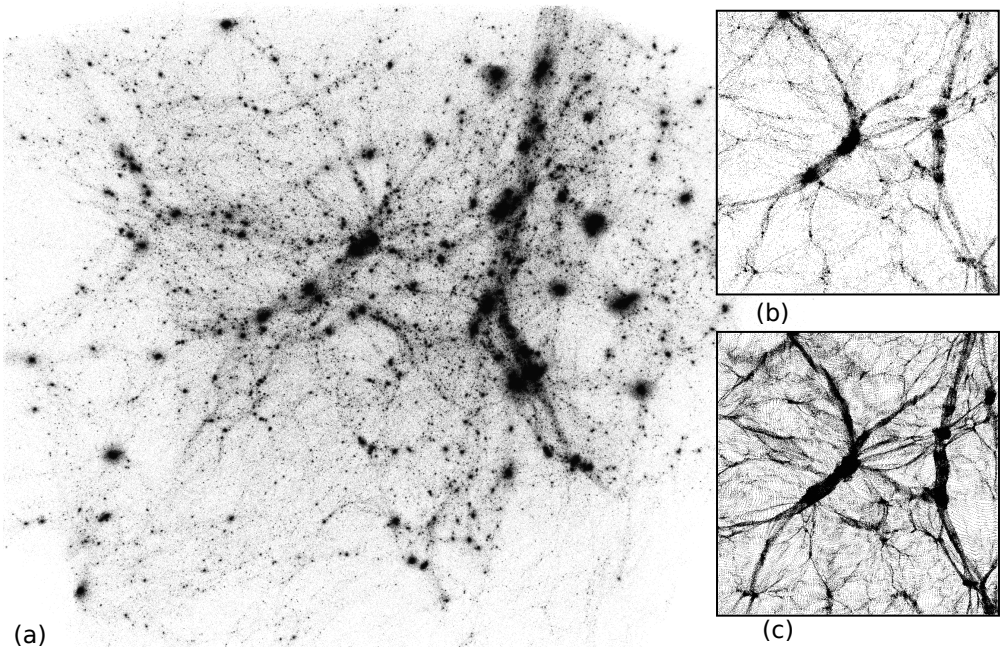


Figure 8.14: The Dark Sky dataset reduced to 5 % using the uniform void-and-cluster technique (a). A slice of the dataset is shown in (b), with the corresponding slice from the original dataset in (c).

After random sampling and uniform void-and-cluster sampling, we have computed the (forward) finite-time Lyapunov exponent. A slice of the FTLE is shown in Figure 8.13. Computing the FTLE after sampling with the uniform void-and-cluster strategy yields better results compared to random sampling, even though the same number of samples have been taken.

8.5.5 *Dark Sky*

The Dark Sky simulations are a series of cosmological N-body simulations of the evolution of the large-scale universe [293]. We study a subset that consists of 111 million particles with a position, velocity, and unique identifier. Figure 8.14 shows a visualization of the dataset reduced to 5 %. Since the simulations investigate the clustering of particles into galaxies, filaments, and the emergence of cosmic voids, the spatial distribution of the particles is strongly non-uniform. Consequently, a sampled subset of the data should preserve this distribution of cosmological mass. This is not possible using Poisson disk sampling or entropy-based adaptive sampling. In contrast, our uniform void-and-cluster technique optimizes the blue noise property with respect to the spatial density of particles in the dataset. The spatial distribution is thus preserved, whilst the samples are optimally stratified.

8.5.6 *Performance and Scalability*

To assess the performance of our algorithm, we compare the run time of different sampling strategies. For the measurements, we use an Intel Core

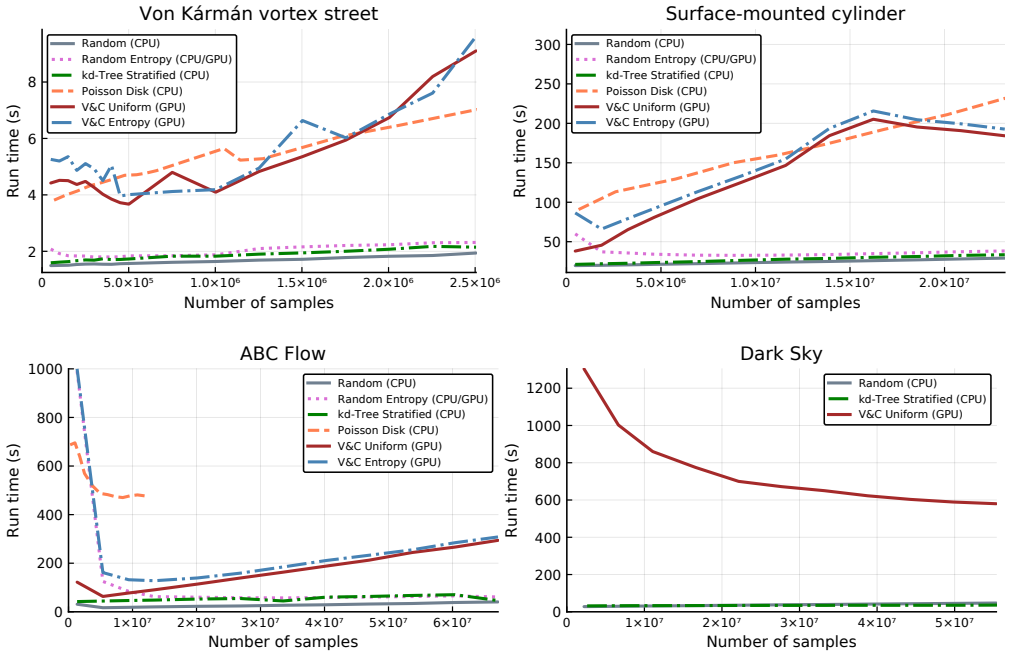


Figure 8.15: The sampling performance on the von Kármán vortex street, the surface-mounted cylinder, the ABC flow, and the Dark Sky dataset.

i7-6700 and an Nvidia Quadro RTX 8000. We enable GPU acceleration where possible.

Measurements for all of our datasets are shown in Figure 8.15. We were not able to measure our implementation of LCCVD for larger datasets since it is computationally demanding and we did not parallelize it. Although Poisson disk sampling is a linear time algorithm, it is inherently sequential and leads to long run times for large data sizes. Random and stratified sampling are fast even though no GPU acceleration is used. In comparison, the void-and-cluster techniques are slower, but considering the data sizes we argue that the performance is acceptable. For example, we take 1,342,177 samples out of 134 million from the ABC flow dataset in 68 seconds. Due to the non-uniform input data of the Dark Sky simulation, a large kernel support is required that leads to a significantly increased run time. The use of adaptive kernel sizes or better suited data structures could potentially improve the efficiency of the neighborhood search for non-uniformly distributed datasets.

The uniform and entropy-based sampling strategies perform similar. However, for small sampling percentages the entropy computation is noticeably slower since the computation depends on the kernel size h, which increases for smaller sampling percentages, and scales with the input data size. This is especially visible in the ABC flow where the run-time of the entropy computation increases dramatically for smaller sampling percentages due to the neighborhood lookup limiting the GPU efficiency. In general, the run-time increases when a larger number of samples is taken, which indicates that the void filling step is the bottleneck.

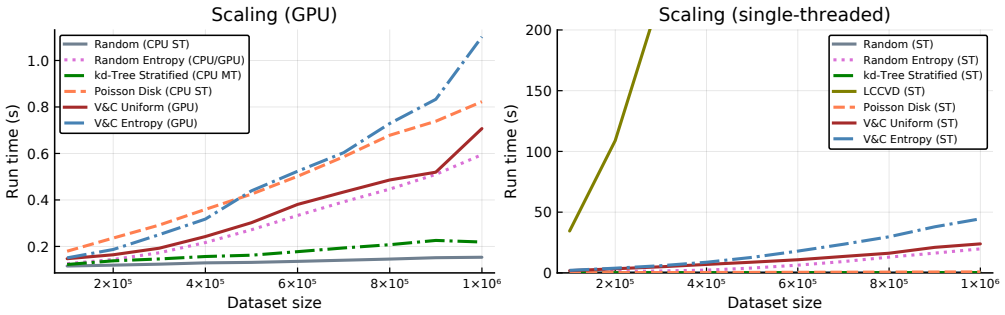


Figure 8.16: We evaluate the scalability of our algorithm using differently sized sinc datasets, whilst always sampling 10 %.

Lastly, to measure the time complexity and scalability of the algorithm, we measure GPU and single-threaded CPU performance with differently sized sinc datasets. The measurements are shown in Figure 8.16. Although our GPU implementation is competitive with random and stratified sampling, the single-threaded CPU implementation is considerably slower. This highlights the benefits of parallelization and GPU acceleration for our algorithm. Compared to LCCVD our single-threaded implementation achieves an enormous speed-up. Although not shown in the plot, LCCVD took more than 2500 seconds to sample a dataset of size 10^6.

8.6 FUTURE WORK: MULTI-NODE PARALLELISM

An important use case that has not been addressed so far is the parallel implementation for distributed memory systems. To scale the sampling technique to multiple nodes on a compute cluster, the spatial domain could be subdivided into uniform tiles. Each compute node then performs void-and-cluster sampling of one tile. Although this will produce tiles that are well distributed according to the blue noise property, the samples in the border region between two or more tiles are not necessarily well separated. If this is not acceptable, then the cluster-and-void optimization step can be applied again to the whole dataset. However, this would have to be performed on a single node which might not be possible, for example due to memory constraints. The extension of the proposed algorithm for multi-node parallelism is thus still an open problem.

8.7 DISCUSSION

In this chapter, we introduce a novel sampling strategy for reducing large scattered data and trajectories. Our void-and-cluster technique optimizes sample distributions with respect to the blue noise property and thus produces samples that evenly cover the spatiotemporal domain. In addition, we discuss adaptive sampling, for example based on the entropy of the value domain. Compared to prior work, our technique is thus able to improve the accuracy of operations such as scattered-data interpolation.

Our technique can be employed in situ or as a preprocess to reduce large datasets to more manageable, but representative datasets. Compared to other approaches to data reduction (Section 3.2), this avoids the curse of dimensionality and still allows for computing derived variables, such as the FTLE. Additionally, the introduced ordering of samples enables progressive data loading and a continuous level of detail that is used for interactive visualization. As our results show, this is also effective in reducing visual clutter in glyph-based visualizations.

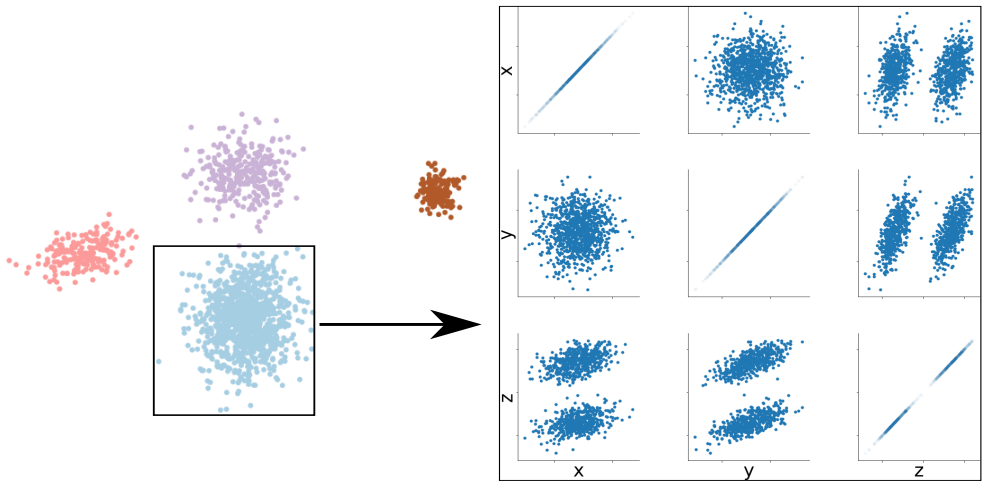
9 PROBABILISTIC SUMMARIES

To deal with large amounts of data, recent approaches employ probabilistic data summaries to represent blocks of volumetric data as probability distributions, cf. Section 3.2. These approaches have been mostly limited to univariate, volumetric data. In this chapter, we present a representation that supports arbitrarily structured, time-dependent, and multivariate data defined in a two- or three-dimensional spatial domain. Compared to our sampling approach presented in Chapter 8, this probabilistic representation limits the computation of derived variables, such as the FTLE, but enables the interactive visual analysis of extreme-scale data, for example those produced from state-of-the art cosmological simulations.

To this end, the data needs to be partitioned, i.e. clustered into spatially coherent regions. In each cluster, we make use of Gaussian mixture models (GMMs) to compactly represent a probability distribution of the data using a weighted combination of Gaussian components. However, multivariate data requires modeling high-dimensional distributions, which suffer from the curse of dimensionality. Our approach is based on the observation that representations of low-dimensional marginal distributions suffice to visualize the data. All common visualizations, such as scatter plots, histograms, and parallel coordinate plots, require only 1D or 2D distributions. The exception is the spatial domain of scattered data in 3D for which we employ a 3D distribution. Thus, we represent the marginal distributions of all individual dimensions and pairs of dimensions as well as the spatial 3D distribution. We introduce our approach formally in Section 9.1.

For large data, common item-based visualizations, such as scatter and parallel coordinate plots, are challenged by overdraw and cluttering. Frequency-based visualizations are a viable alternative in this case [182, 229]. Density estimation [292] is a frequency-based approach commonly used in statistics. However, its usage in interactive visualization has been limited due to performance considerations. Although our approach supports all common visualization techniques, it is especially well suited to density-based techniques since our modeled distributions are already an estimate of density. In Section 9.3, we discuss the efficient visualization and interaction with density-based plots using our compact representation. Additionally, we consider time-dependent histograms that would otherwise be infeasible to produce for large datasets. In this view, we can interactively brush over different time steps.

To visualize the uncertainty introduced by our data representation, we propose an error metric based on the cumulative distribution function, similar to statistical goodness of fit tests. A level-of-detail mechanism allows scientists to drill down on interesting or uncertain regions in the data. Finally, we discuss the explicit visualization of outliers, which are not handled well by density-based visualizations.



(a) Clustered data (b) Scatter plot matrix

Figure 9.1: From a given clustering of the data (a), we model each cluster using combinations of low-dimensional distributions, similar to a scatter plot matrix (b).

Our last contribution is the visualization of spatial density distributions (Section 9.2). Since drawing and rendering samples from the GMMs would be infeasible for large, scattered datasets, we directly render 3D Gaussians. We derive a closed-form solution to integrate anisotropic Gaussians using a splatting approach. Back-to-front splatting has the disadvantage that it assumes non-overlapping Gaussians. Therefore, we employ moment-based order-independent transparency [223] for datasets where this is not an acceptable assumption.

Lastly, we apply and evaluate our approach on a synthetic and three real-world datasets (Section 9.4).

9.1 PROBABILISTIC SUMMARIES

We now describe the creation of probabilistic data summaries for multivariate, scattered data. We assume that the data is clustered into spatially coherent regions, see Figure 9.1. In Section 9.4, we discuss both domain specific and standard clustering techniques for scattered data. Similar to previous work, we employ Gaussian mixture models to represent data distributions in each cluster. However, these have not been applied to multivariate data. High-dimensional Gaussian mixture models require immense computational effort and due to the curse of dimensionality, there are not enough samples to cover a multi-dimensional space extensively.

Our approach is based on the observation that we do not require more than three data-dimensions at once to employ common interactive visual analysis techniques. In fact, the visualization of the spatial distribution is the only aspect considering correlations of three dimensions. Therefore, our approach is to only generate GMMs for the relevant combinations of dimensions. By default, these are all individual dimensions, all pairs of dimensions (cf. Figure 9.1)

and all vectorial attributes. As for high-dimensional GMMs, the storage cost grows quadratically with the number of dimensions. To better reason about our approach, we first introduce it more formally.

9.1.1 *Data Model*

Our data consists of $n \in \mathbb{N}$ samples, where each sample has a position $p \in \mathbb{R}^d$. In this chapter, we assume $d = 3$, although the approach is also applicable to the simplified case of scattered data in 1D or 2D space. Additionally, each sample references $m_v - 1 \in \mathbb{N}_0$ vectorial attributes and $m_s \in \mathbb{N}_0$ scalar attributes from the multivariate value range V. We denote the data for sample $i \in \{0, \dots, n-1\}$ by:

- $v_{i,0} \in \mathbb{R}^{1\times 3}$ for the position,
- $v_{i,j} \in \mathbb{R}^{1\times 3}$ for vectorial attribute $j \in \{1, \dots, m_v - 1\}$,
- $s_{i,j} \in \mathbb{R}$ for scalar attribute $j \in \{0, \dots, m_s - 1\}$.

To define our probabilistic summaries, we concatenate all attributes for sample $i \in \{0, \dots, n-1\}$ into a single vector with $m_u := 3m_v + m_s$ entries to enable linear indexing:

$$u_i := (v_{i,0}, v_{i,1}, \dots, v_{i,m_v-1}, s_{i,0}, \dots, s_{i,m_s-1}) \in \mathbb{R}^{1\times m_u}.$$

GMMs are generated for each given cluster $\mathbb{I} \subseteq \{0, \dots, n-1\}$ and for each relevant combination of dimensions. First, we generate 1D models for each attribute $j \in \{0, \dots, m_u - 1\}$:

$$(u_{i,j})_{i\in\mathbb{I}} \in \mathbb{R}^{|\mathbb{I}|\times 1}.$$

Then, we generate 2D models for each pair of dimensions $j, k \in \{0, \dots, m_u - 1\}$ with $j < k$:

$$(u_{i,j}, u_{i,k})_{i\in\mathbb{I}} \in \mathbb{R}^{|\mathbb{I}|\times 2}.$$

Finally, we generate 3D models for each vectorial attribute $j \in \{0, \dots, m_v - 1\}$:

$$(v_{i,j})_{i\in\mathbb{I}} \in \mathbb{R}^{|\mathbb{I}|\times 3}.$$

Of course, the generation of GMMs for particular combinations of attributes may be skipped if the analysis of their mutual dependence is of no interest in a specific application.

Our probabilistic summary is the combination of all these low-dimensional GMMs for all clusters. They capture all information needed for common interactive visual analysis techniques but limit the analysis of higher dimensional correlations. By modeling only low-dimensional distributions, the curse of dimensionality does not apply. Ultimately, this limitation enables us to create reliable models of multivariate data.

9.1.2 *Gaussian Mixture Models*

We use GMMs because they offer a compact and efficient representation of the target distributions, see Section 3.2. In the following, we provide more details on our fitting procedure.

We generate a GMM for each combination of a cluster $\mathbb{I}$ and a relevant subset of attributes $\mathbb{J} \subseteq \{0, \dots, m_u - 1\}$. As explained above, the number of attributes $|\mathbb{J}|$ is one, two, or three. A GMM is indexed by a pair $g := (\mathbb{I}, \mathbb{J})$ and consists of $n_g \in \mathbb{N}$ weighted Gaussians. Gaussian $l \in \{0, \dots, n_g - 1\}$ is given by its weight $w_{g,l} > 0$, its mean $\mu_{g,l} \in \mathbb{R}^{|\mathbb{J}|}$, and its covariance $\Sigma_{g,l} \in \mathbb{R}^{|\mathbb{J}| \times |\mathbb{J}|}$. The density of a GMM at $p \in \mathbb{R}^{|\mathbb{J}|}$ is the weighted sum of the individual Gaussian densities:

$$\rho_g(p) := \sum_{l=0}^{n_g-1} \frac{w_{g,l}}{\sqrt{|2\pi\Sigma_{g,l}|}} \exp\left(-\frac{(p-\mu_{g,l})^\mathsf{T}\Sigma_{g,l}^{-1}(p-\mu_{g,l})}{2}\right).$$

We compute the parameters of a GMM from a sequence of input samples with the expectation maximization (EM) procedure.

9.1.3 *Fast Selection of GMM Components*

The EM algorithm takes the number of Gaussian components n_g as input. With more components, the target distribution can be modeled better. However, too many components may not significantly improve the model, but increase the storage overhead. We adaptively select the appropriate number of components, but propose approximations to significantly reduce the computational complexity.

We iteratively fit GMMs with an increasing number of components up to a user specified maximum and select the GMM with the best Bayesian information criterion (BIC) [280]. For our data model, the BIC is defined using the number of free parameters k_{GMM} in the GMM as

$$k_{\text{GMM}} \log |\mathbb{I}| - 2\log(\mathcal{L}_p),$$

where $\mathcal{L}_p$ denotes the maximized likelihood and k_{GMM} is given by

$$k_{\text{GMM}} := n_g \left(\frac{|\mathbb{J}|(|\mathbb{J}|+1)}{2} + |\mathbb{J}|\right) + |\mathbb{J}| - 1.$$

The iterative computation of GMMs with different numbers of components is computationally challenging, especially for large clusters. To speed up the selection of the best n_g, we propose two approximations: First, we take a random subset $\mathbb{I}_S \subset \mathbb{I}$ of our cluster whilst iteratively estimating the GMMs. After we have selected the best n_g based on the BIC, we recompute the GMM with n_g components for the whole cluster $\mathbb{I}$.

Second, after we have selected the number of components $\{n_0, \dots, n_{m_u-1}\}$ for all one-dimensional GMMs, we use them as lower and upper bounds for

the two- and three-dimensional GMMs. In detail, for a subset of attributes $\mathbb{J} \subseteq \{0, \dots, m_u - 1\}$, we define the lower bound as

$$n^{\min}_{\mathbb{I},\mathbb{J}} := \min_{j \in \mathbb{J}} n_{\mathbb{I},\{j\}}$$

and an approximate upper bound as

$$n^{\max}_{\mathbb{I},\mathbb{J}} := \Pi_{j \in \mathbb{J}} n_{\mathbb{I},\{j\}}.$$

This implies that the higher-dimensional GMMs include at least the complexity of lower dimensions, whilst being bound by all combinations of all lower dimensional Gaussian components. In Section 9.4.5, we show that the bounds introduce no error, whilst the subsampling introduces a small error on our datasets.

Lastly, it is possible that some clusters contain only a small number of data samples. Although such a clustering may not seem optimal, it is quite likely to occur for scattered data. For very small clusters, e.g. $|\mathbb{I}| \leq 20$, fitting a GMM is problematic since the target distribution may be underdetermined. In this case, we fit a single Gaussian to these clusters.

9.2 SPATIAL VISUALIZATION

In this section, we discuss the visualization of the spatial density distribution. Although we could reconstruct the original data by drawing samples from the GMM of each cluster, this would require rendering a large amount of scattered data. Instead, we derive an efficient formulation to directly splat three-dimensional Gaussians. Additionally, we consider the application of a transfer function to a one-dimensional value distribution in each cluster.

9.2.1 *Integrating Visibility for Gaussians*

To render a trivariate Gaussian distribution, we integrate along a view ray $o + xd$ starting at $o \in \mathbb{R}^3$ in normalized direction $d \in \mathbb{R}^3$ with $x \in \mathbb{R}$. The Gaussian is given by its weight $w := w_{g,l}$, mean $\mu := \mu_{g,l} \in \mathbb{R}^3$, and covariance $\Sigma := \Sigma_{g,l} \in \mathbb{R}^{3\times3}$. To derive a general solution, we integrate over $[a, b]$ by substituting the ray equation into the trivariate Gaussian distribution:

$$I(a,b) := \int_a^b \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{(o + xd - \mu)^\mathsf{T}\Sigma^{-1}(o + xd - \mu)}{2}\right) \mathrm{d}x.$$

Through integration by substitution, see Section A.3, we obtain the following closed-form solution:

$$I(a,b) = c\frac{\sqrt{\pi}}{\sqrt{c_{d,d}}}\left[\frac{1}{2}\operatorname{erf}(y)\right]^{\sqrt{c_{d,d}}\left(b+\frac{c_{o,d}}{c_{d,d}}\right)}_{\sqrt{c_{d,d}}\left(a+\frac{c_{o,d}}{c_{d,d}}\right)}, \tag{9.1}$$

with

$$
\begin{aligned}
c_{d,d} &:= \frac{1}{2} d^\mathsf{T} \Sigma^{-1} d, \\
c_{o,d} &:= \frac{1}{2} (o - \mu)^\mathsf{T} \Sigma^{-1} d, \\
c &:= \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{1}{2} (o - \mu)^\mathsf{T} \Sigma^{-1} (o - \mu) + \frac{c_{o,d}^2}{c_{d,d}} \right).
\end{aligned}
$$

When integrating over all of $\mathbb{R}$, this result simplifies to

$$
I(-\infty, \infty) = c \frac{\sqrt{\pi}}{\sqrt{c_{d,d}}}. \tag{9.2}
$$

We could use this result inside a ray tracer, possibly with ray tracing GPUs [173]. It only has to identify relevant Gaussians per pixel, ray marching for integration becomes unnecessary. In the following, we discuss our approach using GPU rasterization, which works efficiently on widely available hardware.

9.2.2 *Back-to-Front Splatting*

To splat scattered 3D Gaussians, we sort them from back-to-front based on their mean distance to the camera. Then, we integrate the Gaussians along the viewing direction using Equation 9.2. Integrating from $-\infty$ to ∞ is generally a reasonable approximation, but it is possible that we incorrectly evaluate a Gaussian if the camera is positioned within its support. Alternatively, we could employ Equation 9.1, but this is far more expensive and only gives a benefit in rare cases.

To render a single 3D Gaussian, we first compute the principal components of the distribution to fit a bounding box along the principal axes. By default, we limit the size of the bounding box in each dimension by 3 standard deviations. This box is then rasterized and for each resulting fragment, we integrate the Gaussian along the viewing direction in a fragment shader using Equation 9.2. Finally, we tone-map the resulting density (see Equation 9.4) to better convey the high-dynamic range.

We did experience some numerical issues with some of our datasets due to very large or small spatial and value domains. We were able to address these issues by switching to a more numerically stable eigen decomposition [89, Algorithm 8.2.3]. Lastly, we make use of the Cholesky decomposition to invert covariance matrices, which behaves robustly even for nearly singular matrices [309, p. 176].

9.2.3 *Order-Independent Transparency*

The splatting approach assumes that distributions do not overlap since this could lead to visible flickering between frames when the order changes. Depending on the clustering of the data, this assumption is not always acceptable. For this reason, we propose the use of an order-independent transparency (OIT) approach to avoid sorting semi-transparent Gaussians. Although, a large

number of Gaussians are problematic for most OIT approaches, moment-based order-independent transparency (MBOIT) [223] is well-suited for this application.

9.2.4 *Uncertainty Transfer Function*

Lastly, we discuss how to apply a transfer function to the distribution of an attribute $j \in \{n_0, \ldots, n_{m_u-1}\}$ to obtain a color and opacity for each cluster $\mathbb{I}$. The one-dimensional value dimension is modeled separately from the cluster as a GMM $g = (\mathbb{I}, \{j\})$ with n_g components. For each cluster, we compute an expected color and opacity [270] by convolving the transfer function f with the value distribution:

$$E[f|g] := \int_{-\infty}^{\infty} f(p)\rho_g(p)\mathrm{d}p.$$

We insert the Gaussian mixture model into this equation and rearrange:

$$E[f|g] = \sum_{l=0}^{n_g-1} w_{g,l} \int_{-\infty}^{\infty} f(p) \frac{1}{\sqrt{2\pi\sigma_{g,l}^2}} \exp\left(-\frac{(p-\mu_{g,l})^2}{2\sigma_{g,l}^2}\right) \mathrm{d}p. \tag{9.3}$$

We efficiently evaluate this equation by precomputing the integrand, which is simply a convolution of the transfer function with differently parametrized Gaussians. The resulting 2D lookup table thus depends on the transfer function and is parameterized by mean and variance.

9.3 VISUAL ANALYSIS

Now that we are able to render the spatial distribution of our data, we move on to the visual exploration and analysis of additional data dimensions using our representation. This includes multiple views with brushing and linking coupled with a focus and context visualization to emphasize brushed values.

9.3.1 *Sample-Based Visualization*

Although we mostly focus on density-based visualization techniques due to their advantages for visualizing large datasets, we also want to support common item-based visualizations. Specifically, we obtain scatter and parallel coordinate plots by drawing samples from our probability distributions, see Figure 9.10 (c).

Since the clusters may have originally contained a different amount of data samples, we weight the number of samples that we generate for each cluster accordingly. The total number of samples to generate is surprisingly hard to determine. Although we want to avoid drawing too many samples for a faster and less cluttered visualization, we have to draw enough samples to appropriately represent the data distributions. Since an optimal choice is dependent on the data, we set the total number of samples to the original number of data samples n times a user-defined factor between 0 and 1. We

further create frequency-based visualizations, such as the 2D histogram in Figure 9.10 (c), from the generated samples or alternatively using density estimation.

9.3.2 *Density-Based Visualization*

Since we already have an estimate of density in the form of our GMMs, we efficiently construct density-based visualizations that are costly to compute otherwise. To obtain the density, we evaluate and accumulate the Gaussian distributions from the GMMs in all clusters. Since the distribution in each cluster is normalized, we additionally weight each cluster $\mathbb{I}$ by the normalized number of samples it represents $\frac{1}{n}|\mathbb{I}|$.

