Trajectory-Based Analysis of Flow Simulation Data

Vom Fachbereich Informatik der
Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von

Mathias Hummel

Datum der wissenschaftlichen Aussprache: 30. Oktober 2018

Dekan:	Prof. Dr. Stefan Deßloch
Berichterstatter:	Prof. Dr. Christoph Garth
Berichterstatter:	Prof. Dr. Bernd Hamann

D 386

Abstract

The simulation of physical phenomena involving the dynamic behavior of fluids and gases has numerous applications in various fields of science and engineering. Of particular interest is the material transport behavior, the tendency of a flow field to displace parts of the medium. Therefore, many visualization techniques rely on particle trajectories.

Lagrangian Flow Field Representation. In typical *Eulerian* settings, trajectories are computed from the simulation output using numerical integration schemes. Accuracy concerns arise because, due to limitations of storage space and bandwidth, often only a fraction of the computed simulation time steps are available. Prior work has shown empirically that a *Lagrangian*, trajectory-based representation can improve accuracy [Agr+14]. Determining the parameters of such a representation in advance is difficult; a relationship between the temporal and spatial resolution and the accuracy of resulting trajectories needs to be established. We provide an error measure for upper bounds of the error of individual trajectories. We show how areas at risk for high errors can be identified, thereby making it possible to prioritize areas in time and space to allocate scarce storage resources.

Comparative Visual Analysis of Flow Field Ensembles. Independent of the representation, errors of the simulation itself are often caused by inaccurate initial conditions, limitations of the chosen simulation model, and numerical errors. To gain a better understanding of the possible outcomes, multiple simulation runs can be calculated, resulting in sets of simulation output referred to as *ensembles*. Of particular interest when studying the material transport behavior of ensembles is the identification of areas where the simulation runs agree or disagree. We introduce and evaluate an interactive method that enables application scientists to reliably identify and examine regions of agreement and disagreement, while taking into account the local transport behavior within individual simulation runs.

Particle-Based Representation and Visualization of Uncertain Flow Data Sets. Unlike simulation ensembles, where uncertainty of the solution appears in the form of different simulation runs, moment-based Eulerian multi-phase fluid simulations are probabilistic in nature. These simulations, used in process engineering to simulate the behavior of bubbles in liquid media, are aimed toward reducing the need for real-world experiments. The locations of individual bubbles are not modeled explicitly, but stochastically through the properties of locally defined bubble populations. Comparisons between simulation results and physical experiments are difficult. We describe and analyze an approach that generates representative sets of bubbles for moment-based simulation data. Using our approach, application scientists can directly, visually compare simulation results and physical experiments.

Kurzfassung

Die Simulation des dynamischen Verhaltens von Flüssigkeiten und Gasen findet Anwendung in vielen Gebieten von Wissenschaft und Technik. Von besonderem Interesse ist das Transportverhalten, die Tendenz eines Strömungsfeldes, Teile des Mediums zu bewegen. Daher basieren viele Visualisierungstechniken auf Partikellaufbahnen.

Lagrangesche Strömungsfelddarstellung. In typischen *Eulerschen* Szenarien werden Laufbahnen mittels numerischer Integration aus den Simulationsdaten berechnet. Dabei treten Ungenauigkeiten auf, weil oft wegen Beschränkungen von Speicherplatz und Bandbreite nur ein Teil der Zeitschritte verfügbar ist. Es wurde bereits empirisch gezeigt, dass eine *Lagrangesche*, laufbahnbasierte Darstellung die Genauigkeit verbessern kann [Agr+14]. Die Parameter dieser Darstellung lassen sich kaum im Voraus bestimmen, da der Zusammenhang zwischen zeitlicher und räumlicher Auflösung und Genauigkeit unklar ist. Wir stellen ein Fehlermaß für einzelne Laufbahnen vor. Wir zeigen, wie Bereiche hohen Fehlerrisikos ermittelt werden können, wodurch es möglich wird, räumliche und zeitliche Bereiche bei der Zuweisung knapper Platzressourcen zu priorisieren.

Vergleichende, Visuelle Analyse von Strömungsensembles. Unabhängig von der Darstellung entstehen Fehler der eigentlichen Simulation oft durch ungenaue Anfangsbedingungen, Beschränkungen des Simulationsmodells und numerische Fehler. Zur besseren Ermittlung der möglichen Ergebnisse werden mehrere Simulationsdurchläufe berechnet, die Reihen von Simulationsausgaben, sogenannte *Ensembles*, zum Ergebnis haben. Von besonderem Interesse bei der Untersuchung des Transportverhaltens ist die Erkennung von Bereichen, in denen sich die Simulationsläufe gleichen oder unterscheiden. Wir stellen eine Methode vor, durch die zuverlässig Unstimmigkeiten erkannt werden können, wobei das lokale Transportverhalten innerhalb einzelner Simulationsläufe berücksichtigt wird.

Partikelbasierte Visualisierung Unsicherer Strömungsdaten. Im Gegensatz zu Ensembles, wo Unsicherheit der Lösung in Form unterschiedlicher Simulationsläufe auftritt, sind momentbasierte, Eulersche Mehrphasenströmungssimulationen von Natur aus probabilistisch. Diese Ansätze dienen in der Verfahrenstechnik dazu, den Bedarf an physikalischen Experimenten mit Blasen in flüssigen Medien zu reduzieren. Die Positionen einzelner Blasen werden nicht explizit, sondern stochastisch über die Eigenschaften örtlich definierter Blasenpopulationen modelliert. Vergleiche zwischen den Ergebnissen und physikalischen Experimenten sind schwierig. Wir beschreiben und analysieren einen Ansatz, bei dem stellvertretende Mengen von Blasen für momentbasierte Simulationsdaten erzeugt werden. Durch unseren Ansatz können Simulationsergebnisse und physikalische Experimente direkt und visuell verglichen werden.

Acknowledgements

First and foremost, I would like to extend my deepest gratitude and appreciation to Christoph Garth. As my long-time academic advisor, he has been a steady and reliable source of advice, motivation, and constructive criticism. Additionally, I wish to thank Hans Hagen and Ken Joy for their immeasurable advice and inspiration, and their continued hospitality.

I am immensely grateful to Harald Obermaier and Roxana Bujack, who, during their time as postdoctoral researchers, collaborated on a majority of the research presented in this dissertation, and coauthored the corresponding publications. I especially thank them for their monumental patience.

For their part in the bubble project, I would like to extend special appreciation to my repeated coauthors Mark Hlawitschka, Lisa Jöckel, and Jan Schäfer. To Mady Gruys – I thank you for years of support with handling administrative issues, and for single-handedly keeping the atmosphere of our AG family alive.

This journey would never have been the same without my friends and colleagues: Farangis Khoso Anjom, Tim Biedert, Hugo Braun, Christopher Costa, Roger Daneker, Klaus Denker, Achim Ebert, Tobias Gauweiler, Florian Höpel, Lars Hüttenberger, Taimur Khan, Aaron Knoll, Bernd Lietzow, Franca Rupprecht, Inga Scheler, Frederik Walk, Sophie Zakrzewski. You have always been there to provide me with ample advice, companionship, and oftentimes assistance of both academic and administrative nature. Thank you!

I am deeply grateful to the International Research Training Group (IRTG) 2057 and the German Research Foundation (DFG). By providing the academic environment, training, extremely valuable opportunities for exchange with my peers both local and abroad, and funding, the IRTG 2057 has made the research for this dissertation possible in the first place.

Lastly, I would like to acknowledge my family. To my parents and my grandparents: I thank you for your encouragement and inspiration in my pursuits and travels. I am especially thankful to my mother, Birgit, for fostering my curiosity and love for science, and to my father, Markus, for inspiring my enthusiasm for technology and engineering.

For his continuous, unwavering company and emotional support I thank our dog, Argos, and award him a token of my gratitude: one (1) banana, to be dispensed immediately after the submission of this dissertation. Most importantly, I am deeply grateful to my amazing wife, Olga. Thank you for weathering all those storms with me! Without you, none of this would have been possible. You are my rock!

Contents

List of Figures **xiii**

List of Mathematical Symbols **xv**

1 Introduction **1**

2 Methodical Background and Prior Work **7**

2.1 Flow Simulation 7

2.1.1 Eulerian Flow Simulation 8

2.1.2 Lagrangian Flow Simulation 8

2.2 Flow Field Representation 9

2.2.1 Computational Grids 11

2.2.2 Interpolation 13

2.2.3 Cell Location 17

2.2.4 Time-Varying Data 18

2.3 Integral Curves 18

2.3.1 Mathematical Definition 19

2.3.2 Numerical Approximation 19

2.4 Integral Surfaces 22

2.5 Lagrangian Transport 23

2.5.1 The Finite-Time Lyapunov Exponent (FTLE) 23

2.5.2 Lagrangian Coherent Structures (LCS) 23

2.6 Uncertainty 26

3 Error Estimates for a Lagrangian Flow Field Representation **29**

3.1 Related Work 31

3.2 Interval Flow Maps:
A Lagrangian Flow Field Representation 32

3.3 Theoretical Error Estimate 34

3.3.1 One-step Integration Method 34

3.3.2 Local Truncation Error 35

3.3.3 Global Truncation Error 36

3.3.4 Intra-Interval Error . . . 38
3.4 Application to Discrete Flow Data . . . 38
3.4.1 Local Truncation Error . . . 39
3.4.2 Regional Lipschitz constant . . . 40
3.4.3 Error Within the Sections . . . 41
3.5 Experimental Results . . . 42
3.5.1 Evaluation . . . 45
3.5.2 Visualization . . . 45
3.6 Discussion . . . 49

4 Comparative Visual Analysis of Lagrangian Transport in CFD Ensembles 51
4.1 Related Work . . . 52
4.2 Background: Ensembles and Material Transport . . . 54
4.2.1 Ensembles . . . 54
4.2.2 Material Transport . . . 56
4.2.3 FTLE and the Flow Map . . . 57
4.3 Lagrangian Comparison of Flow Ensembles . . . 57
4.3.1 Divergence of Generalized Neighborhoods . . . 58
4.3.2 Lagrangian Neighborhoods for Ensembles . . . 60
4.3.3 Naïve Transport Comparison . . . 61
4.3.4 Variance Classification . . . 62
4.3.5 Classification Space . . . 62
4.3.6 Outliers and Trends . . . 65
4.4 Visualization . . . 66
4.4.1 Classification Space . . . 66
4.4.2 Ensemble Domain . . . 67
4.4.3 Interaction and Exploration . . . 68
4.5 Experiments . . . 69
4.5.1 Implementation . . . 69
4.5.2 Data Sets . . . 72
4.5.3 Results and Analysis . . . 75
4.6 Discussion . . . 77

5 Visualization of Uncertain Multi-Phase Fluid Simulation Data 79
5.1 Background: Method of Moments CFD Simulation . . . 80
5.1.1 Tasks and Requirements . . . 82
5.2 Related Work . . . 82
5.3 A Parametric Bubble Model . . . 83

5.4 Bubble Distribution: Sampling and Updating 86
5.4.1 Filtering 86
5.4.2 Initial Sampling 88
5.4.3 Updating the Bubble Positions 90
5.4.4 Measuring the Bubble Density 91
5.5 Implementation 93
5.6 Case Study: Bubble Column 95
5.7 Discussion 103

6 Conclusion and Future Research **105**

References **109**

Publications **121**

Curriculum Vitae **123**

List of Figures

2.1 Analytical flow fields: double gyre and ABC flow . . . 10
2.2 Examples: Regular grid and irregular grid . . . 11
2.3 VTK unstructured grid cell types . . . 12
2.4 Interpolation: Reference elements for quadrilaterals and hexahedra . . . 14
2.5 Example: integral surface in flow behind an ellipsoid . . . 22
2.6 Influence of integration time on FTLE in 2D convection data set . . . 24

3.1 Constructing a trajectory based on a sequence of short-time flow maps . . . 33
3.2 Local and global truncation errors . . . 35
3.3 Intra-interval error . . . 37
3.4 Double gyre: error rectangles and cones . . . 43
3.5 ABC: error cones . . . 44
3.6 Comparison of estimated and average error over time . . . 46
3.7 Double gyre: estimated and average errors in the domain . . . 47
3.8 Convection: estimated and average errors in the domain . . . 48

4.1 Sources of trajectory endpoint variance . . . 57
4.2 Comparison of FTLE and PCA . . . 59
4.3 Classification space . . . 63
4.4 Influence of individual variance on joint variance . . . 64
4.5 Minimum spanning tree for trend detection . . . 66
4.6 Variances in 3D Rayleigh-Taylor data set and 3D convection data set . . . 69
4.7 Comparison of transport in 2D convection data set . . . 70
4.8 Interactive brushing in 2D convection data set . . . 71
4.9 Joint variance and trends in 2D convection data set . . . 71
4.10 Setting and joint variance in 2D mixing data set . . . 73
4.11 Comparison of transport in 2D mixing data set . . . 74
4.12 Trends in 2D mixing data set . . . 75

5.1 Bubble shape representatives . . . 84
5.2 Barycentric interpolation of bubble shapes . . . 85
5.3 Constructing a bubble using a B-Spline curve . . . 85

5.4 Initializing the bubble set using rejection sampling 87
5.5 Updating the bubble set using rejection sampling 89
5.6 Measuring bubble set density field using box kernels 92
5.7 Advancing bubble positions between intervals 94
5.8 Overview of the sampling based visualization approach 95
5.9 Comparison of color plots and bubble set 96
5.10 Experimental setup; Comparison of physical experiment and bubble set rendering 97
5.11 Density measurement error for increasing number of samples 98
5.12 Density measurement error and number of changes for different kernel width values 99
5.13 Density measurement for different kernel width values 100
5.14 Bubble additions and removals for different kernel width values 101

List of Mathematical Symbols

$\mathbf{x}$	spatial position
$\mathbf{x}_0$	starting position
$\mathbf{t}$	time
$\mathbf{t}_0$	starting time
$\mathbf{x}(t)$	trajectory
$\mathbf{u}(\mathbf{x},t)$	velocity field, Eulerian representation
h	step width for numerical integration; kernel width
$L_{\mathbf{x}}$	linear interpolation with respect to $\mathbf{x}$
N_i	shape function for multilinear interpolation
$F_{t_0}^t(\mathbf{x})$	flow map spanning the time interval $[t_0,t]$
$\sigma_{t_0}^t(\mathbf{x})$	Finite Time Lyapunov Exponent for the time interval $[t_0,t]$
τ_j	local truncation error of trajectory construction for time interval $[t_{j-1},t_j]$
$e_j(x)$	global truncation error of trajectory construction for time interval $[t_{j-1},t_j]$
$s_j(x)$	intra-interval error of trajectory construction for time interval $[t_{j-1},t_j]$
$L(x)$	Lipschitz constant of the velocity field
Ω	spatial domain
I	temporal domain
E	flow field ensemble
Σ	covariance matrix
$N_{\mathbf{x}}$	generalized neighborhood
$P(\mathbf{x})$	neighborhood predicate
C	classification space
$M_r(\mathbf{x},t)$	rth moment of the bubble size disribution
α_{disp}	volume fraction of the dispersed phase
Eo, Re	characteristic numbers Eötvös number, Reynolds number
V	volume
n	number of bubbles
$w(\mathbf{v})$	kernel for density measurement
IMSE	integrated mean squared error

1 Introduction

Many problems in natural sciences and engineering involve the interaction of fluids, gases, and solid boundaries. Classic examples for such scenarios include weather phenomena or the flow around aircraft, vehicles, and ship hulls. Theoretical descriptions of the physical processes governing various aspects of fluid flow have existed for a long time. In practice, however, the equations stemming from any but the most trivial scenarios cannot be solved exactly due to their complexity.

Historically, to be able to make predictions about the performance of new designs, scale models of a design were built and then tested by placing them in wind tunnels or test tanks. If the performance of a model turned out not to be satisfactory, the design and the model had to be modified and the experiment repeated. Clearly, such a process can be extremely time consuming and expensive, or not feasible at all for some applications, such as weather prediction. Theoretical models describing the dynamic behavior of liquids or gases have existed for a long time: the Euler equations were described in the 18th century; the more general Navier-Stokes equations were introduced in the 19th century. In practice, however, these models were not applicable to many complex problems. Only in the mid-20th century, together with the appearance of modern computers, the field of computational fluid dynamics made it possible to calculate reasonably accurate approximate solutions. Such flow simulation methods are used in many fields of science and engineering. Together with Computer Aided Design (CAD), simulating the performance of an aicraft or vehicle design under a specific scenario can be achieved much faster and less costly than building physical models and performing real-world experiments.

In the most general sense, a common workflow is to model the scenario, including the definition of initial and boundary conditions and the choice of the simulation model itself, and then execute the simulation in small, incrementing time steps. With increasing complexity of the simulated scenario, sufficient accuracy under time constraints can only be achieved by an increase of computational power. To this end, simulations are often run in batches on high performance computers such as clusters and supercomputers. The output of such a simulation run consists of a number of fields which define the values of physical quantities relevant to the theoretical model, such as pressure or velocity, defined over the previously specified spatial domain for a number of time steps. The time steps, written to persistent storage, are often also referred to as *restart points*: they completely describe the state of the

simulation at a given point in time and can be used to restart the simulation if problems are encountered.

The primary purpose of the simulation output is to provide answers to the main questions about the simulated scenario. Some of these questions, such as whether or not a threshold of a specific quantity is exceeded, may be answered directly by looking at the output values. However, a lot of information contained in the simulation output cannot be accessed in this way: for an in-depth analysis of the simulation data, visualization methods are a necessity. One of the physical quantities among the simulation output is the velocity field, which defines the momentary speed of movement and direction for each parcel of fluid involved in the simulation. The velocity field is of special interest to application scientists and engineers, because it describes important properties such as the material transport behavior. To provide insight through visual analysis, a large number of methods and technologies have been developed in the active research field of flow visualizaton. Among others, areas of interest include the locations of specific flow features, such as vortices, and the material transport behavior.

In real-world wind tunnel experiments, features and the general flow behavior can be in part revealed by injecting smoke before or around an obstacle. If the flow is steady, i.e., the velocities do not change over time, the resulting curves coincide with the trajectories of the smoke particles and are called stream lines. The same concept can be applied to flow fields resulting from a flow simulation: stream lines, curves that are tangent to the flow direction, are a basic but powerful tool for the visualization of steady flows. These trajectories of massless particles that are advected with the flow are closely related to the material transport behavior, which is why many more advanced visualization methods are based on trajectories.

Lagrangian Flow Field Representation Some visualization methods, such as the detection of material barriers (Lagrangian Coherent Structures), require a particularly large number of trajectories, which have to be computed using numerical integration schemes. Since computing many trajectories is computationally expensive, this step can result in long wait times for visualizations on regular workstation hardware. At the same time, with simulations running on a supercomputer or cluster, accuracy concerns can arise: In many cases, due to restricted availability of storage bandwidth or space, only a fraction of the simulation time steps can be written to persistent storage, while the rest are discarded. One approach that aims to deal with both issues is to leverage the computing power of the supercomputer running the simulation to calculate large numbers of trajectories in advance, alongside the simulation, while using every time step.

The results can then be stored in a Lagrangian representation of the flow field that describes particle trajectories instead of momentary velocities. Such a representation can then be used to quickly reconstruct large numbers of trajectories without the need for numerical integration of the original velocity field. It has also been shown in the past that using a Lagrangian representation, higher accuracy can be achieved given the same space constraints [Agr+14]. However, while error bounds of trajectories computed using numerical integration are well understood, this is not the case for the Lagrangian representation. As a result, chosing the output frequency and resolution for a simulation in advance is difficult. The first goal of this dissertation is to enable application scientists to evaluate the impact of using a trajectory-based flow field representation for the output of a simulation, and to facilitate the choice of parameters for this representation.

Flow Field Ensembles Even with a solid grip on the errors of individual trajectories in a given velocity field, which stems from a single simulation run, it is not immediately clear whether or not the simulation run itself accurately and completely captures the behavior of the modeled scenario. In many cases, even small changes of parameters, boundary conditions, or the simulation model can potentially result in wildly different results and, subsequently, a different transport behavior. In an attempt to capture as many possible variations as possible, collections consisting of multiple simulation runs, so-called *ensembles*, are computed increasingly often. For example, in weather forecasting, ensembles are used to compensate for uncertainty of the initial conditions. Ensemble approaches are made possible by the the ever increasing amount of computing power provided by high performance computers.

Flow simulation ensembles pose additional challenges for the visualization. Established visualization tools that are used to study the transport behavior, such as integral curves or Lagrangian Coherent Structures (LCS), can be applied to individual simulation runs. It is not immediately clear how these methods can aid to understand or compare the behavior of entire sets of simulation output. The second goal of this dissertation is to provide new methods that allow application scientists to compare the material transport behavior of flow field ensembles.

Probabilistic Multi-Phase Flow Simulation Data Probabilistic flow fields do not necessarily utilize the concept of ensembles; they can also result from simulations whose underlying mathematical models are probabilistic in nature. In the field of process engineering, so-called *Method of Moments (MoM)* simulations are used to model flow in chemical reactors involving bubbles or droplets, with the final goal of replacing real-world, physical experiments. The probabilistic MoM approach is used to avoid incurring the high cost of

simulating bubbles individually. Instead, a stochastic model of the bubble population is used where the material phases, such as a liquid and a gas, are treated as interpenetrating continua.

As a consequence, in addition to the velocity fields of the involved phases, the resulting output of the simulation consists of a stochastic description of the bubble population, from which a bubble density field and a bubble diameter field can be deduced. For visualization, application scientists typically use color plots of these scalar fields, which are then visually compared to photographs of corresponding physical experiments. Contrary to the color plots, photographs allow a direct visual assessment of the density and bubble diameters. Accordingly, the third goal of this dissertation is to develop methods that enable direct visual comparison between records of physical experiments and simulation data, without disrupting the established work flow of the application scientists.