9.3.2.1 *Density Plots*

To compute a density in 1D or 2D, we evaluate and accumulate the Gaussian distributions, see Figure 9.7 (b) and (c). Since this operation can be parallelized trivially, we make use of GPU acceleration. In the one-dimensional case, we evaluate 1D Gaussians on the GPU and plot a probability density function. For a 2D plot, we render a quadrilateral for each Gaussian, evaluate the Gaussian for each fragment, and additively accumulate the results.

9.3.2.2 *Parallel Coordinate Plots*

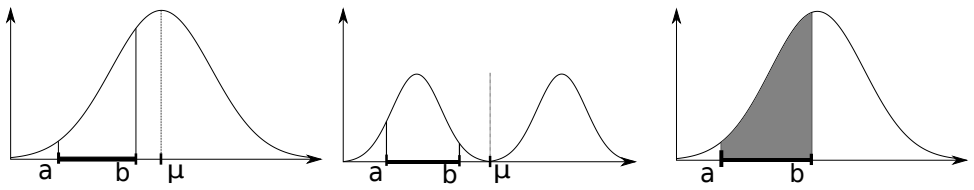
Miller and Wegman [213] formulate parallel coordinate plots for bivariate Gaussian distributions. With this formulation, we splat the 2D distributions for each pair of consecutive dimensions in parallel coordinate space. Specifically, for each pair of axes we draw a quad for each Gaussian by truncating its support to three standard deviations. In the fragment shader, we evaluate the density in parallel coordinate space and additively blend the result with all other Gaussians. A density-based parallel coordinate plot is shown in Figure 9.7 (d).

9.3.2.3 *Mapping Density*

The density-based visualizations described above and the spatial visualization in Section 9.2 all produce a single density ρ per Gaussian and per pixel. For large datasets, this density will have a high dynamic range and needs to be mapped to an opacity between zero and one through a non-linear mapping [153]. We choose a mapping that interprets the density, scaled by a user-controllable parameter $\lambda > 0$, as optical depth. The resulting opacity is

$$1 - \exp(-\lambda\rho). \tag{9.4}$$

With this mapping, multiplying the density of a Gaussian by an integer factor k produces the same result as rendering it k times with alpha blending, which is an intuitive behavior. At the same time, it retains detail even for large densities.



(a) Brushing (mean) (b) Brushing (mean) (c) Brushing (distribution)

Figure 9.2: Brushing of a value range $[a, b]$ applied to several distributions. Brushing only based on the mean value μ would lead to confusing results (a), especially if the distribution is represented by multiple Gaussian components (b). We compute the degree of interest of the brushing operation as the ratio between the integrand (gray) and the total area under the curve (c).

9.3.3 *Brushing Distributions*

To brush a value range of a dimension with our data representation and reflect this in all linked views, we use the clustering information. Although we could brush based on the cluster mean value, this is confusing and not very intuitive, especially when considering a dimension represented by multiple Gaussian components, see Figure 9.2 (a) and (b).

Feng et al. [74] discuss user interaction based on Gaussian distributions in the context of uncertainty visualization. We generalize their work and the concept of smooth brushing [59] to Gaussian mixture models. In detail, we compute the amount a cluster is in focus, the degree of interest, as the ratio between the integrand of the brushed regions of the GMM and the total area, see Figure 9.2 (c). For clusters that contain multiple Gaussian components, we compute the degree of interest as the weighted sum of all components.

9.3.4 *Time-Dependent Visualization*

Brushing in different time steps is a powerful tool for the interactive exploration of time-dependent data [132], but is not practical for large datasets since all time steps have to be processed. Our compact data summaries enable us to interact with multiple time steps at once. We support this interaction in a time histogram [176] where we depict a time-series of a selected dimension as a series of 1D histograms, see Figure 9.9.

If the clustering is fixed over time, we can trivially extend the brushing operation to time-dependent data. This is not possible when clusters change over time, e.g. merge together into larger, or split into smaller clusters. In this case, the relationship of clusters in different time steps has to be explicitly modeled and stored.

For brushing, we need to reassign degrees of interest from frame to frame. To this end, we transfer the degrees of interest to the individual samples uniformly and then reassign them to clusters. Say we have $n_t \in \mathbb{N}$ clusters $\mathbb{I}_{t,0}, \ldots, \mathbb{I}_{t,n_t-1} \subseteq \{0, \ldots, n-1\}$ in frame t and analogously for frame $t+1$. The clusters in frame t have associated degrees of interest $d_{t,0}, \ldots, d_{t,n_t-1} \in [0, 1]$.

Then we define the degree of interest of cluster $k \in \{0, \dots, n_{t+1} - 1\}$ in frame $t + 1$ as

$$d_{t+1,k} := \sum_{l=0}^{n_t-1} \frac{|\mathbb{I}_{t,l} \cap \mathbb{I}_{t+1,k}|}{|\mathbb{I}_{t+1,k}|} d_{t,l} \in [0, 1].$$

The quotient in this sum is the fraction of samples in cluster $\mathbb{I}_{t+1,k}$ that was part of cluster $\mathbb{I}_{t,l}$ in the previous frame. Interest is inherited from the cluster in the previous frame in proportion to that quotient. Note that this method defines a simple linear transform. There is no need to consider all samples at run time. Instead, the transfer coefficients for the degrees of interest can be precomputed and stored in a sparse matrix.

9.3.5 *Uncertainty Visualization*

We introduce an error estimate to convey the uncertainty of the data summaries. By computing and storing an error for each cluster, we are able to visualize the uncertainty interactively during the visual analysis and to support brushing and linking. Prior work measures the error directly between the density of the Gaussian mixture model and the original data. However, this is not robust and suffers from aliasing due to the necessary use of histograms. Instead, we define the error between a Gaussian mixture model and the samples of a cluster $\mathbb{I}$ for a dimension $j \in \{0, \dots, m_u - 1\}$ similar to common statistical goodness of fit tests and our error measure in Section 8.4. In detail, we compute the empirical cumulative distribution functions (CDF) of the data samples

$$F_{\mathbb{I}}(p) := \frac{1}{|\mathbb{I}|} \sum_{i \in \mathbb{I}} \begin{cases} 1 & \text{if } u_{i,j} \leq p, \\ 0 & \text{otherwise} \end{cases} \tag{9.5}$$

and compare it to the CDF F_g of the Gaussian mixture model using the Wasserstein distance [237]:

$$\mathrm{W}(F_{\mathbb{I}}, F_g) := \int_{-\infty}^{\infty} \left| F_{\mathbb{I}}(p) - F_g(p) \right| \mathrm{d}p. \tag{9.6}$$

To visualize the Wasserstein distance, we visualize it together with the CDF, cf. Figure 9.6 (b). A high Wasserstein distance consequently indicates a high uncertainty of the data model.

9.3.6 *Level of Detail and Outliers*

By design, our representation is a simplified model of the data. During the exploration and analysis process, a scientist might want to investigate a subset of the data more closely. For this purpose, we substitute brushed clusters by their original data values. To integrate the data distributions into our frequency-based views, we perform kernel density estimation using Gaussian kernels. We can thus avoid differentiating between the modeled and original data distributions.

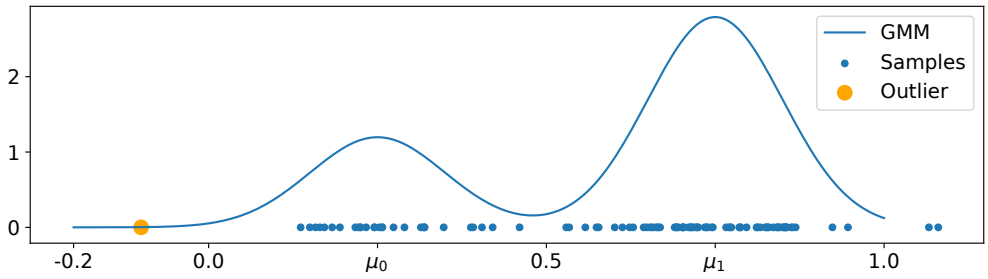


Figure 9.3: To rank samples by their outlyingness, we evaluate the Mahalanobis distance to the closest Gaussian. This measures how many standard deviations a sample is away from the mean of the closest Gaussian.

Moreover, outliers, i.e. isolated samples in regions of low density, tend to get lost in density-based visualizations [210, 229]. To explicitly add outliers to our visualizations, we sort all samples in a cluster in a preprocess according to a measure of outlyingness. Although any measure between a sample and a GMM could be used, we employ the Mahalanobis distance [204] to the closest Gaussian component. This effectively measures how many standard deviations a sample is away from the mean of the closest Gaussian, see Figure 9.3. To visualize outliers, we then take a fixed percentage p_o of outliers from a cluster $\mathbb{I}$ by loading the first $p_o|\mathbb{I}|$ samples. Figure 9.4 (c) shows a spatial visualization with 2% of outliers.

9.4 EVALUATION

In this section, we apply our approach to a synthetic and three real-world datasets.

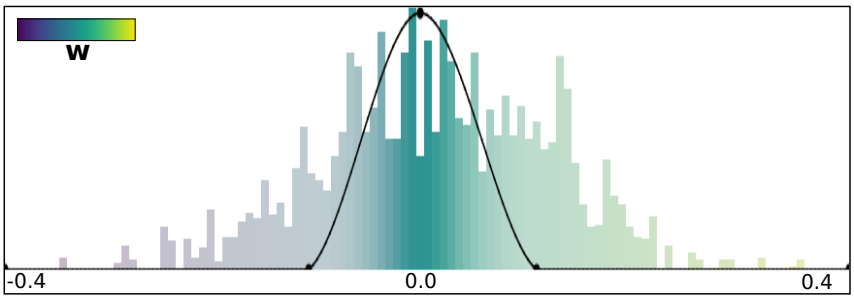
9.4.1 *Synthetic Data*

We first apply our approach to a small synthetic dataset consisting of 10 clusters from a total of 100,000 points. The dataset contains 9 dimensions. The three spatial dimensions in each cluster are normally distributed, but 10 % of the points are distributed uniformly to add noise to the distributions. Figure 9.4 (a) shows this dataset. The 3D Gaussians are shown in Figure 9.4 (b) and in (c) where we explicitly add 2% of outliers from all clusters.

We compare the uncertainty transfer function to a 1D transfer function based on mean values in Figure 9.5. In (a), we set the opacity of the transfer function, where the peak coincides with the mean value. The synthetic dataset in (b) shows the resulting rendering. For our data summaries in (c), the opacity is similarly reduced. This is due to the uncertainty transfer function since it computes the expected opacity with respect to the value distribution. In comparison, the opacity of the Gaussians in (d) using a mean transfer function does not change since the opacity of the mean value is still set to opaque in (a). Thus, changing any of the opacity (or color) values of the transfer function has no influence except if the mean value is changed. An alternative would be

(a) Synthetic data (b) GMMs (c) GMMs & outliers

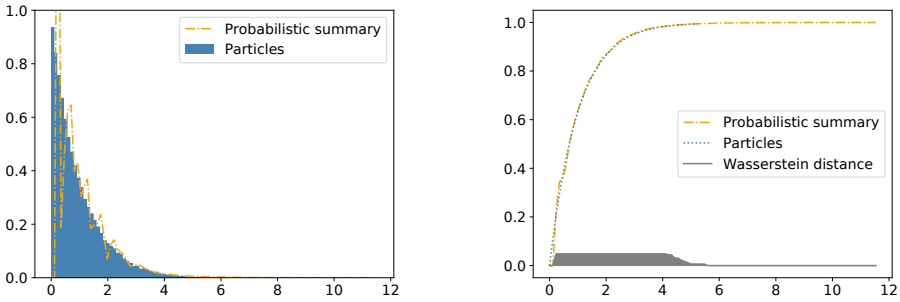
Figure 9.4: Rendering of Gaussians from the synthetic dataset using kernel density estimation (a), with our data model (b), and with 2% of outliers (c).



(a) Transfer function

(b) Synthetic data (c) Expected opacity (d) Mean opacity

Figure 9.5: We set the opacity of the transfer function (a) to visualize the synthetic dataset (b). Our uncertainty transfer function (c) computes the expected opacity (i.e. the integral of the opacity curve), while a 1D transfer function based on the mean value (d) sets all Gaussians to opaque.



(a) Probability density function (b) Cumulative distribution function

Figure 9.6: The exponentially distributed dimension (a) is hard to model using Gaussian components. The cumulative distribution function in (b) conveys the error to the user.

Table 9.1: Overview of the cosmological data from the Illustris simulation.

Dataset	# Dim.	# Particles	# Clusters	Data size	Summaries	Wasserstein dist.
I-3 Gas	15	16,039,182	110,000	2 GB	200 MB	1.77×10^{-6}
I-2 DM	7	319,324,195	841,639	11.3 GB	617 MB	3.10×10^{-7}
I-1 DM	6	2,635,739,426	5,352,571	72.2 GB	1.5 GB	4.56×10^{-8}

the use of a 2D transfer function [171] that offers increased control over the classification, but complicates user interaction.

In Figure 9.6 (a) we illustrate an exponentially distributed dimension of the dataset, which is difficult to model using only Gaussian components. The cumulative distribution function shown in (b), illustrates this error as measured by the Wasserstein distance. By quantifying the error, we can decide if this error is acceptable, or brush and use the level-of-detail mechanism to directly load a subset of the data with a high error.

9.4.2 *Cosmological Data*

The Illustris simulation [225] is a large-scale cosmological hydrodynamical simulation of galaxy formation that aims to predict both dark and baryonic components of matter. In detail, the dynamics of dark matter and gas are simulated with the quasi-Lagrangian code AREPO [295], which employs an unstructured Voronoi tessellation of the domain. After simulation, only the center points of the Voronoi cells are kept and are referred to as particles. Since the simulation has been run in different resolutions and we want to show both dark and baryonic matter, we study multiple separate datasets as shown in Table 9.1. We compare the data based on the 100^{th} time step without descendant or ancestor information.

The Illustris datasets have been clustered into halos using a domain specific approach. The sizes of clusters are extremely irregular and range in between a single particle and up to millions of particles per cluster. Since we cannot fit a GMM to very small clusters, we fit a single Gaussian for clusters of size below 20.

With our approach, we are able to interactively visualize and explore these massive datasets that might not even fit into memory otherwise. Figure 9.7 shows several interactive, linked views of the Illustris-1 dataset. We have brushed the x-, y-, and z-axis in the parallel coordinate plot (d). The brushed regions (green) are then highlighted in red in all other views. The density-based views are free of clutter and clearly show trends and correlations between the dimensions. For example, the parallel coordinate plot in (d) indicates that the brushed values have velocity components that are distributed around zero and are linearly correlated. The spatial visualization depicts 5.3 million clusters that we render and navigate interactively.

Figure 9.8 compares our probabilistic summaries with the original particle data of Illustris-3 Gas. Note that the interactive visualization of Illustris-1 and 2 is not possible on our system due to their data sizes. Although we clearly

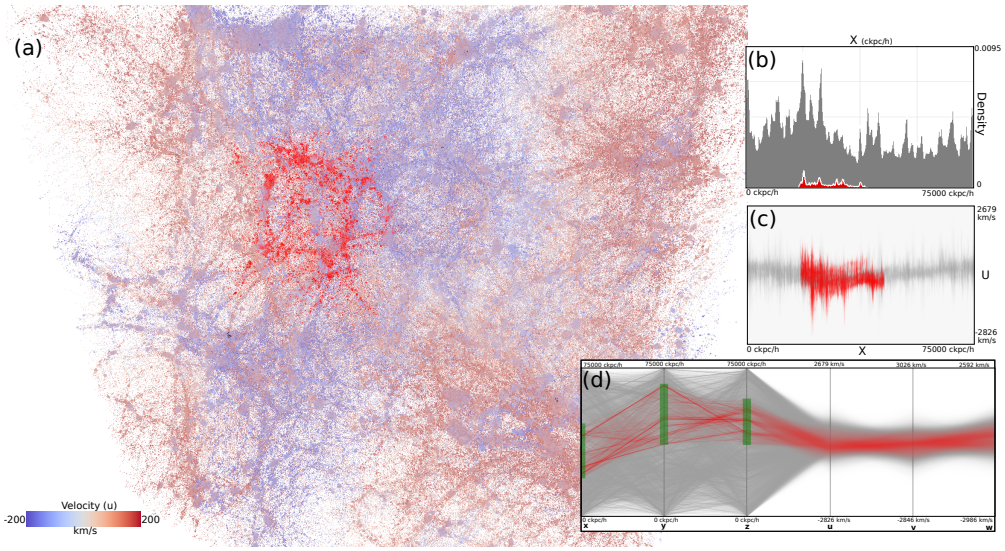


Figure 9.7: Our data representation allows us to interactively visualize the position of particles in the Illustris-1 dataset by splatting 3D Gaussians (a) and to create density-based 1D and 2D plots, depicted in (b) and (c). A density-based parallel coordinate plot is shown in (d). All of those views support interactive navigation and exploration by brushing (red) and linking.

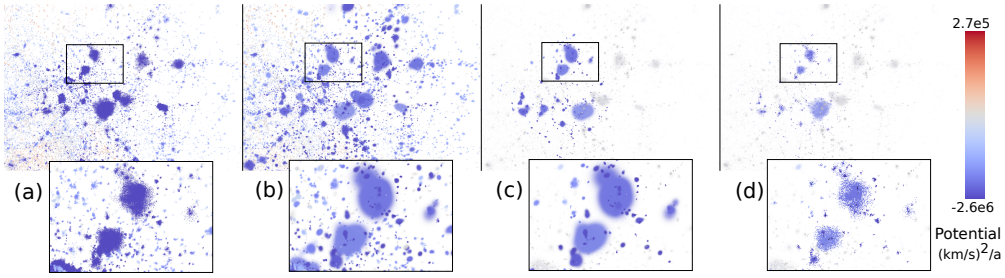
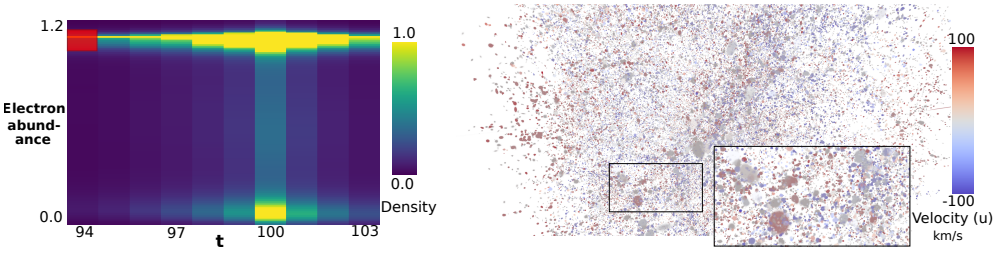


Figure 9.8: The Illustris-3 Gas dataset rendered by splatting particles (a) and 3D Gaussians (b). In (c) we have brushed a region and clusters that are not in focus are shown in a desaturated gray. We load the original particle data of the brushed region and render them together with the context (d).



(a) Time histogram (b) Brushed clusters in the 100th time step

Figure 9.9: A time-histogram of electron abundance in the Illustris-3 Gas dataset is shown in (a). We have brushed (red) in the 94th time step, which affects all linked views in the current, 100th time step. The spatial visualization that highlights the brushed values in the 100th time step is shown in (b). Note that Gaussians are shown as saturated or desaturated, depending on how much they are in focus.

Table 9.2: Overview of the spray nozzle dataset. We show the absolute summary size, relative to the original data size, the average number of GMM components, and the average Wasserstein distance.

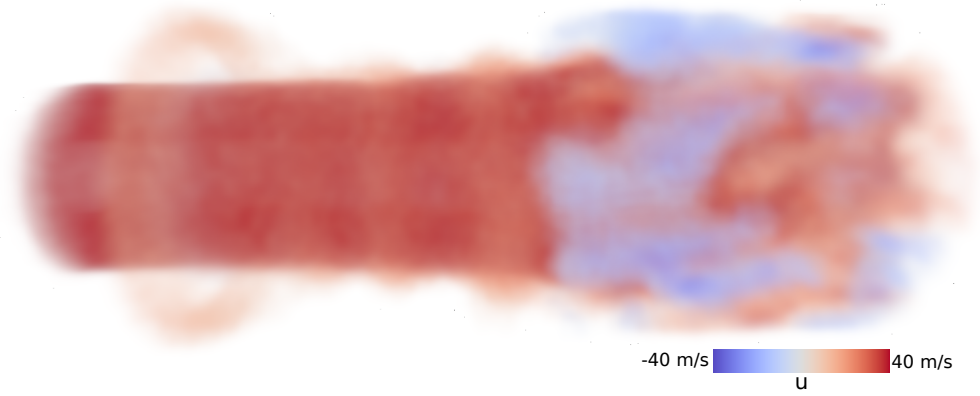
# Clusters	Summaries	Rel. size	GMM comp.	Wasserstein dist.
2,000	6.7 MB	0.006%	1.7 ± 1.19	5.54×10^{-5}
8,000	19.9 MB	0.017%	1.4 ± 0.85	1.57×10^{-5}
32,000	47.5 MB	0.036%	1.2 ± 0.61	4.64×10^{-6}

miss some details in the spatial visualization, we still manage to convey the general structure of the data and the distribution of color-mapped values. Whilst sorting and rendering all 16 million particles as isotropic Gaussians takes 61 ms on our system, the clusters require only 2 ms. Note that for this dataset, the 110,000 clusters are represented by a total of 357,512 Gaussians in 3D. In Figure 9.8 (c), we have brushed a spatial region on the right side which is consequently put into focus. In (d) we have loaded the original particle data of the brushed clusters. Note that all of the linked views are also updated by this operation. Since we only load an additional 240,000 particles, the interactive visualization still takes only 3.8 ms to render. Although other forms of level of detail are possible, this is a powerful way to drill-down from an overview to a detailed view of the data.

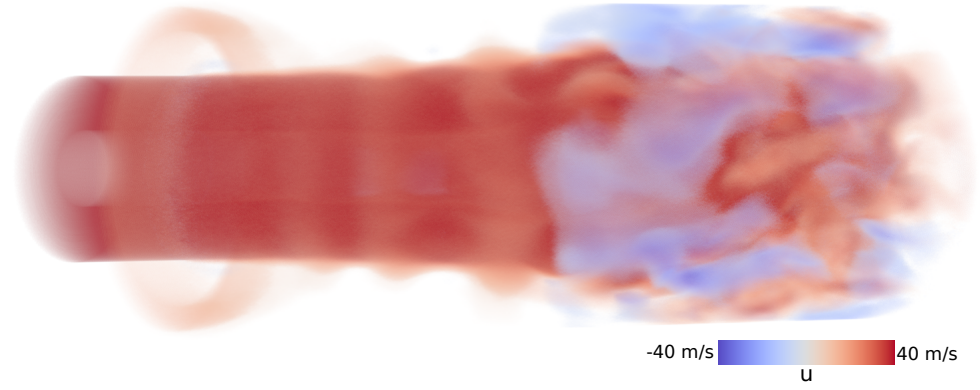
Since the clusters split and merge over time in this dataset, we have precomputed the transfer coefficients for time-dependent visualizations. Figure 9.9 depicts a time-histogram of a selected attribute over several time steps. We have brushed the 94^{th} time step, which is reflected in all views in the current, 100^{th} time step. The spatial visualization in (b) highlights those brushed values. Due to our brush, mostly smaller clusters are in focus. This brushing and linking over time thus allows exploring the selected dimension and its time-dependent behavior. With our data summaries, we can compute the whole time-histogram in just under a second. In comparison, computing a time-histogram from the particle data takes 34 s and is severely I/O limited since the complexity scales with respect to the number of particles instead of the amount of clusters.

9.4.3 *Spray Nozzle*

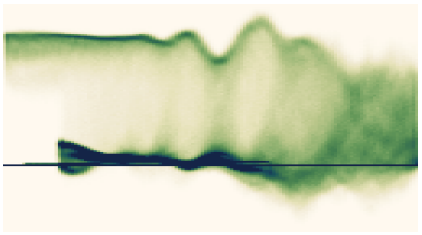
We have applied our technique to a smoothed particle hydrodynamics (SPH) dataset of a fuel spray nozzle simulation [46]. In the context of renewable energy production, biomass is converted into fuel by a gasification process. The quality of the spray is analyzed since it is critical for the efficiency of the gasification. However, the size of the time-dependent data prevents the usage of common interactive visual analysis techniques. In detail, the dataset contains about 43 million particles per time step. Each particle contains a position, velocity, pressure, density, and fluid type for a total of 9 separate



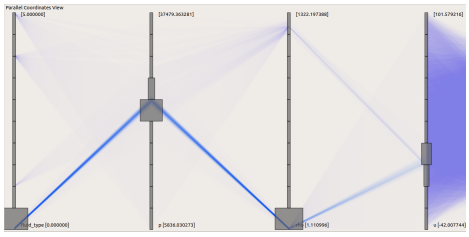
(a) Splatting 3D Gaussians (k-means 32.000)



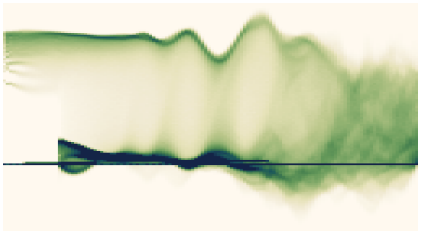
(b) Splatting all particles



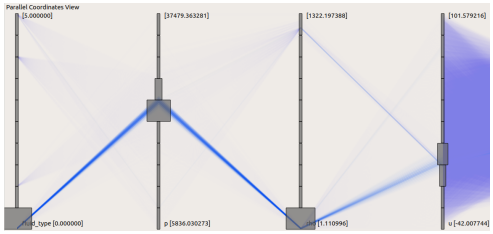
(c) Histogram from samples



(d) PCP from samples



(e) Histogram from all particles



(f) PCP from all particles

Figure 9.10: Visualization of a spray nozzle using our approach with the k-means 32.000 clustering by splatting 3D Gaussians (a) and by drawing samples from the GMMs to create a 2D histogram (c) and a parallel coordinate plot (d). In (b), (e), and (f) the corresponding visualizations using the original SPH particle data are shown.

dimensions. The fluid type describes four different categories, including fluid, gas, and two types of boundaries.

We have partitioned the data using a k-means clustering based on the spatial position, fluid type, and velocity magnitude. Table 9.2 shows the data size reduction and average number of GMM components for different numbers of clusters. For this dataset, we fix the maximum number of GMM components to 6. The size of the data summaries increases with the number of clusters. At the same time, the average number of GMM components decreases. This shows that the number of GMM components adapts to the less complex clusters. Moreover, the average Wasserstein distance is reduced for a larger number of clusters.

Figure 9.10 depicts several visualizations created from our representation and from the original SPH data. The spatial visualizations in (a) and (b) depict velocity in u-direction. Our approach does lose some details, especially on the finer structures on the right side of the cylindrical domain. Since item-based visualizations of 43 million particles suffer from strong overdraw and visual clutter, density-based visualizations are preferable for this dataset. These are fast and efficient to create using our representation that is already an estimate of density. The 2D histogram in (b) and the parallel coordinate plot in (c) have been created from samples drawn from the GMMs. Compared to the reference plots in (e) and (f), we achieve nearly identical results. Moreover, it is possible to vary the number of samples, which could be used to create less cluttered visualizations, e. g. for scatter and parallel coordinate plots.

We represent the fluid type, i. e. the categorical dimension, by interpreting it as a scalar dimension. This is possible since the data only consists of four fluid types that we model using an appropriate number of Gaussian components. We could have increased the maximum number of components for all marginal distributions containing a categorical dimension, but this was not necessary for this dataset. Although a small number of categories is common in multiphase fluid simulations, in general, representing categorical dimensions with GMMs does not scale.

9.4.4 *Hurricane Isabel*

The Hurricane Isabel dataset is an atmospheric simulation from the IEEE Visualization Contest 2004, produced by the Weather Research and Forecast (WRF) model. Besides an implicit spatial position and a velocity vector, the time-dependent dataset contains 9 additional scalar quantities on a uniform grid of size $500 \times 500 \times 100$. Since we formulated our approach for scattered data and did not consider the important but special case of gridded data, we disregard the topology of the dataset.

To apply our approach, we either define clusters through uniform blocks or apply k-means clustering based on the spatial position and velocity magnitude. Both clustering procedures require a fixed number of clusters as input. Independent from the clustering, we always store 3D distributions for the spatial position and the velocity vector and compute the respective 2D marginal distribution from these. Apart from that, we model and store all pairwise

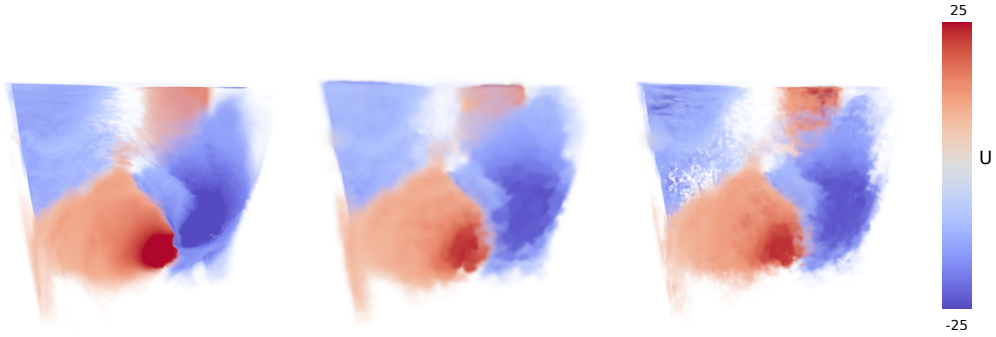
Table 9.3: Overview of the different clustering procedures of the Hurricane Isabel dataset. We show the resulting absolute and relative data size and the average Wasserstein distance.

Model	Clustering	# Clusters	Summaries	Rel. size	Wasserstein dist.
Our	Blocks	1,000	12.1 MB	1.4%	1.20×10^{-4}
HD	Blocks	1,000	5.2 MB	0.6%	1.21×10^{-4}
Our	Blocks	8,000	82.6 MB	9.0%	1.58×10^{-5}
HD	Blocks	8,000	34.7 MB	4.8%	1.56×10^{-5}
Our	Blocks	16,000	146.8 MB	14.8%	8.49×10^{-6}
HD	Blocks	16,000	50.0 MB	5.1%	8.24×10^{-6}
Our	*k*-means	1,000	12.1 MB	1.4%	1.23×10^{-4}
HD	*k*-means	1,000	5.4 MB	0.6%	1.26×10^{-4}
Our	*k*-means	8,000	80.5 MB	8.8%	1.66×10^{-5}
HD	*k*-means	8,000	33.8 MB	3.7%	1.57×10^{-5}
Our	*k*-means	16,000	143.9 MB	14.7%	8.75×10^{-6}
HD	*k*-means	16,000	48.5 MB	5.0%	8.01×10^{-6}

2D distributions and all 15 one-dimensional distributions. Additionally, we compare our representation to modeling each cluster with a high-dimensional Gaussian mixture model.

Table 9.3 shows the data summaries we have created with both clustering procedures, with different cluster sizes, with our approach and using high-dimensional Gaussian mixture models. We have chosen a maximum number of 6 GMM components for the low-dimensional and 32 for the high-dimensional models to achieve a comparable quality. Note that creating the high-dimensional model took nearly 43 hours, cf. Table 9.5. Both approaches can model the data well even though some dimensions are quite challenging. The high-dimensional model performs surprisingly well for this dataset, considering the dimensionality, which is due to high correlations in the dimensions. The low-dimensional representation requires more storage since it cannot make use of these higher-dimensional correlations. In both cases, the two clustering procedures lead to similar results.

Figure 9.11 shows a visualization of wind speed from west to east, i.e. *u*-velocity, by splatting the original data and with our approach. Although our representation loses some details, it conveys the major features of the dataset. Whilst the low-dimensional representation models the spatial position separately, the high-dimensional GMM takes correlations between all dimensions into account. The marginal distribution of the spatial positions is thus also influenced by the other dimensions, which leads to the artifacts in Figure 9.11 (c). This reduces trust in the high-dimensional model since it is unclear if these correlations actually exist in the data or not. Lastly, the high-dimensional



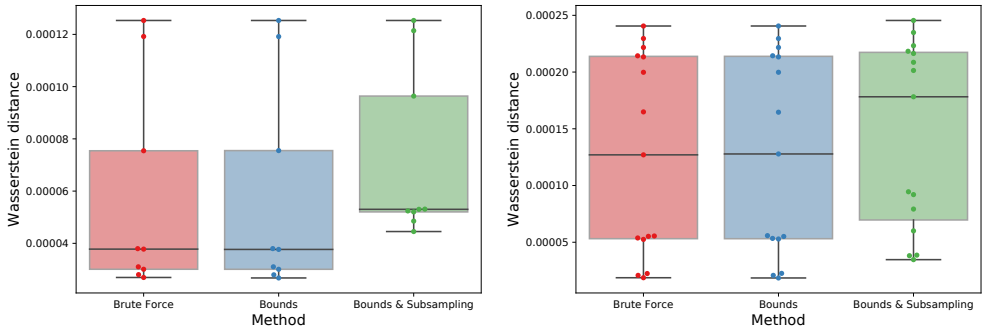
(a) Original data (b) Our low-dim. GMMs (c) High-dim. GMMs

Figure 9.11: Visualization of wind speed from west to east (U) in the Hurricane Isabel data by splatting the original data (a), with the k-means 16.000 clustering of the low-dimensional model (b), and the high-dimensional model (c).

model contains over five times the amount of Gaussian components, which increases the complexity of all visualizations. In comparison, our representation consists of low-dimensional models that are easier to understand and more robust.

9.4.5 *Fast Selection of GMM Components*

We propose a fast selection of GMM components by formulating lower und upper bounds and using subsampling. In Figure 9.12, we illustrate the impact of this approximation on the Spray Nozzle (a) and the Hurricane Isabel (b) dataset compared to a brute-force selection of GMM components. Using only the bounds does not introduce an error. However, using both bounds and the subsampling does lead to a slightly increased error. Not that for subsampling we always take a fixed amount of 200 samples from each cluster. By increasing this fixed number of samples, we lower this error, at the cost of additional computational effort.



(a) Spray Nozzle k=2000 (b) Hurricane Isabel k=1000

Figure 9.12: Comparison of our fast selection of GMM components with the brute-force approach on the Spray Nozzle (a) and the Hurricane Isabel (b) dataset.

Table 9.4: Performance of visualizations with our data summaries.

Dataset	Splatting		PCP	Density	Brushing
	Sorting	OIT		(x, u)	
Illustris-3 Gas	3.9ms	4.3ms	439ms	1.0ms	4ms
Illustris-2 DM	31ms	14ms	421ms	3.6ms	28ms
Illustris-1 DM	196ms	28ms	1241ms	11.2ms	160ms
Hurricane Isabel k=8000	4.6ms	20ms	99ms	1.1ms	2ms
Spray Nozzle k=8000	4.4ms	23ms	47ms	1.4ms	2ms

Table 9.5: Measurements of the data summary preprocessing.

Dataset	Our GMMs	Low-dim. GMMs	High-dim. GMMs
Hurricane Isabel k=1000	2h 54min	9h 31min	42h 55min
Spray Nozzle k=2000	1h 43min	8h 13min	14h 51min

9.4.6 *Performance*

Our evaluations were performed on an Intel i7-6700 with 32 GB of system memory and an NVIDIA GTX 1080 Ti graphics card providing 11 GB of video memory. For GPU acceleration, we make use of both CUDA for general purpose computations and OpenGL for rendering. For our spatial visualization, we have used a screen resolution of 1920×1080. The resolution of our 2D density plots was 200×200 and 800×300 for the parallel coordinate plot (PCP).

Timings for several visualizations are shown in Table 9.4. In general, our prototype allows interactive navigation and creation of all visualizations introduced above. The Illustris-1 and 2 datasets are the most demanding, due to the large number of clusters. Note that the performance of our approach scales with the number of clusters and Gaussian components, not the original data size. The order-independent transparency (OIT) approach performs very well on the cosmological datasets compared to the back-to-front splatting using sorting. Note that the speed varies depending on the number of covered pixels. The sorting approach is faster on the smaller and spatially more compact datasets.

We create our probabilistic data summaries in a preprocessing step using the Python scikit-learn library. This process is trivial to parallelize since all time steps, clusters, and distributions can be processed independently. Due to inherent restrictions imposed by our Python prototype, an implementation in a native language is expected to be significantly faster. The measurements for our prototype are shown in Table 9.5. Our fast GMM component estimation (Section 9.1.2) leads to a significant speedup. Lastly, computing high-dimensional GMMs requires significantly more preprocessing time, making it unsuited for

use in practice. Note that our approximations for a fast estimation of GMM components cannot be used for the high-dimensional data.

9.5 DISCUSSION

We introduce probabilistic data summaries for multivariate scattered data. They enable the interactive visual analysis of large datasets on a single workstation that would not be possible otherwise due to limitations of memory or compute. Although this data representation is a simplified model of the data, we inform the user about this uncertainty and present a level-of-detail and outlier visualization for more detailed investigations. In comparison to the sampling approach introduced in Chapter 8, this representation is significantly more compact and the density-based visualizations are efficient to construct and well suited to visualize massive datasets.

The core insight of our approach is that we only have to model combinations of low-dimensional distributions for visual analysis, which avoids the complexity of modeling high-dimensional distributions. This avoids the curse of dimensionality, but limits higher-dimensional correlations that are still captured using our sampling-based approach. This further limits the computation of derived variables, such as the FTLE, which must be performed on the original dataset.

Although the data must be clustered, we do not make any restrictive assumptions about the clustering procedure. In fact, our evaluation shows that the impact of the clustering on the quality of the representation is less pronounced than expected and is largely offset by the adaptive modeling of GMMs. Note that our sampling technique in Chapter 8 could also be used to construct a clustering procedure.

10

IMAGE-BASED VOLUME VISUALIZATION USING MOMENTS

Continuing progress in high performance computing enables scientists to perform complex simulations in a high spatial and temporal resolution. Exploring and analyzing the resulting datasets is a daunting task. The traditional approach to transfer and analyze the data on an individual workstation is no longer feasible for large datasets due to storage, bandwidth, and compute constraints. In situ visualization [49], where the simulation is tightly coupled with the visualization pipeline, addresses this limitation, but limits exploration and interaction. Image-based approaches (Section 2.2.4) combine in situ visualization with the analysis on low-cost machines. There, the data is represented by images produced from fixed views and predefined parameters. Image-based approaches are also employed to render large or unstructured volumetric data [285] that could not be interactively visualized otherwise. However, the transfer function based exploration of volumetric data is difficult to integrate into image-based approaches as it requires a compact and accurate representation of the signal along each view ray.

In this chapter, we present a novel image-based data representation to visualize large structured and unstructured volumetric data. This representation allows us to store and reconstruct the scalar density along a viewing ray, thus enabling a change of the transfer function similar to recent work [75, 78, 318, 319]. We avoid storing discrete samples or modeling distributions along a ray, instead we compactly represent the density in the Fourier basis. We store Fourier coefficients of a signal bounded between zero and one, also referred to as bounded trigonometric moments. This leads to a sparse and quantizable representation that can be linearly interpolated in space and time. However, reconstructing a signal using a truncated Fourier series causes well-known ringing artifacts. We employ the recently introduced bounded MESE [244] for an efficient and accurate reconstruction of a bounded signal from its moments.

To create a compact representation, we adaptively determine the number of moments per pixel. Specifically, we first ray march the dataset only once and generate a fixed number of moments. We then select an appropriate amount of moments per pixel by utilizing an error measure between all and a prefix of moments. In addition, we introduce a novel coding strategy to compactly store the moments in each pixel. This coding scheme uses information from the lower-order moments in a pixel to predict the next moment. Therefore, we only need to store differences to the actual moments, which are more amenable to quantization and compression. Accounting for the trade-off between quality and data size, we present a parameterizable quantization curve that is pareto optimal. After compression, we store between 20 and 60 bytes per pixel, when using up to 100 moments.

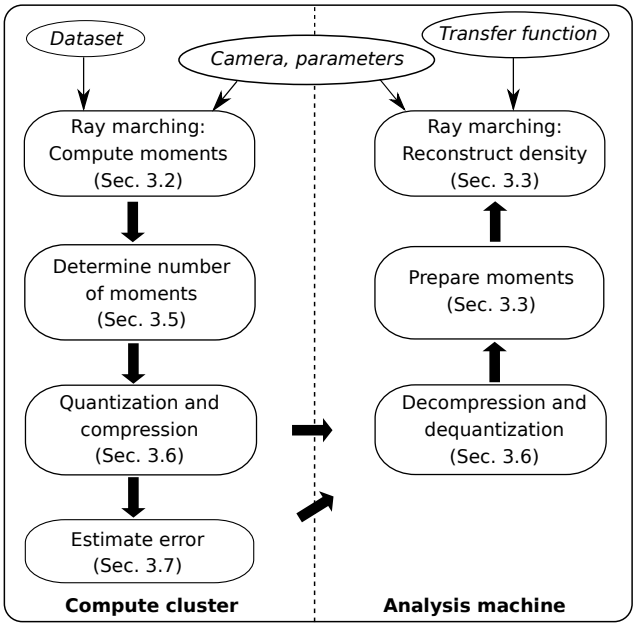


Figure 10.1: A moment image is generated on the compute cluster (left) and is used for interactive volume rendering on the analysis machine (right).

Figure 10.1 gives an overview of our proposed method. Our proxy representation, which we refer to as moment image, is generated on a system with access to the data. The moment image then enables the interactive exploration with a user-controllable transfer function on a low-cost analysis machine.

We begin by reviewing the reconstruction of bounded densities using moments (Section 10.1), before we discuss the computation of moments during ray marching (Section 10.2) to create a moment image. Then, we examine ray marching using a moment image (Section 10.3). Afterwards, we analyze relations between valid sequences of moments (Section 10.4). We employ these insights to select an appropriate number of moments for each pixel (Section 10.5) and for our coding and quantization scheme (Section 10.6). We discuss uncertainty quantification (Section 10.7), single scattering (Section 10.8), and changing the view configuration (Section 10.9). Lastly, we evaluate our approach qualitatively and quantitatively in Section 10.10 on two volumetric and an SPH dataset.

10.1 USING MOMENTS TO RECONSTRUCT BOUNDED DENSITIES

To represent the densities along a viewing ray, we map the length of the ray in the volume to $[-\pi, 0]$ linearly and we assume that the densities are bounded in $[0, 1]$. Since this signal is generally not periodic, we mirror the signal in $[-\pi, 0]$ to a periodic signal in $[-\pi, \pi]$. We represent this signal in the Fourier basis using $m + 1 \in \mathbb{N}$ Fourier coefficients. The Fourier basis, written as a vector, is

$$\mathbf{c}(\varphi) := \frac{1}{2\pi} \left(\exp(-ij\varphi)\right)_{j=0}^{m} \in \mathbb{C}^{m+1}$$

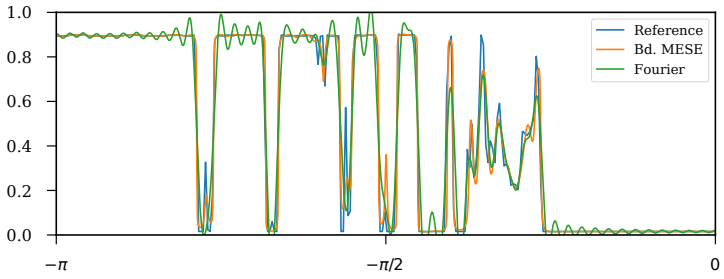


Figure 10.2: Truncated Fourier and bounded MESE reconstruction from the same coefficients.

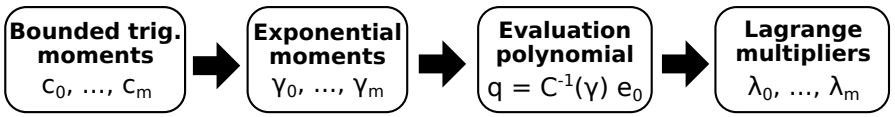


Figure 10.3: Given bounded trigonometric moments, we compute the corresponding exponential moments, solve a linear system, and finally compute Lagrange multipliers, from which we can efficiently reconstruct a bounded signal.

and the Fourier coefficients are

$$c := \int_{-\pi}^{\pi} s(\varphi)\mathbf{c}(\varphi)\,\mathrm{d}\varphi \in \mathbb{R}^{m+1}.$$

Note that the coefficients are real since the mirroring makes the signal even. These Fourier coefficients compactly approximate a signal. However, the reconstruction of a truncated Fourier series does not guarantee bounded values and exhibits well-known ringing artifacts as shown in Figure 10.2.

The Fourier coefficients of a bounded signal are also known as bounded trigonometric moments, or simply moments in the scope of this paper. Peters et al. [244] introduce the bounded maximum entropy spectral estimate (MESE), which reconstructs a signal from bounded trigonometric moments. As illustrated in Figure 10.3, the bounded trigonometric moments are first transformed to exponential moments $\gamma \in \mathbb{C}^{m+1}$. For this we define

$$\check{\gamma}_0 := \frac{1}{4\pi}\exp\left(\pi i\left(c_0 - \frac{1}{2}\right)\right) \in \mathbb{C}.$$

Let $\Re z$ denote the real part of a complex number $z \in \mathbb{C}$. Then, the exponential moments are defined as

$$\begin{aligned}\gamma_0 &:= 2\Re\check{\gamma}_0 \in \mathbb{R},\\ \gamma_l &:= \frac{2\pi i}{l}\left(l\check{\gamma}_0 c_l + \sum_{j=1}^{l-1}(l-j)\gamma_j c_{l-j}\right) \in \mathbb{C},\end{aligned} \tag{10.1}$$

where $l \in \{1,\dots,m\}$. The exponential moments γ are the trigonometric moments of an unbounded signal, which is dual to our bounded signal s. This transformation is invertible.

The next steps utilize the Hermitian Toeplitz matrix

$$C(\gamma) := \frac{1}{2\pi} \begin{pmatrix} \gamma_0 & \overline{\gamma_1} & \cdots & \overline{\gamma_m} \\ \gamma_1 & \gamma_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \overline{\gamma_1} \\ \gamma_m & \cdots & \gamma_1 & \gamma_0 \end{pmatrix} \in \mathbb{C}^{(m+1)\times(m+1)}.$$

We can reconstruct a valid unbounded density from the exponential moments γ if and only if $C(\gamma)$ is positive definite. In this case, the evaluation polynomial $q := C^{-1}(\gamma)e_0$, with $e_0 := (1, 0, \ldots, 0)^\mathsf{T}$ holds coefficients for an unbounded density known as MESE [242].

The bounded MESE is the bounded dual of the MESE [244]. To evaluate it efficiently, we transform the evaluation polynomial into Lagrange multipliers $\lambda \in \mathbb{R}^{m+1}$. We denote $\check{\gamma}_j = \gamma_j$ for all $j \in \{1, \ldots, m\}$. Then for all $l \in \{0, \ldots, m\}$

$$\lambda_l := \frac{1}{\pi i q_0} \sum_{k=0}^{m-l} \check{\gamma}_k \sum_{j=0}^{m-k-l} \overline{q_{j+k+l}} q_j \in \mathbb{R}. \tag{10.2}$$

Now, the bounded MESE is given by

$$\hat{s}(\varphi) = \frac{1}{\pi} \arctan\left(\Re\lambda_0 + 2\Re \sum_{l=1}^{m} \lambda_l \exp(-il\varphi) \right) + \frac{1}{2}. \tag{10.3}$$

Since arctan maps to $(-\frac{\pi}{2}, \frac{\pi}{2})$, the reconstructed density $\hat{s}$ is always in $(0, 1)$. Moreover, the Lagrange multipliers are constructed to ensure that

$$\int_{-\pi}^{\pi} \hat{s}(\varphi)\mathbf{c}(\varphi)\,\mathrm{d}\varphi = c, \tag{10.4}$$

i.e. the bounded trigonometric moments are accounted for exactly. A truncated Fourier series also satisfies Equation 10.4 but does not exploit knowledge about the bounds. As shown in Figure 10.2 the bounded MESE captures complicated signals well while being less prone to ringing than a truncated Fourier series.

10.2 MOMENTS OF RAY DENSITIES

We now discuss the creation of our proxy representation, the moment image. For every pixel, this image contains the moments of a scalar density sampled during ray marching, see Figure 10.4. To this end, we first compute the intersections t_0 and t_1 of a viewing ray $r(t)$ with the volume. We map the parameterized ray $r(t)$ with $t \in [t_0, t_1]$ to phase space $[-\pi, 0]$ linearly and sample the volume at $\varphi_0, \ldots, \varphi_{n-1}$, giving us the densities $s_0, \ldots, s_{n-1} \in [0, 1]$.

To compute $m + 1$ bounded trigonometric moments during ray marching, we assume linear interpolation and perform a quadrature [244, Appendix C]. At each ray marching step, we compute the gradient a_l and y-intercept b_l:

$$a_l := \frac{s_{l+1} - s_l}{\varphi_{l+1} - \varphi_l}, \qquad b_l := s_l - a_l\varphi_l.$$

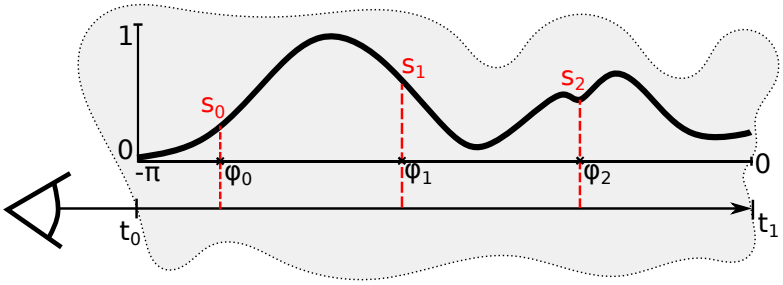


Figure 10.4: During ray marching, we sample a scalar density along the ray parameterized by $[-\pi, 0]$. We compute $m+1$ moments from this signal.

Then the interpolated signal for all $\varphi \in [\varphi_l, \varphi_{l+1}]$ is

$$s(\varphi) := a_l \varphi + b_l.$$

The bounded trigonometric moments are iteratively computed as:

$$c_0 := \frac{1}{\pi} \sum_{l=0}^{n-2} \left[\frac{a_l}{2} \varphi^2 + b_l \varphi \right]_{\varphi_l}^{\varphi_{l+1}} \in \mathbb{R},$$
$$c_j := \frac{1}{\pi} \Re \sum_{l=0}^{n-2} \left[\left(a_l \frac{1 + ij\varphi}{j^2} + b_l \frac{i}{j} \right) \exp(-ij\varphi) \right]_{\varphi_l}^{\varphi_{l+1}} \in \mathbb{R}.$$

Since we mirror the signal, the moments are real and we do not have to compute or store the imaginary parts.

Lastly, we compute a lower bound $s_{\min} \leq s(\varphi) - \epsilon$ and an upper bound $s(\varphi) + \epsilon \leq s_{\max}$ during ray marching. Here, ϵ is a small number to relax the bounds. For example, we set $\epsilon = 0.005$ for our datasets. This relaxation improves the reconstruction since it stays away from 0 and 1, see Equation 10.3. After ray marching, we transform the moments to these bounds:

$$c_0^t := \frac{c_0 - s_{\min}}{s_{\max} - s_{\min}},$$
$$c_j^t := \frac{c_j}{s_{\max} - s_{\min}}. \tag{10.5}$$

Although we have to store the bounds for each pixel to invert this transformation after reconstruction, we found that it improves the reconstruction and enables a more aggressive quantization of the moments (Section 10.6).

10.3 INTERACTIVE RENDERING

To render using a moment image, we first note that it is possible to linearly interpolate the moments of neighboring pixels or even between different time steps. For example, we can increase the resolution of the moment image using bilinear interpolation to render in a higher resolution. Although other types of interpolation are possible, negative weights in filter kernels may invalidate moments and should be avoided.

To ray march a moment image, we compute the Lagrange multipliers for each pixel upon loading. Subsequently, we perform ray marching using a ray $r(t)$ in the interval $[t_0, t_1]$, which we map to $[-\pi, 0]$. During ray marching, we use the Lagrange multipliers to efficiently reconstruct a density at each $\varphi \in [-\pi, 0]$ by evaluating Equation 10.3. Afterwards, we invert the transformation from Equation 10.5 using the bounds $[s_{\min}, s_{\max}]$, apply a transfer function, and perform compositing. We employ a preintegrated transfer function [72]. Ray marching a moment image is fast, taking only a few milliseconds (cf. Table 10.1), which enables the interactive exploration with different transfer functions.

10.4 RELATIONS BETWEEN MOMENTS

It is viable to store moment images using one 32-bit float per moment but storage requirements are considerable. Thus, we strive to reduce the number of moments adaptively and to quantize the remaining moments to only a few bits. Our methods benefit from the underlying theory of the bounded MESE [244]. In this section, we cover the relevant mathematical results.

Recall from Section 10.1 that we construct the evaluation polynomial $q := C^{-1}(\gamma)e_0$ from the exponential moments $\gamma \in \mathbb{C}^{m+1}$. The Toeplitz matrix C has a special structure. Levinson's algorithm exploits this structure to solve for q in time $\mathcal{O}(m^2)$ instead of $\mathcal{O}(m^3)$ [243]. At the same time, it produces intermediate results that aid our quantization scheme. For all $l \in \{1, \dots, m\}$ Levinsons's algorithm computes:

$$q_0^{(0)} := \frac{1}{\gamma_0}, \tag{10.6}$$

$$u^{(l)} := \sum_{k=0}^{l-1} q_k^{(l-1)} \gamma_{l-k}, \tag{10.7}$$

$$q^{(l)} := \frac{\left(q_0^{(l-1)}, \dots, q_{l-1}^{(l-1)}, 0\right) - u^{(l)} \left(0, \overline{q_{l-1}^{(l-1)}}, \dots, \overline{q_0^{(l-1)}}\right)}{1 - |u^{(l)}|^2}. \tag{10.8}$$

Then

$$q = C^{-1}(\gamma)e_0 = 2\pi \left(q_0^{(m)}, \dots, q_m^{(m)}\right).$$

Since the Toeplitz matrix is positive definite, we know $|u^{(l)}| < 1$ [243]. Combined with Equation 10.7, this inequality forces γ_l to reside in a disk of radius

$$r_l := \frac{1}{q_0^{(l-1)}},$$

with center

$$\mathring{\gamma}_l := -\frac{1}{q_0^{(l-1)}} \sum_{k=1}^{l-1} q_k^{(l-1)} \gamma_{l-k} \in \mathbb{C}.$$

Inverting Equation 10.1 shows that the bounded trigonometric moment

$$c_l = \frac{\gamma_l}{2\pi i \check{\gamma}_0} - \frac{1}{l\check{\gamma}_0} \sum_{j=1}^{l-1} (l-j)\gamma_j c_{l-j} \tag{10.9}$$

lies in a disk as well. We base our coding strategy in Section 10.6 on this observation.

Incidentally, the center of this disk also has a compelling relation to the bounded MESE $\hat{s}$. In the case $l = m+1$, we find

$$\mathring{c}_l := \frac{\mathring{\gamma}_l}{2\pi i \check{\gamma}_0} - \frac{1}{l\check{\gamma}_0} \sum_{j=1}^{l-1} (l-j)\gamma_j c_{l-j} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{s}(\varphi) \exp(-il\varphi) \, \mathrm{d}\varphi.$$

In other words, the bounded MESE places every unknown moment in the center of the disk where it must reside [244, Lemma B.2]. This behavior is in stark contrast to a truncated Fourier series, which just sets unknown Fourier coefficients to zero.

Applying this insight repeatedly lets us compute the full Fourier expansion of the bounded MESE [244, Proposition B.3]. In terms of the exponential moments, we obtain the linear recurrence

$$\gamma_{m+1+k} = -\frac{1}{q_0} \sum_{j=0}^{m} \gamma_{j+k} q_{m+1-j}, \tag{10.10}$$

for all $k \in \mathbb{N}$. Mapping these exponential moments to bounded moments through Equation 10.9, we obtain all unknown moments of the bounded MESE $\hat{s}$.

10.5 DETERMINING THE NUMBER OF MOMENTS

To adaptively reduce the number of moments per pixel, we measure the error between a prefix of $c_0, \ldots, c_n$ moments and the full vector of moments $c_0, \ldots, c_m$, where $n < m$. Based on the assumption that the full set of $m+1$ moments accurately captures the signal, we compute the error without accessing the original data or ray marching the dataset again.

To this end, we use the recurrence from Equation 10.10 and Equation 10.9 to compute the missing $m-n$ moments from the prefix of $n+1$ moments. In this manner, we obtain the exact moments $\hat{c}_{n+1}, \ldots, \hat{c}_m$ of the bounded MESE $\hat{s}$, assuming that the moments $c_{n+1}, \ldots, c_m$ have been discarded. We measure the error introduced by discarding these moments using the relative RMSE

$$\mathrm{rRMSE}((c_{n+1}, \ldots, c_m), (\hat{c}_{n+1}, \ldots, \hat{c}_m)) = \frac{1}{c_0} \sqrt{\sum_{i=n+1}^{m} (c_i - \hat{c}_i)^2}.$$

The rRMSE normalizes the root mean squared error (RMSE) with respect to the average value of the signal, i.e. the zeroth moment c_0.

To determine the number of moments for each pixel, we find a value of n so that the error is just below a user-defined threshold. We keep only the first $n+1$ moments per pixel to create a compact moment image. Computing the

moments of the bounded MESE $\hat{c}_{n+1}, \ldots, \hat{c}_m$ takes time $\mathcal{O}(m(m-n))$ and we have to redo this work for each value of n that we try. To keep the overall cost low, we use bisection. It finds a suitable n in $\mathcal{O}(\log m)$ trials. Since the error is not guaranteed to decrease monotonically with n, the found n is not known to be minimal but it certainly satisfies the requested error threshold.