All techniques introduced in this dissertation rely on a number of basic concepts and methods related to computational fluid dynamics and visualization. Before each of the above mentioned goals is treated in a separate chapter of this thesis, Chapter 2 describes the necessary methodical background of flow simulation, data representation, and fundamental visualization techniques that form the basis of the methods and techniques described later in this dissertation. In addition, the chapter lists a number of tangential but relevant related works from the field of scientific visualization.

In Chapter 3, a Lagrangian representation that has been shown to perform well in the past – concatenation and subsequent evaluation of flow maps – is examined regarding its error behavior. Theoretical bounds of the error that arises when trajectories are reconstructed from the Lagrangian representation are derived. The chapter goes on to describe how this error estimate can be calculated in practice. For a number of example data sets, the calculated error estimate is compared to actual error measurements, thereby demonstrating the capability of the method.

Chapter 4 introduces a new method for the comparative analysis of the transport behavior in flow field ensembles. In a basic sense, the transport behavior of two or more simulation runs can be considered to be different for a specific location in space if trajectories started at that position end up in different final locations. For a specific end time, the difference can be quantified by collecting the end points of the trajectories started at the same location and time in all simulation runs, and then computing a scalar variance value from the point cloud. In this work, principal component analysis (PCA) of the set of end points is used to obtain the magnitude of the first principal component, which describes the axis of maximal variation. Unfortunately, taken by itself, the resulting scalar field is not sufficient to distinguish similar from dissimilar transport behavior aross the ensemble: In areas that feature strong divergence, even small perturbations of the starting position can result in

wildly different trajectory end points. It is, therefore, important to identify such regions when examining the joint variance field. This is achieved using a second variance mesasure, the individual variance, defined for each simulation run as the first principal component of the set of trajectory endpoints that result from small perturbations of the start position. Together, these two variances span a classification space that can be used to interactively identify regions of agreement and disagreement in the spatial domain.

Chapter 5 of this thesis introduces a new visualization method for probabilistic method of moment simulation data of flow involving bubbles or droplets. Based on the stochastic definition of the bubble population, a representative set of bubble positions is created using a sampling approach. A parametric bubble model is used to define the shape of each bubble based on local flow properties, the Reynolds number and the Eötvös number. To enable time-varying visualization, an update mechanism for the bubble set is introduced that uses velocity information to advance bubbles between time steps. The density of the modified bubble set is then measured and compared to the simulation output; existing bubbles are removed and new bubbles added in areas where the measured density field deviates from the density field produced by the simulation to account for bubble coalescence and separation events. The approach is evaluated using data from the simulation of a bubble column. The influence of the only major parameter of the method, the width of the kernel used for density measurement, is evaluated experimentally.

Finally, concluding remarks and an outlook for future research can be found in Chapter 6. In summary, the contributions of this dissertation include:

- A theoretical definition and accompanying method to calculate an upper bound for errors that arise when reconstructing trajectories from a Lagrangian flow field representation. This measure can be used to aid applications scientists with the tedious task of choosing the simulation output frequency and resolution, while providing a clear understanding of the accuracy impact associated with the involved trade-offs.
- A comparative approach for analyzing the transport behavior, i.e., the behavior of trajectories, in flow field ensembles by mapping locations of the spatial domain to a classification space. Using this measure, applications scientists can reliably detect spatial regions of agreement and disagreement in simulation ensembles.
- A method to produce intuitive visualizations of bubbles in probabilistic, moment-based multi-phase fluid simulation data using trajectories to update bubble positions over time. Application scientists can use this method for direct, visual comparisons of physical experiments and simulation data.

2 Methodical Background and Prior Work

To tackle the challenges of trajectory-based visualization outlined in Chapter 1, this thesis introduces and discusses a number of techniques and methods and their evaluation. All of these methods rely on a large body of fundamental concepts and techniques that have been established in the field of flow visualization and in related fields. This chapter serves to provide an overview and introduction to these concepts.

2.1 Flow Simulation

Flow simulation methods are widely used in various fields of science and engineering for diverse scenarios involving the interaction of liquids or gases. In contrast to many physical experiments, simulations can be performed and repeated at relatively low cost and often in a relatively short time. Many different types of simulations have been developed for various scenarios. This section does not serve as a summary or introduction to the large field of Computational Fluid Dynamics; instead, it is aimed at providing only a cursory overview of concepts that are relevant to understanding the applications, methods, and techniques that are introduced and discussed in this thesis.

The primary task at the center of a flow simulation method is to find an approximate solution for the *Navier-Stokes Equations* [CM93], which describe the temporal evolution of physical quantities such as – among others – velocity, density, and pressure. Together with additional constraints, such as conservation of mass and energy, these equations describe the motion of a compressible, viscous fluid. Given a set of boundary conditions and initial values, such as initial pressure, velocity, and the configuration of the domain (shapes of walls, inlets, outlets), a numerical solver determines how these properties change over time in accordance with the Navier-Stokes equations. In different settings, different subsets of these equations may be relevant. For example, a scenario may feature either compressible or incompressible flow. Approximate solutions can be obtained through various approaches. These can roughly be divided into *Eulerian* and *Lagrangian* methods [Bat00]. In Eulerian settings, the relevant quantities are tracked from the perspective of stationary locations. For

example, the velocity field is written as $\mathbf{u}(\mathbf{x},t)$, where $\mathbf{x}$ refers to the position in space and t the time.

Conversely, in Lagrangian settings, quantities are viewed from the perspective of individual parcels of the moving medium. Here, velocity can be written as $\mathbf{v}(\mathbf{x}_0,t)$, where $\mathbf{x}_0$ is the starting position of the parcel at a predefined starting time $t_0 \leq t$. The trajectory of the parcel is given as $\mathbf{x}(t)$, with $\mathbf{x}(t_0) = \mathbf{x}_0$.

The relationship between the Eulerian view and the Lagrangian view can be expressed as follows:

$$\mathbf{u}(\mathbf{x}(t),t) = \mathbf{v}(\mathbf{x}_0,t) = \frac{d}{dt}\mathbf{x}(t), \quad \mathbf{x}(t_0) = \mathbf{x}_0. \tag{2.1}$$

2.1.1 Eulerian Flow Simulation

Most simulations considered in this thesis use the Eulerian view, where variables are tracked over time at a set of fixed locations in space. With Eulerian flow simulation, solutions of the Navier-Stokes equations (or a subset thereof) are approximated using numerical methods. To this end, the spatial domain is subdivided into smaller areas in a discretization step [Hir91], resulting in a regular or irregular mesh. Given this mesh, in addition to definitions of boundary conditions and initial values, numerical solvers embedded in simulation software packages such as OpenFOAM [Wel+98] or Gerris [Pop03] are used to compute the values of the involved physical quantities, defined on the mesh, for increasing time steps. The spacing of the time steps can have great influence on the outcome of the simulation. For iterative, numerical schemes, smaller time steps result in smaller errors. Conversely, too large time steps can result in instability, where numerical errors are amplified, causing wildly wrong results. To preserve accuracy and stability, time steps are often adapted as the simulation proceeds.

2.1.2 Lagrangian Flow Simulation

A fundamentally different concept is used in the Lagrangian setting. Here, the simulation does not track fields of various variables over time; instead, the state of the simulation is stored in the form of particles that interact with each other. One method from this family is the *Smoothed Particle Hydrodynamics* approach (SPH), which was originally developed for applications in astrophysics [GM77; Mon05]. Each particle is assigned a radius of influence. To interpolate physical quantities, each particle within range contributes a portion of its value that is governed by an interpolation kernel. An interpolation kernel is a function,

defined in terms of the distance from the location of an individual particle, that has its maximum at the center and monotonously decreases with increased distance [COJ15]

Even though all topics discussed in this thesis are closely related to trajectories, which are a Lagrangian concept, Lagrangian flow simulations do not play a role. All flow data sets stemming from the various applications are used in an Eulerian representation.

2.2 Flow Field Representation

As seen in the previous section, the output of a flow simulation typically consists of the values of several fields – scalar-valued, vector-valued, or tensor-valued – for a number of time steps or restart points. The velocity field $\mathbf{u}$ defines the motion of the liquid or gaseous medium in the domain, and is, therefore, also referred to as a flow field. Depending on the type of simulation, these fields may be represented in different ways. For an Eulerian simulation, values are often specified at the nodes, faces, and cell centers of the discretization, while values in Lagrangian simulations (e.g., SPH) are defined for sets of particles, which also carry their current position within the domain.

Flow fields do not necessarily originate from simulations. For academic purposes, analytic flow fields are often used that are described completely by a closed-form expression. Examples include the two-dimensional double gyre and Arnold-Beltrami-Childress flows (Fig. 2.1), which are trivial to evaluate without the requirement of lengthy simulation or data storage. They are, therefore, useful tools to explain or evaluate certain aspects of flow-related methods.

Historically, digital representations of flow fields have been obtained by measuring the state of physical experiments. For example, in wind tunnel experiments, smoke is used to reveal flow features. These can be recorded by illuminating the smoke using lasers and then imaging the resulting cross sections using high-speed cameras [YH89].

All methods that are employed later in this thesis rely on Eulerian, numerical representations of the involved flow fields. This section provides an overview of such representations and the tools necessary to access them in visualization applications.

The output of an Eulerian flow simulation typically consists of a set of time steps, each describing the state of the simulation at a specific point in time. For a finite set of locations, covering a two- or three-dimensional domain, the values of the simulation's variables are recorded. Depending on the application, one of a number of available data representations is used to store these values.

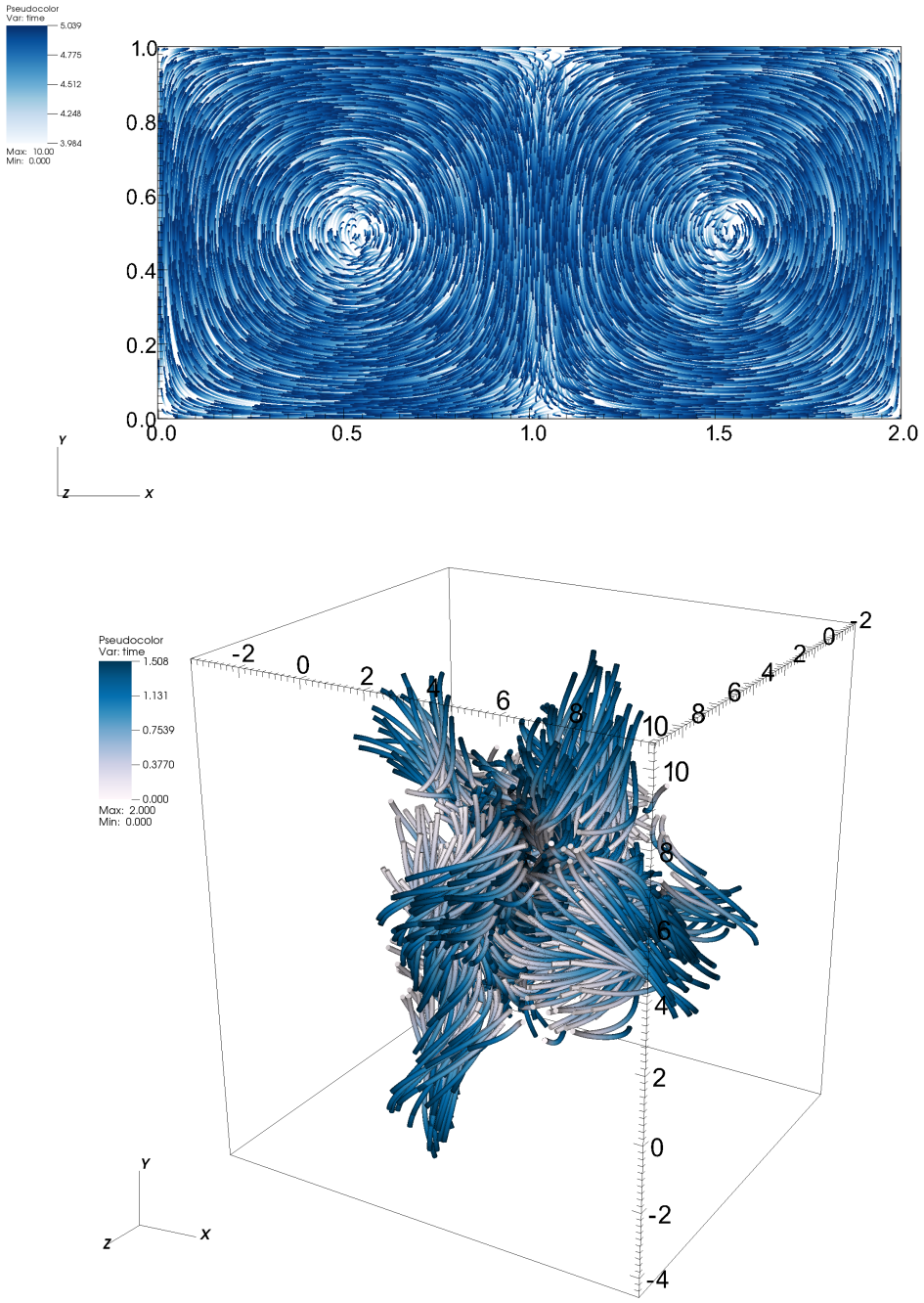

Figure 2.1: Analytically defined flow fields, where the velocity at any point in the domain can be obtained by evaluating a closed-form expression, are useful tools for demonstration and evaluation of visualization methods. The plots show trajectories in two time-varying flow fields. Top: The double gyre flow field is defined on a two-dimensional, closed domain. We use the definition from Shadden et al. [SLM05]. It features two counter-rotating vortices, with their centers oscillating left to right over time. Bottom: The three-dimensional Arnold-Beltrami-Childress (ABC) flow field, as described by Haller [Hal01], is defined on a periodic domain.

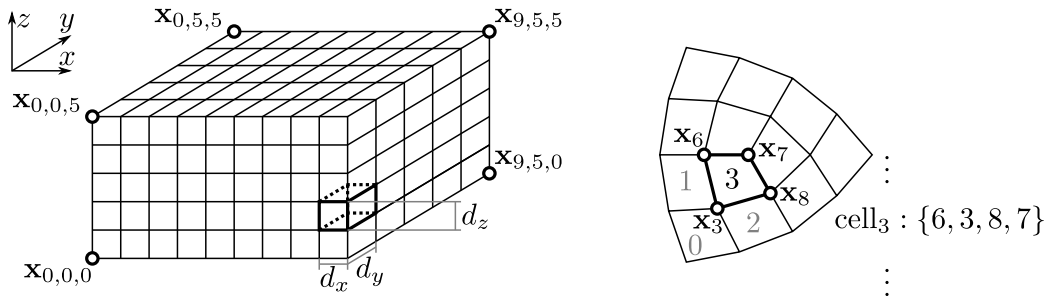

Figure 2.2: The nodes of a regular grid can be accessed through one index per dimension; the connectivity of the cells is given implicitly (left). With irregular grids, this is generally not possible, even if all cells are of the same type (right).

The concepts and algorithms outlined in this section form the basis for the implementations of all methods and techniques in the remainder of this thesis. They are required to access and harness the data contained in the output of Eulerian flow simulations.

2.2.1 Computational Grids

Grids serve as the discrete representation of fields in Eulerian flow simulation. In the field of scientific visualization, they are used wherever discretized fields of values are represented. A prominent example for these representations is given by the Visualization Toolkit's *VTK Legacy File Format* [Avi+10, pp. 472–482], a simple file format that supports most grid types that are commonly found in flow data sets. This paragraph will provide a short description of each grid type, roughly following the documentation of the VTK Legacy File Format.

Structured Grids are composed of quadrilateral cells (in two-dimensions) or hexahedral cells (in three-dimensions). The nodes (also referred to as vertices) forming the corners of the cells are addressed by one integer index per dimension, where the range of the indices i, j, and k, given by the vector (n_x, n_y, n_z), defines the number of nodes and cells in the grid. In the general case, the location of each node has to be provided explicitly:

$$\begin{aligned} \mathbf{x}_{i,j,k} &= (x_i, y_j, z_k), \quad (i,j,k) \in [0, n_x - 1] \times [0, n_y - 1] \times [0, n_z - 1] \\ n_{\text{nodes}} &= n_x n_y n_z \\ n_{\text{cells}} &= (n_x - 1)(n_y - 1)(n_z - 1) \end{aligned}$$

The defining property of a structured grid is that the connectivity between the nodes is completely defined through the nodes' indices.

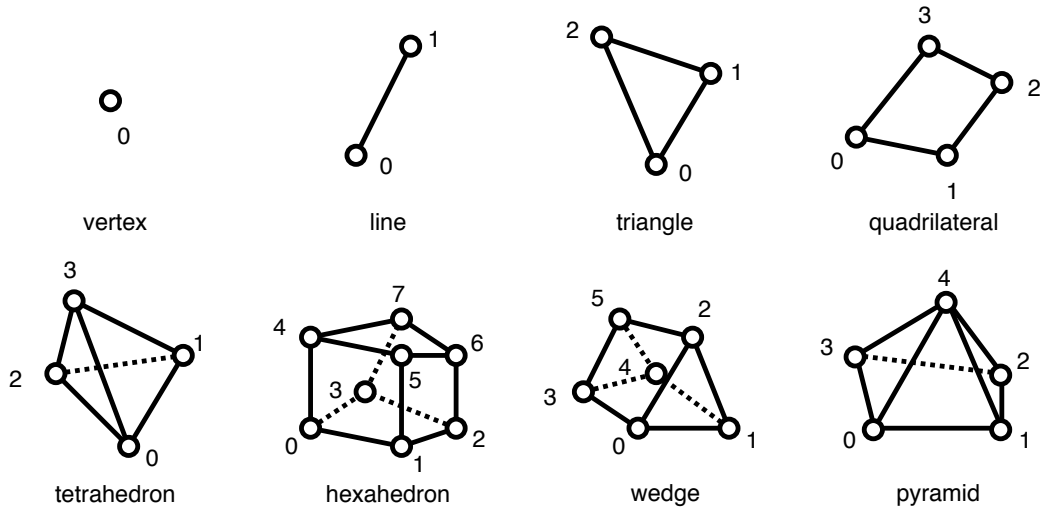

Figure 2.3: Cell types used in unstructured grids, based on the VTK User's Guide [Avi+10, p. 480]

Regular Grids are structured grids where not only the connectivity, but also the spatial position of each node is defined using the indices of the node[1]. To this end, an additional origin point $\mathbf{x}_0$ and cell spacing $\mathbf{d}$ are required:

$$\mathbf{x}_{i,j,k} = \mathbf{x}_0 + (i,j,k) \cdot \mathbf{d} = (x_0 + i \cdot d_x, y_0 + j \cdot d_y, z_0 + k \cdot d_z)$$

A regular grid where $\mathbf{x}_0 = (0,0,0)$ and $\mathbf{d} = (1,1,1)$ is also sometimes referred to as a *Cartesian grid*.

Unstructured Grids are a more general type of grid where the connectivity between nodes is completely independent of the node indices. Typically, nodes are represented as a list of their spatial positions:

$$\mathbf{x}_i = (x_i, y_i, z_i), \quad i \in [0, n-1]$$

Connectivity is then specified as a list of cells. Unlike structured grids, unstructured grids are not limited to quadrilaterals and hexahedra. Two-dimensional data sets typically consist of triangles and quadrilaterals, while in three dimensions, tetrahedra, wedges, or pyramids are often encountered. Figure 2.3 provides an overview of these common cell types and

[1] In VTK, a regular grid is referred to as a *structured points* data set.

the ordering of their indices. The cells of an unstructured grid are defined through lists of integers:

$$\begin{aligned}\text{cell}_0 &= \{j_{0,0}, j_{0,1}, j_{0,2}, \dots\} \\ \text{cell}_1 &= \{j_{1,0}, j_{1,1}, j_{1,2}, \dots\} \\ \text{cell}_2 &= \{j_{2,0}, j_{2,1}, j_{2,2}, \dots\} \\ &\vdots\end{aligned}$$

Each j refers to the index of a node in the node list. In addition to the node indices, for each cell, a cell type has to be specified explicitly.

As a side note, a simple type of unstructured grid that finds frequent use in this thesis is the piece-wise linear representation of trajectories. Such a piece-wise linear curve consists of a set of spatial positions, while connectivity is defined through sets of line segments.

Field Values are provided as lists of floating-point values. They can either be defined for the nodes or for the cells of a data set. Typical field specifications use either scalar, vector-valued, or tensor-valued entries. For the purposes of this thesis, a vector is a tuple of d values, where d is the dimensionality of the field's spatial domain. In the context of visualization, the term "tensor" is often used to refer to matrices of size $d \times d$, even though the term technically encompasses scalars, vectors, and objects of higher dimensionality as well.

2.2.2 Interpolation

Most methods for visualization and analysis require a continuous representation of the input data, meaning that they have to be able to access values at arbitrary locations within the spatial and temporal domain. Typically, however, simulation output values are only available at a finite set of discrete locations, given by the vertices of a collection of cells. Interpolation is used to retrieve values within a cell in a continuous way, while enforcing continuity constraints across cell boundaries. For the applications described in this thesis, linear and multi-linear interpolation are used [HH12].