10.6 COMPRESSION AND QUANTIZATION

Moment images might be produced and archived in large quantities and are transferred over network to the analysis machine. Therefore, small file sizes are paramount. A baseline approach directly quantizes bounded trigonometric moments $c_0 \in [0,1]$ and $c_j \in [-\frac{1}{\pi}, \frac{1}{\pi}]$. Since the representation of moments is essential to reduce storage requirements, we propose a novel coding scheme, a pareto optimal quantization curve, and discuss lossless compression to pack more information into significantly fewer bits.

10.6.1 *Coding*

In Section 10.4 we observed that each exponential moment γ_l lies in a disk characterized by the previous moments $\gamma_0, \ldots, \gamma_{l-1}$. We exploit this constraint in our coding strategy by only storing the difference to the center of the disk. These differences exhibit lower entropy compared to the moments and are better suited for quantization and compression.

First, we explicitly store c_0, then we transform the bounded trigonometric moments c to exponential moments γ. We execute Levinson's algorithm and at each step store the difference between γ_l and the center of the disk $\mathring{\gamma}_l$ given by the previous exponential moments, scaled relative to the radius r_l:

$$u^{(l)} = \frac{\gamma_l - \mathring{\gamma}_l}{r_l} \in \mathbb{C}.$$

To revert this encoding, we simply execute Levinson's algorithm with γ_0 and the stored values of $u^{(l)}$ to solve the system for q. Furthermore, we reconstruct the exponential moments as

$$\gamma_l = \mathring{\gamma}_l + r_l u^{(l)}.$$

10.6.2 *Transformation*

We perform two important transformations that enable a more aggressive quantization. First, for real moments, the corresponding $u^{(l)} \in \mathbb{C}$ with $l > 0$ vary only along the axis aligned with $i\check{\gamma}_0$. We transform $u^{(l)}$ to the real axis:

$$u_T^{(l)} := u^{(l)} \frac{|i\check{\gamma}_0|}{i\check{\gamma}_0} \in (-1,1).$$

The values for $u_T^{(l)} \in (-1,1)$ are distributed mostly around zero, see Figure 10.5. Therefore, we compute $\min(u_T^{(l)})$ and $\max(u_T^{(l)})$ for each $l \in [1,m]$

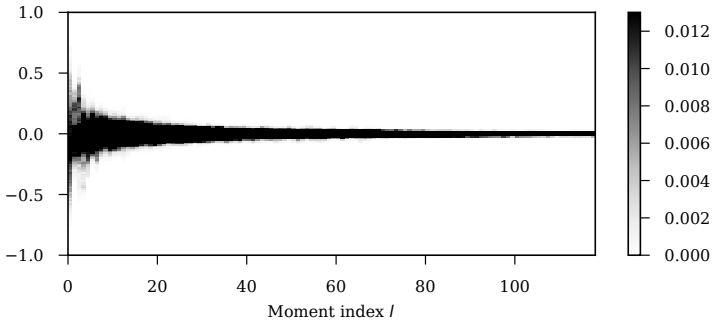


Figure 10.5: Distribution of values $u_T^{(l)}$ that we store for each moment at index l.

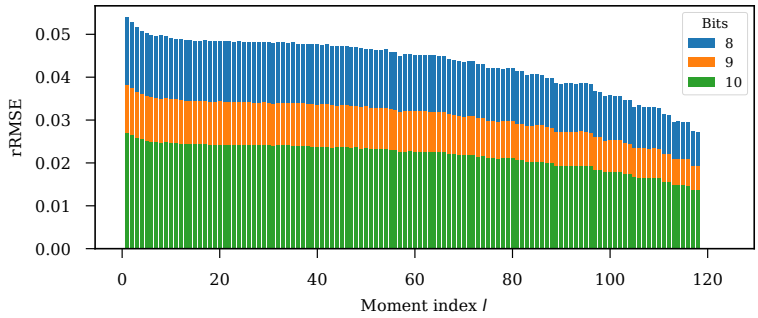


Figure 10.6: We quantize only a single moment index l at a time and measure the resulting error. This shows that the error from quantization depends on the index l.

once per moment image and transform the $u_T^{(l)}$ from this range to (0,1) before quantization. Especially for large l, this transformation leads to a significantly more accurate representation.

10.6.3 *Quantization*

First, we quantize the zeroth moment c_0 to 16-bit in $[0, 1]$. Although we might be able to use fewer bits in some cases, the quality can deteriorate quickly. In general, quantization errors are amplified from coefficients with a lower to a higher index. This is shown in Figure 10.6, where we apply quantization to only a single moment at a time. Then, we measure the rRMSE between the bounded trigonometric moments of the quantized and the original image. Consequently, this shows how much each moment influences the error of the reconstruction. We use this observation to determine the number of bits for quantizing a moment at index $l \in [1, m]$. Specifically, we want to quantize such that at each index we apply approximately the same error. The quantization curve, i.e. the number of bits used to quantize each index is thus determined from the errors shown in Figure 10.6.

Depending on the trade-off between acceptable error and image size, we want to use a different quantization curve. By selecting the number of bits b_1 to quantize the moment at index 1, the resulting error determines the whole

Algorithm 1 Computing a quantization curve

```
procedure QuantizationCurve(Moment image M, b_1)
    Quantization table t
    t[0] ← 16
    t[1] ← b_1
    M' ← quantize(M, 1, b_1)
    e_T ← rRMSE(moments(M), moments(M')) ▷ Determine error threshold
    for l ← 2, m do
        t[l] ← t[l − 1] ▷ Reduce number of bits until we reach the threshold
        for all b ∈ {t[l − 1], ..., 1} do
            M' ← quantize(M, l, b)              ▷ Quantize index l with b bits
            e ← rRMSE(moments(M), moments(M'))
            if e ≥ e_T then
                break
            end if
            t[l] ← b
        end for
    end for
    return t
end procedure
```

curve. We thus employ b_1 as a parameter to create different quantization curves. The algorithm to determine a quantization curve is illustrated in Algorithm 1. At each index l, the algorithm tries to decrease the number of bits as much as possible, buts stays below the error threshold determined from b_1.

We illustrate several quantization curves in Figure 10.7. To determine whether these curves are optimal, we sample randomly perturbed quantization curves and plot the total number of bits against the error in Figure 10.8. This shows that our proposed quantization curves are pareto optimal, i.e. no change of the curve leads to a reduction in both error and size.

Although the quantization curves are always qualitatively similar, they still differ between different datasets and view configurations. We thus determine an optimal quantization curve for each moment image that we generate using Algorithm 1. For moment images with a large resolution, this computation can

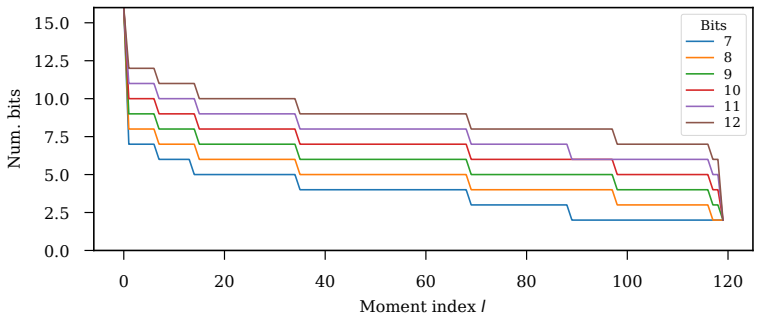


Figure 10.7: Quantization curves for different parameter values.

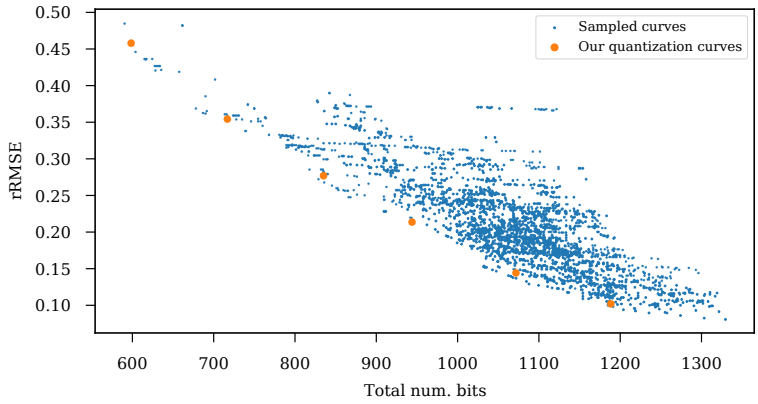


Figure 10.8: Comparison of our pareto optimal quantization curves (see Figure 10.7) and randomly perturbed curves.

be quite expensive. Therefore, we recommend to use a downsampled image of size 64×64, which gives nearly identical results and is fast to compute.

10.6.4 *Compression*

To further reduce the size of a moment image, we apply lossless compression to the quantized coefficients. Figure 10.5 shows that the frequency of values for each moment is non-uniform. Thus, entropy encoding can reduce the data size by storing variable-length codewords for each coefficient at index l containing 2^{b_l} symbols. The length of the codewords is selected proportional to the frequency of occurrence. We employ arithmetic coding [333], which can estimate the frequencies during encoding. Although the compression rate depends on the dataset and quantization, we achieve a $20 - 40\%$ reduction in size. Lastly, we perform fast dictionary coding on the resulting byte stream using the LZ4 [1] library, further reducing the image size (cf. Table 10.2).

10.7 UNCERTAINTY QUANTIFICATION

Since our approach introduces information loss, we want to convey the resulting uncertainty. For example, we visualize the uncertainty using a heat map, temporal animation [194, 202], or by integrating the uncertainty in the transfer function classification [270].

Due to space constraints, we are limited to few statistics about the distribution of errors along a viewing ray in each pixel. Computing these statistics is the only step, and an optional one, of our method that requires ray marching the dataset a second time. In detail, at each step during ray marching, the signal is reconstructed from our representation and is compared to the original dataset.

Although the root-mean squared error (RMSE) can be used to measure the average error, it gives little guarantees about the distribution of errors. In contrast, the maximum error is a conservative bound, but is not robust to

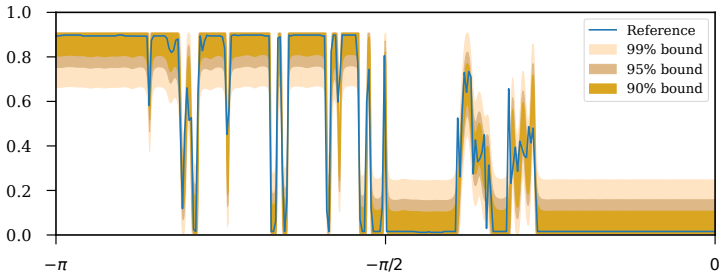


Figure 10.9: We compute error percentiles for the bounded MESE from Figure 10.2 and visualize these as error bounds along the reconstructed signal.

small outliers. We found that this makes the error bounds too conservative and thus not informative. Instead, we propose to compute a single or select percentiles of the error. These order statistics can be used to create robust and expressive error bounds, see Figure 10.9. However, order statistics are expensive to compute and would require $\mathcal{O}(n)$ space during ray marching, which is problematic for parallel computation on a cluster or GPU. Therefore, we employ the heuristic P^2-algorithm [147]. This algorithm performs online estimation of a percentile with fixed storage requirements.

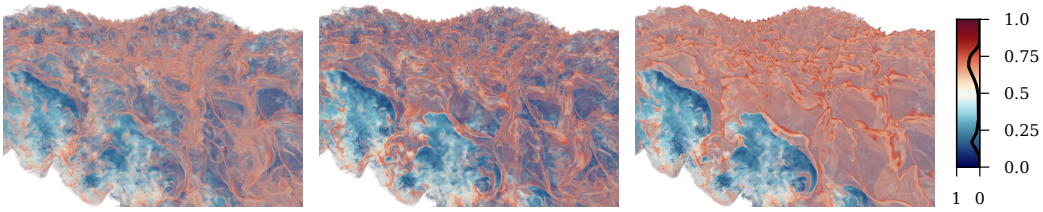
10.8 SINGLE SCATTERING

Volumetric shadows can improve the perception of spatial depth in direct volume visualizations [189]. Although different illumination models exist, the physically-based single scattering model [209] is often employed. At each step during ray marching, this requires evaluating the transmittance to the light sources.

To incorporate single scattering, we create a moment image from the perspective of the light source similar to shadow mapping [332]. For directional light sources, we use an orthographic view projection. Moreover, we recommend to use a smaller number of moments and a lower resolution since single scattering illumination is generally of lower frequency. During rendering, we could ray march along secondary rays from the single scattering moment image to each sample point, but this is computationally expensive. Instead, we ray march the single scattering image, compute the transmittance, and cache it on a regular grid. Then, we sample the cached transmittance during ray marching.

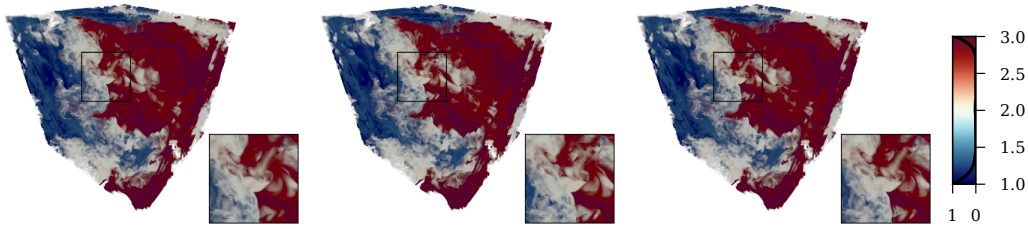
10.9 VIEW PROJECTION

The method presented thus far is limited to a static camera. However, a moment image encodes the entire volume within the view frustum. We exploit this insight to enable the exploration of different views. In detail, we ray march starting from the changed camera and project each point that is inside the view frustum to the moment image. Then, we perform bilinear interpolation of the Lagrange multipliers and reconstruct a density. The interpolation makes the



(a) Reference (b) Ours (Bounded MESE) (c) Ours (Fourier)

Figure 10.10: Entropy field of a Richtmyer-Meshkov instability rendered with direct volume rendering (a), with our approach using the bounded MESE reconstruction (b), and using the Fourier reconstruction (c).



(a) Reference (b) Ours (Bounded MESE) (c) Ours (Fourier)

Figure 10.11: The Rayleigh-Taylor instability shows the density of two mixing fluids. We render a ray marching reference (a), our approach with the bounded MESE reconstruction (b), and with Fourier reconstruction (c).

rendering quite expensive, but is necessary to avoid aliasing artifacts. For faster rendering at the expense of memory requirements, we resample a moment image to a regular grid. To avoid aliasing, we either require a sufficiently high image resolution or we upsample the moment image.

10.10 EVALUATION

We evaluate our approach on three real-world datasets. The entropy field of a Richtmyer-Meshkov instability (Figure 10.10) is given in a resolution of $2048 \times 2048 \times 1920$ with 8 bits per cell. The Rayleigh-Taylor instability dataset (Figure 10.11) consists of a density field showing two mixing fluids. It is stored in single-precision in a resolution of 1024^3. Lastly, the Turbine dataset stems from a smoothed particle hydrodynamics (SPH) simulation [60] of 100 million particles per time step, where each particle consists of a position and a scalar value in single-precision. We sample the scattered data by performing SPH interpolation with a cubic spline kernel. We employ a uniform grid for accelerating the neighborhood search during ray marching.

We evaluate our approach on these datasets qualitatively (Section 10.10.1), quantitatively (Section 10.10.2), and discuss the performance (Section 10.10.3). We compare the bounded MESE with the truncated Fourier reconstruction, which are both applicable to our moment images. Furthermore, we compare our approach to the ray-histogram approach by Wang et al. [319].

If not noted otherwise, we use an image resolution of 1024×768 with a maximum of 100 moments per pixel. We employ our novel coding technique and quantize the moments as discussed in Section 10.6 with the quantization curve given by $b_1 = 10$. We further evaluate the impact of our coding and quantization scheme (Section 10.10.4), and discuss selecting the number of moments (Section 10.10.5).

10.10.1 *Qualitative Evaluation*

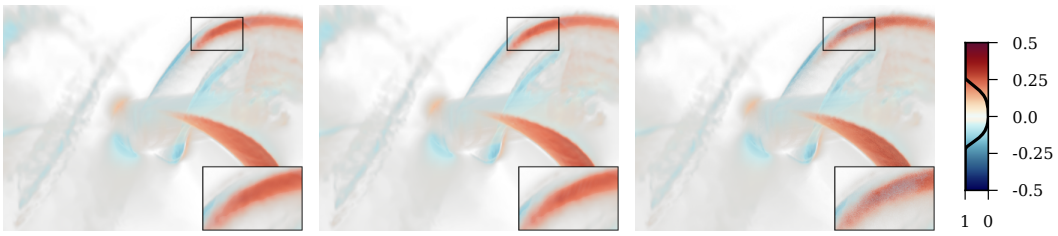
The Richtmyer-Meshkov instability is shown in Figure 10.10 using direct volume rendering (a), our approach using the bounded MESE (b), and with the Fourier reconstruction (c). This complicated dataset is difficult to represent, see e.g. the reconstruction of a single pixel in Figure 10.2. Although the Fourier reconstruction generally leads to good results, it introduces strong ringing artifacts for this dataset. Figure 10.11 shows the Rayleigh-Taylor instability rendered with direct volume rendering (a), with our approach (b), and with the Fourier reconstruction (c). Here, the Fourier method and the bounded MESE produce visually similar results.

The Turbine dataset is shown in Figure 10.12 using direct volume rendering (a), with our approach (b), and with ray-histograms (c). The transfer function, illustrated on the right, maps low and high velocities to non-zero opacities. This reveals the rotating turbine blades and indicates the presence of several vortices. Whilst our approach shows no obvious artifacts, the ray-histogram in (c) contains noise, for example on the upper right side. These artifacts might be due to the depth ordering of samples that is not considered in their approach. They might also stem from quantization of the floating point values due the use of histograms. Note that our approach does not quantize the sampled densities. Instead, we quantize the moments from which we reconstruct a smooth signal.

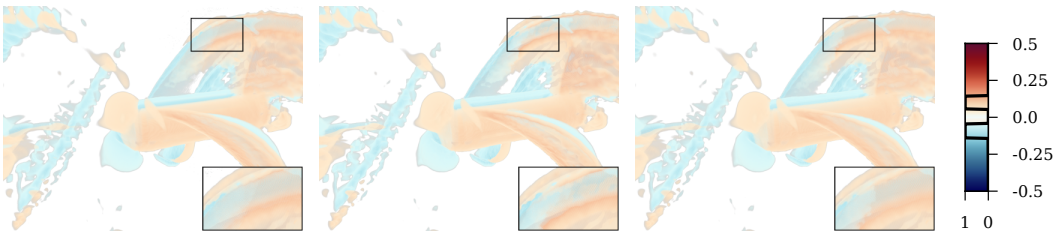
In Figure 10.12 (d), we change the transfer function to show two small intervals. Our bounded MESE (e) and Fourier (f) reconstructions accurately incorporate this transfer function and lead to results that are nearly indistinguishable from the reference. Note that the transfer function can be changed interactively, whilst the reference performs expensive SPH interpolation during rendering. The image-based approaches enable the interactive exploration of this large dataset.

We use a single moment image to render Figure 10.13 (a). Then, we rotate the view to the other side of the volume (b). We reproject our moment image (c), which still leads to accurate results. Some regions of the volume are outside the view frustum of the moment image. These regions are shown in green since we do not have any information in these areas.

In Figure 10.14 we show the Turbine dataset with direct volume rendering and single scattering using brute-force ray marching (a), using a moment image for single scattering (b), and using moment images for both single scattering and volume rendering (c). The single scattering moment image is computed in a resolution of 512^2 with a maximum of 50 moments. Using our

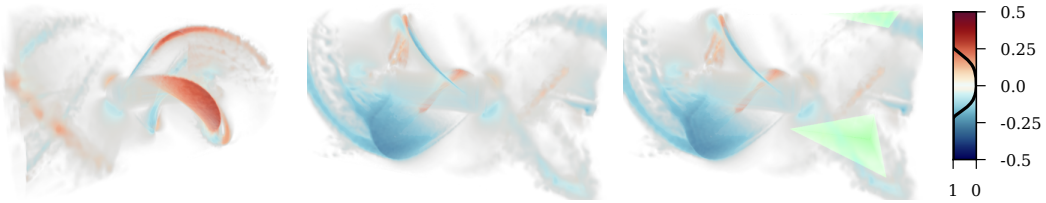


(a) Reference (b) Ours (Bounded MESE) (c) Ray-histograms



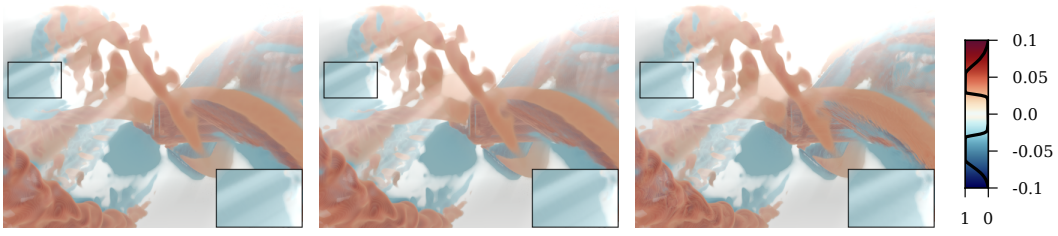
(d) Reference (e) Ours (Bounded MESE) (f) Ours (Fourier)

Figure 10.12: The Turbine dataset with direct volume rendering (a), using our approach (b), and using ray-histograms (c). We change the transfer function and render with direct volume rendering (d), using our approach (e), and using Fourier reconstruction (f).



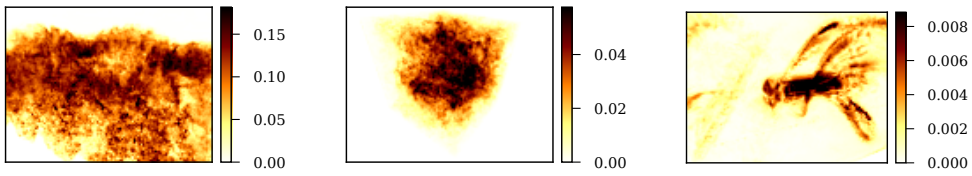
(a) Ours (original view) (b) Reference (changed view) (c) Ours (changed view)

Figure 10.13: We create a single moment image of the Turbine (a), change the view configuration (b), and use the moment image to render from this view (c). Regions of the volume that are outside the view frustum in (a) cannot be reconstructed and are shown in green in (c).



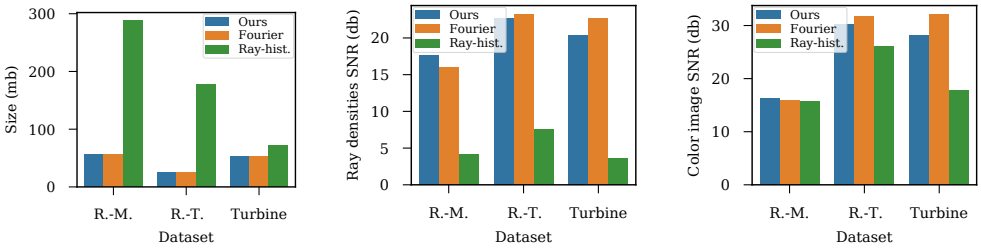
(a) Reference (b) Reference, bounded MESE single scattering (c) Ours (Bounded MESE)

Figure 10.14: The Turbine dataset with direct volume rendering and single scattering using brute-force ray marching (a), using a moment image to compute single scattering from the directional light source (b), and using moment images for both (c).



(a) Richtmyer-Meshkov (b) Rayleigh-Taylor (c) Turbine

Figure 10.15: Visualization of the 90th error percentile for our datasets.



(a) Proxy image sizes (b) Accuracy of ray densities (c) Accuracy of images

Figure 10.16: Comparison of proxy image sizes (a), the accuracy of reconstructed densities (b), and the quality of composited color images (c).

coding and quantization scheme it is only 5.3 MB in file size, but the single scattering is nearly indistinguishable from the reference.

We visualize the 90th error percentile in Figure 10.15 for all three datasets. Compared to the Rayleigh-Taylor and Turbine datasets, the Richtmyer-Meshkov dataset shows a higher error, due to its higher complexity.

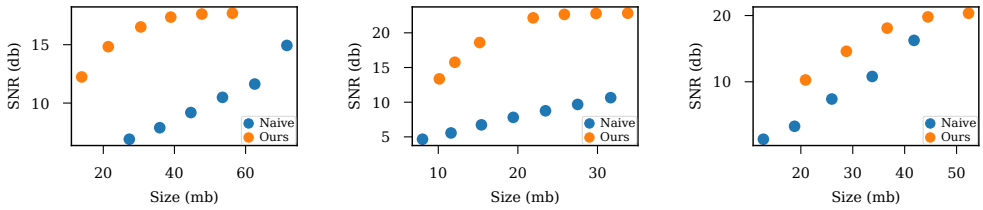
10.10.2 *Quantitative Evaluation*

Figure 10.16 compares the image storage size (a), the quality of reconstructed densities (b), and the quality of the composited images (c). We use the signal-to-noise ratio (SNR) in the logarithmic scale (db) to assess the quality. The original size of the Richtmyer-Meshkov dataset is 8193 MB, the Rayleigh-Taylor dataset is 4294 MB, and the Turbine 1609 MB for a single time step.

Compared to the original data, our moment images are between two and three orders of magnitude smaller in size. Our moment images are also significantly smaller compared to the ray-histogram approach. At the same time, the accuracy of our approach is better, as shown in Figure 10.16 (b). Depending on the transfer function, the quality of the composited images is more similar, see Figure 10.16 (c). Lastly, the Fourier reconstruction generally performs similarly to the bounded MESE, but leads to the best results for the Turbine dataset.

Table 10.1: Run-time measurements from our datasets.

Richtmyer-Meshkov		
Reference	Ray marching	28.8 ms
Generation	Ray marching	1025.6 ms
	Determining moment subset	1722.7 ms
	Coding	924.0 ms
	Quantization curve	930.3 ms
	Lossless encoding	281.1 ms
Reconstruction	Lossless decoding	510.7 ms
	Moment preparation	848.1 ms
	Ray marching	135.9 ms
Rayleigh-Taylor		
Reference	Ray marching	21.0 ms
Generation	Ray marching	183.5 ms
	Determining moment subset	836.0 ms
	Coding	424.0 ms
	Quantization curve	432.6 ms
	Lossless encoding	137.2 ms
Reconstruction	Lossless decoding	266.8 ms
	Moment preparation	391.6 ms
	Ray marching	16.2 ms
Turbine		
Reference	Ray marching	6203.1 ms
Generation	Ray marching	18678.0 ms
	Determining moment subset	1798.5 ms
	Coding	822.3 ms
	Quantization curve	849.5 ms
	Lossless encoding	319.0 ms
Reconstruction	Lossless decoding	557.9 ms
	Moment preparation	765.9 ms
	Ray marching	48.9 ms



(a) Richtmyer-Meshkov dataset (b) Rayleigh-Taylor dataset (c) Turbine dataset

Figure 10.17: Comparison of our coding and compression scheme with a naive approach that quantizes and compresses the coefficients.

10.10.3 *Performance Analysis*

For comparability, we measure the generation and reconstruction steps of our approach on the same system: An AMD Ryzen 5 3600 with 16GB RAM and a NVIDIA GeForce 1080Ti. We accelerate all steps using CUDA, except for the lossless compression. The performance measurements are shown in Table 10.1.

In comparison to a reference ray marching implementation, the generation of a moment image is several times slower. Since each step is trivial to parallelize, generating the moment images on a compute cluster would be significantly faster or could produce multiple images in parallel. Moreover, ray marching SPH data is extremely costly and not suited for interactive rendering. With our approach, the rendering step is decoupled from data access and the SPH interpolation and is thus interactive. Note that the performance of the Fourier reconstruction is comparable to our approach.

10.10.4 *Quantization and Compression*

Figure 10.17 compares our coding scheme to a naive approach that quantizes the coefficients with a fixed number of bits and then performs lossless compression. We vary the amount of quantization and compare the resulting image size and accuracy.

A moment image of the Richtmyer-Meshkov dataset without compression or quantization is 294.7 MB in size. The naive quantization reduces the image size to 27.3 or up to 71.9 MB, depending on the quantization. Our approach achieves between 13.9 and 56.3 MB at a consistently better quality. A moment image without compression or quantization is 133.5 MB for the Rayleigh-Taylor dataset and 266.6 MB for the Turbine. As shown in Figure 10.17 (b) and (c), our quantization scheme significantly reduces the data sizes for both datasets. In comparison to a naive quantization approach, our coding scheme achieves a reduction in size as well as an increase in accuracy.

Table 10.2 shows the effect of lossless compression methods on moment images quantized with our approach. Whilst the fast dictionary coding using `lz4` achieves mostly a small reduction in size, the entropy coding achieves a greater compression. However, the entropy coding is computationally more expensive to compute. We combine both methods to achieve the best compression rate.

Table 10.2: Image storage size after entropy coding, dictionary coding, and using both lossless compression methods.

Dataset	Entropy	Dictionary	Combined
Rayleigh-Taylor	78.3%	91.3%	69.6%
Richtmyer-Meshkov	58.5%	69.4%	50.6%
Turbine	82.7%	99.2%	81.9%

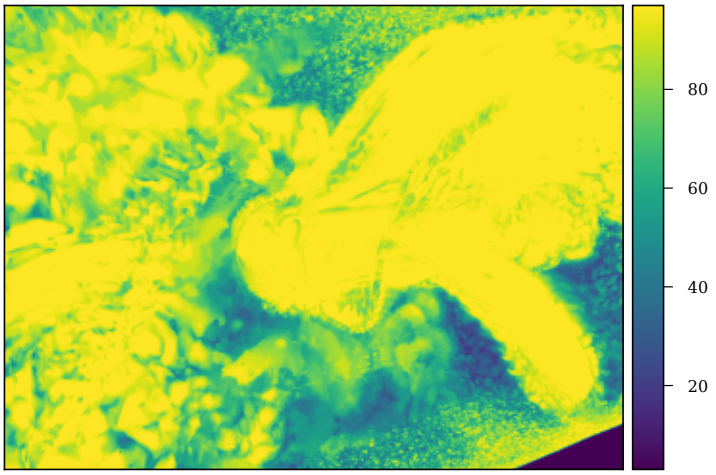


Figure 10.18: Number of moments per pixel in the Turbine dataset.

10.10.5 *Number of Moments*

Figure 10.18 illustrates the number of moments in each pixel in the moment image of the Turbine dataset. The amount of moments adapts to the complexity of the data. For example the turbine blade and swirling regions require more moments, whilst the surrounding volume requires fewer. Note that this adaptation is independent of the employed transfer function.

In Figure 10.19, we change the maximum number of moments for all datasets and measure the error. As the maximal number of moments is increased, the accuracy also improves. We recommend to use at least 50 moments to achieve sufficient quality. Otherwise, this parameter is a trade-off between the required computational effort and the accuracy of the representation.

10.11 DISCUSSION

Our image-based representation enables the interactive exploration of large and arbitrarily structured volumetric data by decoupling the access to the data from interactive rendering and exploration. Our evaluation shows that our proposed moment image representation is both compact and accurate. Selecting the number of moments per pixel, encoding, and quantizing the moments is key to achieve small image sizes. Our technique enables scientists

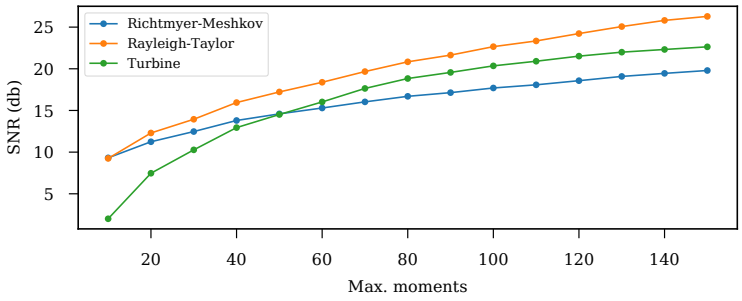


Figure 10.19: The error from using a different maximum number of moments.

to create a large amount of images from different views or simulation time steps.

Compared to ray-histograms [319], our representation is both smaller and more accurate since we do not store distributions, which cannot reconstruct the ordering of densities. Compared to the bounded MESE, Fourier reconstruction yields surprisingly good results, especially for smooth signals such as the SPH dataset. However, ringing can cause strong artifacts for the Fourier reconstruction. In comparison, the bounded MESE yields good and more predictable results. Therefore, we recommend the Fourier reconstruction for known smooth datasets, such as SPH data, and the bounded MESE otherwise. Note that most of the concepts discussed in this paper, including our coding and quantization strategy, are equally applicable to the Fourier reconstruction.

We have shown that moment images can be used for single scattering illumination. Since that requires less accuracy, the corresponding moment images take only few megabytes. Lastly, we were surprised how changing the view configuration still leads to accurate results. This shows that our representation is a volumetric representation, but compactly encoded in an image.

10.12 FUTURE WORK

In the future, we want to investigate whether subdividing a ray into less complex signals can increase the accuracy or decrease the total number of moments per pixel. Moreover, less complex signals could give more guarantees about the accuracy, which would help in constructing tighter error bounds.

Lastly, multiple moment images could be combined to reconstruct a complete volume with high accuracy. This would require choosing a minimal set of images, e. g. using an approach similar to Lukasczyk et al. [201]. Then, a volumetric representation could be reconstructed in a preprocess or on-the-fly during rendering.

11 CONCLUSION

In this thesis, we discuss the visual analysis of particle data to facilitate knowledge discovery in large and unstructured datasets. To this end, we present novel methods for Lagrangian flow visualization and data reduction.

Parts of our flow visualization methods have been developed and evaluated together with domain experts working in the field of thermal turbomachinery. This has already spawned an exciting publication [54]. We believe that other fields where particle data is generated can similarly benefit from our proposed analysis methods. For example, our flow visualization methods could offer new and interesting insights in cosmological simulations, such as the Illustris simulations [225]. This simulation code studies both dark matter and baryonic components in the evolution of the universe. Our method can visualize these different components, similar to multiphase fluid simulations [246]. Whilst the concept of Lagrangian coherent structures has already been applied to cosmological data resampled to a volume [81], our novel formulation (Chapter 5) should be able to produce faster and more accurate results.

Although our flow visualization and analysis technique approach is already employed to understand complex real-world flows (Chapter 6), we believe that the visualization of uncertainties in unsteady flows is pivotal in understanding the chaotic nature of turbulent flows. Our proposed visualization of uncertainties in the transport and mixing behavior (Chapter 7) is only a first step in this direction. Especially the sensitivity of the flow dynamics with respect to slight changes in the initial or boundary conditions is only addressed partially. Recent work explore the parameter space of ensemble simulations by learning a continuous parameter space [17, 122]. The application to transport and mixing behavior seems promising. Due to the inherent difficulties that arise in the visualization of uncertain and ensemble data, more research is needed.

By researching and developing these visualization methods specific to particle data, scientists and engineers are more likely to consider particle simulations for solving their problems. Furthermore, Lagrangian representations are becoming popular as a compact representation to store and post-process unsteady flows [3, 34]. The Lagrangian representation is advantageous since no numerically sensitive integration has to be performed, thus allowing lower temporal and spatial resolutions. However, common data reduction approaches are often limited to volumetric data. This emphasizes the need for data reduction techniques specific to the Lagrangian representation.

For large-scale simulations, performing the simulation on a supercomputer and the post-processing and analysis on individual workstations is no longer practical. The presented probabilistic data reduction methods (Chapter 8, Chapter 9) address this by creating a compact representation that is well-suited for interactive visualization of large particle data. These techniques

achieve a meaningful reduction in data size by omitting some information, which introduces uncertainty. By measuring and visualizing these uncertainties and by representing probabilistic data our work is thus in large parts also applicable to visualize uncertain data. The visualization of uncertain data is still a challenging topic in the field of scientific visualization [27, 30, 251, 316]. In the future, we would like to apply and extend our methods to uncertain and specifically ensemble data.

In comparison to these probabilistic data reduction methods, we also introduce an image-based approach (Chapter 10) to visualize large-scale particle data. This approach represents a dataset from a fixed view. Although our image-based proxy representation enables transfer function exploration for unstructured volumetric data, it is still limited in comparison to object-space data reduction. However, it is especially well-suited for in situ visualization, where the images are created on a compute cluster whilst the simulation executes and the data can be immediately explored on low-cost analysis machines. The image-based paradigm is also beneficial to compactly summarize ensemble data, where a small set of images might be able to sufficiently represent an ensemble member. Additionally, particle data that requires costly interpolation during volume rendering strongly benefits from image-based approaches.

Lastly, whilst we expect simulations and measurement devices to continue to produce growing data sizes, we also expect them to become more complex. In the future, we expect multifaceted data [165] to become even more prominent. Especially fluid simulations can benefit from both the Lagrangian and Eulerian point of view [275, 276, 295]. This heterogeneity presents new challenges as well as opportunities for visualization and analysis methods.

Part III

APPENDIX

A

APPENDIX

A.1 VOID-AND-CLUSTER SAMPLING ALGORITHM

Our void-and-cluster sampling algorithm for scattered data is shown in detail in Algorithm 2. We split the sampling algorithm into the initial random sampling, optimization, and the void filling steps.

A.2 INDEXING A LOWER TRIDIAGONAL MATRIX

We derive how to index the non-zero elements of a lower tridiagonal matrix $A \in \mathbb{R}^{n \times n}$ given a linear index $k \in \{0, \ldots, \frac{n(n-1)}{2}\}$. We define i as the i-th column and j as the j-th row of the lower tridiagonal matrix A, i.e. the non-zero elements. Note that going from i and j to the linear index k is easier: $k = \frac{(i-1)i}{2} + j$. We re-order and solve the equation for i:

$$\begin{aligned}
&\frac{(i-1)i}{2} + j = k \\
\Rightarrow\ &(i-1)i = 2(k-j) \\
\Rightarrow\ &i^2 - i = 2(k-j) \\
\Rightarrow\ &i^2 - i - 2(k-j) = 0 \\
\Rightarrow\ &i = \frac{1}{2} \pm \sqrt{\frac{1}{4} - 2(k-j)}.
\end{aligned}$$

Only the positive solution is of interest to us. Now, we can compute i by setting $j = 0$ and using the floor function since i must be a natural number:

$$i = \left\lfloor \frac{1}{2} + \sqrt{\frac{1}{4} - 2k} \right\rfloor. \tag{A.1}$$

Finally, we insert i in the equation we started from to compute j:

$$j = k - \frac{(i-1)i}{2}. \tag{A.2}$$

Due to floating point inaccuracies for large k, we evaluate the square root in double-precision.

Algorithm 2 Void-and-cluster sampling algorithm

procedure VOIDANDCLUSTER($P \subset \mathbb{R}^d, v : P \to V, h_P, q \in (0,1]$)
  $h \leftarrow \sqrt[d]{\frac{h_P}{q}}$ ▷ Kernel size for samples
  $n \leftarrow q|P|$ ▷ Number of samples
  $\phi \leftarrow \text{IMPORTANCE}(v, h)$ ▷ From entropy or const.
  $S, r, \rho_P, \lambda_S \leftarrow \text{INITIALRANDOMSAMPLING}(P, \phi, h)$
  $\text{OPTIMIZESAMPLES}(S, r, \lambda_S, \rho_P, h)$
  $\text{VOIDFILLING}(S, r, \lambda_S, \rho_P, h, n)$
  return $S, \frac{1}{\phi}, r$ ▷ Return samples, weights, and rank
end procedure

procedure INITIALRANDOMSAMPLING(P, ϕ, h)
  for all $p \in P$ **do** ▷ Compute point density ρ_P
    $\rho_P \leftarrow \text{ADDDENSITY}(\phi(p), h)$
  end for
  $S, r \leftarrow \text{RANDOMSAMPLING}(\phi)$
  for all $s \in S$ **do** ▷ Compute sample density λ_S
    $\lambda_S \leftarrow \text{ADDDENSITY}(\rho_P(s)^{-1}, h)$
  end for
  return S, r, ρ_P, λ_S
end procedure

procedure OPTIMIZESAMPLES($S, r, \lambda_S, \rho_P, h$)
  while true **do**
    $s_{\max} \leftarrow \arg\max_{s \in S}\{\lambda_S(s)\}$ ▷ Find tightest cluster
    $\lambda_S \leftarrow \text{ADDDENSITY}(-\rho_P(s_{\max})^{-1}, h)$
    $p_{\min} \leftarrow \arg\min_{p \in P \setminus S}\{\lambda_S(p)\}$ ▷ Find largest void
    $\lambda_S \leftarrow \text{ADDDENSITY}(\rho_P(p_{\min})^{-1}, h)$
    $r[p_{\min}] \leftarrow r[s_{\max}]$, $r[s_{\max}] \leftarrow \infty$ ▷ Exchange rank
    if $p_{\min} = s_{\max}$ **then**
      break
    end if
  end while
end procedure

procedure VOIDFILLING($S, r, \lambda_S, \rho_P, h, n$)
  for $i \leftarrow |S|, n$ **do**
    $p_{\min} \leftarrow \arg\min_{p \in P \setminus S}\{\lambda_S(p)\}$ ▷ Find largest void
    $\lambda_S \leftarrow \text{ADDDENSITY}(\rho_P(p_{\min})^{-1}, h)$
    $S \leftarrow S \cup p_{\min}$ ▷ Add sample
    $r[p_{\min}] \leftarrow i$
  end for
end procedure

A.3 3D GAUSSIAN RAY INTEGRATION

We integrate a trivariate Gaussian distribution along a ray $o + xd$ starting at $o \in \mathbb{R}^3$ in normalized direction $d \in \mathbb{R}^3$ with $x \in \mathbb{R}$. The Gaussian is given by its mean $\mu \in \mathbb{R}^3$ and covariance $\Sigma \in \mathbb{R}^{3\times3}$. To derive a general solution, we integrate over $[a, b]$ by substituting the ray equation into the trivariate Gaussian distribution:

$$I(a,b) := \int_a^b \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{(o+xd-\mu)^\mathsf{T}\Sigma^{-1}(o+xd-\mu)}{2}\right) \mathrm{d}x. \tag{A.3}$$

Note that $|2\pi\Sigma| = (2\pi)^3|\Sigma|$ for trivariate Gaussians, which we prefer due to its compactness. We start by simplifying the equation:

$$\begin{aligned}
I(a,b) &= \int_a^b \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{((o-\mu)+xd)^\mathsf{T}\Sigma^{-1}((o-\mu)+xd)}{2}\right) \mathrm{d}x \\
&= \int_a^b \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-(o-\mu)^\mathsf{T}\Sigma^{-1}(o-\mu)\right) \\
&\qquad \exp\left(-2x\underbrace{\frac{1}{2}(o-\mu)^\mathsf{T}\Sigma^{-1}d}_{c_{o,d}}\right) \exp\left(-x^2\underbrace{\frac{1}{2}d^\mathsf{T}\Sigma^{-1}d}_{c_{d,d}}\right) \mathrm{d}x \\
&= \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-(o-\mu)^\mathsf{T}\Sigma^{-1}(o-\mu)\right) \int_a^b \exp\left(-2xc_{o,d} - x^2c_{d,d}\right) \mathrm{d}x.
\end{aligned} \tag{A.4}$$

We substitute $r := x + \frac{c_{o,d}}{c_{d,d}}$ and thus rewrite and simplify the integrand as follows:

$$\begin{aligned}
I(a,b) &= \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-(o-\mu)^\mathsf{T}\Sigma^{-1}(o-\mu)\right) \\
&\qquad \int_{a+\frac{c_{o,d}}{c_{d,d}}}^{b+\frac{c_{o,d}}{c_{d,d}}} \exp\left(-2\left(r-\frac{c_{o,d}}{c_{d,d}}\right)c_{o,d} - \left(r-\frac{c_{o,d}}{c_{d,d}}\right)^2 c_{d,d}\right) \mathrm{d}r \\
&= \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-(o-\mu)^\mathsf{T}\Sigma^{-1}(o-\mu)\right) \\
&\qquad \int_{a+\frac{c_{o,d}}{c_{d,d}}}^{b+\frac{c_{o,d}}{c_{d,d}}} \exp\left(-2rc_{o,d} + 2\frac{c_{o,d}^2}{c_{d,d}} - r^2c_{d,d} + 2rc_{o,d} - \frac{c_{o,d}^2}{c_{d,d}}\right) \mathrm{d}r \\
&= \underbrace{\frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-(o-\mu)^\mathsf{T}\Sigma^{-1}(o-\mu)\right) \exp\left(\frac{c_{o,d}^2}{c_{d,d}}\right)}_{c} \int_{a+\frac{c_{o,d}}{c_{d,d}}}^{b+\frac{c_{o,d}}{c_{d,d}}} \exp\left(-r^2c_{d,d}\right) \mathrm{d}r.
\end{aligned} \tag{A.5}$$

Now, we perform another substitution using $p := r\sqrt{c_{d,d}}$:

$$I(a,b) = c\frac{1}{\sqrt{c_{d,d}}} \int_{\sqrt{c_{d,d}}\left(a+\frac{c_{o,d}}{c_{d,d}}\right)}^{\sqrt{c_{d,d}}\left(b+\frac{c_{o,d}}{c_{d,d}}\right)} \exp(-p^2)\,\mathrm{d}p. \tag{A.6}$$

The integrand can now be expressed by its antiderivative, which leads us to the following solution:

$$I(a,b) = c\frac{1}{\sqrt{c_{d,d}}}\left[\frac{\sqrt{\pi}}{2}\operatorname{erf}(p)\right]_{\sqrt{c_{d,d}}\left(a+\frac{c_{o,d}}{c_{d,d}}\right)}^{\sqrt{c_{d,d}}\left(b+\frac{c_{o,d}}{c_{d,d}}\right)}. \tag{A.7}$$

Although no closed-form solution exists for the error function, fast and accurate numerical approximations of this well known function are available. Lastly, when integrating over $(-\infty,\infty)$, the error function disappears:

$$\left[\frac{\sqrt{\pi}}{2}\operatorname{erf}(p)\right]_{-\infty}^{\infty} = \sqrt{\pi},$$

which leads to

$$I(-\infty,\infty) = c\frac{\sqrt{\pi}}{\sqrt{c_{d,d}}}. \tag{A.8}$$

BIBLIOGRAPHY

[1] LZ4. Accessed: 30. Sept. 2020. URL: https://lz4.github.io/lz4 (cit. on p. 125).

[2] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas. "Adaptively Sampled Particle Fluids." In: *ACM Transactions on Graphics* 26.3 (2007). ISSN: 0730-0301. DOI: 10.1145/1275808.1276437 (cit. on p. 13).

[3] A. Agranovsky, D. Camp, C. Garth, E. W. Bethel, K. I. Joy, and H. Childs. "Improved Post Hoc Flow Analysis via Lagrangian Representations." In: *IEEE 4th Symposium on Large Data Analysis and Visualization*. 2014, pp. 67–75. DOI: 10.1109/LDAV.2014.7013206 (cit. on pp. 33, 78, 135).

[4] A. Agranovsky, C. Garth, and K. I. Joy. "Extracting Flow Structures Using Sparse Particles." In: *Vision, Modeling, and Visualization*. 2011, pp. 153–160. DOI: 10.2312/PE/VMV/VMV11/153-160 (cit. on pp. 33, 51).

[5] 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: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '14. 2014, pp. 424–434. ISBN: 978-1-4799-5500-8. DOI: 10.1109/SC.2014.40 (cit. on p. 12).

[6] H. Akaike. "A New Look at the Statistical Model Identification." In: *IEEE Transactions on Automatic Control* 19.6 (1974), pp. 716–723. DOI: 10.1109/TAC.1974.1100705 (cit. on p. 22).

[7] G. Akinci, N. Akinci, M. Ihmsen, and M. Teschner. "An Efficient Surface Reconstruction Pipeline for Particle-Based Fluids." In: *Workshop on Virtual Reality Interaction and Physical Simulation*. 2012. ISBN: 978-3-905673-96-8. DOI: 10.2312/PE/vriphys/vriphys12/061-068 (cit. on p. 13).

[8] M. Ankerst, S. Berchtold, and D. A. Keim. "Similarity Clustering of Dimensions for an Enhanced Visualization of Multidimensional Data." In: *Proceedings IEEE Symposium on Information Visualization*. 1998, pp. 52–60. DOI: 10.1109/INFVIS.1998.729559 (cit. on p. 14).

[9] A. O. Artero, M. C. F. de Oliveira, and H. Levkowitz. "Uncovering Clusters in Crowded Parallel Coordinates Visualizations." In: *IEEE Symposium on Information Visualization*. 2004, pp. 81–88. DOI: 10.1109/INFVIS.2004.68 (cit. on p. 16).

[10] S. Bachthaler and D. Weiskopf. "Continuous Scatterplots." In: *IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), pp. 1428–1435. ISSN: 1077-2626. DOI: 10.1109/TVCG.2008.119 (cit. on p. 17).

[11] I. Baeza Rojo, M. Gross, and T. Günther. "Fourier Opacity Optimization for Scalable Exploration." In: *IEEE Transactions on Visualization and Computer Graphics* (2019), pp. 1–1. DOI: 10.1109/TVCG.2019.2915222 (cit. on p. 8).

[12] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola. "TTHRESH: Tensor Compression for Multidimensional Visual Data." In: *IEEE Transactions on Visualization and Computer Graphics* 26.9 (2020), pp. 2891–2903. DOI: 10.1109/TVCG.2019.2904063 (cit. on p. 19).

[13] M. Balsa Rodríguez, E. Gobbetti, J. A. Iglesias Guitián, M. Makhinya, F. Marton, R. Pajarola, and S. K. Suter. "State-of-the-Art in Compressed GPU-Based Direct Volume Rendering." In: *Computer Graphics Forum* 33.6 (2014), pp. 77–100. DOI: 10.1111/cgf.12280 (cit. on pp. 11, 19).

[14] M. Balzer, T. Schlömer, and O. Deussen. "Capacity-constrained Point Distributions: A Variant of Lloyd's Method." In: *ACM Transactions on Graphics* 28.3 (2009), 86:1–86:8. ISSN: 0730-0301. DOI: 10.1145/1531326.1531392 (cit. on p. 26).

[15] S. Barakat, C. Garth, and X. Tricoche. "Interactive Computation and Rendering of Finite-Time Lyapunov Exponent Fields." In: *IEEE Transactions on Visualization and Computer Graphics* 18.8 (2012), pp. 1368–1380. ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.33 (cit. on p. 32).

[16] R. A. Becker and W. S. Cleveland. "Brushing Scatterplots." In: *Technometrics* 29.2 (1987), pp. 127–142. DOI: 10.1080/00401706.1987.10488204 (cit. on p. 15).

[17] W. Berger, H. Piringer, P. Filzmoser, and E. Gröller. "Uncertainty-Aware Exploration of Continuous Parameter Spaces Using Multivariate Prediction." In: *Computer Graphics Forum* 30.3 (2011), pp. 911–920. DOI: 10.1111/j.1467-8659.2011.01940.x (cit. on p. 135).

[18] E. Bertini and G. Santucci. "Give Chance a Chance: Modeling Density to Enhance Scatter Plot Quality through Random Data Sampling." In: *Information Visualization* 5.2 (2006), pp. 95–110. DOI: 10.1057/palgrave.ivs.9500122 (cit. on p. 16).

[19] J. Beyer, M. Hadwiger, and H. Pfister. "State-of-the-Art in GPU-Based Large-Scale Volume Visualization." In: *Computer Graphics Forum* 34.8 (2015), pp. 13–37. DOI: 10.1111/cgf.12605 (cit. on pp. 11, 19).

[20] H. Bhatia, S. Jadhav, P. Bremer, G. Chen, J. A. Levine, L. G. Nonato, and V. Pascucci. "Flow Visualization with Quantified Spatial and Temporal Errors Using Edge Maps." In: *IEEE Transactions on Visualization and Computer Graphics* 18.9 (2012), pp. 1383–1396. ISSN: 2160-9306. DOI: 10.1109/TVCG.2011.265 (cit. on p. 36).

[21] W. G. Bickley. "LXXIII. The Plane Jet." In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 23.156 (1937), pp. 727–731. DOI: 10.1080/14786443708561847 (cit. on p. 32).

[22] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006 (cit. on pp. 6, 22).

[23] A. Biswas, S. Dutta, E. Lawrence, J. Patchett, J. C. Calhoun, and J. Ahrens. "Probabilistic Data-Driven Sampling via Multi-Criteria Importance Analysis." In: *IEEE Transactions on Visualization and Computer Graphics* (2020), pp. 1–1. ISSN: 1941-0506. DOI: 10.1109/TVCG.2020.3006426 (cit. on p. 25).

[24] A. Biswas, S. Dutta, J. Pulido, and J. Ahrens. "In Situ Data-Driven Adaptive Sampling for Large-Scale Simulation Data Summarization." In: *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. ACM. 2018, pp. 13–18. DOI: 10.1145/3281464.3281467 (cit. on pp. 25, 73, 77).

[25] J. Blaas, C. Botha, and F. Post. "Extensions of Parallel Coordinates for Interactive Exploration of Large Multi-Timepoint Data Sets." In: *IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), pp. 1436–1451. ISSN: 1077-2626. DOI: 10.1109/TVCG.2008.131 (cit. on p. 16).

[26] J. Bonet and T.-S.L. Lok. "Variational and Momentum Preservation Aspects of Smooth Particle Hydrodynamic Formulations." In: *Computer Methods in Applied Mechanics and Engineering* 180.1 (1999), pp. 97 –115. ISSN: 0045-7825. DOI: 10.1016/S0045-7825(99)00051-1 (cit. on p. 41).

[27] G.-P. Bonneau, H.-C. Hege, C. R. Johnson, M. M. Oliveira, K. Potter, P. Rheingans, and T. Schultz. "Overview and State-of-the-Art of Uncertainty Visualization." In: *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization*. Springer London, 2014, pp. 3–27. ISBN: 978-1-4471-6497-5. DOI: 10.1007/978-1-4471-6497-5_1 (cit. on pp. 35, 136).

[28] R. P. Botchen, D. Weiskopf, and T. Ertl. "Texture-based Visualization of Uncertainty in Flow Fields." In: *IEEE Visualization*. 2005, pp. 647–654. DOI: 10.1109/VISUAL.2005.1532853 (cit. on p. 36).

[29] R. Bridson. "Fast Poisson Disk Sampling in Arbitrary Dimensions." In: *ACM SIGGRAPH 2007 Sketches*. SIGGRAPH 2007. ACM, 2007, p. 22. DOI: 10.1145/1278780.1278807 (cit. on pp. 26, 83).

[30] K. Brodlie, R. A. Osorio, and A. Lopes. "A Review of Uncertainty in Data Visualization." In: *Expanding the Frontiers of Visual Analytics and Visualization*. Springer London, 2012, pp. 81–109. ISBN: 978-1-4471-2804-5. DOI: 10.1007/978-1-4471-2804-5_6 (cit. on pp. 35, 136).

[31] L. Brookshaw. "A Method of Calculating Radiative Heat Diffusion in Particle Simulations." In: *Publications of the Astronomical Society of Australia* 6.2 (1985), pp. 207–210 (cit. on p. 30).

[32] R. Bruckschen, F. Kuester, B. Hamann, and K. I. Joy. "Real-Time Out-of-Core Visualization of Particle Traces." In: *Proceedings of the Symposium on Parallel and Large-Data Visualization and Graphics*. 2001, pp. 45–149 (cit. on p. 34).

[33] S. L. Brunton and C. W. Rowley. "Fast Computation of Finite-Time Lyapunov Exponent fields for Unsteady Flows." In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 20.1 (2010), p. 017503. DOI: 10.1063/1.3270044 (cit. on p. 32).

[34] R. Bujack and K. I. Joy. "Lagrangian Representations of Flow Fields with Parameter Curves." In: *IEEE 5th Symposium on Large Data Analysis and Visualization*. 2015, pp. 41–48. DOI: 10.1109/LDAV.2015.7348070 (cit. on pp. 7, 33, 135).

[35] R. Bujack, L. Yan, I. Hotz, C. Garth, and B. Wang. "State of the Art in Time-Dependent Flow Topology: Interpreting Physical Meaningfulness Through Mathematical Properties." In: *Computer Graphics Forum* (2020). ISSN: 1467-8659. DOI: 10.1111/cgf.14037 (cit. on pp. 1, 31).

[36] K. Bürger, P. Kondratieva, J. Krüger, and R. Westermann. "Importance-Driven Particle Techniques for Flow Visualization." In: *IEEE Pacific Visualization Symposium*. 2008, pp. 71–78. DOI: 10.1109/PACIFICVIS.2008.4475461 (cit. on p. 35).

[37] K. Bürger, J. Schneider, P. Kondratieva, J. H. Krüger, and R. Westermann. "Interactive Visual Exploration of Unsteady 3D Flows." In: *Eurographics/IEEE-VGTC Symposium on Visualization*. 2007, pp. 251–258. DOI: 10.2312/VisSym/EuroVis07/251-258 (cit. on p. 34).

[38] R. Bürger and H. Hauser. "Visualization of Multivariate Scientific Data." In: *Eurographics 2007 - State of the Art Reports*. The Eurographics Association, 2007. DOI: 10.2312/egst.20071057 (cit. on p. 13).

[39] R. Bürger, P. Muigg, H. Doleisch, and H. Hauser. "Interactive Cross-Detector Analysis of Vortical Flow Data." In: *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization*. 2007, pp. 98–110. DOI: 10.1109/CMV.2007.12 (cit. on p. 53).

[40] M. Burtscher, H. Mukka, A. Yang, and F. Hesaaraki. "Real-Time Synthesis of Compression Algorithms for Scientific Data." In: *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2016, pp. 264–275. DOI: 10.1109/SC.2016.22 (cit. on p. 20).

[41] L. Carpenter. "The A-Buffer, an Antialiased Hidden Surface Method." In: *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH. Association for Computing Machinery, 1984, pp. 103–108. DOI: 10.1145/800031.808585 (cit. on p. 8).

[42] E. Catmull and R. Rom. "A Class of Local Interpolating Splines." In: *Computer Aided Geometric Design*. Academic Press, 1974, pp. 317 –326. ISBN: 978-0-12-079050-0. DOI: 10.1016/B978-0-12-079050-0.50020-5 (cit. on p. 7).