Generally speaking, at the core of interpolation is a function that maps the cell onto a reference element with a parameterization that facilitates interpolation. Given such a parameterization, to interpolate a quantity that is defined for the cell's nodes at a spatial

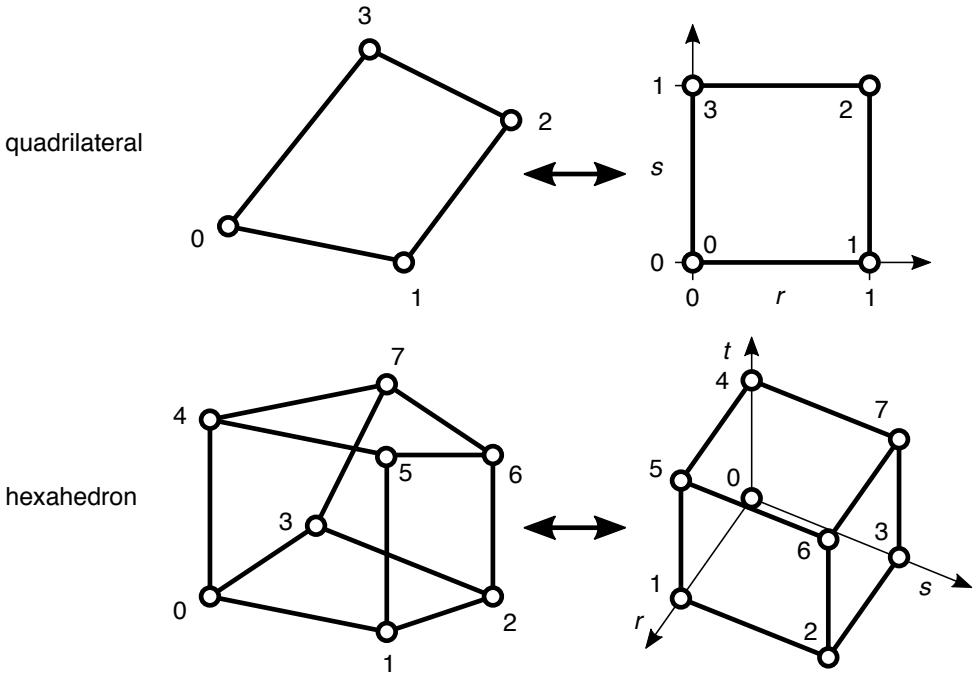

Figure 2.4: For the purpose of interpolation, quadrilaterals and hexahedra are mapped to an axis-aligned, unit sized reference element parameterized by $\mathbf{p} = (r,s)$ and (r,s,t), respectively.

position $\mathbf{x}$ within the cell, the position $\mathbf{x}$ is expressed as a convex combination of the node positions $\mathbf{x}_i$:

$$\mathbf{x} = \sum_i N_i \mathbf{x}_i, \qquad \text{where } N_i \leq 1 \text{ and } \sum_i N_i = 1. \tag{2.2}$$

Following a concept from the Finite Element Method (FEM), finding the weights N_i is achieved using *shape functions*, which are defined in terms of the parameterization of the reference element [ZTZ05]. The weight associated with each node quantifies the influence of that node on the interpolated result.

Barycentric Linear Interpolation is used for triangles and tetrahedra, where any position within the cell can be expressed by barycentric coordinates. The weights N_i, $i \in [0..2]$ (triangle), $i \in [0..3]$ (tetrahedron) of a point $\mathbf{x}$ inside the cell are defined such that:

$$\begin{aligned} \mathbf{x} &= \sum N_i \mathbf{x}_i \\ 1 &= \sum N_i \\ \mathbf{x} = \mathbf{x}_i \Leftrightarrow & \begin{cases} N_j = 1 & i = j \\ N_j = 0 & \text{otherwise} \end{cases} \end{aligned}$$

The values of the barycentric coordinates L_i for the point $\mathbf{x}$ inside the cell can be obtained by solving the equation:

$$\begin{pmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} N_0 \\ N_1 \\ N_2 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{x} \qquad \text{(triangle)}$$

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ z_0 & z_1 & z_2 & z_3 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} N_0 \\ N_1 \\ N_2 \\ N_3 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \mathbf{x} \qquad \text{(tetrahedron)}$$

Finally, to obtain the interpolated value at $\mathbf{x}$ of a field defined on the nodes through values v_i, the barycentric coordinates N_i can be used directly:

$$v = \sum N_i v_i \tag{2.3}$$

Multi-Linear Interpolation is used where linear interpolation using barycentric coordinates is not available, e.g., with quadrilaterals and cubes. The general approach is to define shape functions on a reference element that is defined on a specific interval (here: $[0,1]$) in each dimension, resulting in a parameterization of the reference element. In two-dimensional quadrilaterals, two parameters are used ($\mathbf{p} = (r,s)$); three-dimensional hexahedra require three parameters ($\mathbf{p} = (r,s,t)$).

The weights N_i of each node $\mathbf{x}_i$ are calculated for a given point $\mathbf{p}$ within the reference element by evaluating the shape functions defined for the cell type. For the four corner nodes of a two-dimensional quadrilateral cell, bi-linear interpolation is used with the shape functions:

$$\mathbf{p} = (r,s) \qquad \begin{aligned} N_0 &= (1-r)\cdot(1-s) \\ N_1 &= \quad r \quad \cdot(1-s) \\ N_2 &= \quad r \quad \cdot \quad s \\ N_3 &= (1-r)\cdot \quad s \end{aligned} \tag{2.4}$$

For a three-dimensional hexahedron, which has eight corner nodes, the tri-linear shape functions are defined as:

$$\mathbf{p} = (r,s,t) \qquad \begin{aligned} N_0 &= (1-r)\cdot(1-s)\cdot(1-t) \\ N_1 &= \quad r \quad \cdot(1-s)\cdot(1-t) \\ N_2 &= \quad r \quad \cdot \quad s \quad \cdot(1-t) \\ N_3 &= (1-r)\cdot \quad s \quad \cdot(1-t) \\ N_4 &= (1-r)\cdot(1-s)\cdot \quad t \\ N_5 &= \quad r \quad \cdot(1-s)\cdot \quad t \\ N_6 &= \quad r \quad \cdot \quad s \quad \cdot \quad t \\ N_7 &= (1-r)\cdot \quad s \quad \cdot \quad t \end{aligned} \tag{2.5}$$

The shape functions for other cell types, such as pyramids and wedges, have similar definitions.

When given a parameter $\mathbf{p}$, the interpolated quantity v, defined as v_i on the nodes, can be obtained in a straightforward manner according to Eq. 2.3. However, to interpolate at a given spatial position $\mathbf{x}$, the parameter $\mathbf{p}$ has to be found first. In the special case of a regular grid, where the node positions are defined through their indices, and all cells are axis-aligned rectangles (2D) or cuboids (3D), this can be achieved easily. Given the mesh origin $\mathbf{x}_0$ and spacing $\mathbf{d}$, both the correct cell indices **cellid** and the position $\mathbf{p}$ in the reference element can be determined:

$$\begin{aligned} \mathbf{cellid} &= \left\lfloor \frac{\mathbf{x}-\mathbf{x}_0}{\mathbf{d}} \right\rfloor \\ \mathbf{p} &= \mathbf{cellid} - \frac{\mathbf{x}-\mathbf{x}_0}{\mathbf{d}} \end{aligned}$$

In the general case, where cells may not be rectangular or cuboid, finding the parameter $\mathbf{p}$ is a bit more involved. Finding a correct value of $\mathbf{p}$ requires solving the equation

$$\begin{pmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 \end{pmatrix} \begin{pmatrix} N_0 \\ N_1 \\ N_2 \\ N_3 \end{pmatrix} = \mathbf{x}.$$

Together with the definitions of the shape functions (Eqs. 2.4, 2.5), this is a non-linear system that can be solved approximately using Newton's method.

2.2.3 Cell Location

To obtain the value of a field at an arbitrary location, values at the vertices of a cell that contains the location are interpolated. In regular grids, the indices of the cell that contains the given location can be trivially identified thanks to the grid's implicit topology:

$$\mathbf{cellid} = (i, j, k) = \left\lfloor \frac{(\mathbf{x} - \mathbf{x}_0)}{\mathbf{d}} \right\rfloor$$

In non-regular structured grids and in unstructured grids, identifying the cell containing a given location $\mathbf{x}$ is a bit more involved. Naïvely, one could simply iterate over all cells and test whether or not $\mathbf{x}$ is contained within the cell. This approach becomes extremely costly when used in combination with algorithms that require many evaluations, such as the computation of trajectories. Instead, spatial subdivision methods using tree structures are employed to identify the correct cells.

An early approach uses *octrees* [WV92], where each leaf of the tree stores a set of indices representing all cells that intersect the leaf. Leaves are subdivided until a certain criterion, such as the number of cells per leaf not exceeding a threshold, is fulfilled. To locate the cell that contains a given point $\mathbf{x}$, one first traverses the octree to find the leaf that contains $\mathbf{x}$. If the leaf is associated with multiple cells, $\mathbf{x}$ is checked against each of these cells. Due to the nature of octrees, such an approach can require an unnecessarily large number of subdivisions (and thereby increase the storage overhead) or, depending on the leaf size, fail to adequately narrow down the number of cell candidates that have to be checked individually.

Oriented bounding box trees (*OBBTrees*) [GLM96] offer a different approach: They compute a tree structure of nested bounding boxes, which are not necessarily aligned with the coordinate axes of the domain.

Another, more recent spatial subdivision approach is the *cell tree* [GJ10], which is based on the idea of bounding interval hierarchies. In this spatial subdivision structure, nodes can overlap spatially, which makes it possible to avoid storing objects in multiple leaves. Cell trees are utilized in the implementations of the techniques introduced and discussed in this thesis.

2.2.4 Time-Varying Data

In the output of flow simulations, time is represented by storing the values of the involved fields as a sequence of time steps:

$$\{t_0, \ldots, t_n\}, \quad t_i < t_{i+1}$$

To obtain values of a quantity $\mathbf{u}(\mathbf{x},t)$ at a location $\mathbf{x}$ and for a specific point in time t, which may not match any of the available time steps, interpolation between time steps is used. First, the two time steps enclosing t are identified, so that $t_i \leq t < t_{i+1}$. Then, the quantity is evaluated at the location $\mathbf{x}$ using spatial interpolation in both time steps t_i and $t_i + 1$, resulting in two values $\mathbf{u}_{t_i}$ and $\mathbf{u}_{t_i+1}$. Finally, the value at time t is obtained using linear interpolation:

$$L_{\mathbf{x},t}\mathbf{u}(\mathbf{x},t) = \frac{t - t_i}{t_{i+1} - t_i} L_{\mathbf{x}}\mathbf{u}_{\mathbf{t_i}}(\mathbf{x}) + \frac{t_{i+1} - t}{t_{i+1} - t_i} L_{\mathbf{x}}\mathbf{u}_{\mathbf{t_{i+1}}}(\mathbf{x})$$

In practice, output written by a simulation is often sub-sampled in time. Because the available computing power on supercomputers often outpaces storage bandwidth [Chi+10], and to reduce the requirement for storage overall, only a fraction of the computed simulation time steps is stored in the output. Such practice is appropriate for the original purpose of providing restart points, records of the simulation state at a given point in time from which the simulation can be restarted after it is interrupted for any reason. For visualization, however, the omission of time steps can compromise the accuracy of trajectories that are computed based on the essentially incomplete velocity data. To mitigate this issue, Lagrangian storage methods that can take advantage of in situ computing capabilities have been studied in the past [Agr+14].

2.3 Integral Curves

Integral curves have long served as powerful tools for intuitive visualizations [McL+09]. They are closely related to lines of smoke that are used in real-world wind tunnel experiments to identify flow features [Tho+04]. Three basic types of integral curves are commonly used for visualization: path lines, streak lines, and time lines. A path line describes the trajectory of a (hypothetical) massless particle, which was released into the flow (seeded) at a specific time and location. Streak lines and time lines can be defined as the union of infinitesimally close path line end points. If new path lines are continuously seeded at the same location for a certain duration, the union of their endpoints forms a streak line. This makes streak lines a close approximation of smoke lines used in real-world experiments. In contrast, if the path lines were seeded at the same time along a pre-defined seeding curve, their endpoints form

a time line. For time-invariant flow fields, where the velocity is constant over time, path lines and streak lines are identical. In this special case, they are called streamlines. The straight-forward, intuitive interpretation of integral curves allows comparisons to classic flow visualization methods that have been used in physical experiments, e.g., lines of smoke in wind tunnels.

2.3.1 Mathematical Definition

A path line or trajectory $\mathbf{x}(t)$ is a curve defined by a spatial and temporal starting point $\mathbf{x}_0$, t_0 and by the property of being tangential to the velocity field at each point:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{u}(\mathbf{x},t), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{2.6}$$

According to the Picard-Lindelöf theorem, a solution of ordinary differential equations of this form exists and is unique if the velocity field $\mathbf{u}(\mathbf{x},t)$ is Lipschitz-continuous [Har02]:

$$\exists M \quad \left\|\mathbf{u}(\mathbf{x},t) - \mathbf{u}(\mathbf{x}',t)\right\| \leq M \left\|\mathbf{x} - \mathbf{x}'\right\| \quad \forall \mathbf{x}' \text{ in a neighborhood of } \mathbf{x}$$

For the purpose of visualization, where velocity fields are defined through multi-linear interpolation over a set of cells, this property holds in practice.

2.3.2 Numerical Approximation

To obtain a path line for a given velocity field, the ordinary differential equation (2.6) is typically solved using numerical integration methods.

The Euler Method is arguably the simplest and most straight-forward method to approximate the solution of Equation 2.6. Beginning at the starting point $\mathbf{x}_0$, new positions along the path line are found by adding the velocity vector at the current position, multiplied by a step width factor h, to the current position:

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\mathbf{u}(\mathbf{x}_t,t) \tag{2.7}$$

Effectively, the path line position is advanced through time by adding straight line segments, each of which is tangential to the velocity field in its starting point. Unfortunately, this property is the major limitation of the Euler method, since changes of the velocity field along the straight line segments in space or in time are not incorporated, causing the calculated path line to drift in relation to the theoretical, exact solution. Because the position of each

$$
\begin{array}{c|cccc}
0 & & & & \\
c_2 & a_{21} & & & \\
c_3 & a_{31} & a_{32} & & \\
\vdots & \vdots & & \ddots & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} \quad b_s
\end{array}
\qquad
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

Table 2.1: The coefficients of a Runge-Kutta method can be arranged in a so-called Butcher tableau, which makes them easier to document and implement. The example on the right describes a fixed step size, fourth-order Runge-Kutta method (RK4).

point on the path line depends on the positions of previous points, these errors accumulate and get worse over time. The integration error is closely related to the value of the step with h; Chapter 3 discusses estimates for the upper bounds of the errors of one-step integration methods such as the Euler method. Generally speaking, choosing a smaller value for h results in a smaller integration error. In turn, very small values for h require a large number of integration steps, each of which requires the velocity to be evaluated at a certain position and time. Depending on the representation of the flow field, the large number of velocity field evaluations can greatly increase the computational cost.

Runge-Kutta Methods are a widely used generalization and extension of the Euler method [Pre+07]. The basic idea is to incorporate multiple values of the vector field in a single step to increase accuracy.

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\sum_{i=1}^{s} b_i k_i \tag{2.8}$$

$$
\begin{aligned}
k_1 &= \mathbf{u}(\mathbf{x}_t, t) \\
k_2 &= \mathbf{u}(\mathbf{x}_t + h(a_{21}k_1), t + c_2 h) \\
k_3 &= \mathbf{u}(\mathbf{x}_t + h(a_{31}k_1 + a_{32}k_2), t + c_3 h) \\
&\vdots \\
k_s &= \mathbf{u}(\mathbf{x}_t + h(a_{s1}k_1 + a_{s2}k_2 + \cdots + a_{s,s-1}k_{s-1}), t + c_s h)
\end{aligned}
$$

The coefficients a_i, b_i, c_i can be arranged using so-called Butcher tableaus. Table 2.1 shows the tableau for a fixed step width Runge-Kutta method of fourth order that is sometimes referred to as *the* Runge-Kutta method.

Adaptive Integration Methods In practice, there is usually no obvious choice for the step width h of a numerical integration scheme. Choosing a large value minimizes the computational cost, but may result in low accuracy in areas with rapidly changing velocities. Conversely, a very small value may improve accuracy in such regions, while performing unnecessarily many steps in other regions. A common solution to this predicament is presented by adaptive integration methods, where the step size is adapted based on a local estimate of the integration error. The basic approach is to perform the same integration step using two integration schemes of different expected accuracy. If the difference between the results exceeds a pre-defined upper threshold, the step is repeated with a smaller step size. If, on the other hand, the difference is below another pre-defined lower threshold, the step size is increased in the next step. The result is an integration method that controls its own step size while keeping the estimated error within pre-defined acceptable limits.

One very straight-forward adaptive integration scheme is step doubling, where the result of a single step is compared to the result of two steps with half the step size. This approach has the disadvantage that a large number of function evaluations are required, which makes the method slow [Pre+07].

In contrast, the Runge-Kutta-Fehlberg method (often referred to as RKF or RK4(5)) consists of a fifth-order method with the convenient property that the same function evaluations can be used to construct a fourth order method [Feh69]. Another adaptive Runge-Kutta method with similar properties but improved error behavior was proposed by Dormand and Prince [DP80]. Their method, often referred to as DOPRI5, was used to compute all trajectories in this work. Specifically, our implementation of the DOPRI5 method is based on the one presented by Press et al. [Pre+07].

Even though integral curves have proven a powerful tool for flow visualization, they can suffer from a number of limitations when applied directly. Small numbers of integral curves may not capture important flow features, while large numbers of curves can lead to visual clutter and occlusion. In three-dimensional settings, the small profile of a curve can make it difficult to place visual depth cues that enable an observer to perceive the shape of the curve.

The trajectories represented by integral curves can provide a building block for more complex and powerful visualization methods. Over the years, there have been multiple approaches that incorporate trajectories to produce visualizations. Depth perception in three-dimensional curves can be improved using illumination [ZSH96; Mat+03]; alternatively, curves can be displayed as shaded tubes [Dic89]. Carefully selected seed points can help reduce clutter [TB96]. Surveys reveal that the majority of current state-of-the-art flow visualization techniques utilize advection [Lar+04; Lar+07; Sal+08b; McL+09; Pob+11; Bra+12].

2.4 Integral Surfaces

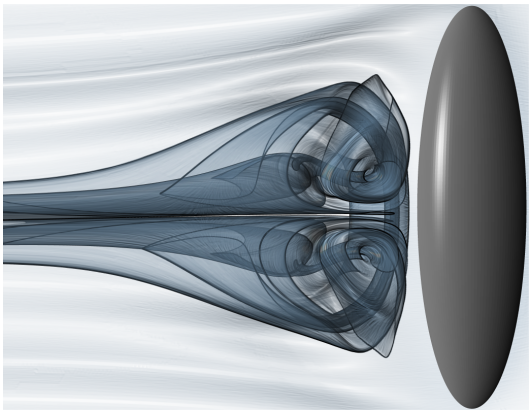

Figure 2.5: Integral surfaces, such as this stream surface depicting the flow behind an ellipsoid, allow for highly intuitive visualization of complex three-dimensional flow features. They are highly dependent on trajectory information.

In comparison to curves, the three-dimensional shape of a surface can be conveyed to an observer with relative ease. With hand-drawn illustrations, the fundamental visual limitations of curves have been avoided by using surfaces to explain complex flow phenomena [Dal83].

In essence, an integral surface is the union of a set of infinitesimally close integral curves. Analogous to integral curves, integral surfaces can be grouped into three basic types: path surfaces, streak surfaces, and time surfaces. While an integral curve is seeded on a single point (path line, streak line) or a seed curve (time line), an integral surfaces uses a seed curve (path line, streak line) or a seed surface (time surface). In the special case of time-invariant flow, path surfaces and streak surfaces for the same seed curve are identical; they are called stream surfaces. Early work on integral surfaces focused on these stream surfaces [Hul90]. The main idea was to seed a large number of stream lines along the seed curves and to connect these curves using triangle strips. Compared to integral curves, a major challenge when computing integral surfaces is the fact that the commonly encountered strong anisotropy of flow fields can cause adjacent integral curves to quickly diverge from each other, resulting in degenerate triangles that impact the visual quality of the surface. This is circumvented by inserting additional stream lines as soon as a triangle strip triggers a refinement criterion, e.g., a distance threshold.

Hultquist's original, recursive formulation of the stream surface algorithm [Hul90], though widely adapted, made it difficult to extend to large, time-varying flow fields or to add more advanced refinement criteria. These limitations were addressed in later work with iterative algorithms for path surfaces [Gar+08] as well as time and streak surfaces [KGJ09].

Compared to integral curves, integral surfaces can be combined with a wider range of visualization techniques such as transparency and textures [Löf+97]. Especially the use of transparency turned out to be necessary due to the self-occluding tendency of the surfaces. With increasing complexity of the visualized flow fields, and therefore increasing complexity of integral surfaces, new rendering methods became necessary to reveal complex flow features in many layers of the surfaces. This was achieved using illustrative techniques such as silhouettes and clever placement of cuts as well as curvature-modulated transparency and stretching-independent texturing [Bor+10; Hum+10](Fig. 2.5).

2.5 Lagrangian Transport

One aspect that is of special interest during the analysis of simulation data is the transport behavior of flow, i.e., information regarding the travel of material that starts out in a specific spatial location or area and is then advected by the flow.

2.5.1 The Finite-Time Lyapunov Exponent (FTLE)

A basic construct useful for the analysis of transport behavior is the *flow map*. For a given time interval $[t_0, t]$, it maps a start position $\mathbf{x}$ to the final position on the corresponding trajectory:

$$F_{t_0}^{t}(\mathbf{x}_0) = \mathbf{x}(t_0 + t), \quad \text{with } \mathbf{x}(t_0) = \mathbf{x}_0 \tag{2.9}$$

Based on a flow map, the Finite Time Lyapunov Exponent is a measure for how quickly the distance between two infinitesimally close particles, released at the same time, can grow [Hal01]. The FTLE value at $\mathbf{x}(t_0)$ for advection time $T \in \mathbb{R}$ is given as the logarithm of the largest singular value of $\nabla\phi$ (which is identical to the square root of the largest eigenvalue of the Cauchy Green tensor $\nabla\phi^T\nabla\phi$):

$$\sigma_{t_0}^{t}(\mathbf{x}(t_0)) = \frac{1}{t - t_0} \log \sqrt{\lambda_{\max}\left(\nabla F_{t_0}^{t}(\mathbf{x}_0)^T \nabla F_{t_0}^{t}(\mathbf{x}_0)\right)}. \tag{2.10}$$

2.5.2 Lagrangian Coherent Structures (LCS)

Even though not used directly in this thesis, the detection of so-called Lagrangian Coherent Structures (LCS) is the main motivation for the computation of the Finite-Time Lyapunov Exponent and a fundamental concept for the visualization of time-varying flow fields.