[43] D. D. Chamberlin and R. F. Boyce. "SEQUEL: A Structured English Query Language." In: *Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*. SIGFIDET '74. 1974, pp. 249–264. ISBN: 9781450374156. DOI: 10.1145/800296.811515 (cit. on p. 15).

[44] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical Methods for Data Analysis*. Wadsworth & Brooks/Cole Publishing Company, 1983 (cit. on p. 13).

[45] A. Chaudhuri, T. H. Wei, T. Y. Lee, H. W. Shen, and T. Peterka. "Efficient Range Distribution Query for Visualizing Scientific Data." In: *IEEE Pacific Visualization Symposium*. 2014, pp. 201–208. DOI: 10.1109/PacificVis.2014.60 (cit. on p. 24).

[46] G. Chaussonnet, S. Braun, T. Dauch, M. Keller, J. Kaden, C. Schwitzke, T. Jakobs, R. Koch, and H.-J. Bauer. "Three-Dimensional SPH Simulation of a Twin-Fluid Atomizer Operating at High Pressure." In: *14th Triennial International Conference on Liquid Atomization and Spray Systems* (2018) (cit. on pp. 2, 29, 107).

[47] B. Chen, A. Kaufman, and Q. Tang. "Image-Based Rendering of Surfaces from Volume Data." In: *Volume Graphics*. The Eurographics Association, 2001. ISBN: 3-211-83737-X. DOI: 10.2312/VG/VG01/281-299 (cit. on p. 12).

[48] C.-M. Chen, A. Biswas, and H. Shen. "Uncertainty Modeling and Error Reduction for Pathline Computation in Time-Varying Flow Fields." In: *IEEE Pacific Visualization Symposium*. 2015, pp. 215–222. DOI: 10.1109/PACIFICVIS.2015.7156380 (cit. on p. 36).

[49] H. Childs. "Data Exploration at the Exascale." In: *Supercomputing frontiers and innovations* 2.3 (2015), pp. 5–13. DOI: 10.14529/jsfi150301 (cit. on pp. 12, 115).

[50] C. S. Co, B. Heckel, H. Hagen, B. Hamann, and K. I. Joy. "Hierarchical Clustering for Unstructured Volumetric Scalar Fields." In: *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*. 2003, pp. 43–. ISBN: 0-7695-2030-8. DOI: 10.1109/VISUAL.2003.1250389 (cit. on p. 10).

[51] W. G. Cochran. *Sampling Techniques*. 3rd edition. John Wiley & Sons, 1977 (cit. on p. 25).

[52] C. Conti, D. Rossinelli, and P. Koumoutsakos. "GPU and APU Computations of Finite Time Lyapunov Exponent fields." In: *Journal of Computational Physics* 231.5 (2012), pp. 2229–2244. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2011.10.032 (cit. on p. 32).

[53] C. D. Correa, Y. H. Chan, and K. L. Ma. "A Framework for Uncertainty-Aware Visual Analytics." In: *IEEE Symposium on Visual Analytics Science and Technology*. 2009, pp. 51–58. DOI: 10.1109/VAST.2009.5332611 (cit. on p. 36).

[54] T. F. Dauch, C. Ates, T. Rapp, M. C. Keller, G. Chaussonnet, J. Kaden, M. Okraschevski, R. Koch, C. Dachsbacher, and H.-J. Bauer. "Analyzing the Interaction of Vortex and Gas–Liquid Interface Dynamics in Fuel Spray Nozzles by Means of Lagrangian-Coherent Structures (2D)." In: *Energies* (2019). ISSN: 1996-1073. DOI: 10.3390/en12132552 (cit. on pp. iv, 4, 54, 135).

[55] T. F. Dauch, T. Rapp, G. Chaussonnet, S. Braun, M. C. Keller, J. Kaden, R. Koch, C. Dachsbacher, and H.-J. Bauer. "Highly Efficient Computation of Finite-Time Lyapunov Exponents (FTLE) on GPUs Based on Three-Dimensional SPH Datasets." In: *Computers & Fluids* (2018). ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2018.07.015 (cit. on pp. 3, 46).

[56] B. Delaunay. "Sur la Sphère Vide." French. In: *Bull. Acad. Science URSS* 6 (1934), pp. 793–800 (cit. on p. 5).

[57] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data Via the EM Algorithm." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22. DOI: 10.1111/j.2517-6161.1977.tb01600.x (cit. on p. 23).

[58] A. Dix and G. Ellis. "By Chance Enhancing Interaction with Large Data Sets Through Statistical Sampling." In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. 2002, pp. 167–176. ISBN: 1-58113-537-8. DOI: 10.1145/1556262.1556289 (cit. on pp. 16, 25).

[59] H. Doleisch and H. Hauser. "Smooth Brushing For Focus+Context Visualization Of Simulation Data In 3D." In: *Journal of WSCG*. 2001, pp. 147–154 (cit. on pp. 15, 101).

[60] *DualSPHysics*. https://dual.sphysics.org/. Accessed: 2021-01-14 (cit. on p. 127).

[61] S. Dutta, A. Biswas, and J. Ahrens. "Multivariate Pointwise Information-Driven Data Sampling and Visualization." In: *Entropy* 21.7 (2019), p. 699 (cit. on p. 26).

[62] 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." In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 811–820. ISSN: 1077-2626. DOI: 10.1109/TVCG.2016.2598604 (cit. on pp. 23, 24).

[63] S. Dutta and H. W. Shen. "Distribution Driven Extraction and Tracking of Features for Time-varying Data Analysis." In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), pp. 837–846. ISSN: 1077-2626. DOI: 10.1109/TVCG.2015.2467436 (cit. on pp. 23, 24).

[64] 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: *IEEE Pacific Visualization Symposium*. 2017, pp. 111–120. DOI: 10.1109/PACIFICVIS.2017.8031585 (cit. on p. 23).

[65] I. Eames and J. B. Flor. "New Developments in Understanding Interfacial Processes in Turbulent Flows." In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 369.1937 (2011), pp. 702–705. DOI: 10.1098/rsta.2010.0332 (cit. on p. 27).

[66] M. Edmunds, R. S. Laramee, G. Chen, N. Max, E. Zhang, and C. Ware. "Surface-Based Flow Visualization." In: *Computers & Graphics* 36.8 (2012). Graphics Interaction Virtual Environments and Applications 2012, pp. 974 –990. ISSN: 0097-8493. DOI: 10.1016/j.cag.2012.07.006 (cit. on p. 34).

[67] S. Eichelbaum, G. Scheuermann, and M. Hlawitschka. "PointAO - Improved Ambient Occlusion for Point-based Visualization." In: *EuroVis Short Papers*. The Eurographics Association, 2013. ISBN: 978-3-905673-99-9. DOI: 10.2312/PE.EuroVisShort.EuroVisShort2013.013-017 (cit. on p. 8).

[68] G. Ellis, E. Bertini, and A. Dix. "The Sampling Lens: Making Sense of Saturated Visualisations." In: *CHI 2005 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '05. Association for Computing Machinery, 2005, pp. 1351–1354. ISBN: 1595930027. DOI: 10.1145/1056808.1056914 (cit. on p. 16).

[69] G. Ellis and A. Dix. "Enabling Automatic Clutter Reduction in Parallel Coordinate Plots." In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 717–724. DOI: 10.1109/TVCG.2006.138 (cit. on p. 16).

[70] G. Ellis and A. Dix. "A Taxonomy of Clutter Reduction for Information Visualisation." In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007), pp. 1216–1223. DOI: 10.1109/TVCG.2007.70535 (cit. on p. 16).

[71] D. Ellsworth, B. Green, and P. Moran. "Interactive Terascale Particle Visualization." In: *IEEE Visualization*. 2004, pp. 353–360. DOI: 10.1109/VISUAL.2004.55 (cit. on pp. 10, 20, 34).

[72] K. Engel, M. Kraus, and T. Ertl. "High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading." In: *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*. HWWS '01. Association for Computing Machinery, 2001, pp. 9–16. DOI: 10.1145/383507.383515 (cit. on p. 120).

[73] M. Farazmand and G. Haller. "Computing Lagrangian Coherent Structures from their Variational Theory." In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22.1 (2012), p. 013128. DOI: 10.1063/1.3690153 (cit. on pp. 44, 47).

[74] D. Feng, L. Kwock, Y. Lee, and R. Taylor. "Matching Visual Saliency to Confidence in Plots of Uncertain Data." In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 980–989. DOI: 10.1109/TVCG.2010.176 (cit. on p. 101).

[75] O. Fernandes, S. Frey, F. Sadlo, and T. Ertl. "Space-Time Volumetric Depth Images for In-Situ Visualization." In: *IEEE 4th Symposium on Large Data Analysis and Visualization*. 2014, pp. 59–65. DOI: 10.1109/LDAV.2014.7013205 (cit. on pp. 13, 115).

[76] R. Fraedrich, S. Auer, and R. Westermann. "Efficient High-Quality Volume Rendering of SPH Data." In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1533–1540. ISSN: 1077-2626. DOI: 10.1109/TVCG.2010.148 (cit. on pp. 11, 24).

[77] R. Fraedrich, J. Schneider, and R. Westermann. "Exploring the Millennium Run - Scalable Rendering of Large-Scale Cosmological Datasets." In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 1251–1258. ISSN: 1941-0506. DOI: 10.1109/TVCG.2009.142 (cit. on pp. 10, 20).

[78] S. Frey, F. Sadlo, and T. Ertl. "Explorable Volumetric Depth Images from Raycasting." In: *XXVI Conference on Graphics, Patterns and Images*. 2013, pp. 123–130. DOI: 10.1109/SIBGRAPI.2013.26 (cit. on pp. 12, 115).

[79] S. Frey, T. Schlömer, S. Grottel, C. Dachsbacher, O. Deussen, and T. Ertl. "Loose Capacity-Constrained Representatives for the Qualitative Visual Analysis in Molecular Dynamics." In: *IEEE Pacific Visualization Symposium*. 2011, pp. 51–58. DOI: 10.1109/PACIFICVIS.2011.5742372 (cit. on pp. 26, 83).

[80] J. H. Friedman and W. Stuetzle. "John W. Tukey's Work on Interactive Graphics." In: *The Annals of Statistics* 30.6 (2002), pp. 1629–1639. ISSN: 00905364. URL: http://www.jstor.org/stable/1558733 (cit. on p. 14).

[81] L. Fritschi, I. B. Rojo, and T. Günther. "Visualizing the Temporal Evolution of the Universe from Cosmology Simulations." In: *IEEE Scientific Visualization Contest*. 2019. DOI: 10.1109/SciVis47405.2019.8968943 (cit. on p. 135).

[82] R. Fuchs, J. Kemmler, B. Schindler, J. Waser, F. Sadlo, H. Hauser, and R. Peikert. "Toward a Lagrangian Vector Field Topology." In: *Computer Graphics Forum* 29.3 (2010), pp. 1163–1172. DOI: 10.1111/j.1467-8659.2009.01686.x (cit. on p. 31).

[83] J. D. Furst and S. M. Pizer. "Marching Ridges." In: *IASTED International Conference on Signal and Image Processing*. 2001 (cit. on p. 33).

[84] GEBCO Bathymetric Compilation Group 2019. "The GEBCO 2019 Grid - a Continuous Terrain Model for Oceans and Land at 15 Arc-Second Intervals." In: (2019). DOI: 10.5285/836f016a-33be-6ddc-e053-6c86abc0788e (cit. on p. 65).

[85] *GPUSPH*. http://www.gpusph.org. Accessed: 2018-03-26 (cit. on p. 57).

[86] C. Garth, F. Gerhardt, X. Tricoche, and H. Hans. "Efficient Computation and Visualization of Coherent Structures in Fluid Flow Applications." In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007), pp. 1464–1471. DOI: 10.1109/TVCG.2007.70551 (cit. on p. 33).

[87] A. V. Getling. *Rayleigh-Bénard Convection*. World Scientific, 1998. DOI: 10.1142/3097 (cit. on p. 27).

[88] R. A. Gingold and J.J. Monaghan. "Smoothed Particle Hydrodynamics: Theory and Application to Non-Spherical Stars." In: *Monthly notices of the royal astronomical society* 181.3 (1977), pp. 375–389. DOI: 10.1093/mnras/181.3.375 (cit. on pp. 1, 29).

[89] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1993 (cit. on p. 98).

[90] L. J. Gosink, C. Garth, J. C. Anderson, E. W. Bethel, and K. I. Joy. "An Application of Multivariate Statistical Analysis for Query-Driven Visualization." In: *IEEE Transactions on Visualization and Computer Graphics* 17.3 (2011), pp. 264–275. ISSN: 1077-2626. DOI: 10.1109/TVCG.2010.80 (cit. on p. 24).

[91] P. Goswami, P. Schlegel, B. Solenthaler, and R. Pajarola. "Interactive SPH Simulation and Rendering on the GPU." In: *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2010, pp. 55–64. DOI: 10.2312/SCA/SCA10/055-064 (cit. on p. 43).

[92] P. Gralka, I. Wald, S. Geringer, G. Reina, and T. Ertl. "Spatial Partitioning Strategies for Memory-Efficient Ray Tracing of Particles." In: *IEEE 10th Symposium on Large Data Analysis and Visualization*. 2020 (cit. on p. 8).

[93] C. P. Gribble. "Interactive Particle Visualisation." In: *Trends in Interactive Visualization: State-of-the-Art Survey*. London: Springer London, 2009, pp. 79–97. ISBN: 978-1-84800-269-2 (cit. on p. 8).

[94] C. P. Gribble, T. Ize, A. Kensler, I. Wald, and S. G. Parker. "A Coherent Grid Traversal Approach to Visualizing Particle-Based Simulation Data." In: *IEEE Transactions on Visualization and Computer Graphics* 13.4 (2007), pp. 758–768. ISSN: 2160-9306. DOI: 10.1109/TVCG.2007.1059 (cit. on p. 8).

[95] S. Grottel, M. Krone, C. Müller, G. Reina, and T. Ertl. "MegaMol – A Prototyping Framework for Particle-based Visualization." In: *IEEE Transactions on Visualization and Computer Graphics* 21.2 (2015), pp. 201–214. ISSN: 1077-2626. DOI: 10.1109/TVCG.2014.2350479 (cit. on p. 8).

[96] S. Grottel, G. Reina, C. Dachsbacher, and T. Ertl. "Coherent Culling and Shading for Large Molecular Dynamics Visualization." In: *Computer Graphics Forum* 29.3 (2010), pp. 953–962. DOI: 10.1111/j.1467-8659.2009.01698.x (cit. on p. 8).

[97] D. Groß and S. Gumhold. "Advanced Rendering of Line Data with Ambient Occlusion and Transparency." In: *IEEE Transactions on Visualization and Computer Graphics* (2020). ISSN: 1941-0506. DOI: 10.1109/TVCG.2020.3028954 (cit. on p. 8).

[98] T. Günther, M. Gross, and H. Theisel. "Generic Objective Vortices for Flow Visualization." In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 36.4 (2017), 141:1–141:11. DOI: 10.1145/3072959.3073684 (cit. on p. 67).

[99] T. Günther, C. Rössl, and H. Theisel. "Opacity Optimization for 3D Line Fields." In: *ACM Transactions on Graphics* 32.4 (2013), 120:1–120:8. ISSN: 0730-0301. DOI: 10.1145/2461912.2461930 (cit. on p. 8).

[100] T. Günther and H. Theisel. "The State of the Art in Vortex Extraction." In: *Computer Graphics Forum* 37.6 (2018), pp. 149–173. DOI: 10.1111/cgf.13319 (cit. on p. 31).

[101] H. Guo, W. He, T. Peterka, H. Shen, S. M. Collis, and J. J. Helmus. "Finite-Time Lyapunov Exponents and Lagrangian Coherent Structures in Uncertain Unsteady Flows." In: *IEEE Transactions on Visualization and Computer Graphics* 22.6 (2016), pp. 1672–1682. ISSN: 2160-9306. DOI: 10.1109/TVCG.2016.2534560 (cit. on pp. 37, 59, 63).

[102] H. Guo, W. He, S. Seo, H. Shen, E. M. Constantinescu, C. Liu, and T. Peterka. "Extreme-Scale Stochastic Particle Tracing for Uncertain Unsteady Flow Visualization and Analysis." In: *IEEE Transactions on Visualization and Computer Graphics* 25.9 (2019), pp. 2710–2724. ISSN: 2160-9306. DOI: 10.1109/TVCG.2018.2856772 (cit. on p. 37).

[103] H. Guo, X. Yuan, J. Huang, and X. Zhu. "Coupled Ensemble Flow Line Advection and Analysis." In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2733–2742. ISSN: 1941-0506. DOI: 10.1109/TVCG.2013.144 (cit. on p. 37).

[104] A. Hadjighasem, M. Farazmand, D. Blazevski, G. Froyland, and G. Haller. "A Critical Comparison of Lagrangian Methods for Coherent Structure Detection." In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.5 (2017), p. 053104. DOI: 10.1063/1.4982720 (cit. on p. 33).

[105] M. Hadwiger, J. M. Kniss, C. Rezk-salama, D. Weiskopf, and K. Engel. *Real-time Volume Graphics*. A. K. Peters, Ltd., 2006. ISBN: 1568812663 (cit. on p. 10).

[106] M. Hadwiger, M. Mlejnek, T. Theußl, and P. Rautek. "Time-Dependent Flow seen through Approximate Observer Killing Fields." In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pp. 1257–1266. ISSN: 2160-9306. DOI: 10.1109/TVCG.2018.2864839 (cit. on p. 30).

[107] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, and M. Gross. "Real-time Ray-Casting and Advanced Shading of Discrete Isosurfaces." In: *Computer Graphics Forum*. Vol. 24. 3. Wiley Online Library. 2005, pp. 303–312. DOI: 10.1111/j.1467-8659.2005.00855.x (cit. on p. 13).

[108] G. Haller. "Distinguished Material Surfaces and Coherent Structures in Three-Dimensional Fluid Flows." In: *Physica D: Nonlinear Phenomena* 149.4 (2001), pp. 248–277. DOI: 10.1016/S0167-2789(00)00199-8 (cit. on pp. 33, 44).

[109] G. Haller. "A Variational Theory of Hyperbolic Lagrangian Coherent Structures." In: *Physica D: Nonlinear Phenomena* 240.7 (2011), pp. 574–598. ISSN: 0167-2789. DOI: 10.1016/j.physd.2010.11.010 (cit. on pp. 33, 44).

[110] G. Haller. "Lagrangian Coherent Structures." In: *Annual Review of Fluid Mechanics* 47 (2015), pp. 137–162. DOI: 10.1146/annurev-fluid-010313-141322 (cit. on pp. 1, 31, 33).

[111] G. Haller, D. Karrasch, and F. Kogelbauer. "Material Barriers to Diffusive and Stochastic Transport." In: *Proceedings of the National Academy of Sciences* 115.37 (2018), pp. 9074–9079. ISSN: 0027-8424. DOI: 10.1073/pnas.1720177115 (cit. on pp. 59–61).

[112] G. Haller, D. Karrasch, and F. Kogelbauer. "Barriers to the Transport of Diffusive Scalars in Compressible Flows." In: *SIAM Journal on Applied Dynamical Systems* 19.1 (2020), pp. 85–123. DOI: 10.1137/19M1238666 (cit. on pp. 59, 60).

[113] G. Haller and G. Yuan. "Lagrangian Coherent Structures and Mixing in Two-Dimensional Turbulence." In: *Physica D: Nonlinear Phenomena* 147.3 (2000), pp. 352–370. DOI: 10.1016/S0167-2789(00)00142-1 (cit. on p. 32).

[114] J. Han, J. Tao, H. Zheng, H. Guo, D. Z. Chen, and C. Wang. "Flow Field Reduction Via Reconstructing Vector Data From 3D Streamlines Using Deep Learning." In: *IEEE Computer Graphics and Applications* 39.4 (2019), pp. 54–67. ISSN: 1558-1756. DOI: 10.1109/MCG.2018.2881523 (cit. on p. 20).

[115] M. Han, I. Wald, W. Usher, Q. Wu, F. Wang, V. Pascucci, C. D. Hansen, and C. R. Johnson. "Ray Tracing Generalized Tube Primitives: Method and Applications." In: *Computer Graphics Forum* 38.3 (2019), pp. 467–478. DOI: 10.1111/cgf.13703 (cit. on p. 8).

[116] P. Hanrahan. "VizQL: A Language for Query, Analysis and Visualization." In: *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*. SIGMOD '06. Association for Computing Machinery, 2006, p. 721. ISBN: 1595934340. DOI: 10.1145/1142473.1142560 (cit. on p. 15).

[117] H. Hauser. "Generalizing Focus+Context Visualization." In: *Scientific Visualization: The Visual Extraction of Knowledge from Data*. Springer Berlin Heidelberg, 2006, pp. 305–327. ISBN: 978-3-540-30790-7. DOI: 10.1007/3-540-30790-7_18 (cit. on p. 15).

[118] H. Hauser, F. Ledermann, and H. Doleisch. "Angular Brushing of Extended Parallel Coordinates." In: *Proceedings of the IEEE Symposium on Information Visualization*. InfoVis '02. IEEE Computer Society, 2002, p. 127. ISBN: 076951751X. DOI: 10.1109/INFVIS.2002.1173157 (cit. on p. 15).

[119] S. Hazarika, A. Biswas, and H. W. Shen. "Uncertainty Visualization Using Copula-Based Analysis in Mixed Distribution Models." In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 934–943. ISSN: 1077-2626. DOI: 10.1109/TVCG.2017.2744099 (cit. on p. 24).

[120] S. Hazarika, A. Biswas, P. Wolfram, E. Lawrence, and N. Urban. "Relationship-aware Multivariate Sampling Strategy for Scientific Simulation Data." In: *IEEE VIS Short Papers*. 2020. URL: https://arxiv.org/abs/2008.13306 (cit. on p. 26).

[121] S. Hazarika, S. Dutta, H. Shen, and J. Chen. "CoDDA: A Flexible Copula-based Distribution Driven Analysis Framework for Large-Scale Multivariate Data." In: *IEEE Transactions on Visualization and Computer Graphics* (2019), pp. 1–1. ISSN: 1077-2626. DOI: 10.1109/TVCG.2018.2864801 (cit. on p. 24).

[122] W. He, J. Wang, H. Guo, K. Wang, H. Shen, M. Raj, Y. S. G. Nashed, and T. Peterka. "InSituNet: Deep Image Synthesis for Parameter Space Exploration of Ensemble Simulations." In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 23–33. ISSN: 1941-0506. DOI: 10.1109/TVCG.2019.2934312 (cit. on p. 135).

[123] J. Heinrich, S. Bachthaler, and D. Weiskopf. "Progressive Splatting of Continuous Scatterplots and Parallel Coordinates." In: *Computer Graphics Forum* 30.3 (2011), pp. 653–662. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2011.01914.x (cit. on p. 17).

[124] J. Heinrich and D. Weiskopf. "Continuous Parallel Coordinates." In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 1531–1538. ISSN: 1077-2626. DOI: 10.1109/TVCG.2009.131 (cit. on p. 17).

[125] J. Heinrich and D. Weiskopf. "State of the Art of Parallel Coordinates." In: *Eurographics 2013 - State of the Art Reports*. The Eurographics Association, 2013. DOI: 10.2312/conf/EG2013/stars/095-116 (cit. on p. 14).

[126] J. Helman and L. Hesselink. "Representation and Display of Vector Field Topology in Fluid Flow Data Sets." In: *Computer* 22.08 (1989), pp. 27–36. ISSN: 0018-9162. DOI: 10.1109/2.35197 (cit. on pp. 1, 30).

[127] N. Henze. "Invariant Tests for Multivariate Normality: A Critical Review." In: *Statistical papers* 43.4 (2002), pp. 467–506 (cit. on p. 22).

[128] H. Hersbach, C. Peubey, A. Simmons, P. Berrisford, P. Poli, and D. Dee. "ERA-20CM: A Twentieth-Century Atmospheric Model Ensemble." In: *Quarterly Journal of the Royal Meteorological Society* 141.691 (2015), pp. 2350–2375. DOI: 10.1002/qj.2528 (cit. on p. 36).

[129] M. Hlawatsch, P. Leube, W. Nowak, and D. Weiskopf. "Flow Radar Glyphs - Static Visualization of Unsteady Flow with Uncertainty." In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 1949–1958. ISSN: 2160-9306. DOI: 10.1109/TVCG.2011.203 (cit. on p. 7).

[130] M. Hlawatsch, F. Sadlo, H. Jang, and D. Weiskopf. "Pathline Glyphs." In: *Computer Graphics Forum* 33.2 (2014), pp. 497–506. DOI: 10.1111/cgf.12335 (cit. on p. 7).

[131] M. Hlawatsch, F. Sadlo, and D. Weiskopf. "Hierarchical Line Integration." In: *IEEE Transactions on Visualization and Computer Graphics* 17.8 (2011), pp. 1148–1163. ISSN: 1077-2626. DOI: 10.1109/TVCG.2010.227 (cit. on pp. 36, 78).

[132] H. Hochheiser and B. Shneiderman. "Dynamic Query Tools for Time Series Data Sets: Timebox Widgets for Interactive Exploration." In: *Information Visualization* 3.1 (2004), pp. 1–18. DOI: 10.1057/palgrave.ivs.9500061 (cit. on p. 101).

[133] H. Hochstetter, J. Orthmann, and A. Kolb. "Adaptive Sampling for On-The-Fly Ray Casting of Particle-based Fluids." In: *Eurographics / ACM SIGGRAPH Symposium on High Performance Graphics*. 2016. ISBN: 978-3-03868-008-6. DOI: 10.2312/hpg.20161199 (cit. on p. 11).

[134] B. E. Hollister and A. Pang. "Visual Analysis of Transport Similarity in 2D CFD Ensembles." In: *Electronic Imaging* 2016.1 (2016), pp. 1–11. DOI: 10.2352/ISSN.2470-1173.2016.1.VDA-508 (cit. on p. 36).

[135] W. Hong, N. Neophytou, K. Mueller, and A. Kaufman. "Constructing 3D Elliptical Gaussians for Irregular Data." In: *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*. Springer Berlin Heidelberg, 2009, pp. 213–225. ISBN: 978-3-540-49926-8. DOI: 10.1007/b106657_11 (cit. on p. 10).

[136] M. Hopf and T. Ertl. "Herarchical Splatting of Scattered Data." In: *IEEE Visualization*. 2003, pp. 433–440. DOI: 10.1109/VISUAL.2003.1250404 (cit. on pp. 9, 20).

[137] M. Hopf, M. Luttenberger, and T. Ertl. "Hierarchical Splatting of Scattered 4D Data." In: *IEEE Computer Graphics and Applications* 24.4 (2004), pp. 64–72. ISSN: 1558-1756. DOI: 10.1109/MCG.2004.7 (cit. on pp. 9, 20).

[138] D. A. Huffman. "A Method for the Construction of Minimum-Redundancy Codes." In: *Proceedings of the IRE* 40.9 (1952), pp. 1098–1101. ISSN: 2162-6634. DOI: 10.1109/JRPROC.1952.273898 (cit. on p. 19).

[139] M. Hummel, R. Bujack, K. I. Joy, and C. Garth. "Error Estimates for Lagrangian Flow Field Representations." In: *Proceedings of the Eurographics / IEEE VGTC Conference on Visualization: Short Papers*. EuroVis '16. Eurographics Association, 2016, pp. 7–11. DOI: 10.2312/eurovisshort.20161153 (cit. on p. 36).

[140] M. Hummel, H. Obermaier, C. Garth, and K. I. Joy. "Comparative Visual Analysis of Lagrangian Transport in CFD Ensembles." In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2743–2752. ISSN: 2160-9306. DOI: 10.1109/TVCG.2013.141 (cit. on p. 36).

[141] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak. "Out-of-Core Compression and Decompression of Large n-Dimensional Scalar Fields." In: *Computer Graphics Forum* 22.3 (2003), pp. 343–348. DOI: 10.1111/1467-8659.00681 (cit. on p. 20).

[142] O. Igouchkine, Y. Zhang, and K. Ma. "Multi-Material Volume Rendering with a Physically-Based Surface Reflection Model." In: *IEEE Transactions on Visualization and Computer Graphics* 24.12 (2018), pp. 3147–3159. ISSN: 1941-0506. DOI: 10.1109/TVCG.2017.2784830 (cit. on p. 13).

[143] M. Ihmsen, N. Akinci, M. Becker, and M. Teschner. "A Parallel SPH Implementation on Multi-Core CPUs." In: *Computer Graphics Forum* 30.1 (2011), pp. 99–112. DOI: 10.1111/j.1467-8659.2010.01832.x (cit. on p. 43).

[144] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner. "SPH Fluids in Computer Graphics." In: *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014. DOI: 10.2312/egst.20141034 (cit. on pp. 29, 42).

[145] A. Inselberg. "The Plane with Parallel Coordinates." In: *The visual computer* 1.2 (1985), pp. 69–91. DOI: 10.1007/BF01898350 (cit. on p. 14).