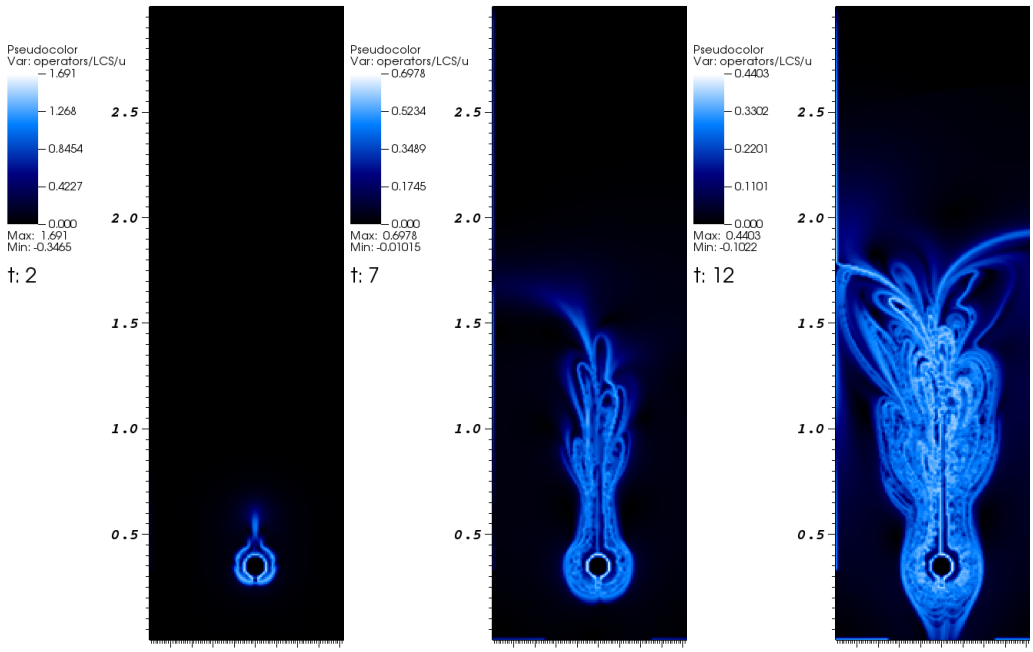

Figure 2.6: Plots of the finite-time Lyapunov exponent (FTLE) provide clues about the transport behavior in a two-dimensional simulation of convection flow around a heated cylinder. Depicted is the FTLE field for the time interval spanning the beginning of the simulation to the end times $t = 2, 7, 12$s.

Years of active research on LCS have resulted in a large body of work both on the theoretical foundations of LCS and on the practical problems of efficient computation of the FTLE field, extraction of LCS, and applications in visualization. This section will provide only a cursory overview of the basic ideas and concepts.

The main property of LCS that makes them useful for visualization is that, in theory, they provide material barriers that are not crossed by flow. This means that two particles that are released into the flow at opposite sides of an LCS will never cross the LCS and therefore remain on separate sides. This property provides useful information about the transport behavior of flow field.

Definitions Shadden et al. [SLM05] define Lagrangian Coherent Structures for finite time intervals as ridges of the FTLE field, or, more precisely, "special gradient lines of the FTLE field that are transverse to the direction of minimum curvature". Instead of directly using a curvature-based formulation, they provide a second, slightly simplified definition, the *second-derivative ridge*. For two-dimensional settings, the ridge is defined as a curve $\mathbf{c}(s)$ that fulfills the requirements of being parallel to the gradient of the FTLE field σ, and that, for each point on the curve, the smaller eigenvector of the Hessian of σ is orthogonal to $\mathbf{c}$ and corresponds to a negative eigenvalue. In effect, the second requirement enforces that

$\mathbf{c}(t)$ follows a local minimum of the second derivative, and, thereby, a ridge of the second derivative.

Shadden et al. then derive an expression of the flux across this curve to verify the material barrier property of the LCS. They find that the flux is negligible for well-defined LCS, and that flux is inversely proportional to the FTLE integration time.

Galilean Invariance and Objectivity In classical mechanics, the principle of *Galilean Invariance* states that the same laws of motion hold under Galilean transformations. A transformation is Galilean if it is affine, preserves intervals of time, and preserves the distance of simultaneous events [Arn89]. Specifically, a Galilean transformation can be written as a combination of a translation, a rotation, and a uniform motion.

Many techniques used for visualization of flow fields are not Galilean invariant. For example, classical topology of time-invariant flow fields is not Galilean invariant; it is highly dependent on the frame of reference [BHJ16]. In contrast, it can be seen that the FTLE field is Galilean invariant: For a fixed time interval, both a constant translation and uniform motion merely result in a constant translation of the trajectory endpoints, while a rotation of the frame causes the endpoints to be rotated. Since the actual calculation uses only the derivatives of the end positions, the translations, being constant offsets, have no influence whatsoever. The rotation does cause a rotation of the Jacobian, which results in rotated eigenvectors of the Cauchy-Green tensor. However, apart from the rotation of the domain itself, this rotation also does not affect the FTLE field, since only the eigenvalues – specifically, the largest eigenvalue – are relevant (Eq. 2.10). Therefore, Lagrangian Coherent Structures, being defined based on the FTLE field, can be considered to be Galilean invariant.

Objectivity, the "requirement that material response be independent of the observer" [Gur81], is a slightly stricter invariance property since it allows for the translation and rotation to change over time. Haller [Hal15] shows that the FTLE field fulfills this requirement, but notes that many techniques for the extraction of LCS do not.

Lagrangian Coherent Structures and Vector Field Topology in Steady Flows For static flow fields, vector field topology (VFT) serves as a powerful visualization tool. Through the identification of critical points and separatrices connecting them, the domain can be subdivided into areas of equivalent flow behavior. At first glance, this indicates a close relation to the material lines and surfaces described by Lagrangian Coherent Structures. In fact, Sadlo [Sad15] observes that the VFT separatrices are sometimes very similar to LCS. However, while LCS are Galilean invariant, the critical points and separatrices forming

classic vector field topology are not. This can easily be seen: Simply adding a constant velocity vector to the flow field causes all critical points, which are defined as points of zero velocity, to vanish. Bujack et al. [BHJ16] approach this problem by considering critical points from a number of different frames of reference, identified through the determinant of the Jacobian.

Efficient Computation of FTLE and LCS Analyzing Lagrangian Coherent Structures in three-dimensional, time-varying flow fields can be computationally expensive. In naïve approaches, large amounts of trajectories are required to achieve sufficient spatial resolutions. Several works have dealt with this issue. Garth et al. [Gar+07b] have proposed an incremental refinement method to reduce the number of required particle trajectories for calculating the FTLE field. In a similar approach, using Adaptive Mesh Refinement (AMR), Sadlo and Peikert [SP07] have proposed a method for efficient, direct extraction of LCS. A hierarchical representation of the FTLE field is also used by Barakat et al. [BGT12]; they introduced a view-based, GPU-accelerated method to accelerate interactive visual analysis of LCS. Sadlo et al. [SRP11] presented a grid-based method to efficiently compute time series of the FTLE field in regions related to predefined regions.

Brunton and Rowley [BR10] accelerate FTLE computation using concatenated, short-time flow maps. Conti et al. [CRK12] discussed how many-core architectures, such as GPUs, can be utilized for fast computation of FTLE fields along with Brunton and Rowley's intermediate flow map approach.

2.6 Uncertainty

The visualization methods discussed so far consider the simulation results as a sort of ground truth: the goal is to present information contained in the available data as closely as possible. In practice, however, any simulation data are already subject to errors or noise in various forms.

One source of uncertainty that appears in many scenarios – such as ocean or weather simulations – is the fact that the initial conditions cannot be known exactly, simply because it is not possible to measure them at a high resolution. In addition, simulation relies on spatial discretizations, which, by their nature, are only an approximation of reality. A detailed overview of sources for uncertainty and error in simulations of real-world physical processes, as well as methods for verification, validation, and uncertainty quantification, can be found in [Nat12]. The field of uncertainty visualization aims to aid analysis and decision

making by making uncertainty and its impact in scientific data sets visible. This section provides an overview of relevant basic concepts and prior work.

Early methods focused on plots and charts of stochastic distributions. A commonly used method for the visualization of distributions is the box plot, which summarizes a one-dimensional distribution using a glyph that encodes the distribution's quartiles [Tuk77]. Multiple extensions and variants for encoding additional information have been presented, for example violin plots, which add a histogram to the glyph [HN98]. Potter et al. introduced one-dimensional summary plots — adding higher-order moments and a distribution fit — and two-dimensional summary plots for encoding multiple, correlated datasets [Pot+10]. Potter et al. also presented an interactive visualization tool for two-dimensional probability density functions [Pot+12].

Statistical and stochastic descriptions of fields or data with numerical errors are often regarded as data with uncertainties [Ger98] and visualized accordingly. Pang et al. [PWL97] and Griethe and Schumann [GS06] give an overview of visualization techniques such as error-bars, pseudo-coloring, and geometry generation. Similarly, Johnson and Sanderson [JS03] investigate existing and future visualization techniques for uncertain data, suggesting improved interaction with statistical data.

Specific to uncertain flow fields, Ferstl et al. [FBW16] presented a method for visualizing the geometry of uncertain streamlines.

An overview of uncertainty visualization methods was provided by Bonneau et al. [Bon+14].

Examining the relation between simulation results and parameter values, Sedlmair et al. [Sed+14] survey a number of existing techniques and arrange them in a conceptual framework for visual parameter space analysis.

Stochastic Modeling and Probabilistic Simulation Uncertainty appears not only in the form of errors and inaccuracies, but can also result from simulations that incorporate a stochastic model. Such probabilistic approaches are often used for processes that may exhibit a large variation of possible behaviors at a small scale, which may be too computationally expensive to simulate explicitly. Examples include stochastic modeling of fibers [Göt+06], bubbly flows in chemical reactors [Hla+16b], or even stochastic models for turbulence [SP10].

In a slightly different direction, stochastic methods can also be utilized to produce specific instances of processes that follow a probabilistic description. For example, synthetic data are used to validate an automatic measurement method for fiber thickness in scanning electron microscopy images [Eas+16]. Another example of such an approach is presented

in Chapter 5 of this thesis, where instances of probabilistically described bubble sets are generated for the purpose of visualization.

3 Error Estimates for a Lagrangian Flow Field Representation

Trajectory information is an integral basis of many visualization methods, especially those meant to analyze material transport. For flow simulation data that are represented in an Eulerian way, i.e., physical quantities are observed over time from the perspective of fixed locations in space, trajectory information has to be obtained through numerical integration. Computing a large number of trajectories using numerical integration directly from the Eulerian velocity field can be computationally expensive. In case of simulations running on a high performance supercomputer, an additional issue arises from the fact that in modern HPC architecture, the computational capacity frequently exceeds the available input/output bandwidth. To avoid spatial subsampling of the Eulerian flow field, which would result in reduced accuracy for visualization methods that focus on single time steps, temporal subsampling is used [Chi+10]. It is, in fact, not uncommon to write only every 100th time step to persistent storage. This situation leads to the concern that, in typical scenarios, the accuracy of computed trajectories may be compromised.

A possible solution lies in Lagrangian representations of the flow field. In contrast to Eulerian representations, a Lagrangian flow field representation directly incorporates the trajectories of individual fluid parcels. Such a representation could be computed in situ, i.e., alongside the simulation on the same high performance computer, leveraging its superior computing power and using every time step of the simulation.

Based on their theoretical definitions, the Eulerian and Lagrangian views are equivalent – both encode velocity information and can be converted into each other. From an Eulerian flow field representation, the trajectories needed for the Lagrangian view can be obtained using integration, while differentiation of the trajectories yields the momentary velocities needed for the Eulerian view. For example, the flow map, a Lagrangian flow field representation that maps the starting positions $\mathbf{x}_0$ of fluid parcels at a given starting time t_0 to their end

positions at a later point in time t, provides trajectories but can also, through differentiation, yield the velocity field:

$$\begin{aligned}(\mathbf{x}_0, t_0) &\mapsto F_{t_0}^{t}(\mathbf{x}_0) \\ \mathbf{x}(t) &= F_{t_0}^{t}(\mathbf{x}_0) \\ \mathbf{u}(\mathbf{x}, t_0) &= \frac{d}{dt} F_{t_0}^{t}(\mathbf{x})\end{aligned} \tag{3.1}$$

Despite this theoretical equivalence, a conversion is not possible in practice without some loss of accuracy. Both Eulerian and Lagrangian representations are typically stored in a discretized form, resulting in numerical errors during integration and differentiation. The resolution of the discretization itself, both in space and time, determines the magnitude of these errors.

In the past, Agranovsky et al. [Agr+14] have shown that, for the purpose of obtaining trajectories, a Lagrangian flow representation constructed by concatenating a number of short-time, single-interval flow maps can result in higher accuracy than a typical, temporally sub-sampled Eulerian representation of the same size. In this representation, long trajectories can be constructed by evaluating the short-time flow maps in sequence, each taking the output of its predecessor as input. Bujack et al. [BJ15] then showed that the trajectory construction process can be interpreted as a one-step numerical integration method. Based on the well-understood error behavior of this class of algorithms, they defined global upper bounds for the local truncation error (the error that arises from performing a single integration step) and the global truncation error (the cumulative error at a certain point in time) for all reconstructed trajectories in a given data set.

Given that typical flow data sets feature a large variety of flow behavior across their temporal and spatial domain, it can be expected that the often large errors predicted by the global error estimate may only occur in relatively few trajectories. Extending on the work of Bujack et al., this chapter introduces a new definition for an upper bound of the error of individual trajectories. Specifically, error measures are introduced that make use of the flow map's spatial and temporal derivatives to estimate the upper bound of specific trajectories' error as they are constructed. An additional error measure is defined for the temporal linear interpolation of the trajectory between adjacent time slices.

To evaluate these measures, they are applied to a number of time-varying flow fields. The estimated error bounds are compared to measurements of the actual errors, obtained by comparing trajectories constructed from the Lagrangian representation to those obtained by numerical integration of a non-subsampled Eulerian representation. In summary, the contributions of this chapter are[1]:

[1] The results described in this chapter were published in [Hum+16].

- *Local measure for the local and global truncation error:* The measures for the truncation errors τ and e described in this chapter make it possible to calculate an upper bound for the error of individual trajectories as they are being constructed using multilinear interpolation from a set of consecutive flow maps.
- *Measure for the intra-interval error of the temporal piecewise linear approximation:* Using the measure for the error s between the time slices, an upper bound for the error of the piecewise linear trajectory representation can be calculated.
- *Approach for approximate calculation of the error measures in practice:* A description of the implementation explains how the theoretically defined error measures can be calculated in practice.
- *Evaluation of the proposed error measures based on example flow fields:* Based on analytically defined and simulated flow data sets, the error measures are evaluated by comparing the calculated, estimated upper bounds to actual measured errors.

The chapter is organized as follows: Section 3.1 will provide a short overview of related work specific to errors of trajectories. The Lagrangian flow field representation is explained in Section 3.2. The theoretical definition of the error estimate is given in Section 3.3. Section 3.4 describes how an implementation can make use of the theoretical estimate to make predictions for actual data sets, followed by a description of the experimental results in Section 3.5. Finally, Section 3.6 provides a conclusion of the findings.

3.1 Related Work

Lagrangian methods have been long used for efficient particle tracer computation [Bru+01], the calculation of Lagrangian Coherent Structures, [HY00; Hal02], or to incorporate flow maps into the Eulerian representation of a flow field [SS06; Sal+08a; JEH02]. Brunton and Rowley [BR10] used concatenated, short-time flow maps to accelerate computation of the Finite-Time Lyapunov Exponent (FTLE) field.

Hlawatsch et al. [HSW11] employ a hierarchical scheme to construct longer integral lines from previously computed partial solutions. Chandler et al. [COJ15] construct trajectories in smoothed particle hydrodynamics (SPH) [Mon05] data sets, which – by their nature – are already represented in a Lagrangian way. Agranovsky et al. [Agr+14] show empirically that the Lagrangian representation described in Section 3.2 is more accurate than Eulerian advection methods. Bujack and Joy [BJ15] extend this idea by providing the first theoretical estimate of the overall error in this type of Lagrangian flow field representations. In this chapter, the findings of Bujack and Joy are refined to derive a local error for each path line,

which can be used for the visualization of the uncertainty of the method as well as for the in situ adjustment of the resolution chosen for storage.

3.2 Interval Flow Maps: A Lagrangian Flow Field Representation

The flow map provides a straight-forward Lagrangian representation of a flow field. For a specific start time t_0 and end time t, this function maps each position $\mathbf{x}_0$ of the spatial domain to the position at time t of a trajectory that passes through $\mathbf{x}_0$ at time t_0 :

$$F_{t_0}^{t_1}(\mathbf{x}_0) = \mathbf{x}(t), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.2}$$

Even though a flow map is, in theory, defined as a continuous function, in practice it is usually approximated by a set of discrete values for a specific time interval $[t_0, t]$. A straight-forward method is to define a regular grid on the spatial domain and then use a numerical integration scheme such as DOPRI5 to calculate a trajectory. The end point of each trajectory is then stored at the node of its starting point. For example, in a three-dimensional, regular grid, each flow map value is associated with its starting node through a triplet of indices:

$$F_{i,j,k} = F_{t_0}^{t}(\mathbf{x}_{i,j,k}) \tag{3.3}$$

Values in between these vertices, within the grid cells, can then be retrieved using multilinear interpolation.

As-is, a single flowmap can only be used to retrieve the trajectory end points. If intermediate points are required, e.g., to reconstruct integral curves, it makes sense to divide the time range $[t_{\text{start}}, t_{\text{end}}]$ into a set of $n+1$ time points $\{t_0 \dots t_n\}$ with $t_0 = t_{\text{start}}$ and $t_n = t_{\text{end}}$ and then provide flow maps mapping to each of these time points. Two basic approaches come to mind:

1. Each flow map maps from the the start time t_0 to its end time t_{i+1}, resulting in the sequence
$$(F_{t_0}^{t_1}, F_{t_0}^{t^2}, \dots, F_{t_0}^{t_n}). \tag{3.4}$$

2. The flow maps cover consecutive time intervals $[t_i \dots t_{i+1}]$, $i = 0 \dots n$, resulting in the sequence
$$(F_{t_0}^{t_1}, F_{t_1}^{t_2}, \dots, F_{t_{n-1}}^{t_n}). \tag{3.5}$$

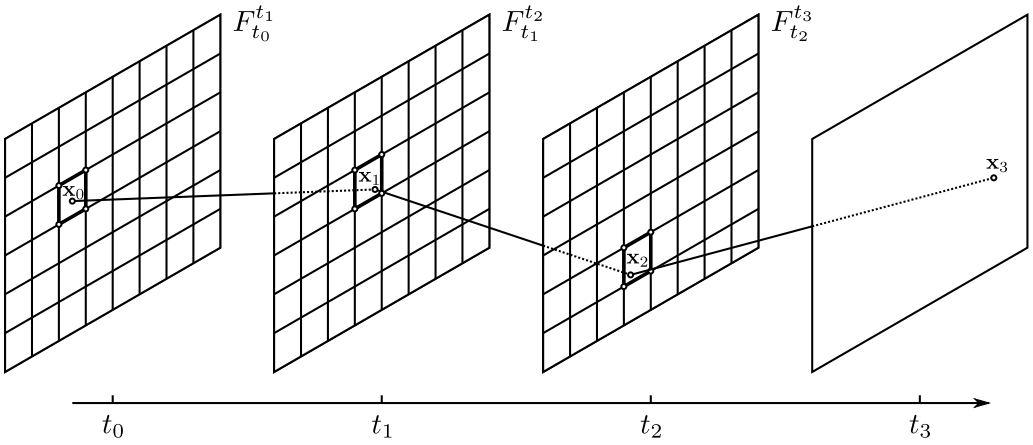

Figure 3.1: Based on a sequence of flow maps F_i^{i+1} spanning the time intervals $[t_i, t_{i+1}]$, each of which is defined on a regular grid, a trajectory is constructed through consecutive evaluation. Starting at $\mathbf{x}_0$, the start position of the trajectory at time t_0, the first flow map F_0^1 is evaluated by interpolating the values stored at the nodes of the surrounding cell (bold). The result, $\mathbf{x}_1$, is in turn used as input for the second flow map F_1^2. After each time interval has been crossed by evaluating its corresponding flow map, the end point $\mathbf{x}_3$ of the trajectory is reached.

At first glance, version 1 appears to be more practical: a piecewise linear path line starting at t_0 from a given seed point $\mathbf{x}_0$ can easily be constructed using the positions

$$\mathbf{x}_i = F_{t_0}^{t_i}(\mathbf{x}_0), \quad i \in [0 \ldots n]. \tag{3.6}$$

However, it immediately becomes clear that this approach only supports path lines which begin at t_0. In addition, since the trajectories of consecutive flow maps increase in length, flow effects such as divergence have stronger influence as i is increased. This can result in accuracy problems since the flow maps are represented by sets of discrete positions.

We therefore choose the second version (Fig. 3.1): by having each flow map cover a relatively small time interval, path lines can be generated starting at any time t_i. Further, since these flow maps are independent from each other, computing them in parallel becomes a trivial task. Constructing a piecewise linear path line requires the same number of steps as with version 1. Here, however, each path line point depends on the previous points:

$$\mathbf{x}_i = (F_{t_{i-1}}^{t_i} \circ F_{t_{i-2}}^{t_{i-1}} \circ \cdots \circ F_{t_0}^{t_1})(\mathbf{x}_0), \quad i \in [0 \ldots n]. \tag{3.7}$$

Because of this dependency, interpolation errors accumulate along the constructed trajectory.

3.3 Theoretical Error Estimate

In this section, we will derive a regional error estimate for the Lagrangian flow representation using the method of concatenating interpolated, short-interval flow maps. Bujack and Joy [BJ15] have shown that this method is a special kind of numerical one-step integration method and have given an overall error estimate. For the entire spatial domain, their global estimate provides a single value that represents the maximum of the error of any possible trajectory. Since the error is exponential in terms of the integration time, the maximal error can grow very fast, owing to large values of the involved derivatives near areas of high divergence.

In practice, many data sets exhibit areas of high divergence only in a portion of the domain, which leads to the expectation that in the remaining areas, errors should remain well below the all-encompassing, global error measure. Based on this assumption, we formulate error measures that make it possible to assess the expected error of the flow field representation for each constructed trajectory individually.

The trajectories, and, as a consequence, the errors themselves, live in two- or three-dimensional space. In order to keep the notation concise, we will define the errors for the one-dimensional case, under the assumption that each component of the actual, multi-dimensional error can be treated separately. For now, this assumption remains unproven; it does, however, open an interesting avenue for future research. Section 3.4 will explain how this one-dimensional definition can be used to estimate the multi-dimensional error.

Trajectories are constructed from the Lagrangian flow representation by interpolating sequential flow maps. The sequence $x(t_0),\dots,x(t_n)$ of exact path line points is approximated by the sequence $x_0,\dots,x_n$ with $x_0 = x(t_0)$. Each point x_j is obtained by interpolating the flow map $F_{t_{j-1}}^{t_j}$ at the previous point x_{j-1}.

We will first show how this construction method can be viewed as a one-step integration method before we describe the error measure.

3.3.1 One-step Integration Method

In numerics of ordinary differential equations [Sch02; GH10], for a given initial value problem

$$x'(t) = f(t,x(t)), \quad x_0 = x(t_0), \tag{3.8}$$

a one-step integration method produces a sequence of points $x_0,\dots,x_n$ by the rule

$$x_j = S(x_{j-1}) = x_{j-1} + hA(t_{j-1},x_{j-1},f,h), \tag{3.9}$$

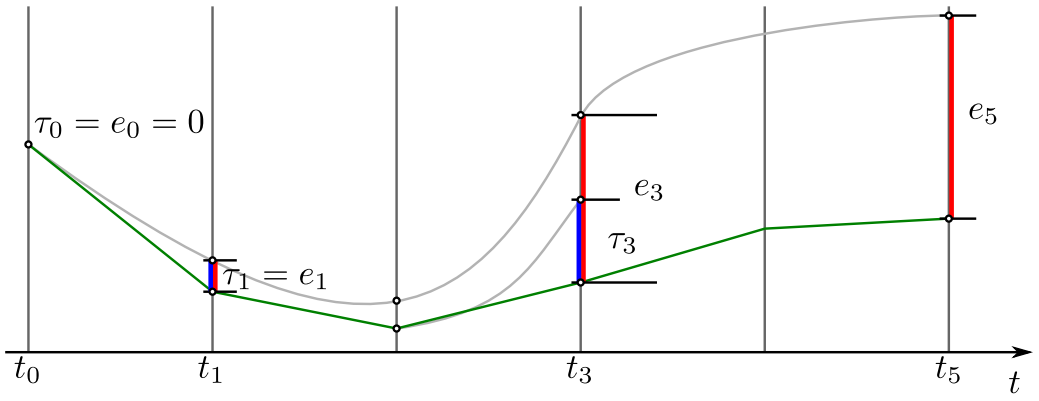

Figure 3.2: A one-dimensional example is used to illustrate the difference between the two types of truncation errors, describing how the approximated trajectory (green) diverges from the correct trajectory (gray). The local truncation error τ_i (blue) is the additional error incurred when crossing the time interval $[t_{i-1}, t_i]$, while the global truncation error e_i (red) is the cumulative error incurred at all intervals up to t_i. Not depicted in the illustration is how the error is caused by the interpolation.

where S is the rule to advance from one step to the next and A is the increment function. It can be seen that patching together the interpolated trajectories is a special kind of numerical one-step integration method with increment function

$$A(t_{j-1}, x_{j-1}, f, h) = L_x \dot{F}^t_{t_{j-1}}(x_{j-1}) \tag{3.10}$$

from using the Taylor expansion [GL10] with respect to time

$$\begin{aligned} L_x F^{t_j}_{t_{j-1}}(x) &\overset{\text{Taylor}}{=} L_x F^{t_{j-1}}_{t_{j-1}}(x) + h_t L_x \dot{F}^t_{t_{j-1}}(x) \\ &= x + h_t L_x \dot{F}^t_{t_{j-1}}(x), \end{aligned} \tag{3.11}$$

where the dot represents the temporal derivative with respect to the end time and $t \in [t_{j-1}, t_j]$.

3.3.2 Local Truncation Error

The local truncation error τ_j for a step j and an exact position $x(t_{j-1})$ is defined by

$$\tau_j(x(t_{j-1})) := |x(t_j) - S(x(t_{j-1}))| \tag{3.12}$$

and describes the error that the one-step method makes advancing one step. For our method, the local truncation error is the error that we make within each of the stored time slices when we interpolate the endpoints of the spanning trajectories in space to generate trajectories

starting at arbitrary spatial locations. It coincides with the linear interpolation error [HH12] of a point in the interval $x \in [x_0, x_1]$, which follows from the Taylor series

$$\begin{aligned}\tau_j(x) =& |L_x F^{t_j}_{t_{j-1}}(x) - F^{t_j}_{t_{j-1}}(x)| \\ \overset{\text{Taylor}}{\leq}& \frac{1}{2}(x_1 - x)(x - x_0) \max_{\zeta \in [x_0, x_1]} |(F^{t_j}_{t_{j-1}})''(\zeta)|.\end{aligned} \tag{3.13}$$

3.3.3 Global Truncation Error

The global truncation error e_j up to the step j is defined by

$$e_j(x_0) := |x(t_j) - x_j|. \tag{3.14}$$

The notation $e_n(x_0)$ was chosen to stress that it is valid for the sequence $x_0, \ldots, x_n$ that starts at x_0. It suffices

$$\begin{aligned}e_j(x_0) \overset{(3.14)}{=}& |x(t_j) - x_j| \\ \leq& |x(t_j) - S(x(t_{j-1}))| + |S(x(t_{j-1})) - x_j| \\ \overset{(3.12)}{=}& \tau_j(x(t_{j-1})) + |S(x(t_{j-1})) - x_j| \\ \overset{(3.9)}{=}& \tau_j(x(t_{j-1})) + |S(x(t_{j-1})) - S(x_{j-1})| \\ \overset{(3.9)}{=}& \tau_j(x(t_{j-1})) + |x(t_{j-1}) + hA(t_{j-1}, x(t_{j-1}), f, h) \\ & - x_{j-1} - hA(t_{j-1}, x_{j-1}, f, h)| \\ \overset{(3.14)}{=}& \tau_j(x(t_{j-1})) + e_{j-1} \\ & + h|A(t_{j-1}, x(t_{j-1}), f, h) - A(t_{j-1}, x_{j-1}, f, h)|.\end{aligned} \tag{3.15}$$

As is typically done in the analysis of numerical methods for solving ordinary differential equations, the overall maximal error can be derived by using the global maximum of τ_j and the global Lipschitz constant L:

$$\forall t \in \mathbb{R}, \forall x, y \in \mathbb{R}^n : |A(t, x, f, h) - A(t, y, f, h)| \leq L|x - y|. \tag{3.16}$$

In this application, however, this coarse approximation is not sufficient. Instead, an approximation of the error for the specific integration sequence $x_0, \ldots, x_n$, corresponding to a single trajectory, is needed. This is achieved by not looking at the maximal possible errors across the entire domain, but at the maximal possible errors that are expected to occur for the area around the point in the sequence x_{j-1} that is big enough to contain the true value $x(t_{j-1})$. Since an upper bound for the global truncation error of the trajectory at the previous step, $e_{j-1}(x_0)$, is already available, this upper bound can be used to define the

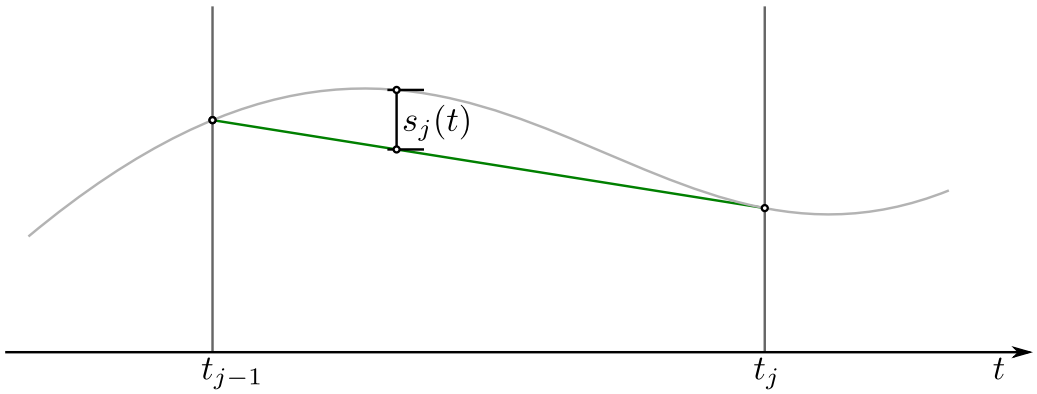

Figure 3.3: The intra-interval error s, defined for every point in time within the interval, is caused by temporal linear interpolation of the constructed trajectory. The illustration does not show the truncation errors τ and e.

search area. Without depending on the true value $x(t_{j-1})$, the regional Lipschitz constant $L(x_{j-1})$ sufficing

$$\begin{aligned} &\forall x,y \in [x_{j-1}-e_{j-1}, x_{j-1}+e_{j-1}]: \\ &|A(t_{j-1},x,f,h)-A(t_{j-1},y,f,h)| \leq L(x_{j-1})|x-y|, \end{aligned} \tag{3.17}$$

as well as the regional value of the local truncation error $\bar{\tau}_j(x_{j-1})$ that bounds $\tau_j(x(t_{j-1})$ from above

$$\bar{\tau}_j(x_{j-1}) := \max_{x\in[x_{j-1}-e_{j-1}, x_{j-1}+e_{j-1}]} \tau_j(x) \tag{3.18}$$

are used. Since this value depends on the global truncation error $e_{j-1}(x_0)$ calculated thus far along the trajectory, it can – unlike the local truncation error (3.12) – not be calculated from only looking at one time step.

Finally, the global truncation error for the trajectory starting at x_0 is given by

$$\begin{aligned} e_j(x_0) &\overset{(3.15)}{\leq} \tau_j(x(t_{j-1})) + e_{j-1} \\ &\quad + h|A(t_{j-1},x(t_{j-1}),f,h) - A(t_{j-1},x_{j-1},f,h)| \\ &\overset{(3.17),(3.18)}{\leq} \bar{\tau}_j(x_{j-1}) + e_{j-1} + hL(x_{j-1})|x(t_{j-1}) - x_{j-1}| \\ &\overset{(3.14)}{=} \bar{\tau}_j(x_{j-1}) + e_{j-1} + hL(x_{j-1})e_{j-1} \\ &= \bar{\tau}_j(x_{j-1}) + e_{j-1}(1 + hL(x_{j-1})). \end{aligned} \tag{3.19}$$

This is the main theoretical result of this chapter.

3.3.4 Intra-Interval Error

The truncation errors calculated so far represent the errors of the sequence $x_0, \ldots, x_n$. A continuous curve is constructed from the sequence through linear interpolation for each interval. This temporal interpolation causes additional error between the stored time steps t_j.

$$L_t F^t_{t_{j-1}}(x_{j-1}) = \frac{(t - t_{j-1})x_j + (t_j - t)x_{j-1}}{t_j - t_{j-1}}. \tag{3.20}$$

Due to the uncertainty of the one-step points, the additional approximation error for arbitrary times $t \in [t_{j-1}, t_j]$ is then given by

$$\begin{aligned} s_j(x_0, t) :=& |L_t F^t_{t_{j-1}}(x_{j-1}) - F^t_{t_{j-1}}(x_{j-1})| \\ \leq& \max_{x \in [x_{j-1} - e_{j-1}, x_{j-1} + e_{j-1}]} |L_t F^t_{t_{j-1}}(x) - F^t_{t_{j-1}}(x)| \\ \overset{\text{Taylor}}{\leq}& \max_{x \in [x_{j-1} - e_{j-1}, x_{j-1} + e_{j-1}]} \\ & \frac{1}{2}(t - t_{j-1})(t_j - t) \max_{\theta \in [t_{j-1}, t_j]} |\ddot{F}^\theta_{t_{j-1}}(x)|. \end{aligned} \tag{3.21}$$

It is added to the error (3.19) to approximate the absolute error of the Lagrangian representation

$$a_j(x_0, t) := e_j(x_0) + s_j(x_0, t). \tag{3.22}$$

3.4 Application to Discrete Flow Data

The theoretical description of the truncation errors τ and e provided in Section 3.3 shows how an upper bounds of these errors can be defined for the Lagrangian interval flow map representation outlined in Section 3.2. In practical applications, the theoretical estimates are calculated based on derivatives that have to be approximated from data known only at discrete positions. Depending on whether the method is used in situ (alongside the simulation) or post-hoc, different amounts of information are available for computing the required derivatives. It might make sense to compute high-resolution flow maps in situ, utilize it to calculate the error estimates, and then write a sub-sampled version to storage. This section shows how the error estimates can take the possibly reduced resolution of the stored flow maps into account while using a high-resolution flow map for calculation. It is also shown how a less accurate estimate can be calculated post-hoc, where only the lower-resolution flow maps are available.

In the following, the high temporal and spatial resolutions of the simulation are denoted by h_t^{sim} and h_x^{sim}; The lower resolutions of the data, as they are written to storage, are referred

to as h_t and h_x respectively. Further, x_i^{sim} are the full resolution grid points of the simulation, x_i the coarse grid points that are stored on disk, and especially x_0, x_1 the two closest coarse grid points that span the interval containing an arbitrary point x.

3.4.1 Local Truncation Error

In-situ, we can use the points x_i on the high resolution grid to approximate the second derivative in the formula for the local truncation error (3.13) using the second order finite differences

$$|(F_{t_{j-1}}^{t_j})''(x_i^{sim})| \approx \frac{|F_{t_{j-1}}^{t_j}(x_{i-1}^{sim}) - 2F_{t_{j-1}}^{t_j}(x_i^{sim}) + F_{t_{j-1}}^{t_j}(x_{i+1}^{sim})|}{{h_x^{sim}}^2} \tag{3.23}$$

with $x_{i\pm1}^{sim}$ being the left and right neighbor of grid point x_i. We can also just use these grid points to evaluate the regional local truncation error

$$\begin{aligned} \bar{\tau}_j(x_{j-1}) &\overset{(3.18)}{=} \max_{x\in[x_{j-1}-e_{j-1}, x_{j-1}+e_{j-1}]} \tau_j(x) \\ &\approx \max_{x_i^{sim}\in[x_{j-1}-e_{j-1}, x_{j-1}+e_{j-1}]} \tau_j(x_i^{sim}) \\ &\overset{(3.13)}{\approx} \frac{1}{2} \max_{x_i^{sim}\in[x_{j-1}-e_{j-1}, x_{j-1}+e_{j-1}]} \\ &\quad (x_1 - x_i^{sim})(x_i^{sim} - x_0)|(F_{t_{j-1}}^{t_j})''(x_i^{sim})|. \end{aligned} \tag{3.24}$$

Post-hoc, we can only use the grid points that were stored to disk to approximate the derivative

$$|(F_{t_{j-1}}^{t_j})''(x_i)| \approx \frac{|F_{t_{j-1}}^{t_j}(x_{i-1}) - 2F_{t_{j-1}}^{t_j}(x_i) + F_{t_{j-1}}^{t_j}(x_{i+1})|}{{h_x}^2}. \tag{3.25}$$

We are then faced with two options for the construction of the regional local truncation error. If we want a coarse, quick approximation, we can use

$$(x_1 - x)(x - x_0) \leq \frac{h_x^2}{4} \tag{3.26}$$

to get

$$\bar{\tau}_j(x_{j-1}) \overset{(3.18)}{\approx} \frac{h_x^2}{4} \max_{x_i\in[x_{j-1}-e_{j-1}, x_{j-1}+e_{j-1}]} |(F_{t_{j-1}}^{t_j})''(x_i)| \tag{3.27}$$

and if we prefer a more exact, but computationally more costly estimate, we can linearly interpolate the values of the derivative (3.25) in an arbitrarily higher resolution grid x_i^{high} and evaluate the factors for $\bar{\tau}_j(x_{j-1})$ on that grid by

$$\begin{aligned}\bar{\tau}_j(x_{j-1}) \approx & \frac{1}{2} \max_{x_i^{high} \in [x_{j-1}-e_{j-1}, x_{j-1}+e_{j-1}]} \\ & (x_1 - x_i^{high})(x_i^{high} - x_0)\big((x_i^{high} - x_0)|(F_{t_{j-1}}^{t_j})''(x_1)| \\ & + (x_1 - x_i^{high})|(F_{t_{j-1}}^{t_j})''(x_0)|\big).\end{aligned} \tag{3.28}$$

3.4.2 Regional Lipschitz constant

The localized Lipschitz constant of the increment function can be estimated by

$$\begin{aligned}L(x_{j-1}) &\overset{(3.10)}{\leq} \max_{x_1,x_2 \in [x_{j-1}-e_{j-1}, x_{j-1}+e_{j-1}]} \max_{t \in [t_{j-1}, t_j]} \\ &\qquad \frac{|L_x \dot{F}_{t_{j-1}}^{\tau}(x_1) - L_x \dot{F}_{t_{j-1}}^{\tau}(x_2)|}{|x_1 - x_2|} \\ &\leq \max_{x_1,x_2 \in [x_{j-1}-e_{j-1}, x_{j-1}+e_{j-1}]} \max_{t \in [t_{j-1}, t_j]} \\ &\qquad \frac{|\dot{F}_{t_{j-1}}^{t}(x_1) - \dot{F}_{t_{j-1}}^{t}(x_2)|}{|x_1 - x_2|}.\end{aligned} \tag{3.29}$$

Due to the piecewise linear interpolation of the flow field, it is sufficient to look for the maximal regional Lipschitz constant between neighboring grid points in space.

During the simulation, when still all time steps t^{sim} with the high resolution of h_t^{sim} are known, the Lipschitz constant L can be approximated by the temporally discrete formula

$$\begin{aligned}L(x_{j-1}) &\overset{(3.29)}{\approx} \max_{x_i^{sim} \in [x_{j-1}-e_{j-1}, x_{j-1}+e_{j-1}]} \max_{t^{sim} \in [t_{j-1}, t_j]} \\ &\qquad \frac{|\dot{F}_{t_{j-1}}^{t^{sim}}(x_i^{sim}) - \dot{F}_{t_{j-1}}^{t^{sim}}(x_{i-1}^{sim})|}{h_x^{sim}}.\end{aligned} \tag{3.30}$$

In contrast, when performing post-hoc analysis, the values of the flow field between the stored time steps are no longer known. We can, therefore, only approximate L using the values at the stored time slices t_j and the stored grid points x_i

$$L(x_{j-1}) \overset{(3.29)}{\approx} \max_{x_i \in [x_{j-1}-e_{j-1}, x_{j-1}+e_{j-1}]} \frac{|\dot{F}_{t_{j-1}}^{t_j}(x_i) - \dot{F}_{t_{j-1}}^{t_j}(x_{i-1})|}{h_x}. \tag{3.31}$$

Depending on the type of the simulation, the derivative $\dot{F}_{t_{j-1}}^{t^{sim}}(x_i)$ would be known in situ, because it is exactly the flow velocity $u(x_i^{sim}, t^{sim})$ at position x_i^{sim} and time t^{sim}. In the post-hoc analysis, however, the velocity field is not known exactly, but can be approximated using finite differences up to an error of order $O(h_t)$ as

$$\dot{F}_{t_{j-1}}^{t_j}(x_i) \approx \frac{F_{t_{j-1}}^{t_j}(x_i) - F_{t_{j-1}}^{t_{j-1}}(x_i)}{h_t}. \tag{3.32}$$

3.4.3 Error Within the Sections

During the simulation, we can approximate the second derivative with respect to time in (3.21) at each of the high resolution grid points x_i^{sim} and high resolution time steps t^{sim} using the second order finite differences

$$\begin{aligned}\ddot{F}_{t_{j-1}}^{t^{sim}}(x_i^{sim}) \approx &\frac{1}{(h_t^{sim})^2}(F_{t_{j-1}}^{t^{sim}-h_t^{sim}}(x_i^{sim}) \\ &- 2F_{t_{j-1}}^{t^{sim}}(x_i^{sim}) + F_{t_{j-1}}^{t^{sim}+h_t^{sim}}(x_i^{sim})).\end{aligned} \tag{3.33}$$

If the velocities are known in situ, we can use the symmetric finite differences of first order instead of (3.33)

$$\begin{aligned}\ddot{F}_{t_{j-1}}^{t^{sim}}(x_i^{sim}) =& \dot{u}(x_i^{sim}, t^{sim}) \\ \approx& \frac{|u(x_i^{sim}, t^{sim} + h_t^{sim}) - u(x_i^{sim}, t^{sim} - h_t^{sim})}{2h_t^{sim}}.\end{aligned} \tag{3.34}$$

Either approximation can the be used to evaluate the formula (3.21) at the in situ time steps and grid points

$$\begin{aligned}s_j(x_0, t^{sim}) \approx &\frac{1}{2}(t^{sim} - t_{j-1})(t_j - t^{sim}) \max_{\theta^{sim} \in [t_{j-1}, t_j]} \\ &\max_{x_i^{sim} \in [x_{j-1} - e_{j-1}, x_{j-1} + e_{j-1}]} |\ddot{F}_{t_{j-1}}^{\theta^{sim}}(x_i^{sim})|.\end{aligned} \tag{3.35}$$

Again, during post-hoc analysis, the values of the flow map between the stored time steps are not known. Due to the structure of the sectional spanning trajectories, we do not have three consecutive points on one trajectory either. That means we can only approximate the second derivative using the values on the reconstructed path lines

$$\begin{aligned}\ddot{F}_{t_{j-1}}^{t_j}(x_{j-1}) \approx &\frac{F_{t_{j-1}}^{t_{j+1}}(x_{j-1}) - 2F_{t_{j-1}}^{t_j}(x_{j-1}) + F_{t_{j-1}}^{t_{j-1}}(x_{j-1})}{{h_t}^2} \\ \approx &\frac{x_{j+1} - 2x_j + x_{j-1}}{{h_t}^2}.\end{aligned} \tag{3.36}$$

As a result for the visualization of the error within the sections, we can choose arbitrary sample points $t \in \mathbb{R}$ to evaluate (3.21)

$$s_j(x_0,t) \approx \frac{(t-t_{j-1})(t_j-t)}{2{h_t}^2} \max_{k\in\{j-1,j\}} |x_{k+1}-2x_k+x_{k-1}|. \tag{3.37}$$

3.5 Experimental Results

Multiple data sets were chosen to evaluate the theoretical error measures presented in this chapter. We used a Runge-Kutta integration scheme to compute sets of consecutive flow maps using the high in situ resolution for each scenario. The trajectory endpoints were then stored on a regular grid, as explained in Section 3.2. Then, a number of seed points were chosen randomly in the spatial domain of the data sets. A set of path lines was then generated through consecutive interpolation of the stored flow maps for each interval. In addition, a set of ground truth path lines was obtained by applying the numerical integration method DOPRI5 to the original flow field in its high resolution. From these two sets of path lines, the actual reconstruction errors were calculated for each seed point as the Euclidean distance between the integrated and the interpolated position. Applying our error estimate to the reconstructed path lines from integration allowed us to directly compare the actual errors to the theoretically predicted errors.