[146] M. Isenburg, P. Lindstrom, and J. Snoeyink. "Lossless Compression of Predicted Floating-Point Geometry." In: *Computer-Aided Design* 37.8 (2005), pp. 869–877. ISSN: 0010-4485. DOI: 10.1016/j.cad.2004.09.015 (cit. on p. 20).

[147] R. Jain and I. Chlamtac. "The P^2 Algorithm for Dynamic Calculation of Quantiles and Histograms without Storing Observations." In: *Communications of the ACM* 28.10 (1985), pp. 1076–1085. ISSN: 0001-0782. DOI: 10.1145/4372.4378 (cit. on p. 126).

[148] J. Jakob, M. Gross, and T. Günther. "A Fluid Flow Data Set for Machine Learning and its Application to Neural Flow Map Interpolation." In: *IEEE Transactions on Visualization and Computer Graphics* (2020), pp. 1–1. ISSN: 1941-0506. DOI: 10.1109/TVCG.2020.3028947 (cit. on p. 32).

[149] Y. Jang, R. P. Botchen, A. Lauser, D. S. Ebert, K. P. Gaither, and T. Ertl. "Enhancing the Interactive Visualization of Procedurally Encoded Multifield Data with Ellipsoidal Basis Functions." In: *Computer Graphics Forum* 25.3 (2006), pp. 587–596. DOI: 10.1111/j.1467-8659.2006.00978.x (cit. on p. 10).

[150] Y. Jang, R. Fuchs, B. Schindler, and R. Peikert. "Volumetric Evaluation of Meshless Data From Smoothed Particle Hydrodynamics Simulations." In: *IEEE/EG Symposium on Volume Graphics*. 2010. ISBN: 978-3-905674-23-1. DOI: 10.2312/VG/VG10/045-052 (cit. on p. 11).

[151] Y. Jang, M. Weiler, M. Hopf, J. Huang, D. S. Ebert, K. P. Gaither, and T. Ertl. "Interactively Visualizing Procedurally Encoded Scalar Fields." In: *Proceedings of the Sixth Joint Eurographics - IEEE TCVG Conference on Visualization*. VISSYM'04. 2004, pp. 35–44. ISBN: 3-905673-07-X. DOI: 10.2312/VisSym/VisSym04/035-044 (cit. on p. 10).

[152] J. Jeong and F. Hussain. "On the Identification of a Vortex." In: *Journal of Fluid Mechanics* 285 (1995), pp. 69–94. DOI: 10.1017/S0022112095000462 (cit. on p. 34).

[153] J. Johansson, P. Ljung, M. Jern, and M. Cooper. "Revealing Structure Within Clustered Parallel Coordinates Displays." In: *IEEE Symposium on Information Visualization, 2005*. 2005, pp. 125–132. DOI: 10.1109/INFVIS.2005.1532138 (cit. on p. 100).

[154] C. R. Johnson and A. R. Sanderson. "A Next Step: Visualizing Errors and Uncertainty." In: *IEEE Computer Graphics and Applications* 23.5 (2003), pp. 6–10. ISSN: 1558-1756. DOI: 10.1109/MCG.2003.1231171 (cit. on p. 35).

[155] C. Jones, K.-L. Ma, S. Ethier, and W.-L. Lee. "An Integrated Exploration Approach to Visualizing Multivariate Particle Data." In: *Computing in Science & Engineering* 10.4 (2008), pp. 20–29. DOI: 10.1109/MCSE.2008.88 (cit. on p. 35).

[156] D. Juba and A. Varshney. "Modelling and Rendering Large Volume Data with Gaussian Radial Basis Functions." In: *University of Maryland, Technical Report No. UMIACS-TR-2007-22* (2007) (cit. on p. 10).

[157] R. Kaehler, T. Abel, and H.-C. Hege. "Simultaneous GPU-Assisted Raycasting of Unstructured Point Sets and Volumetric Grid Data." In: *Eurographics/IEEE VGTC Symposium on Volume Graphics*. 2007. ISBN: 978-3-905674-03-3. DOI: 10.2312/VG/VG07/049-056 (cit. on p. 8).

[158] F. Kahlert and S. Gumhold. "Partial Matching of Trajectories with Particle Orientation for Exploratory Trajectory Visualization." In: *Vision, Modeling, and Visualization*. The Eurographics Association, 2020. ISBN: 978-3-03868-123-6. DOI: 10.2312/vmv.20201193 (cit. on p. 7).

[159] J. Kalojanov and P. Slusallek. "A Parallel Algorithm for Construction of Uniform Grids." In: *Proceedings of the Conference on High Performance Graphics 2009*. HPG '09. ACM, 2009, pp. 23–28. ISBN: 978-1-60558-603-8. DOI: 10.1145/1572769.1572773 (cit. on p. 42).

[160] Y. Kanamori, Z. Szego, and T. Nishita. "GPU-based Fast Ray Casting for a Large Number of Metaballs." In: *Computer Graphics Forum* 27.2 (2008), pp. 351–360. DOI: 10.1111/j.1467-8659.2008.01132.x (cit. on p. 8).

[161] E. Kandogan. "Star Coordinates: A Multi-dimensional Visualization Technique with Uniform Treatment of Dimensions." In: *In Proceedings of the IEEE Information Visualization Symposium, Late Breaking Hot Topics*. 2000, pp. 9–12 (cit. on p. 14).

[162] E. Kandogan. "Visualizing Multi-Dimensional Clusters, Trends, and Outliers Using Star Coordinates." In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '01. 2001, pp. 107–116. ISBN: 158113391X. DOI: 10.1145/502512.502530 (cit. on p. 14).

[163] M. Kanzler, M. Rautenhaus, and R. Westermann. "A Voxel-Based Rendering Pipeline for Large 3D Line Sets." In: *IEEE Transactions on Visualization and Computer Graphics* 25.7 (2019), pp. 2378–2391. ISSN: 1941-0506. DOI: 10.1109/TVCG.2018.2834372 (cit. on p. 8).

[164] G. K. Karch, F. Sadlo, D. Weiskopf, and T. Ertl. "Visualization of 2D Unsteady Flow Using Streamline-Based Concepts in Space-Time." In: *Journal of Visualization* 19.1 (2016), pp. 115–128. ISSN: 1875-8975. DOI: 10.1007/s12650-015-0284-z (cit. on p. 31).

[165] J. Kehrer and H. Hauser. "Visualization and Visual Analysis of Multifaceted Scientific Data: A Survey." In: *IEEE Transactions on Visualization and Computer Graphics* 19.3 (2013), pp. 495–513. ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.110 (cit. on pp. 13, 136).

[166] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. "Visual Analytics: Definition, Process, and Challenges." In: *Information Visualization: Human-Centered Issues and Perspectives*. Springer Berlin Heidelberg, 2008, pp. 154–175. ISBN: 978-3-540-70956-5. DOI: 10.1007/978-3-540-70956-5_7 (cit. on p. 17).

[167] M. Kern, C. Neuhauser, T. Maack, M. Han, W. Usher, and R. Westermann. "A Comparison of Rendering Techniques for 3D Line Sets with Transparency." In: *IEEE Transactions on Visualization and Computer Graphics* (2020), pp. 1–1. DOI: 10.1109/TVCG.2020.2975795 (cit. on p. 8).

[168] B. Kim and T. Günther. "Robust Reference Frame Extraction from Unsteady 2D Vector Fields with Convolutional Neural Networks." In: *Computer Graphics Forum* 38.3 (2019), pp. 285–295. DOI: 10.1111/cgf.13689 (cit. on p. 31).

[169] G. Kindlmann, C. Chiw, T. Huynh, A. Gyulassy, J. Reppy, and P.-T. Bremer. "Rendering and Extracting Extremal Features in 3D Fields." In: *Computer Graphics Forum* 37.3 (2018), pp. 525–536. DOI: 10.1111/cgf.13439 (cit. on p. 33).

[170] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Vol. 23. Springer Science & Business Media, 2013 (cit. on p. 60).

[171] J. Kniss, G. Kindlmann, and C. Hansen. "Multidimensional Transfer Functions for Interactive Volume Rendering." In: *IEEE Transactions on Visualization and Computer Graphics* 8.3 (2002), pp. 270–285. DOI: 10.1109/TVCG.2002.1021579 (cit. on p. 105).

[172] A. Knoll, Y. Hijazi, R. Westerteiger, M. Schott, C. Hansen, and H. Hagen. "Volume Ray Casting with Peak Finding and Differential Sampling." In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 1571–1578. ISSN: 1941-0506. DOI: 10.1109/TVCG.2009.204 (cit. on p. 13).

[173] A. Knoll, R. K. Morley, I. Wald, N. Leaf, and P. Messmer. "Efficient Particle Volume Splatting in a Ray Tracer." In: *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*. Apress, 2019. Chap. 29, pp. 533–541. ISBN: 978-1-4842-4427-2. DOI: 10.1007/978-1-4842-4427-2_29 (cit. on pp. 9, 98).

[174] A. Knoll, I. Wald, P. Navratil, A. Bowen, K. Reda, M. E. Papka, and K. Gaither. "RBF Volume Ray Casting on Multicore and Manycore CPUs." In: *Computer Graphics Forum* 33.3 (2014), pp. 71–80. DOI: 10.1111/cgf.12363 (cit. on p. 11).

[175] R. Koch, S. Braun, L. Wieth, G. Chaussonnet, T. Dauch, and H.-J. Bauer. "Prediction of Primary Atomization using Smoothed Particle Hydrodynamics." In: *European Journal of Mechanics - B/Fluids* 61, Part 2 (2017). Rotating Flows, pp. 271–278. ISSN: 0997-7546. DOI: 10.1016/j.euromechflu.2016.10.007 (cit. on p. 29).

[176] R. Kosara, F. Bendix, and H. Hauser. "Time Histograms for Large, Time-Dependent Data." In: *Proceedings of the Sixth Joint Eurographics-IEEE TCVG conference on Visualization*. 2004, pp. 45–54 (cit. on p. 101).

[177] J. Krüger, P. Kipfer, P. Konclratieva, and R. Westermann. "A Particle System for Interactive Visualization of 3D Flows." In: *IEEE Transactions on visualization and computer graphics* 11.6 (2005), pp. 744–756. DOI: 10.1109/TVCG.2005.87 (cit. on p. 34).

[178] S. Kumar, S. Petruzza, W. Usher, and V. Pascucci. "Spatially-Aware Parallel I/O for Particle Data." In: *Proceedings of the 48th International Conference on Parallel Processing*. ICPP 2019. Association for Computing Machinery, 2019. ISBN: 9781450362955. DOI: 10.1145/3337821.3337875 (cit. on p. 25).

[179] Norman Kuring. 2005. URL: https://earthobservatory.nasa.gov/images/5432/the-gulf-stream (cit. on p. 31).

[180] W. Kutta. "Beitrag zur Näherungsweisen Integration Totaler Differentialgleichungen." In: *Zeitschrift für Mathematik und Physik.* 46 (1901), pp. 435–453 (cit. on p. 29).

[181] S. Lakshminarasimhan, N. Shah, S. Ethier, S.-H. Ku, C. S. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. "ISABELA for Effective In Situ Compression of Scientific Data." In: *Concurrency and Computation: Practice and Experience* 25.4 (2013), pp. 524–540. DOI: 10.1002/cpe.2887 (cit. on pp. 2, 20).

[182] O. D. Lampe and H. Hauser. "Interactive Visualization of Streaming Data with Kernel Density Estimation." In: *IEEE Pacific Visualization Symposium*. 2011, pp. 171–178. DOI: 10.1109/PACIFICVIS.2011.5742387 (cit. on p. 93).

[183] D. A. Lane. "UFAT - A Particle Tracer for Time-Dependent Flow Fields." In: *Proceedings Visualization '94*. 1994, pp. 257–264. DOI: 10.1109/VISUAL.1994.346311 (cit. on p. 34).

[184] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post. "Topology-Based Flow Visualization, The State of the Art." In: *Topology-based Methods in Visualization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–19. ISBN: 978-3-540-70823-0. DOI: 10.1007/978-3-540-70823-0_1 (cit. on p. 30).

[185] D. J. Lehmann and H. Theisel. "Orthographic Star Coordinates." In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2615–2624. ISSN: 1941-0506. DOI: 10.1109/TVCG.2013.182 (cit. on p. 15).

[186] A. Lež, A. Zajic, K. Matković, A. Pobitzer, M. Mayer, and H. Hauser. "Interactive Exploration and Analysis of Pathlines in Flow Data." In: *Proceedings of the 19th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2011), pp. 17–24 (cit. on pp. 35, 53).

[187] G. Li, J. Xu, T. Zhang, G. Shan, H. Shen, K. Wang, S. Liao, and Z. Lu. "Distribution-Based Particle Data Reduction for In-situ Analysis and Visualization of Large-scale N-body Cosmological Simulations." In: *IEEE Pacific Visualization Symposium*. 2020, pp. 171–180. DOI: 10.1109/PacificVis48177.2020.1186 (cit. on p. 24).

[188] S. Li, N. Marsaglia, C. Garth, J. Woodring, J. Clyne, and H. Childs. "Data Reduction Techniques for Simulation, Visualization and Data Analysis." In: *Computer Graphics Forum* 37.6 (2018), pp. 422–447. DOI: 10.1111/cgf.13336 (cit. on p. 19).

[189] F. Lindemann and T. Ropinski. "About the Influence of Illumination Models on Image Comprehension in Direct Volume Rendering." In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 1922–1931. ISSN: 1941-0506. DOI: 10.1109/TVCG.2011.161 (cit. on p. 126).

[190] P. Lindstrom. "Fixed-Rate Compressed Floating-Point Arrays." In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2674–2683. DOI: 10.1109/TVCG.2014.2346458 (cit. on p. 19).

[191] P. Lindstrom and M. Isenburg. "Fast and Efficient Compression of Floating-Point Data." In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 1245–1250. ISSN: 1941-0506. DOI: 10.1109/TVCG.2006.143 (cit. on p. 20).

[192] M. B. Liu and G. R. Liu. "Smoothed Particle Hydrodynamics (SPH): An Overview and Recent Developments." In: *Archives of Computational Methods in Engineering* 17.1 (2010), pp. 25–76. DOI: 10.1007/s11831-010-9040-7 (cit. on p. 46).

[193] S. Liu, W. Cui, Y. Wu, and M. Liu. "A Survey on Information Visualization: Recent Advances and Challenges." In: *The Visual Computer* 30.12 (2014), pp. 1373–1393. ISSN: 1432-2315. DOI: 10.1007/s00371-013-0892-3 (cit. on p. 13).

[194] S. Liu, J. A. Levine, P. T. Bremer, and V. Pascucci. "Gaussian Mixture Model Based Volume Visualization." In: *IEEE 2nd Symposium on Large Data Analysis and Visualization*. 2012, pp. 73–77. DOI: 10.1109/LDAV.2012.6378978 (cit. on p. 125).

[195] S. Liu, D. Maljovec, B. Wang, P. Bremer, and V. Pascucci. "Visualizing High-Dimensional Data: Advances in the Past Decade." In: *IEEE Transactions on Visualization and Computer Graphics* 23.3 (2017), pp. 1249–1268. ISSN: 1077-2626. DOI: 10.1109/TVCG.2016.2640960 (cit. on p. 13).

[196] 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. 3. Wiley Online Library. 2016, pp. 669–691. DOI: 10.1111/cgf.12934 (cit. on p. 7).

[197] W. E. Lorensen and H. E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm." In: *SIGGRAPH Computer Graphics* 21.4 (1987), pp. 163–169. ISSN: 0097-8930. DOI: 10.1145/37402.37422 (cit. on pp. 13, 44).

[198] E. N. Lorenz. "Deterministic Nonperiodic Flow." In: *Journal of the Atmospheric Sciences* 20.2 (1963), pp. 130–141 (cit. on pp. 1, 35, 59).

[199] L. B. Lucy. "A Numerical Approach to the Testing of the Fission Hypothesis." In: *The Astronomical Journal* 82 (1977), pp. 1013–1024. DOI: 10.1086/112164 (cit. on pp. 1, 29).

[200] J. Lukasczyk, C. Garth, M. Larsen, W. Engelke, I. Hotz, D. Rogers, J. Ahrens, and R. Maciejewski. "Cinema Darkroom: A Deferred Rendering Framework for Large-Scale Datasets." In: *IEEE 10th Symposium on Large Data Analysis and Visualization*. 2020. URL: https://arxiv.org/abs/2010.03936 (cit. on p. 12).

[201] J. Lukasczyk, E. Kinner, J. Ahrens, H. Leitte, and C. Garth. "VOIDGA: A View-Approximation Oriented Image Database Generation Approach." In: *IEEE 8th Symposium on Large Data Analysis and Visualization*. 2018, pp. 12–22. DOI: 10.1109/LDAV.2018.8739204 (cit. on pp. 12, 134).

[202] C. Lundström, P. Ljung, A. Persson, and A. Ynnerman. "Uncertainty Visualization in Medical Volume Rendering Using Probabilistic Animation." In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007), pp. 1648–1655 (cit. on p. 125).

[203] C. Lundström, P. Ljung, and A. Ynnerman. "Local Histograms for Design of Transfer Functions in Direct Volume Rendering." In: *IEEE Transactions on Visualization and Computer Graphics* 12.6 (2006), pp. 1570–1579. ISSN: 1077-2626. DOI: 10.1109/TVCG.2006.100 (cit. on p. 24).

[204] P. C. Mahalanobis. "On the Generalized Distance in Statistics." In: National Institute of Science of India. 1936 (cit. on p. 103).

[205] O. Mallo, R. Peikert, C. Sigg, and F. Sadlo. "Illuminated Lines Revisited." In: *IEEE Visualization*. 2005, pp. 19–26. DOI: 10.1109/VISUAL.2005.1532772 (cit. on pp. 8, 34).

[206] A. R. Martin and M. O. Ward. "High Dimensional Brushing for Interactive Exploration of Multivariate Data." In: *Proceedings of the 6th Conference on Visualization*. VIS '95. IEEE Computer Society, 1995, p. 271. ISBN: 0818671874 (cit. on p. 15).

[207] Ivan Marusic and Susan Broomhall. "Leonardo da Vinci and Fluid Mechanics." In: *Annual Review of Fluid Mechanics* 53.1 (2021), pp. 1–25. DOI: 10.1146/annurev-fluid-022620-122816 (cit. on p. 28).

[208] M. Mathur, G. Haller, T. Peacock, J. E. Ruppert-Felsot, and H. L. Swinney. "Uncovering the Lagrangian Skeleton of Turbulence." In: *Physical Review Letters* 98.14 (2007), p. 144502. DOI: 10.1103/PhysRevLett.98.144502 (cit. on p. 53).

[209] N. Max. "Optical Models for Direct Volume Rendering." In: *IEEE Transactions on Visualization and Computer Graphics* 1.2 (1995), pp. 99–108. ISSN: 1941-0506. DOI: 10.1109/2945.468400 (cit. on pp. 10, 126).

[210] A. Mayorga and M. Gleicher. "Splatterplots: Overcoming Overdraw in Scatter Plots." In: *IEEE Transactions on Visualization and Computer Graphics* 19.9 (2013), pp. 1526–1538. ISSN: 1077-2626. DOI: 10.1109/TVCG.2013.65 (cit. on pp. 16, 103).

[211] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. "Over Two Decades of Integration-Based, Geometric Flow Visualization." In: *Computer Graphics Forum*. Vol. 29. 6. Wiley Online Library. 2010, pp. 1807–1829. DOI: 10.1111/j.1467-8659.2010.01650.x (cit. on p. 33).

[212] C. J. Mecklin and D. J. Mundfrom. "A Monte Carlo Comparison of the Type I and Type II Error Rates of Tests of Multivariate Normality." In: *Journal of Statistical Computation and Simulation* 75.2 (2005), pp. 93–107. DOI: 10.1080/0094965042000193233 (cit. on p. 22).

[213] J. J. Miller and E. J. Wegman. "Construction of Line Densities for Parallel Coordinate Plots." In: *Computing and Graphics in Statistics* 36 (1991), pp. 107–123 (cit. on p. 100).

[214] V. Molchanov, A. Fofonov, and L. Linsen. "Continuous Representation of Projected Attribute Spaces of Multifields over Any Spatial Sampling." In: *Computer Graphics Forum* 32.3pt3 (2013), pp. 301–310. DOI: 10.1111/cgf.12117 (cit. on p. 17).

[215] V. Molchanov, A. Fofonov, S. Rosswog, P. Rosenthal, and L. Linsen. "SmoothViz: An Interactive Visual Analysis System for SPH Data." In: *Proceedings of the 8th International SPHERIC Workshop*. 2013, pp. 350–356 (cit. on pp. 15, 35).

[216] J. J. Monaghan. "Why Particle Methods Work." In: *SIAM Journal on Scientific and Statistical Computing* 3.4 (1982), pp. 422–433. DOI: 10.1137/0903027 (cit. on p. 84).

[217] J. J. Monaghan. "Smoothed Particle Hydrodynamics." In: *Annual review of astronomy and astrophysics* 30.1 (1992), pp. 543–574. DOI: 10.1146/annurev.aa.30.090192.002551 (cit. on p. 76).

[218] J. J. Monaghan. "Simulating Free Surface Flows with SPH." In: *Journal of Computational Physics* 110.2 (1994), pp. 399 –406. ISSN: 0021-9991. DOI: 10.1006/jcph.1994.1034 (cit. on p. 29).

[219] G. E. Moore. "Cramming More Components onto Integrated Circuits." In: *Electronics Magazine* (1965) (cit. on p. 2).

[220] N. Morrical, W. Usher, I. Wald, and V. Pascucci. "Efficient Space Skipping and Adaptive Sampling of Unstructured Volumes Using Hardware Accelerated Ray Tracing." In: *IEEE Visualization*. 2019, pp. 256–260. DOI: 10.1109/VISUAL.2019.8933539 (cit. on p. 10).

[221] K. Mueller, T. Moller, and R. Crawlis. "Splatting Without the Blur." In: *IEEE Visualization*. 1999, pp. 363–544. DOI: 10.1109/VISUAL.1999.809909 (cit. on p. 9).

[222] K. Mueller, N. Shareef, J. Huang, and R. Crawfis. "IBR-Assisted Volume Rendering." In: *Late Breaking Hot Topics, Proceedings of IEEE Visualization*. 1999, pp. 5–8 (cit. on p. 12).

[223] C. Münstermann, S. Krumpen, R. Klein, and C. Peters. "Moment-Based Order-Independent Transparency." In: *Proc. ACM Comput. Graph. Interact. Tech.* 1.1 (2018), 7:1–7:20. ISSN: 2577-6193. DOI: 10.1145/3203206 (cit. on pp. 8, 94, 99).

[224] E. A. Nadaraya. "On Estimating Regression." In: *Theory of Probability & Its Applications* 9.1 (1964), pp. 141–142 (cit. on p. 5).

[225] D. Nelson et al. "The Illustris Simulation: Public Data Release." In: *Astronomy and Computing* 13 (2015), pp. 12–37. DOI: 10.1016/j.ascom.2015.09.003 (cit. on pp. 3, 105, 135).

[226] N. Neophytou and K. Mueller. "Space-Time Points: 4D Splatting on Efficient Grids." In: *Symposium on Volume Visualization and Graphics, IEEE / ACM SIGGRAPH*. 2002, pp. 97–106. DOI: 10.1109/SWG.2002.1226515 (cit. on p. 10).

[227] N. Neophytou, K. Mueller, K. T. McDonnell, W. Hong, X. Guan, H. Qin, and A. Kaufman. "GPU-Accelerated Volume Splatting with Elliptical RBFs." In: *Proceedings of the Eighth Joint Eurographics / IEEE VGTC Conference on Visualization*. 2006, pp. 13–20. ISBN: 3-905673-31-2. DOI: 10.2312/VisSym/EuroVis06/013-020 (cit. on p. 10).

[228] J. Nickolls, I. Buck, M. Garland, and K. Skadron. "Scalable Parallel Programming with CUDA." In: *Queue* 6.2 (2008), pp. 40–53. ISSN: 1542-7730. DOI: 10.1145/1365490.1365500 (cit. on p. 43).

[229] M. Novotny and H. Hauser. "Outlier-Preserving Focus+Context Visualization in Parallel Coordinates." In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 893–900. ISSN: 1077-2626. DOI: 10.1109/TVCG.2006.170 (cit. on pp. 16, 93, 103).

[230] J. Onderik, M. Chládek, and R. Ďurikovič. "SPH with Small Scale Details and Improved Surface Reconstruction." In: *Proceedings of the 27th Spring Conference on Computer Graphics*. SCCG '11. 2013, pp. 29–36. ISBN: 978-1-4503-1978-2. DOI: 10.1145/2461217.2461224 (cit. on p. 13).

[231] J. Orthmann, M. Keller, and A. Kolb. "Topology-Caching for Dynamic Particle Volume Raycasting." In: *Vision, Modeling, and Visualization*. The Eurographics Association, 2010. ISBN: 978-3-905673-79-1. DOI: 10.2312/PE/VMV/VMV10/147-154 (cit. on p. 11).

[232] R. S. Allendes Osorio and K. W. Brodlie. "Uncertain Flow Visualization using LIC." In: *Theory and Practice of Computer Graphics*. The Eurographics Association, 2009. ISBN: 978-3-905673-71-5. DOI: 10.2312/LocalChapterEvents/TPCG/TPCG09/215-222 (cit. on p. 36).

[233] M. Otto, T. Germer, H.-C. Hege, and H. Theisel. "Uncertain 2D Vector Field Topology." In: *Computer Graphics Forum* 29.2 (2010), pp. 347–356. DOI: 10.1111/j.1467-8659.2009.01604.x (cit. on p. 36).

[234] M. Otto, T. Germer, and H. Theisel. "Closed Stream Lines in Uncertain Vector Fields." In: *Proceedings of the 27th Spring Conference on Computer Graphics*. Association for Computing Machinery, 2011, pp. 87–94. ISBN: 9781450319782. DOI: 10.1145/2461217.2461235 (cit. on p. 36).

[235] M. Otto, T. Germer, and H. Theisel. "Uncertain Topology of 3D Vector Fields." In: *IEEE Pacific Visualization Symposium*. 2011, pp. 67–74. DOI: 10.1109/PACIFICVIS.2011.5742374 (cit. on p. 36).

[236] M. Otto and H. Theisel. "Vortex Analysis in Uncertain Vector Fields." In: *Computer Graphics Forum* 31 (2012), pp. 1035–1044. DOI: 10.1111/j.1467-8659.2012.03096.x (cit. on p. 36).

[237] V. M. Panaretos and Y. Zemel. "Statistical Aspects of Wasserstein Distances." In: *Annual Review of Statistics and Its Application* 6.1 (2019), pp. 405–431. DOI: 10.1146/annurev-statistics-030718-104938 (cit. on pp. 81, 102).

[238] A. T. Pang, C. M. Wittenbrink, and S. K. Lodha. "Approaches to Uncertainty Visualization." In: *The Visual Computer* 13.8 (1997), pp. 370–390. ISSN: 1432-2315. DOI: 10.1007/s003710050111 (cit. on p. 35).

[239] R. J. Pattenden, S. R. Turnock, and X. Zhang. "Measurements of the Flow over a Low-Aspect-Ratio Cylinder Mounted on a Ground Plane." In: *Experiments in Fluids* 39.1 (2005), pp. 10–21. ISSN: 1432-1114. DOI: 10.1007/s00348-005-0949-9 (cit. on p. 50).

[240] K. Pearson. "Contributions to the Mathematical Theory of Evolution." In: *Philosophical Transactions of the Royal Society of London. A* 185 (1894), pp. 71–110. ISSN: 02643820. URL: http://www.jstor.org/stable/90667 (cit. on p. 21).

[241] A. E. Perry and M. S. Chong. "Topology of Flow Patterns in Vortex Motions and Turbulence." In: *Applied Scientific Research* 53.3-4 (1994), pp. 357–374. DOI: 10.1007/BF00849110 (cit. on p. 30).

[242] C. Peters, J. Klein, M. B. Hullin, and R. Klein. "Solving Trigonometric Moment Problems for Fast Transient Imaging." In: *ACM Transactions on Graphics* 34.6 (2015). ISSN: 0730-0301. DOI: 10.1145/2816795.2818103 (cit. on p. 118).

[243] C. Peters, S. Merzbach, J. Hanika, and C. Dachsbacher. "Spectral Rendering with the Bounded MESE and sRGB Data." In: *Workshop on Material Appearance Modeling*. The Eurographics Association, 2019. ISBN: 978-3-03868-080-2. DOI: 10.2312/mam.20191304 (cit. on p. 120).

[244] C. Peters, S. Merzbach, J. Hanika, and C. Dachsbacher. "Using Moments to Represent Bounded Signals for Spectral Rendering." In: *ACM Transactions on Graphics* 38.4 (2019), 136:1–136:14. ISSN: 0730-0301. DOI: 10.1145/3306346.3322964 (cit. on pp. 115, 117, 118, 120, 121).

[245] C. Petz, K. Pöthkow, and H.-C. Hege. "Probabilistic Local Features in Uncertain Vector Fields with Spatial Correlation." In: *Computer Graphics Forum* 31 (2012), pp. 1045–1054. DOI: 10.1111/j.1467-8659.2012.03097.x (cit. on p. 36).