Double Gyre One of the data sets used for the evaluation of the theoretical error measure in this chapter is the well known double gyre. It is a time-varying, two-dimensional vector field with a closed boundary. We use the analytical definition from Shadden et al. [SLM05] to calculate the velocity field. Example trajectory can be found in Figure 3.4. The global truncation error (3.19) plotted to densely spread seedpoints all over the domain for the 8-th time interval of 16 can be seen in Figure 3.7.

ABC The Arnold Beltrami Childress (ABC) vector field, as defined in [Hal01], is a time-varying, three-dimensional vector field that features periodic boundary conditions. Figure 3.5 shows a set of 1000 path lines that were obtained from 16 consecutive flow map intervals.

Convection The convection dataset stems from a simulation of a two-dimensional flow around a small, heated cylinder. As heat is transferred from the cylinder, the surrounding air begins to rise. Figure 3.4 shows a selection of path lines. The distribution of the global truncation error over the domain for the 12-th time interval of 16 can be found in Figure 3.8.

Figure 3.4: Double gyre. Top: A reconstructed path line is shown with the estimated global truncation error e from (3.19) represented by rectangles. Bottom: Five reconstructed path lines are represented as tubes with a radius equal to the absolute estimated error a from (3.22), which also determines the color. On both sides, the corresponding correct path lines are plotted in black.

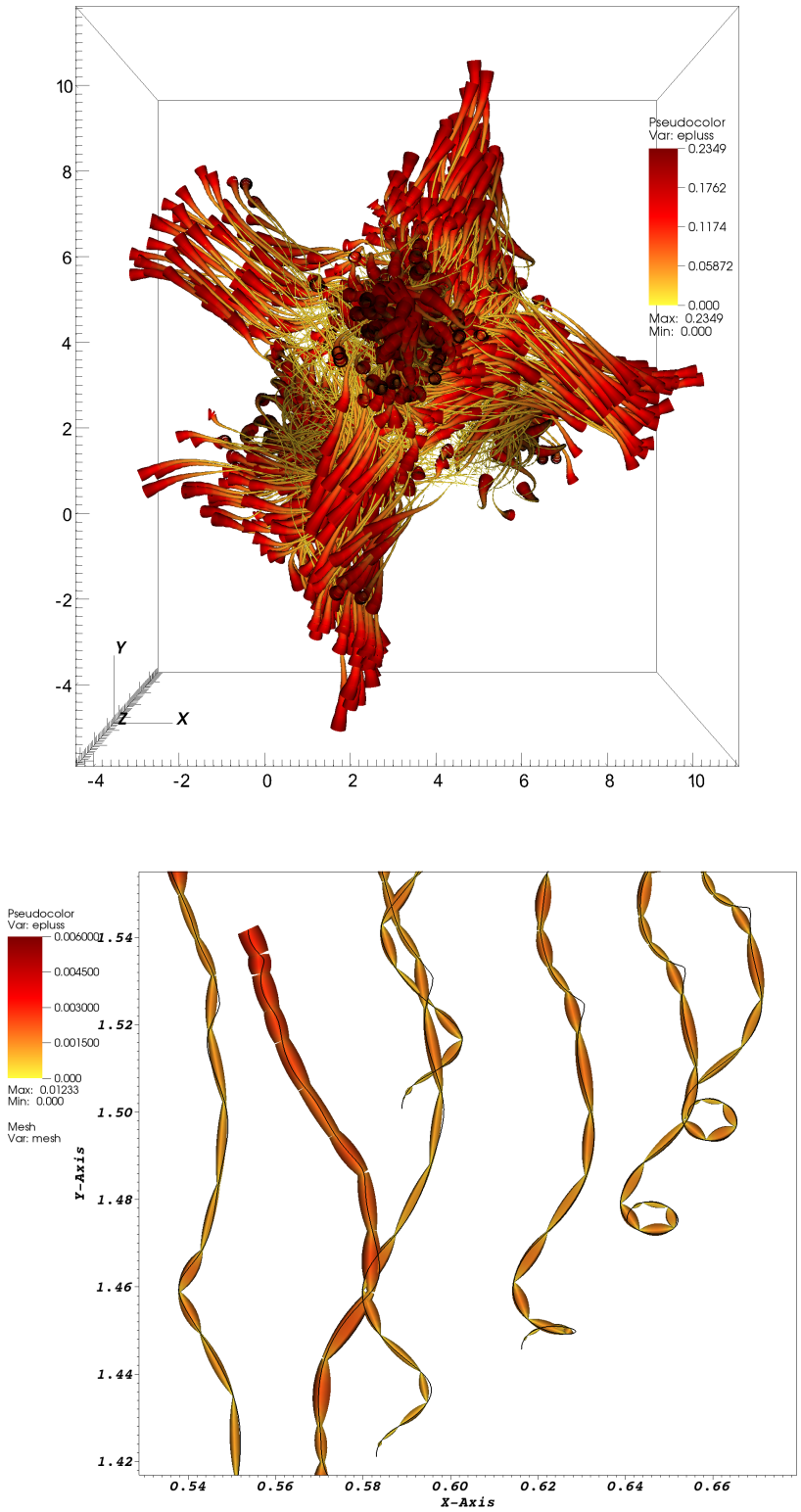

Figure 3.5: As in Figure 3.4, the estimated absolute error (3.22) is represented by the thickness and color of the tubes. Top: 1000 path lines in the ABC data set reconstructed from 16 flow map intervals. Bottom: Several path lines in the convection flow field with 32 intervals.

3.5.1 Evaluation

The behavior of the average and the maximal estimated as well as measured global truncation errors (3.19) for the different data sets can be found in Figure 3.6. It shows the errors for 100,000 randomly seeded path lines as they progress through 16 time intervals. As theoretically predicted, the error increases exponentially over time, which is why we depict the results using a logarithmic scale. Our theoretical error estimate is constructed to bound the actual error from above, which can be well perceived in the charts. The real error is usually smaller than the estimated one. This can be attributed to the fact that the error estimate represents only an upper bound for the error. In addition, errors from consecutive time steps are able to cancel each other out. Since the upper bound of the error increases exponentially with time, the difference between the estimated measured errors also increases with each interval.

The relationship between measured and estimated errors becomes clearer in Figure 3.7. Estimated errors (left column) and measured errors (right column) clearly exhibit the same features, clearly demonstrating that the estimate does not only provide an upper bound, but also correlates with the measured error.

3.5.2 Visualization

In a d-dimensional flow field, the truncation errors are d-dimensional vectors. The global truncation error (3.19) at the endpoints of the sections per component mapped to a rectangle can be found in Figure 3.4. The real positions of the path line can not be located outside the rectangles. It shows how the global truncation error increases exponentially over time.

In order to also visualize the error between the stored time steps, we decided to encode the Euclidean norm of the absolute error vector (3.22) in the radius and color of a tube that is centered around the reconstructed path line. This representation of the path lines as tubes resembles the uncertainty visualization method *UFLOW* as presented by Lodha et al. [Lod+96a]. The tubes have a very intuitive interpretation, because the actual path line has to lie inside of it. For long advection times, the radii of the tubes may become very large, which can cause visual clutter. This is why we offer to only encode the error in the color and draw a line instead of a tube to get a more sparse representation of the reliability of the reconstructed path lines. Figure 3.4 shows five path lines in the double gyre data set that were reconstructed using a set of 16 consecutive flow map intervals. The image shows well how the error does not only increase over time, but also grows between each two consecutive stored time intervals to become smaller as we reach the next stored position.

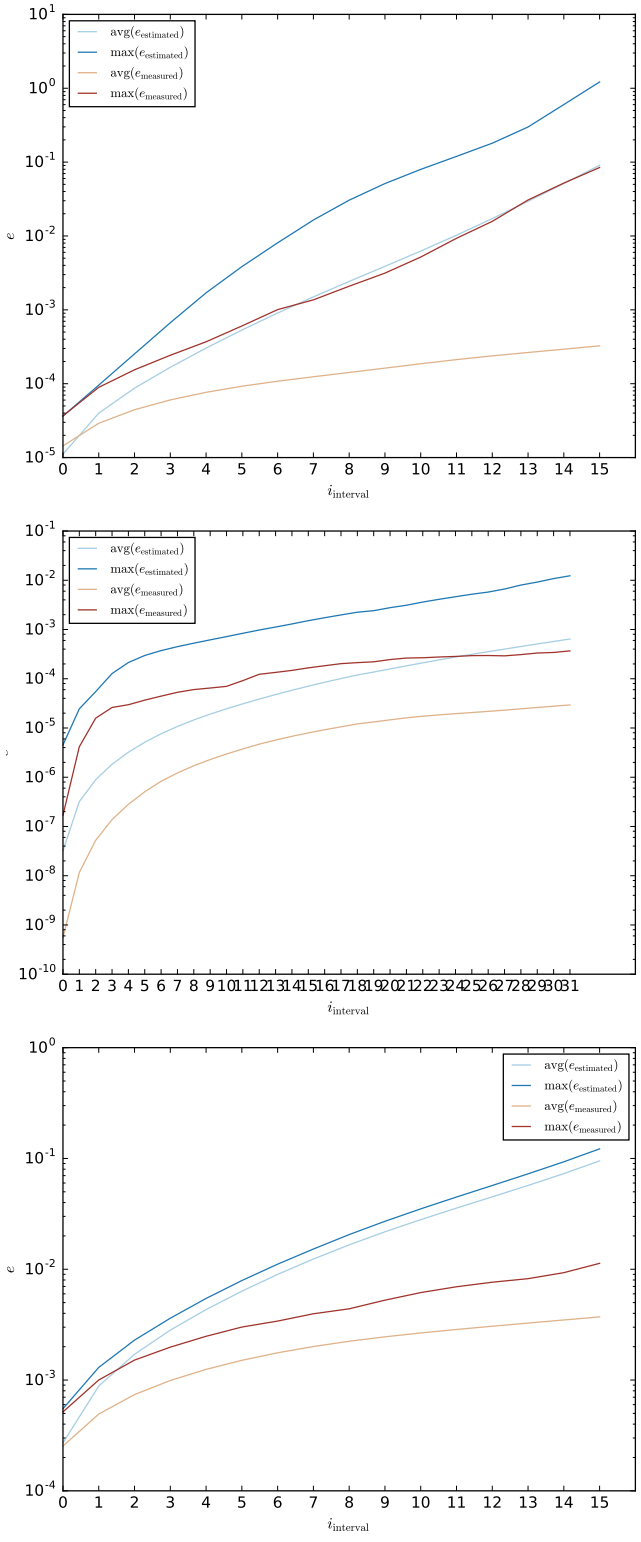

Figure 3.6: Averages and maxima of estimated and measured error over time: double gyre (top), convection (center), ABC (bottom).

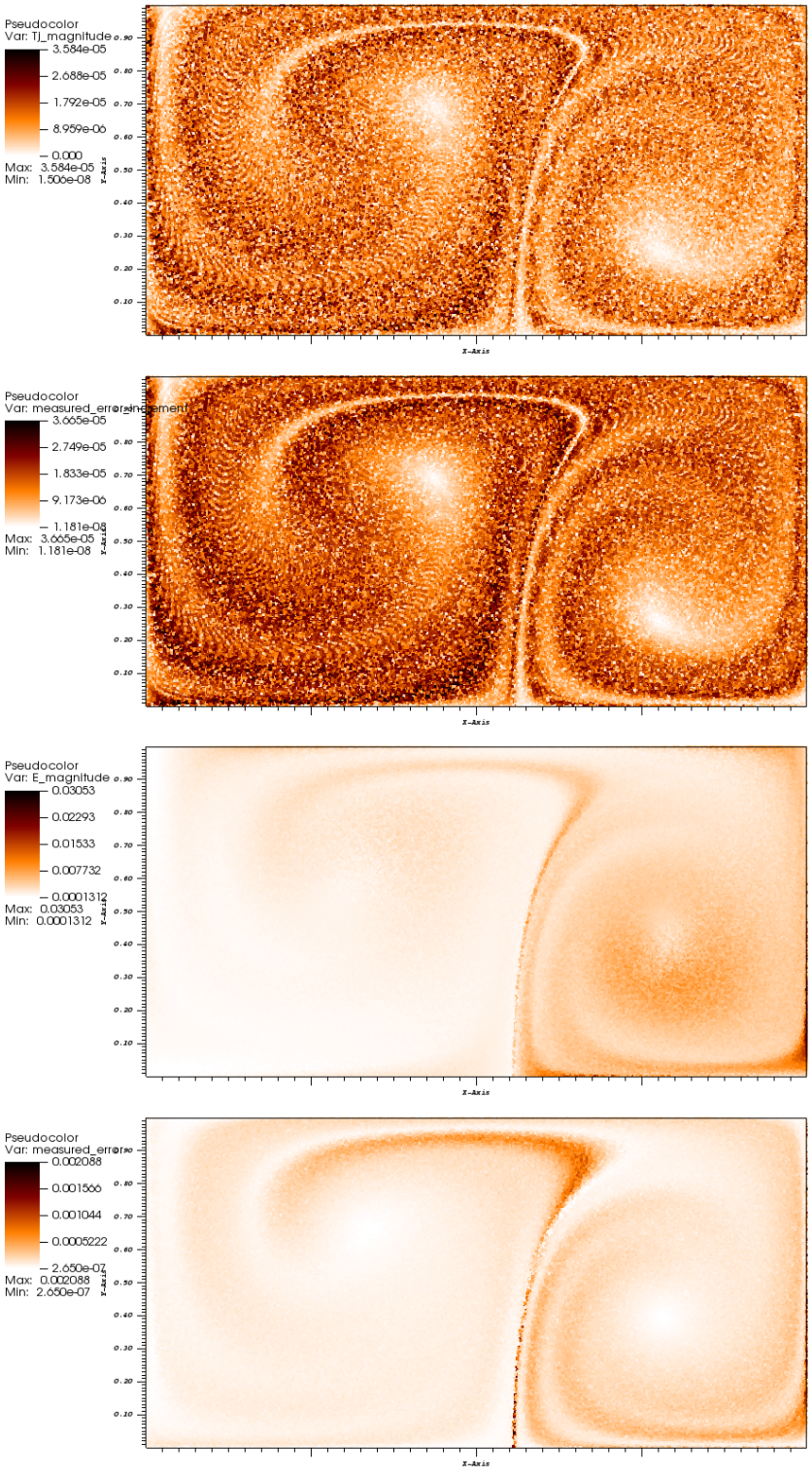

Figure 3.7: Double gyre: Estimation and measurement of the local truncation error τ (a,b) and of the global truncation error e (c,d).

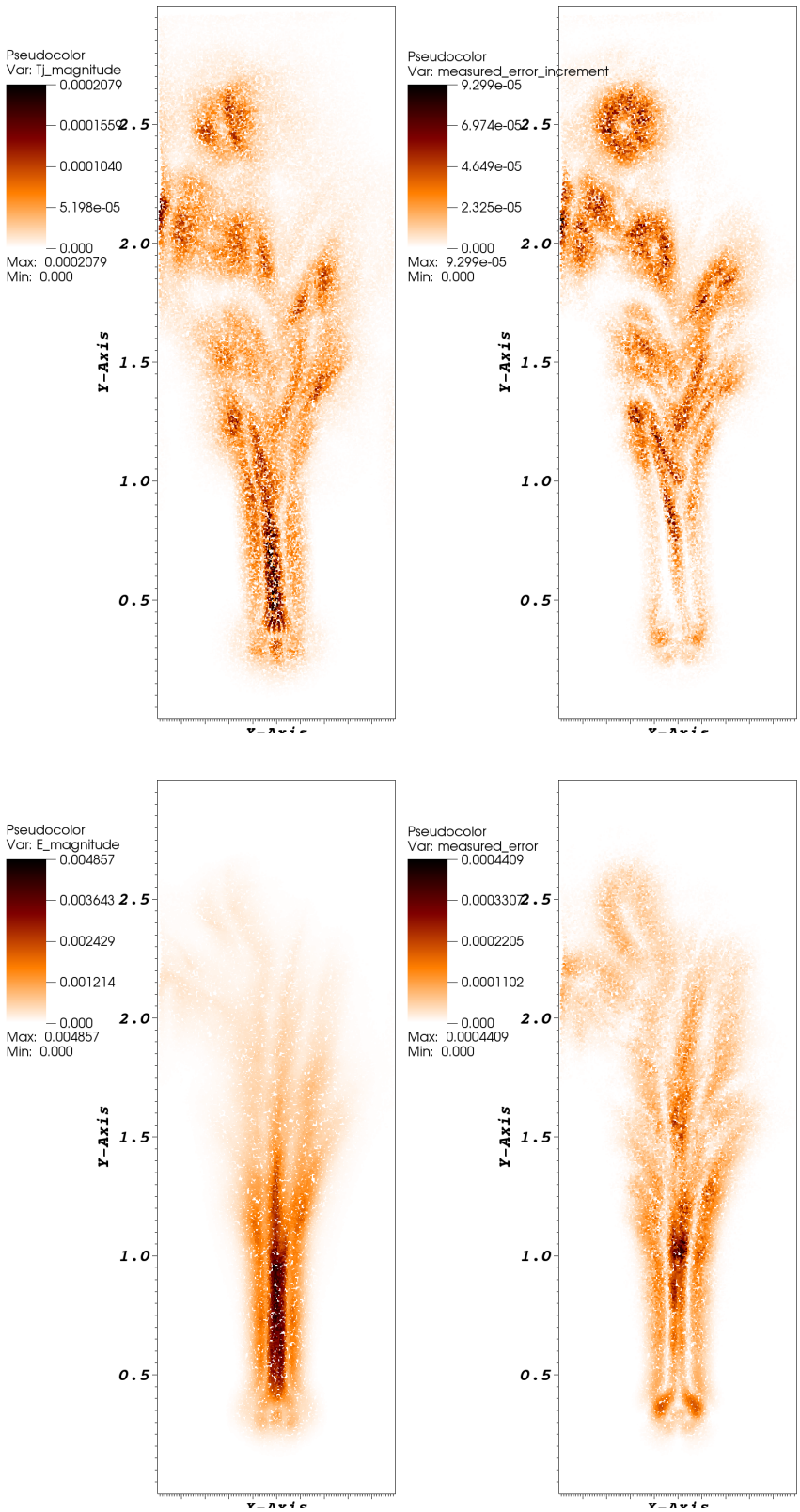

Figure 3.8: Convection: Estimation and measurement of the local truncation error τ (left pair) and of the global truncation error e (right pair).

3.6 Discussion

For application scientists who design and execute flow simulations, choosing the right output frequency is tedious at best. Discarding a large number of time steps can seriously compromise the accuracy of any trajectory-based analysis, while storing all time steps is simply not feasible due to time constraints. Even though it has been shown in the past that the errors can be reduced using a Lagrangian flow field representation, the same trade-off still applies.

This chapter has introduced a method to estimate the error of individual trajectories that are reconstructed from that Lagrangian flow field representation. We have compared the theoretical estimate to the actual errors for path lines in a number of example flow data sets. The results show that using our method, a reasonably accurate upper bound of the reconstruction error can be computed. Further, our experiments have confirmed that the error estimates correlate with the actual errors and reveal similar features.

Using our estimates, the error behavior of the Lagrangian representation can be assessed as the simulation progresses. Not only will this enable the application scientist to evaluate the quality of trajectories that are later used for visualization, but it also makes it possible to define error bounds that can then be used to automatically steer both the frequency and spatial resolution of stored flow data. In some applications, the maximal error may not even be of very high relevance: the end points of trajectories seeded in areas with a very large reconstruction error are highly sensitive to perturbations of the start point. It could make sense to accept these usually localized errors, and use a measure other than the maximum, such as a percentile, to control the resolution. Our measures provide the means to implement and monitor such a scheme.

In the future, we will examine how this error estimate can be used in situ to control multi-resolution storage schemes.

4 Comparative Visual Analysis of Lagrangian Transport in CFD Ensembles

The error measures described in Chapter 3 serve to assess the accuracy of a Lagrangian flow field representation, with the ultimate goal of representing the output of a simulation as closely as possible given storage or bandwidth restrictions.

When using simulation methods to make predictions about real-world, physical phenomena, a single simulation run can not capture the entire range of possible outcomes. Instead, the results are highly dependent on parameters, boundary conditions, and the mathematical simulation model itself. In many cases, the setting cannot be exactly specified beforehand and has to be estimated.

Recent increases in computational power and the availability of high performance parallel computing have made it feasible to run numerical simulations repeatedly for large sets of different parameters in reasonable amounts of time [Smi+09]. The purpose of such an approach to simulation is to compensate for the complexity of real-world physical phenomena by capturing a large number of behaviors. The underlying assumption is that the insight gained from multiple alternative mathematical models allows for a more complete understanding of the real-world phenomenon and how it relates to computational models. A set of output that results from multiple simulation runs is referred to as an *ensemble*.

Ensembles of time-varying vector fields, as generated by computational fluid dynamics simulations, pose severe challenges to the field of comparative visualization. For ensembles of (stationary) scalar fields, a localized Eulerian comparison can provide appropriate means for comparative analysis. In vector fields, however, the property of central interest, namely the transport of material, is a Lagrangian characteristic of the field. Here, local comparisons are complicated by the wide range of diverging and converging flow behaviors that may be present in individual fields. An analysis of this type of behavior therefore requires the development of novel approaches to efficient ensemble visualization.

This chapter introduces a trajectory-based, Lagrangian approach to the comparison of the transport behavior in flow field ensembles. To identify areas of agreement and disagreement across the ensemble members, a classification strategy based on two types of variance is

introduced. The approach examines the behavior of Lagrangian neighborhoods in individual simulation runs and relates those to joint behaviors present in the ensemble. Both cases heavily rely on trajectory information obtained through path line integration techniques, as known from methods such as Finite Time Lyapunov Exponent (FTLE) analysis, and analyze properties of the advected neighborhood using *Principal Components Analysis* (PCA).

Results are provided to show how the techniques introduced in this chapter enable a robust visual comparison of the transport behaviors present in flow field ensembles, and provide means to identify prevalent deterministic transport behaviors. The effectiveness of interaction and querying techniques for such a comparison in a selection of 2D and 3D time-varying benchmark data sets is demonstrated. In summary, this chapter makes the following contributions[1]:

- *Computation of individual and joint vector field variance:* We provide a definition of generalized Lagrangian neighborhoods in flow fields that allows the computation of variance-based flow divergence on arbitrary scales. Furthermore, our work establishes a notion of flow divergence for multiple fields. The proposed joint vector field variance can be seen as a form of multi-field FTLE.
- *Definition of Lagrangian comparability:* We propose a classification space for robust comparison of Lagrangian transport by connecting intra- and inter-vector field transport variance.
- *A framework for the comparison of time-varying flow fields in an ensemble:* Together with trends analysis, interaction, and comparability classification, this chapter presents a framework for Lagrangian transport comparison in CFD ensembles.

The chapter proceeds with a presentation of related work and definitions in sections 4.1 and 4.2, respectively. We then detail Lagrangian properties and our approach to comparing transport in time varying flow fields in Section 4.3. Section 4.4 shows how we applied our approach to the comparative visualization of vector fields in an ensemble. Sections 4.5 and 4.6 present results of our work and conclusions.

4.1 Related Work

This chapter is focused on the comparison of a specific property of time-varying flow fields, namely material transport. Since this property is not directly encoded in the form of scalar fields, we are confronted with the challenge to compare sets of flow trajectories across

[1] The results described in this chapter were published in [Hum+13].

CFD ensembles. As a consequence, our work extends the state-of-the-art in ensemble visualization to 2D and 3D ensembles of time-varying flow fields.

Stationary vector fields may also be compared based on their topological skeletons as demonstrated by Otto et al. [Ott+10; OGT11]. For time varying data sets, however, no definition of a topological decomposition exists. Instead, separating stable and unstable material lines may be extracted over finite amounts of time, by extracting the Finite Time Lyapunov Exponent (FTLE) field. As discussed in Section 2.5, extremal values of this field indicate that flow trajectories seeded within an infinitesimal neighborhood diverge strongly over time. Semantically, such an FTLE field represents a comparison measure for fluid transport within a spatial neighborhood. In visualization, this type of divergence measure has been extracted from dense sets of trajectories [Gar+07a], as well as in a localized fashion [Kas+09; Obe+09]. Recently, Schneider et al. [Sch+12] proposed the use of Principal Component Analysis (PCA) to compute variance based flow divergence in uncertain vector fields. Our work makes use of the notion of FTLE as a representation of integration reliability and extends PCA based variance estimation to neighborhoods of particles within a single flow field as well as across members of an ensemble.

Uncertain data often consists of data samples drawn from a set of simulation runs, and as such is closely related to ensemble data sets, a data type that has recently come into focus of the visualization community. A prime application area for ensemble visualization is climate research, where multiple climate prediction models are used to produce climate forecasts. Potter et al. [Pot+09a; Pot+09b] investigate visual analysis of such ensembles through trends and contour or *Spaghetti Plots* and color-mapped per-point means and averages [WP09]. Generalizations of such contour plots are used by Sanyal et al. [San+10] alongside uncertainty glyphs to analyze scalar-valued weather forecasts. In simulation ensembles, where clear trends in qualitative outcome are detectable, an ensemble clustering step can be beneficial before performing statistical data averaging, as demonstrated by Smith et al. [Smi+06]. Our work employs per-point averaging and trends analysis techniques within the context of Lagrangian flow trajectories. As common in ensemble visualization, we also make use of summary statistic plots. A complete comparison of members of an ensemble is a challenging and complex task that requires the implementation of linked components for data comparisons on multiple levels. For 2D scalar field ensembles in powertrain system development, such a complex system may consist of domain, feature, and member-based comparison components [Pir+12]. Jarema et al. [Jar+15] proposed a method to visualize distributions in 2D vector field ensembles using Gaussian mixture models, glyphs, and pairwise differences. Petz et al. [PPH12] investigate spatial distributions of local features such as critical points and cores of swirling motion in uncertain vector fields.

Trajectory based comparison techniques [Lod+96b] allow for the analysis of streamlines generated by different numerical integration schemes. Uncertainty of particle traces in fluid flow is encoded using envelopes, ribbons, and various glyphs arranged along the trajectory. Verma and Pang [VP04] extend this work by performing pairwise comparisons of streamlines from two different vector fields. They note that distinguishing between errors caused by numerical integration and differences present in flow fields is crucial for effective comparison. The approach presented in our work verifies this observation and generalizes the notion of integration reliability or uncertainty and field differences by distinguishing between joint and individual transport variance in ensembles of time-varying vector fields.

Botchen et al. [BWE05] presented a texture-based method for the visualization of uncertainty in velocity fields. For fluid flows, dense texture-based visualization techniques of uncertainties in flow fields have proven to be especially suitable for 2D vector fields [BWE06]. Combined with overlaying approaches, they can be used to compare small numbers of flow and scalar fields [Urn+06]. Alternatively, geometric features, such as vector glyphs [WPL96] can be extended to encode uncertain information, such as variability in direction and magnitude. Hlawatsch et al. [Hla+11a] proposed glyphs for uncertain, time-varying vector fields.

4.2 Background: Ensembles and Material Transport

The visual analysis of ensemble data sets has gained in popularity during the past decade [WP09]. In this chapter, we focus on the visual comparison of transport behaviors present in an ensemble of vector fields. We presently recapitulate formal definitions of such data and motivate the desire to perform a visual analysis of flow ensembles.

4.2.1 Ensembles

Ensemble data sets pose multiple challenges for visualization. Owing to the fundamental idea of running multiple simulations, the amount of data contained in an ensemble data set is typically much larger than that of a single simulation run. This leads to issues with storage, transfer, and processing before visualization can even be attempted. Secondly, the sheer amount of information contained in an ensemble is difficult to handle. In essence, an additional dimension is added to the often already existing three spatial dimensions and the temporal dimension. A naïve approach would be to apply traditional visualization methods separately to each ensemble member and then compare the results. Such an approach is not

likely to succeed, because the results of many visualization methods (i.e., images of curves or scalar fields) do not easily lend themselves to direct comparisons.

Formal Notation A d-dimensional time-varying flow field

$$\mathbf{u} : \Omega \times I \rightarrow \mathbb{R}^d \tag{4.1}$$

is a vector-valued function defined over the spatial domain $\Omega \subseteq \mathbb{R}^d$ and over an interval in time $I \subseteq \mathbb{R}$. In this thesis, a flow field *ensemble* E is defined as a collection of m such vector fields defined over a common domain, $\Omega_E = \Omega_1 \cap ... \cap \Omega_m$ and $I_E = I_1 \cap ... \cap I_m$:

$$E : \{1, \ldots, m\} \times \Omega_E \times I_E \rightarrow \mathbb{R}^d. \tag{4.2}$$

We call the first (discrete) parameter dimension of this mapping the *ensemble dimension*. In order to establish meaningful semantics for a vector field comparison, all $\mathbf{u}_i = E(i,.,.)$ are assumed to correspond to solutions of models for the same physical phenomenon. Because this work does not aim at explicit parameter optimization, our definition of an ensemble does not require that the simulations used to create individual $\mathbf{u}_i$ share a common parameter space.

A time-varying flow field $\mathbf{u}$ describes the trajectories $\mathbf{x}$ of infinitesimal particles according to the following differential equation

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{u}(\mathbf{x}(t), t) \tag{4.3}$$

with initial particle position $\mathbf{x}(t_0) = \mathbf{x}_0$. In the following we denote an individual particle or trajectory simply as $\mathbf{x}$ or $\mathbf{x}_i$. Numerical integration of equation 4.3, e.g., using 4th-order Runge-Kutta techniques or adaptive methods such as DOPRI5, allows the explicit computation of particle trajectories. In a Lagrangian setting, flow field properties such as pressure, speed, and particle neighborhoods are explicitly transported along with the flow and stored at such moving particle locations, rather than being position-bound in the form of a Eulerian representation. In the following, we recognize transport as a central Lagrangian property of flow fields.

For a single flow field, analysis of transport behavior can be achieved by evaluating such Lagrangian properties over a spatial neighborhood around a flow particle. These comparisons allow for the analysis of differences in particle behavior and are detailed in the following sections.

4.2.2 Material Transport

Numerical simulation models for complex physical phenomena involve large sets of simulation parameters. For fluid simulations, typical parameters include physical properties such as boundary conditions (e.g., pressure or temperature), viscosity, stiffness, or other internal forces as well as simulation parameters, such as simulation accuracy or resolution. In order to reproduce a physical behavior, this parameter space needs to be explored (cf. [BM10]) until simulation results converge to experimental observations. As simulation behavior in this parameter-space is highly nonlinear, unguided manual exploration of parameter space requires tedious and repeated comparisons between simulation outputs. In order to expedite this process, domain experts increasingly make use of ensemble techniques.

For vector fields, the output of fluid simulations, the most characteristic property is their transport behavior. This transport behavior describes where material is transported, defines (topological) boundaries or transport barriers, regions of flow divergence and convergence, and the deformation and displacement of material elements in general. Consequently, a natural question arising in the visualization of vector field ensembles is how to extract and visualize similarities and differences in Lagrangian transport. Figure 4.1 illustrates transport behavior in multiple vector fields and exemplifies the task of Lagrangian comparison. This illustration reveals immediate similarities between transport behavior analysis in an ensemble and divergence analysis for neighborhoods in individual fields. In the following sections we make use of such divergence measures in order to analyze differences and commonalities of transport behavior in vector fields of an ensemble and present a framework for visual comparison of Lagrangian transport behaviors.

Note that ensemble comparison strategies can generally take two different approaches. The most common approach is data-oriented, and involves merging or overlapping the domains of individual runs, followed by a per-point comparison in data space. An alternate approach is to compare members of an ensemble in feature space. Hereby comparisons are not performed on data at a specific location, but over a set of features present in the flow field. Similarities can then be defined by the presence or absence of common feature types, shapes, or properties (see for example [Wei+10; OJ12] for feature based similarity metrics in flow fields). The techniques presented in this chapter represent a hybrid method – comparisons are performed in data-space, while relying on the transport property computed from flow trajectories.

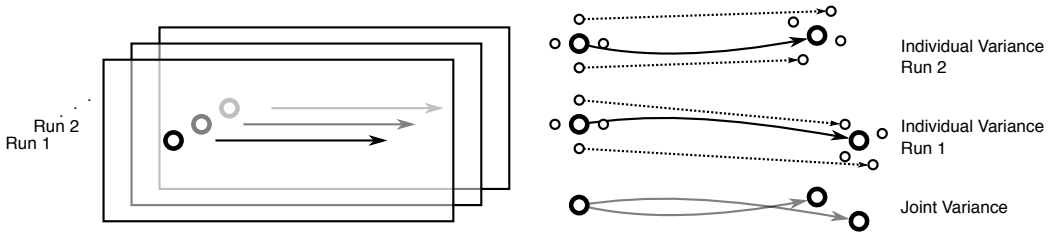

Figure 4.1: A particle started at identical positions in all vector fields of an ensemble is transported to different final positions. Different locations in the ensemble lead to stronger or weaker separation of particle positions. Notice the conceptual similarity between ensemble divergence and individual flow field divergence.

4.2.3 FTLE and the Flow Map

The observation that multi-field comparisons of Lagrangian transport are related to divergence measures in single fields is an important insight for the techniques developed in this chapter. For this reason, the following sections analyze the notion of fluid transport in an individual flow field and existing means for comparison and establish a basis for the subsequent presentation of novel multi-field ensemble comparison techniques.

The best-studied property of Lagrangian neighborhoods is the tendency of particle neighborhoods to diverge or converge over time. The *Finite Time Lyapunov Exponent* (FTLE), which measures exponential stretching of infinitesimal particle neighborhoods over finite intervals in time, is a measure of this characteristic. This measure can be computed directly from the gradient of a displacement map, given by the flow map F, of a vector field. As described in Section 2.5.1, during FTLE computation magnitudes and directions of maximal deformation of an infinitesimal neighborhood can be obtained directly from the eigenvalues and eigenvectors of the left Cauchy Green tensor of ∇F [Obe+09]. Thus, the FTLE is a suitable measure to compute relative stretching and divergence of an infinitesimal neighborhood in a single flow field. However, a generalization of this approach to ensembles is not straightforward.

4.3 Lagrangian Comparison of Flow Ensembles

A possible avenue for comparative transport analysis in flow ensembles is a generalization of Lagrangian measures over particles neighborhoods, which we examine in the following.

4.3.1 Divergence of Generalized Neighborhoods

Let the domain $D \subset \mathbb{R}^d$ of $\mathbf{u}$ be densely sampled by a set of n flow particles $\mathbf{x}_i \in \{\mathbf{x}_1 ... \mathbf{x}_n\}$. Let $\mathbf{x}$ be a specific flow particle. At any point in time $t \in \mathbb{R}$, we can define a generalized neighborhood for $\mathbf{x}$ as follows:

$$N_{\mathbf{x}}(t) = \{\mathbf{x}_i \in \{\mathbf{x}_1 ... \mathbf{x}_n\} \mid P(\mathbf{x}(t), \mathbf{x}_i(t)) = \text{true}\} \tag{4.4}$$

where P is a neighborhood predicate that indicates local adjacency. An example of a spatially isotropic neighborhood predicate P with radius $r \in \mathbb{R}$ is

$$P(\mathbf{x}(t), \mathbf{x}_i(t)) = \begin{cases} 1 & \text{if } \|\mathbf{x}_i(t) - \mathbf{x}(t)\| \leq r \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

Note that these types of spatial neighborhood predicates allow the discussion of an implied *neighborhood scale* based on the choice of r. In Section 4.3.2, we give neighborhood predicates for ensembles that also include non-spatial dimensions. In this generalized definition, the Lagrangian neighborhood is no longer limited to particles that are immediate spatial neighbors of $\mathbf{x}$, but may be a scattered across larger regions in space.

FTLE The applicability of FTLE techniques is directly influenced by properties of this neighborhood. FTLE computations require the local linearization of the deformation map, i.e., the computation of the flow map gradient ∇F, or the velocity gradient tensor [Obe+09]. For this reason, the Cauchy Green tensor derived during FTLE computations is only an accurate representation of deformation if the deformation of the Lagrangian neighborhood can be linearized with sufficient accuracy. For generalized neighborhoods, i.e., particle neighborhoods that extend over a set of simulations or over regions in space, the function describing fluid deformations tends to become increasingly nonlinear and therefore unsuitable for the direct application of FTLE techniques, as shown in Figure 4.2. Note also that while the flow map may be averaged over larger regions of space to obtain FTLE values for spatially large neighborhoods, flow map gradients cannot be computed across multiple vector fields due to a lack of a metric/ordering for the ensemble dimension.

PCA An alternative to computing the linearized deformation or shape change of a neighborhood is to measure geometric or statistical properties of the neighborhood after deformation. Note that these properties are in general absolute measurements of neighborhood properties and stand in contrast to the relative notion of deformation given by FTLE. However, if

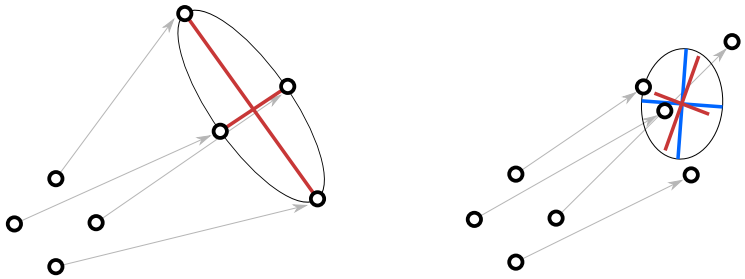

Figure 4.2: Differences between spectral decompositions of the left Cauchy Green tensor and the particle covariance matrix for an identical deformed neighborhood. Left: Both extracted coordinate frames are identical for an isotropic neighborhood with linear deformation. Right: For non-linear deformations, principal components (red) and singular values of the deformation gradient (blue) diverge. The velocity gradient tensor and the Cauchy Green tensor lose accuracy when computed for strongly nonlinear mappings.

the statistical and geometric properties of the neighborhood definition in Equation 4.4 are constant over the complete data, these properties can provide a notion of certain relative neighborhood changes. Let in the following $F_{t_0}^{t}(N_{\mathbf{x}}(t_0)) = \{\mathbf{x}_i(t) \mid \mathbf{x}_i \in N_{\mathbf{x}}(t_0)\}$ denote the set of particle positions obtained by advecting all particles in a neighborhood $N_{\mathbf{x}}(t_0)$ for a time of $t - t_0$.

Principal Components Analysis allows the extraction of extremal directions of this set of displaced point positions. PCA computes the eigenvectors and eigenvalues, i.e., directional variances, of the covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$, given by

$$\Sigma(F_{t_0}^{t}(N_{\mathbf{x}}(t_0)))_{ij} = \frac{1}{n-1}\sum_{k=1}^{n}(x_{ki}(t) - \mu_i)\cdot(x_{kj}(t) - \mu_j) \tag{4.6}$$

where $n = |N_{\mathbf{x}}(t)|$, x_{ki} and x_{kj} are the ith and jth component of a vector $\mathbf{x}$ indicating particle position. The vector μ represents the mean of all positions in $F_{t_0}^{t}(N_{\mathbf{x}}(t_0))$. By construction, eigenvectors of Σ correspond to directions that minimize the squared distance to all points in the point set. Additionally, eigenvalue magnitudes correspond to directional variances. PCA can be viewed as an approximation technique for multidimensional Gaussian distributions and is, therefore, a purely statistical measure. This implies that there are several conceptual differences between PCA and Cauchy Green tensor based spectral decompositions: Rather than representing a relative change in displacement, it measures characteristics of the final neighborhood distribution. PCA is therefore identical for arbitrary permutations of particle positions. Note that there are special cases, where outliers in $F_{t_0}^{t}(N_{\mathbf{x}}(t_0))$ can distort PCA values (see for example [MMY06]). Inspection of outlier behavior is discussed in Section 4.3.6. We make use of PCA in the remainder of this chapter for its capability to give a

measure of flow divergence and represent statistical variance of advected point clouds at the same time.

We propose to combine the presented definitions of generalized Lagrangian neighborhoods and PCA into a novel technique for the Lagrangian comparison of time-varying flow fields in an ensemble E. In the following, we do not distinguish steady from unsteady flow fields, but note that flow field comparison for steady fields can generally rely on simpler techniques, such as a comparison of topological graphs.

It is important to note that deformations of particle neighborhoods carry additional information that may be captured with FTLE-related or PCA-based methods. Eigenvectors of the derived tensors can, for example, provide directions of neighborhood expansion and contraction, adding another level of complexity to the analysis process. In the course of this chapter, we exclusively focus on scalar measures of variance, keeping in line with previous work on ensemble visualization and exploiting it as a well-understood probabilistic measure.

4.3.2 Lagrangian Neighborhoods for Ensembles

As indicated previously, defining a comparison technique that incorporates transport behaviors of multiple flow fields requires the extension of classic Lagrangian neighborhoods across multiple domains. We propose to use the following neighborhood predicate as a neighborhood definition for a complete ensemble

$$P_E(\mathbf{x}(t),\mathbf{x}_i(t)) = \begin{cases} 1 & \text{if } \|\mathbf{x}_i^{\Omega_E}(t) - \mathbf{x}^{\Omega_E}(t)\| \leq r \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

as required by Equation 4.4. The main difference to Equation 4.5 is that particles $\mathbf{x}_i$ and $\mathbf{x}$ may belong to different ensemble members. The positions $\mathbf{x}^{\Omega_E}(.)$ correspond to particle positions projected into the common ensemble domain. This essentially defines a radial neighborhood spanning across all domains $\{\Omega_1 ... \Omega_m\}$, treating particles $\mathbf{x}_i$ and $\mathbf{x}$ as if they originated from the same simulation run. Thus, the predicate results in neighborhoods containing exactly m particles (one for each run) for a radius of zero. We denote a Lagrangian ensemble neighborhood as N_E. The results presented in this work use a constant radius r over the entirety of the ensemble domain Ω_E.