[246] M. Piochowiak, T. Rapp, and C. Dachsbacher. "Stochastic Volume Rendering of Multi-Phase SPH Data." In: *Computer Graphics Forum* 40.1 (2021), pp. 97–109. DOI: 10.1111/cgf.14121 (cit. on pp. 11–13, 18, 135).

[247] A. Pobitzer, R. Peikert, R. Fuchs, B. Schindler, A. Kuhn, H. Theisel, K. Matković, and H. Hauser. "The State of the Art in Topology-Based Visualization of Unsteady Flow." In: *Computer Graphics Forum* 30.6 (2011), pp. 1789–1811. DOI: 10.1111/j.1467-8659.2011.01901.x (cit. on p. 31).

[248] S. Popinet. "Free Computational Fluid Dynamics." In: *ClusterWorld* 2.6 (2004). URL: http://gfs.sf.net/ (cit. on p. 67).

[249] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. "The State of the Art in Flow Visualisation: Feature Extraction and Tracking." In: *Computer Graphics Forum* 22.4 (2003), pp. 775–792. DOI: 10.1111/j.1467-8659.2003.00723.x (cit. on p. 31).

[250] K. Pöthkow and H.-C. Hege. "Nonparametric Models for Uncertainty Visualization." In: *Computer Graphics Forum* 32.3pt2 (2013), pp. 131–140. DOI: 10.1111/cgf.12100 (cit. on p. 36).

[251] K. Potter, P. Rosen, and C. R. Johnson. "From Quantification to Visualization: A Taxonomy of Uncertainty Visualization Approaches." In: *Uncertainty Quantification in Scientific Computing*. 2012, pp. 226–249. ISBN: 978-3-642-32677-6. DOI: 10.1007/978-3-642-32677-6_15 (cit. on pp. 35, 136).

[252] T. J. Purcell, C. Donner, M. Cammarano, H. W. Jensen, and P. Hanrahan. "Photon Mapping on Programmable Graphics Hardware." In: *ACM SIGGRAPH 2005 Courses*. SIGGRAPH '05. New York, NY, USA: ACM, 2005. DOI: 10.1145/1198555.1198797 (cit. on p. 42).

[253] T. Rapp and C. Dachsbacher. "Visualizing Transport and Mixing in Particle-based Fluid Flows." In: *Vision, Modeling and Visualization*. 2019. ISBN: 978-3-03868-098-7. DOI: 10.2312/vmv.20191330 (cit. on pp. 3, 4).

[254] T. Rapp and C. Dachsbacher. "Uncertain Transport in Unsteady Flows." In: *Proceedings of IEEE Visualization*. 2020, pp. 16–20. DOI: 10.1109/VIS47514.2020.00010 (cit. on p. 4).

[255] T. Rapp, C. Peters, and C. Dachsbacher. "Image-based Visualization of Large Volumetric Data Using Moments." Submitted to IEEE Transactions on Visualization and Computer Graphics. 2020 (cit. on p. 4).

[256] T. Rapp, C. Peters, and C. Dachsbacher. "Void-and-Cluster Sampling of Large Scattered Data and Trajectories." In: *IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE Visualization 2019)* 26.1 (2020), pp. 780–789. ISSN: 077-2626. DOI: 10.1109/TVCG.2019.2934335 (cit. on p. 4).

[257] T. Rapp, C. Peters, and C. Dachsbacher. "Visual Analysis of Large Multivariate Scattered Data using Clustering and Probabilistic Summaries." In: *IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE Visualization 2020)* 27.2 (2021), pp. 1580–1590. ISSN: 1941-0506. DOI: 10.1109/TVCG.2020.3030379 (cit. on p. 4).

[258] P. Rautek, M. Mlejnek, J. Beyer, J. Troidl, H. Pfister, T. Theußl, and M. Hadwiger. "Objective Observer-Relative Flow Visualization in Curved Spaces for Unsteady 2D Geophysical Flows." In: *IEEE Transactions on Visualization and Computer Graphics* (2020). ISSN: 1941-0506. DOI: 10.1109/TVCG.2020.3030454 (cit. on p. 31).

[259] N. M. Razali and Y. B. Wah. "Power Comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling Tests." In: *Journal of Statistical Modeling and Analytics* 2.1 (2011), pp. 21–33 (cit. on p. 22).

[260] F. Reichl, M. G. Chajdas, J. Schneider, and R. Westermann. "Interactive Rendering of Giga-Particle Fluid Simulations." In: *Proceedings of High Performance Graphics*. HPG '14. Eurographics Association, 2014, pp. 105–116 (cit. on p. 11).

[261] F. Reichl, M. Treib, and R. Westermann. "Visualization of Big SPH Simulations via Compressed Octree Grids." In: *IEEE International Conference on Big Data*. 2013, pp. 71–78. DOI: 10.1109/BigData.2013.6691717 (cit. on pp. 11, 24).

[262] G. Reina. "Visualization of Uncorrelated Point Data." PhD thesis. University of Stuttgart, 2008. DOI: 10.18419/opus-2649 (cit. on p. 8).

[263] S. Reinhardt, M. Huber, O. Dumitrescu, M. Krone, B. Eberhardt, and D. Weiskopf. "Visual Debugging of SPH Simulations." In: *21st International Conference Information Visualisation*. 2017, pp. 117–126. DOI: 10.1109/iV.2017.20 (cit. on pp. 16, 35).

[264] J. C. Roberts. "State of the Art: Coordinated Multiple Views in Exploratory Visualization." In: *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)*. 2007, pp. 61–71. DOI: 10.1109/CMV.2007.20 (cit. on p. 15).

[265] I. B. Rojo and T. Günther. "Vector Field Topology of Time-Dependent Flows in a Steady Reference Frame." In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 280–290. DOI: 10.1109/TVCG.2019.2934375 (cit. on pp. 31, 67).

[266] F. Sadlo and R. Peikert. "Efficient Visualization of Lagrangian Coherent Structures by Filtered AMR Ridge Extraction." In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007), pp. 1456–1463. DOI: 10.1109/TVCG.2007.70554 (cit. on p. 33).

[267] F. Sadlo and R. Peikert. "Visualizing Lagrangian Coherent Structures and Comparison to Vector Field Topology." In: *Topology-Based Methods in Visualization II* (2009), pp. 15–29. DOI: 10.1007/978-3-540-88606-8_2 (cit. on p. 33).

[268] F. Sadlo, A. Rigazzi, and R. Peikert. "Time-Dependent Visualization of Lagrangian Coherent Structures by Grid Advection." In: *Topological Methods in Data Analysis and Visualization* (2011), pp. 151–165. DOI: 10.1007/978-3-642-15014-2_13 (cit. on p. 32).

[269] F. Sadlo and D. Weiskopf. "Time-Dependent 2-D Vector Field Topology: An Approach Inspired by Lagrangian Coherent Structures." In: *Computer Graphics Forum* 29.1 (2010), pp. 88–100. DOI: 10.1111/j.1467-8659.2009.01546.x (cit. on p. 31).

[270] E. Sakhaee and A. Entezari. "A Statistical Direct Volume Rendering Framework for Visualization of Uncertain Data." In: *IEEE Transactions on Visualization and Computer Graphics* 23.12 (2017), pp. 2509–2520. ISSN: 1077-2626. DOI: 10.1109/TVCG.2016.2637333 (cit. on pp. 99, 125).

[271] M. Salvi, J. Montgomery, and A. Lefohn. "Adaptive Transparency." In: *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*. HPG. 2011, pp. 119–126. DOI: 10.1145/2018323.2018342 (cit. on p. 8).

[272] T. Salzbrunn, C. Garth, G. Scheuermann, and J. Meyer. "Pathline Predicates and Unsteady Flow Structures." In: *The Visual Computer* 24.12 (2008), pp. 1039–1051. ISSN: 1432-2315. DOI: 10.1007/s00371-007-0204-x (cit. on p. 35).

[273] S. Sane, R. Bujack, C. Garth, and H. Childs. "A Survey of Seed Placement and Streamline Selection Techniques." In: *Computer Graphics Forum* (2020). ISSN: 1467-8659. DOI: 10.1111/cgf.14036 (cit. on p. 33).

[274] S. Sanikommu, H. Toye, P. Zhan, S. Langodan, G. Krokos, O. Knio, and I. Hoteit. "Impact of Atmospheric and Model Physics Perturbations On a High-Resolution Ensemble Data Assimilation System of the Red Sea." In: *Journal of Geophysical Research-Oceans, Revision submitted* (2020). eprint: arXiv:2002.01825 (cit. on p. 63).

[275] F. Sauer, J. Xie, and K. L. Ma. "A Combined Eulerian-Lagrangian Data Representation for Large-Scale Applications." In: *IEEE Transactions on Visualization and Computer Graphics* 23.10 (2017), pp. 2248–2261. ISSN: 1077-2626. DOI: 10.1109/TVCG.2016.2620975 (cit. on p. 136).

[276] F. Sauer, H. Yu, and K.-L. Ma. "An Analytical Framework for Particle and Volume Data of Large-Scale Combustion Simulations." In: *Proceedings of the 8th International Workshop on Ultrascale Visualization*. UltraVis '13. Denver, Colorado: ACM, 2013, 1:1–1:8. ISBN: 978-1-4503-2500-4. DOI: 10.1145/2535571.2535590 (cit. on p. 136).

[277] K. Schatz, C. Müller, M. Krone, J. Schneider, G. Reina, and T. Ertl. "Interactive Visual Exploration of a Trillion Particles." In: *IEEE 6th Symposium on Large Data Analysis and Visualization*. 2016, pp. 56–64. DOI: 10.1109/LDAV.2016.7874310 (cit. on p. 8).

[278] B. Schindler, R. Peikert, R. Fuchs, and H. Theisel. "Ridge Concepts for the Visualization of Lagrangian Coherent Structures." In: *Topological Methods in Data Analysis and Visualization II* (2012), pp. 221–235. DOI: 10.1007/978-3-642-23175-9_15 (cit. on p. 33).

[279] D. Schneider, J. Fuhrmann, W. Reich, and G. Scheuermann. "A Variance Based FTLE-Like Method for Unsteady Uncertain Vector Fields." In: *Topological Methods in Data Analysis and Visualization II: Theory, Algorithms, and Applications*. Springer Berlin Heidelberg, 2012, pp. 255–268. ISBN: 978-3-642-23175-9. DOI: 10.1007/978-3-642-23175-9_17 (cit. on pp. 36, 59).

[280] G. Schwarz. "Estimating the Dimension of a Model." In: *The Annals of Statistics* 6.2 (1978), pp. 461–464 (cit. on pp. 22, 96).

[281] S. C. Shadden. "Lagrangian Coherent Structures." In: *Transport and Mixing in Laminar Flows*. John Wiley & Sons, Ltd, 2011. Chap. 3, pp. 59–89. ISBN: 9783527639748. DOI: 10.1002/9783527639748.ch3 (cit. on p. 31).

[282] S. C. Shadden, F. Lekien, and J. E. Marsden. "Definition and Properties of Lagrangian Coherent Structures from Finite-Time Lyapunov Exponents in Two-Dimensional Aperiodic Flows." In: *Physica D: Nonlinear Phenomena* 212.3 (2005), pp. 271–304. ISSN: 0167-2789. DOI: 10.1016/j.physd.2005.10.007 (cit. on pp. 33, 44).

[283] J. Shade, S. Gortler, L.-W. He, and R. Szeliski. "Layered Depth Images." In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '98. ACM, 1998, pp. 231–242. ISBN: 0-89791-999-8. DOI: 10.1145/280814.280882 (cit. on p. 12).

[284] S. S. Shapiro and M. B. Wilk. "An Analysis of Variance Test for Normality (Complete Samples)." In: *Biometrika* 52.3/4 (1965), pp. 591–611. ISSN: 00063444. DOI: 10.2307/2333709 (cit. on p. 22).

[285] N. Shareef, T.-Y. Lee, H.-W. Shen, and K. Mueller. "An Image-Based Modelling Approach To GPU-based Unstructured Grid Volume Rendering." In: *Volume Graphics*. 2006. ISBN: 3-905673-41-X. DOI: 10.2312/VG/VG06/031-038 (cit. on pp. 12, 115).

[286] D. Shepard. "A Two-Dimensional Interpolation Function for Irregularly-Spaced Data." In: *Proceedings of the 1968 23rd ACM National Conference*. ACM. 1968, pp. 517–524 (cit. on p. 5).

[287] 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 Berlin Heidelberg, 2009, pp. 75–88. ISBN: 978-3-540-88606-8. DOI: 10.1007/978-3-540-88606-8_6 (cit. on pp. 35, 53).

[288] L. Shi, L. Zhang, W. Cao, and G. Chen. "Analysis Enhanced Particle-based Flow Visualization." In: *Electronic Imaging* 2017.1 (2017), pp. 12–21. DOI: 10.2352/ISSN.2470-1173.2017.1.VDA-385 (cit. on p. 33).

[289] B. Shneiderman. "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations." In: *IEEE Symposium on Visual Languages*. 1996, pp. 336–343. DOI: 10.1109/VL.1996.545307 (cit. on p. 17).

[290] R. Sicat, J. Krüger, T. Möller, and M. Hadwiger. "Sparse PDF Volumes for Consistent Multi-Resolution Volume Rendering." In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2417–2426. ISSN: 1077-2626. DOI: 10.1109/TVCG.2014.2346324 (cit. on p. 24).

[291] C. Sigg, T. Weyrich, M. Botsch, and M. Gross. "GPU-Based Ray-Casting of Quadratic Surfaces." In: *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*. SPBG'06. Eurographics Association, 2006, pp. 59–65. ISBN: 3905673320 (cit. on p. 8).

[292] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Routledge, 2018 (cit. on p. 93).

[293] Samuel W. Skillman, Michael S. Warren, Matthew J. Turk, Risa H. Wechsler, Daniel E. Holz, and P. M. Sutter. *Dark Sky Simulations: Early Data Release*. 2014. eprint: arXiv:1407.2600 (cit. on pp. 2, 89).

[294] B. Solenthaler, J. Schläfli, and R. Pajarola. "A Unified Particle Model for Fluid - Solid Interactions." In: *Computer Animation and Virtual Worlds* 18.1 (2007), pp. 69–82. ISSN: 1546-427X (cit. on p. 13).

[295] V. Springel. "E Pur Si Muove: Galilean-Invariant Cosmological Hydrodynamical Simulations on a Moving Mesh." In: *Monthly Notices of the Royal Astronomical Society* 401 (2010), pp. 791–851. DOI: 10.1111/j.1365-2966.2009.15715.x (cit. on pp. 105, 136).

[296] J. Staib, S. Grottel, and S. Gumhold. "Visualization of Particle-based Data with Transparency and Ambient Occlusion." In: *Computer Graphics Forum* 34.3 (2015), pp. 151–160. DOI: 10.1111/cgf.12627 (cit. on p. 8).

[297] K. Stockinger, J. Shalf, K. Wu, and E. W. Bethel. "Query-Driven Visualization of Large Data Sets." In: *IEEE Visualization*. 2005, pp. 167–174. DOI: 10.1109/VISUAL.2005.1532792 (cit. on p. 15).

[298] H. A. Sturges. "The Choice of a Class Interval." In: *Journal of the American Statistical Association* 21.153 (1926), pp. 65–66. DOI: 10.1080/01621459.1926.10502161 (cit. on p. 21).

[299] Y. Su, G. Agrawal, J. Woodring, K. Myers, J. Wendelberger, and J. Ahrens. "Effective and Efficient Data Sampling Using Bitmap Indices." In: *Cluster Computing* 17.4 (2014), pp. 1081–1100. ISSN: 1573-7543. DOI: 10.1007/s10586-014-0360-5 (cit. on pp. 25, 73).

[300] P. N. Sun, A. Colagrossi, S. Marrone, and A. M. Zhang. "Detection of Lagrangian Coherent Structures in the SPH framework." In: *Computer Methods in Applied Mechanics and Engineering* 305 (2016), pp. 849 –868. ISSN: 0045-7825. DOI: 10.1016/j.cma.2016.03.027 (cit. on pp. 33, 41).

[301] J. E. Swan, K. Mueller, T. Moller, N. Shareel, R. Crawfis, and R. Yagel. "An Anti-Aliasing Technique for Splatting." In: *IEEE Visualization*. 1997, pp. 197–204. DOI: 10.1109/VISUAL.1997.663882 (cit. on p. 9).

[302] L. Szécsi and D. Illés. "Real-Time Metaball Ray Casting with Fragment Lists." In: *Eurographics (Short Papers)*. 2012. DOI: 10.2312/conf/EG2012/short/093-096 (cit. on p. 8).

[303] S. Taneda. "Visual Study of Unsteady Separated Flows Around Bodies." In: *Progress in Aerospace Sciences* 17 (1977), pp. 287–348. URL: https://ui.adsabs.harvard.edu/abs/1977PrAeS..17..287T (cit. on pp. 47, 48).

[304] J. J. Thomas and K. A. Cook. "A Visual Analytics Agenda." In: *IEEE Computer Graphics and Applications* 26.01 (2006), pp. 10–13. ISSN: 0272-1716. DOI: 10.1109/MCG.2006.5 (cit. on p. 17).

[305] D. Thompson, J. A. Levine, J. C. Bennett, P. T. Bremer, A. Gyulassy, V. Pascucci, and P. P. Pébay. "Analysis of Large-Scale Scalar Data Using Hixels." In: *IEEE 1st Symposium on Large Data Analysis and Visualization*. 2011, pp. 23–30. DOI: 10.1109/LDAV.2011.6092313 (cit. on p. 23).

[306] A. Tikhonova, C. D. Correa, and K.-L. Ma. "An Exploratory Technique for Coherent Visualization of Time-varying Volume Data." In: *Computer Graphics Forum* 29.3 (2010), pp. 783–792. DOI: 10.1111/j.1467-8659.2009.01690.x (cit. on p. 12).

[307] A. Tikhonova, C. D. Correa, and K.-L. Ma. "Explorable Images for Visualizing Volume Data." In: *IEEE Pacific Visualization Symposium*. 2010, pp. 177–184. DOI: 10.1109/PACIFICVIS.2010.5429595 (cit. on p. 12).

[308] A. Tikhonova, C. D. Correa, and K.-L. Ma. "Visualization by Proxy: A Novel Framework for Deferred Interaction with Volume Data." In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1551–1559. DOI: 10.1109/TVCG.2010.215 (cit. on p. 12).

[309] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997 (cit. on p. 98).

[310] J. W. Tukey. "The Future of Data Analysis." In: *Annals of Mathematical Statistics* 33.1 (1962), pp. 1–67. DOI: 10.1214/aoms/1177704711 (cit. on p. 15).

[311] J. W. Tukey. *Exploratory Data Analysis*. Addison–Wesley Publishing Company, Inc., 1977 (cit. on p. 13).

[312] M. Uffinger, F. Sadlo, and T. Ertl. "A Time-Dependent Vector Field Topology Based on Streak Surfaces." In: *IEEE Transactions on Visualization and Computer Graphics* 19.3 (2013), pp. 379–392. ISSN: 1941-0506. DOI: 10.1109/TVCG.2012.131 (cit. on p. 31).

[313] R. A. Ulichney. "Dithering with Blue Noise." In: *Proceedings of the IEEE* 76.1 (1988), pp. 56–79. ISSN: 0018-9219. DOI: 10.1109/5.3288 (cit. on p. 26).

[314] R. A. Ulichney. "Void-and-Cluster Method for Dither Array Generation." In: *Human Vision, Visual Processing, and Digital Display IV*. Vol. 1913. 1993, pp. 332–343. DOI: 10.1117/12.152707 (cit. on pp. 4, 73, 76).

[315] I. Wald, A. Knoll, G. P. Johnson, W. Usher, V. Pascucci, and M. E. Papka. "CPU Ray Tracing Large Particle Data with Balanced P-k-d Trees." In: *IEEE Visualization*. 2015, pp. 57–64. DOI: 10.1109/SciVis.2015.7429492 (cit. on p. 8).

[316] J. Wang, S. Hazarika, C. Li, and H. Shen. "Visualization and Visual Analysis of Ensemble Data: A Survey." In: *IEEE Transactions on Visualization and Computer Graphics* 25.9 (2019), pp. 2853–2872. ISSN: 2160-9306. DOI: 10.1109/TVCG.2018.2853721 (cit. on pp. 35, 136).

[317] K. C. Wang, Kewei Lu, T. H. Wei, N. Shareef, and H. W. Shen. "Statistical Visualization and Analysis of Large Data Using a Value-Based Spatial Distribution." In: *IEEE Pacific Visualization Symposium*. 2017, pp. 161–170. DOI: 10.1109/PACIFICVIS.2017.8031590 (cit. on p. 23).

[318] K.-C. Wang, T.-H. Wei, N. Shareef, and H.-W. Shen. "Ray-Based Exploration of Large Time-Varying Volume Data Using Per-Ray Proxy Distributions." In: *IEEE Transactions on Visualization and Computer Graphics* 26.11 (2020), pp. 3299–3313. ISSN: 1941-0506. DOI: 10.1109/TVCG.2019.2920130 (cit. on pp. 13, 115).

[319] K. Wang, N. Shareef, and H. Shen. "Image and Distribution Based Volume Rendering for Large Data Sets." In: *IEEE Pacific Visualization Symposium*. 2018, pp. 26–35. DOI: 10.1109/PacificVis.2018.00013 (cit. on pp. 13, 115, 127, 134).

[320] K. Wang, J. Xu, J. Woodring, and H. Shen. "Statistical Super Resolution for Data Analysis and Visualization of Large Scale Cosmological Simulations." In: *IEEE Pacific Visualization Symposium*. 2019, pp. 303–312. DOI: 10.1109/PacificVis.2019.00043 (cit. on p. 24).

[321] W. Wang, W. Wang, and S. Li. "From Numerics to Combinatorics: A Survey of Topological Methods for Vector Field Visualization." In: *Journal of Visualization* 19.4 (2016), pp. 727–752. DOI: 10.1007/s12650-016-0348-8 (cit. on p. 31).

[322] Y. Wang, W. Chen, J. Zhang, T. Dong, G. Shan, and X. Chi. "Efficient Volume Exploration Using the Gaussian Mixture Model." In: *IEEE Transactions on Visualization and Computer Graphics* 17.11 (2011), pp. 1560–1573. ISSN: 1077-2626. DOI: 10.1109/TVCG.2011.97 (cit. on p. 24).

[323] G. S. Watson. "Smooth Regression Analysis." In: *Sankhyā: The Indian Journal of Statistics, Series A* (1964), pp. 359–372 (cit. on p. 5).

[324] G. H. Weber and H. Hauser. "Interactive Visual Exploration and Analysis." In: *Scientific Visualization*. Springer London, 2014, pp. 161–173. ISBN: 978-1-4471-6497-5. DOI: 10.1007/978-1-4471-6497-5_15 (cit. on pp. 2, 17).

[325] T. Wei, S. Dutta, and H. Shen. "Information Guided Data Sampling and Recovery Using Bitmap Indexing." In: *IEEE Pacific Visualization Symposium*. 2018, pp. 56–65. DOI: 10.1109/PacificVis.2018.00016 (cit. on pp. 25, 73, 77).

[326] M. Weiler, R. Botchen, S. Stegmaier, T. Ertl, J. Huang, Y. Jang, D. S. Ebert, and K. P. Gaither. "Hardware-Assisted Feature Analysis and Visualization of Procedurally Encoded Multifield Volumetric Data." In: *IEEE Computer Graphics and Applications* 25.5 (2005), pp. 72–81. ISSN: 0272-1716. DOI: 10.1109/MCG.2005.106 (cit. on p. 10).

[327] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. "Boundary Switch Connectors for Topological Visualization of Complex 3D Vector Fields." In: *Proceedings of the Sixth Joint Eurographics - IEEE TCVG Conference on Visualization*. VISSYM'04. Eurographics Association, 2004, pp. 183–192. ISBN: 390567307X (cit. on p. 30).

[328] L. Westover. "Interactive Volume Rendering." In: *Proceedings of the 1989 Chapel Hill Workshop on Volume Visualization*. ACM. 1989, pp. 9–16 (cit. on p. 9).

[329] L. Westover. "Footprint Evaluation for Volume Rendering." In: *SIGGRAPH Computer Graphics* 24.4 (1990), pp. 367–376. ISSN: 0097-8930. DOI: 10.1145/97880.97919 (cit. on p. 9).

[330] S. S. Wilks. *Contributions to Probability and Statistics*. Stanford University Press Stanford, 1960 (cit. on p. 62).

[331] S. S. Wilks. *Collected Papers; Contributions to Mathematical Statistics*. Wiley, 1967 (cit. on p. 62).

[332] L. Williams. "Casting Curved Shadows on Curved Surfaces." In: *SIGGRAPH Computer Graphics* 12.3 (1978), pp. 270–274. ISSN: 0097-8930. DOI: 10.1145/965139.807402 (cit. on p. 126).

[333] I. H. Witten, R. M. Neal, and J. G. Cleary. "Arithmetic Coding for Data Compression." In: *Communications of the ACM* 30.6 (1987), pp. 520–540. ISSN: 0001-0782. DOI: 10.1145/214762.214771 (cit. on pp. 19, 125).

[334] J. Woodring, J. P. Ahrens, J. Patchett, C. Tauxe, and D. H. Rogers. "High-Dimensional Scientific Data Exploration via Cinema." In: *IEEE Workshop on Data Systems for Interactive Analysis (DSIA)*. 2017, pp. 1–5. DOI: 10.1109/DSIA.2017.8339086 (cit. on p. 12).

[335] 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." In: *Computer Graphics Forum* 30.3 (2011), pp. 1151–1160. DOI: 10.1111/j.1467-8659.2011.01964.x (cit. on pp. 25, 73, 83).

[336] A. Yang, H. Mukka, F. Hesaaraki, and M. Burtscher. "MPC: A Massively Parallel Compression Algorithm for Scientific Data." In: *IEEE International Conference on Cluster Computing*. 2015, pp. 381–389. DOI: 10.1109/CLUSTER.2015.59 (cit. on p. 20).

[337] Y. Ye, R. Miller, and K.-L. Ma. "In Situ Pathtube Visualization with Explorable Images." In: *Proceedings of the 13th Eurographics Symposium on Parallel Graphics and Visualization*. EGPGV '13. 2013, pp. 9–16. ISBN: 978-3-905674-45-3. DOI: 10.2312/EGPGV/EGPGV13/009-016 (cit. on p. 12).

[338] B.-L. Yeo. and B. Liu. "Volume Rendering of DCT-Based Compressed 3D Scalar Data." In: *IEEE Transactions on Visualization and Computer Graphics* 1.1 (1995), pp. 29–43. ISSN: 1941-0506. DOI: 10.1109/2945.468390 (cit. on p. 19).

[339] J. Yu and G. Turk. "Reconstructing Surfaces of Particle-based Fluids Using Anisotropic Kernels." In: *ACM Transactions on Graphics* 32.1 (2013), 5:1–5:12. ISSN: 0730-0301. DOI: 10.1145/2421636.2421641 (cit. on p. 13).

[340] M. M. Zdravkovich. *Flow Around Circular Cylinders: Volume I: Fundamentals*. Flow Around Circular Cylinders: A Comprehensive Guide Through Flow Phenomena, Experiments, Applications, Mathematical Models, and Computer Simulations. OUP Oxford, 1997. ISBN: 9780198563969 (cit. on p. 47).

[341] M. Zeidan, T. Rapp, C. Peters, and C. Dachsbacher. "Moment-Based Opacity Optimization." In: *Eurographics Symposium on Parallel Graphics and Visualization*. 2020. ISBN: 978-3-03868-107-6. DOI: 10.2312/pgv.20201072 (cit. on p. 9).

[342] M. Zeyen, J. Ahrens, H. Hagen, K. Heitmann, and S. Habib. "Cosmological Particle Data Compression in Practice." In: *Proceedings of the In Situ Infrastructures on Enabling Extreme-Scale Analysis and Visualization*. ISAV'17. Association for Computing Machinery, 2017, pp. 12–16. ISBN: 9781450351393. DOI: 10.1145/3144769.3144776 (cit. on p. 19).

[343] Y. Zhu and R. Bridson. "Animating Sand As a Fluid." In: *ACM Transactions on Graphics* 24.3 (2005), pp. 965–972. ISSN: 0730-0301 (cit. on p. 13).

[344] T. Zirr and C. Dachsbacher. "Memory-Efficient On-the-Fly Voxelization and Rendering of Particle Data." In: *IEEE Transactions on Visualization and Computer Graphics* 24.2 (2018), pp. 1155–1166. DOI: 10.1109/TVCG.2017.2656897 (cit. on p. 12).

[345] J. Ziv and A. Lempel. "A Universal Algorithm for Sequential Data Compression." In: *IEEE Transactions on Information Theory* 23.3 (1977), pp. 337–343. ISSN: 1557-9654. DOI: 10.1109/TIT.1977.1055714 (cit. on p. 19).

[346] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. "EWA Volume Splatting." In: *IEEE Visualization*. 2001, pp. 29–538. DOI: 10.1109/VISUAL.2001.964490 (cit. on pp. 9, 10).