This neighborhood definition treats all ensemble members equally by collapsing the ensemble dimension.

4.3.3 Naïve Transport Comparison

Given our ensemble neighborhood definition together with PCA-based variance estimation, we are able to make a direct comparison of transport properties of multiple flow fields. An analysis of Lagrangian flow field comparison based on an ensemble neighborhood with radius $r = 0$, a point-based comparison, reveals the following insights: For a given advection time $t - t_0$ and a specific point $\mathbf{x}$ in Ω_E, a PCA of $\Sigma(F(N_{E\mathbf{x}}(t_0)))$ measures the variance in advection across all ensemble members. A large maximal eigenvalue of Σ indicates that the compared vector fields disagree in the final advected point position. As a consequence, the agreement of ensemble members in a neighborhood $N_E\mathbf{x}(t_0)$ is measured by its *joint variance* $\mathrm{Var}_E(N_{\mathbf{x}}(t_0)) = \lambda_{\max}(\Sigma(F^t_{t_0}(N_{E\mathbf{x}}(t_0))))$. High values of joint variance are expected in regions with non-deterministic or strongly varying transport behavior across runs of the ensemble, whereas low values indicate deterministic fluid transport behaviors that are shared across runs. While this is a valid measure of inter-field variance, there are multiple concerns with respect to expressiveness and practical utility:

1. Outliers and grouping or trends are not detected.
2. Susceptibility with regard to simulation noise.
3. Susceptibility with regard to accuracy of the numerical integration technique.

The last two points are mainly due to the fact that intra-field variance is neglected during comparison. The following example illustrates this problem:

Let $\mathbf{u}_i$ be a set of vector fields originating from the same simulation model and with identical parameter settings. Let there be minimal noise in the data due to differences in numerical precision, varying hardware environments, or due to non-deterministic behavior caused by programming errors. Let further $\mathbf{x} = (\mathbf{x}_0, t_0)$ be the location of the point considered for Lagrangian comparison.

Assuming for simplicity that $\mathbf{u}_i$ are truly identical ($\mathbf{u}_i = \mathbf{u}$ for all i), we can model this noise by applying a random infinitesimal offset vector ε_i to $\mathbf{x}_0$ in every simulation. The displacement of every $(\mathbf{x}_0 + \varepsilon_i, t_0) \in N_{E\mathbf{x}}(t_0)$ with respect to $(\mathbf{x}_0, t_0)$ is then

$$F^t_{t_0}(\mathbf{x}_0 + \varepsilon_i) - F^t_{t_0}(\mathbf{x}_0) = \nabla F^t_{t_0}(\mathbf{x}_0) \cdot \varepsilon_i. \tag{4.8}$$

Here, the maximal value is used in the computation of $\sigma^t_{t_0}(\mathbf{x}_0)$. Consequently, the eigenvalue magnitudes of $\Sigma(F^t_{t_0}(N_{E\mathbf{x}}(t_0)))$, the joint ensemble variances, are directly dependent on $\sigma^t_{t_0}(\mathbf{x}_0)$, or the variances of individual fields, $\Sigma(F^t_{t_0}(N_{i\mathbf{x}}(t)))$.

We conclude that there is an immediate relationship between single field divergence/variance and joint ensemble variance, when numerical inaccuracies, as caused by noise or

inaccurate particle tracing [LB98], come into play. In summary, large joint variances may not accurately describe fundamentally different transport behaviors in the field, if high individual variances are present. This insight is central to the development of our joint variance classification scheme developed in this chapter and similar to observations made by Verma and Pang [VP04].

4.3.4 Variance Classification

We conclude from these observations that a one-dimensional space, e.g., one based on joint variance values, is insufficient for reliable comparison and instead utilize a two-dimensional classification space.

To enable a complete comparison of transport behaviors in a vector field ensemble, we propose to combine individual and joint variance analysis into a complete variance classification framework. Instead of exclusively computing $\mathrm{Var}_E(N_{E\mathbf{x}}(t_0))$ for comparison of a Lagrangian neighborhood $N_{E\mathbf{x}}(t_0)$, we express every particle $\mathbf{x}$ with associated ensemble neighborhood in a new classification space C comprised of joint and individual variances:

$$C = \mathrm{Var}_E(N_{E\mathbf{x}}(t_0)) \times \mathrm{Var}(N_{1\mathbf{x}}(t_0)) \times \ldots \times \mathrm{Var}(N_{m\mathbf{x}}(t_0)), \tag{4.9}$$

where $N_{i\mathbf{x}}(t_0)$ is a respective neighborhood in Ω_i. In this space, every neighborhood is expressed as $m+1$-dimensional point. Note that we use a fixed advection time $t - t_0$, which would otherwise further increase dimensionality of this space.

Because the ensemble dimension is not ordered and possesses no metric, we collapse/project this classification space to two dimensions, as illustrated in Figure 4.3.

4.3.5 Classification Space

In summary, every particle neighborhood $F^t_{t_0}(N_{E\mathbf{x}}(t_0))$ has a single joint variance and m individual variances. Using the average of these individual variances, we are able to represent variance for each point x in the ensemble as a single point in the collapsed classification space:

$$C' = \mathrm{Var}_E(N_{E\mathbf{x}}(t_0)) \times \overline{\mathrm{Var}}(N_{\mathbf{x}}(t_0)) \tag{4.10}$$

with

$$\overline{\mathrm{Var}}(N_{\mathbf{x}}(t_0)) = \frac{1}{m}\sum_{i=1}^{m} Var(N_{i\mathbf{x}}(t_0)). \tag{4.11}$$

This location is identical to the mean location of the error-bar representation in Figure 4.3. Other forms of averages, such as the median or mode may be used interchangeably to

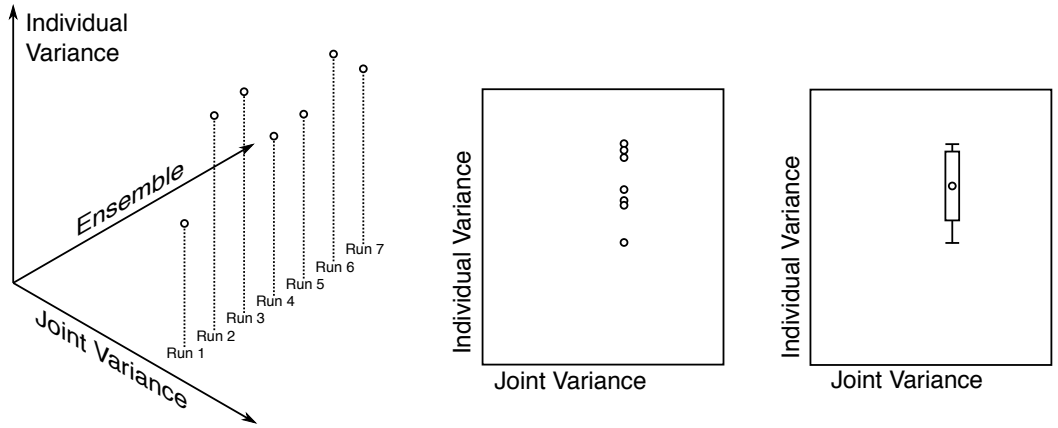

Figure 4.3: Left: The classification space represented as a combination of two-dimensional spaces. Depicted is a single point of the ensemble domain, with a different value of individual variance in each run. Joint variance is defined across the ensemble and, therefore, takes only a single value. The space is then collapsed along the non-spatial dimension of ensemble runs (center), until, finally, the point set is reduced to the properties of its distribution (right). In our experiments, we use the mean as a representative; however, other values, such as the maximum, could also be useful.

reduce the impact of outliers. While the mean is only a basic measure of the distribution of individual variances, it allows for an indication of prevalent flow divergence magnitudes.

The position of a point $\mathbf{x}$, when mapped into the classification space C', can provide insights into the comparability of the flow fields' transport behaviors. More specifically, using this average individual variance allows us to identify different classes of transport comparability. We distinguish first by average individual variance and then consider joint variance to obtain the following distinct cases:

In general, a **low average individual variance** (lower region of C') indicates the tendency of all runs to have numerically stable particle trajectories with low flow divergence. In such a situation, reliable comparison of transport behaviors based on joint variance values, similar to the naïve comparison strategy, is possible:

- *Low Joint Variance:* The vector fields in the ensemble show similar and stable transport behavior. Flow trajectories transport fluid to locations that are spatially close. Furthermore, due to low individual variances, the computed individual flow trajectories are reliable representatives of flow behavior. If the ensemble is used to model a stochastic process, these features correspond to deterministic flow transport.
- *High Joint Variance:* Vector fields show strongly dissimilar transport behavior. Paths of individual flow trajectories are in regions with low divergence and, thus, reliable representatives of the underlying flow in the region. However, at the same time,

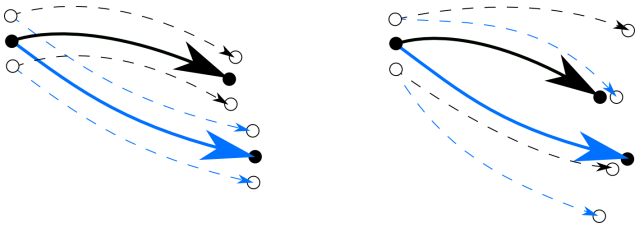

Figure 4.4: Comparing transport behaviors in of two flow fields (black and blue). Left: Both flow fields have low individual variance, as illustrated by similar behavior of neighbors. However, joint variance is high. Right: Both flow fields have a high individual variance. Joint variance is also high. Note how a slight relative displacement of path lines in one of the fields would yield different joint variance values.

trajectories in different flow fields transport fluid matter to spatially separate regions of the flow field. A trend analysis (see Section 4.3.6) can aid in further analysis.

In contrast, in areas of **high average individual variance** (upper region of C'), noise and numerical errors have a strong influence on the displacement of fluid matter. Drawing reliable conclusions about flow comparison is difficult. The values of joint variance are generally not fully independent of the individual variance values: if the neighborhood radii used to compute individual and joint variances are identical, joint variance values tend to be higher than the largest individual variance. The reason for this is that the joint particle neighborhood used for PCA corresponds to a union of the individual displaced particle neighborhoods. Thus, an explicit distinction between different types of high average individual variance based on joint variance values is not always possible. Otherwise, the following tendencies can be identified:

- *Low Joint Variance:* All fields likely show similar transport behavior. All fields transport particles to spatially close regions in the ensemble. Due to numerically unstable trajectories in individual fields, drawing a conclusion about comparability needs further investigation of the fields.
- *High Joint Variance:* The fields likely show dissimilar transport behavior. Due to highly divergent individual trajectories, the difference between transport across the ensemble may be caused by noise or inaccurate integration. Without further investigation, transport behaviors must be regarded as incomparable.

Figure 4.4 illustrates two of the four described cases. In order to allow the investigation of transport behavior in regions with a high joint variance, we implement a transport trends analysis based on particle clustering (Section 4.3.6) and provide means for interactive trajectory seeding (Section 4.4).

4.3.6 Outliers and Trends

As a variance based description of particle distributions after advection, PCA suffers from the same limitations as other statistical analysis techniques that fit normal distributions to arbitrarily distributed data. While its ability to provide a quantitative measure of the spreading of flow particles is suitable for the identification of commonalities and differences in flow transport, it cannot model trends or outliers in the set of flow neighborhood positions. These, however, are important measures for the comparison of ensemble data. For this matter we extend the purely variance-based analysis with a capability to highlight trends and outliers.

For the purpose of this work we rely on a single abstract definition that covers trends and outliers. *Trends* or *outliers* in transport behavior are present in a particle neighborhood, if $F^t_{t_0}(N_{E\mathbf{x}}(t_0))$ contains one (outlier) or multiple (trend) flow neighborhood locations S that are (jointly) separate from the rest of the advected neighborhood, $F^t_{t_0}(N_{E\mathbf{x}}(t_0)) \setminus S$. In practice, we detect and quantify trends by performing a *Minimum Spanning Tree* (MST) cluster analysis of $F^t_{t_0}(N_{E\mathbf{x}}(t_0))$.

Trends and outliers become relevant for locations in the ensemble that show a large joint variance. In these cases we perform a trends analysis as follows:

1. Construct one MST for every set of particles in $F^t_{t_0}(N_{E\mathbf{x}}(t_0))$ that originates from the same run.
2. From these individual MSTs, construct one joint MST, such that it connects all particles in $F^t_{t_0}(N_{E\mathbf{x}}(t_0))$.
3. Remove edges created by the joint MST that are longer than Δsep.
4. Count connected components in the resulting graph.

Since the MST is cycle free, the number of connected components is one higher than the number of edges created by the joint MST whose lengths exceeds Δsep. We chose Δsep empirically to be a fraction of the average trajectory lengths of particles in $F^t_{t_0}(N_{E\mathbf{x}}(t_0))$. Figure 4.5 provides an illustration of our trends and outlier detection. Components that contain only particles from one member of the ensemble represent a run whose transport behavior is an outlier. A graph component with particles from two or more runs constitutes a trend. The number of connected components may be used as simple visual indicator of transport trends and can enrich purely variance based visual comparison.

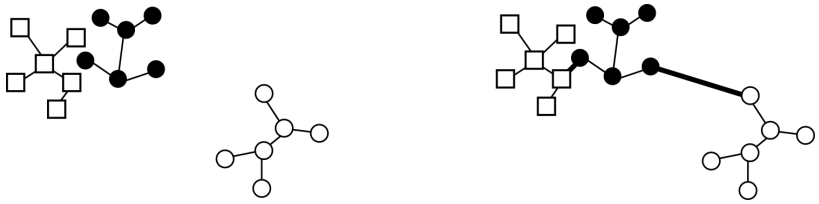

Figure 4.5: For trend and outlier detection MSTs are first created for members of individual runs in $F^t_{t_0}(N_{E\mathbf{x}}(t_0))$; here $m = 3$ (left). In a second step, lengths of new edges in the joint MST provide a measure of distance between ensemble neighborhoods (right).

4.4 Visualization

Our visualization system consists of two components: a statistical representation of variance distributions in the form of a scatter plot, accompanied by a rendering of the flow domain. Both representations are semantically linked with regard to interaction. This approach is inspired by the use of multi-dimensional histograms to generate transfer functions for volume rendering [KD98; KKH02].

4.4.1 Classification Space

The collapsed classification space C' can be visualized directly in the form of a scatter plot. Each point $\mathbf{x}$ of the ensemble domain Ω_E is mapped to its position in the classification space:

$$\mathbf{x} \mapsto (\mathrm{Var}_E(N_{E\mathbf{x}}(t_0)), \overline{\mathrm{Var}}(N_{\mathbf{x}}(t_0))). \tag{4.12}$$

Section 4.3.5 described four cases of the variance configuration and their associated interpretation with regard to ensemble transport behavior. To obtain a first impression of ensemble behavior, a color map is constructed using four colors and used to depict the ensemble's behavior, as exemplified in Figure 4.11. While any choice of colors is technically possible, the examples presented in this chapter use the following color scheme for each corner of the classification space.

green: lower left corner. Both individual and joint variance are low.

red: lower right corner. Individual variance is low, while joint variance is high.

blue: upper left corner. Individual variance is high, while joint variance is low.

white: upper right corner. Both individual and joint variance are high.

Note that red and green regions indicate that a comparison of the transport behavior is possible using the joint variance value, while other regions are difficult to compare. However, it can be said that blue regions tend towards similarity, while white regions tend towards dissimilarity.

A second color-coding variant is aimed at visualizing the results of the trend analysis from Section 4.3.6. Here, we choose color based on the number of identified trends.

blue: one trend (no distinguishable trends found)

gray: two trends

yellow: more than two trends

An example can be found in Figure 4.12.

4.4.2 Ensemble Domain

For two-dimensional fields ($\Omega_E \subset \mathbb{R}^2$), a straight-forward visual representation can be produced immediately in the form of a color-mapped planar representation according to the color maps described above. To facilitate the transfer of insights gained by visualizing the classification space back to the ensemble domain, we can directly assign colors in the ensemble domain Ω_E. Each position is assigned its corresponding classification color. The result can be seen in, e.g., Figure 4.11.

Since the green areas (low variances) indicate similar behavior across the entire ensemble, in these areas traditional vector field visualization methods such as *Line Integral Convolution* (LIC) [CL93] can be applied to a representative field to jointly visualize the flow, e.g., from one of the ensemble members. This is achieved by assigning a lower opacity to the green color and then blending the visualization over a LIC-image. An example is provided in Figure 4.7. Note that LIC images are representations of stationary flow and as such have a limited expressiveness in highly time-varying flows. Alternative time-varying options include e.g., *UFLIC*, which was demonstrated to work well in combined visualizations [Gar+09].

For three-dimensional fields ($\Omega_E \subset \mathbb{R}^3$), occlusion renders such a straight-forward method impossible. Instead, we visualize the ensemble domain using volume rendering. The quality and usefulness of volume rendering depends strongly on the choice of a transfer function, that maps the visualized data values to color and opacity [DCH88]. For this reason we define opacity in two stages: Analogously to the two-dimensional case, the low-variance, green areas are assigned a lower opacity than high-variance regions. In a second step, this opacity value is used as input for a user-definable, bump-shaped opacity transfer function that

specifies the final opacity of a voxel in the rendering. This yields a depiction of structures that exhibit either high individual variance, high joint variance, or both, thereby capturing the areas which are of most interest in comparative flow ensemble analysis. An example is depicted in Figure 4.6.

4.4.3 Interaction and Exploration

Beyond the semi-automated color mapping outlined above, it is of course necessary to allow user control of color and opacity mapping. We provide a number of interaction modes of increasing complexity.

First, it is not immediately obvious which variance values should be considered "low" or "high", especially since the maxima of joint and average variance are usually represented by outliers. Also, such a classification may depend on the data set, application domain, and features of interest. To steer the corresponding mapping, we allow a user to select a rectangular area within the visualization of the classification space – initially set to the extent of the classification space – that is used as the basis of the above color mapping. The four corner values are interpolated smoothly within the rectangle, see Figure 4.7. For data points outside the rectangle, the color of the closest point on the rectangle is used. Further, the inside of the rectangle defines a transitional area to cover variance ranges for which a clear classification cannot be made.

Using the techniques described so far, it is possible for a user to gain an overview of an ensemble's joint transport behavior. We provide two further tools for a more specific investigation. First, in classification space, we enable a user to manually highlight specific regions using standard brushing tools. The visualization of the ensemble domain is instantly updated to reflect the change of the color map. This allows a user to closely examine features such as classification-space clusters, that may be apparent in the scatter plot.

Moreover, in the ensemble domain, we support the placement of path lines to visualize the flow locally, as shown in Figure 4.9. For a given seed point, a path line is started in each ensemble member and then displayed in the ensemble domain. While this method is not suitable for visualizing the flow globally, it is highly useful to provide additional insight into the local transport behavior. When applied to regions of low joint variance, the result is a tight bundle of path lines that indicates the flow direction in all ensemble members. Outliers can be identified immediately as they diverge from the bundle.

When a path line is seeded in an area of high joint variance, the curves tend to fan out in different directions. Trends can be identified as clusters of the path line endpoints. This

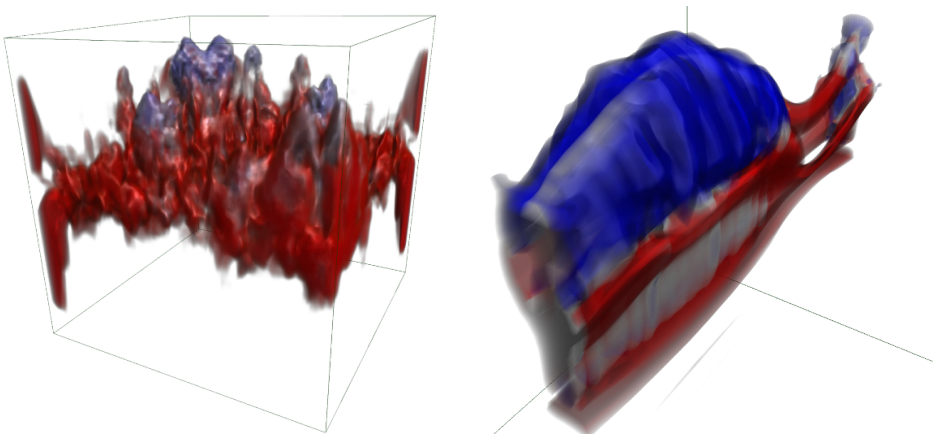

Figure 4.6: Left: Volume rendering of individual variance and joint variance for the classified Rayleigh-Taylor Instability. Transport behavior in the middle of the data set is detected to be strongly dissimilar as expected. Right: Volume rendering of individual and joint variance for the classified 3D convection data set. The classification procedure proposed in this work reveals strong disagreement between ensemble members along lateral sides of the ensemble.

is especially useful in connection with a visualization of the number of trends, since each trend can be explicitly identified.

4.5 Experiments

4.5.1 Implementation

The implementation of our visual analysis framework is comprised of tasks that require different amounts of processing time and memory. For the construction of advected neighborhoods $F_{t_0}^{t}(N_{\mathbf{x}}(t_0))$, we sample the flow fields uniformly and compute a corresponding flow map $F_{t_0}^{t}(.)$ for every field of the ensemble. This pre-computation is identical to the flow map computations required for classic FTLE techniques. Depending on the underlying grid, interpolation scheme, flow map resolution, and the integration time $t - t_0$, computing the flow map can be computationally expensive. In the examples shown here, the computation time of a complete set of flow maps for all members of an ensemble ranged from 2 minutes to 11.5 minutes in total on a low-cost, commodity workstation, leveraging basic OpenMP parallelization with 8 threads (Table 4.1).

With these sampled flow maps, individual and joint variance computation for all points in the ensemble is reduced to performing look ups in $F_{t_0}^{t}(.)$ for particles in $N_{\mathbf{x}}(t_0)$ followed by a PCA of $F_{t_0}^{t}(N_{\mathbf{x}}(t_0))$. These computations are significantly faster than the flow map

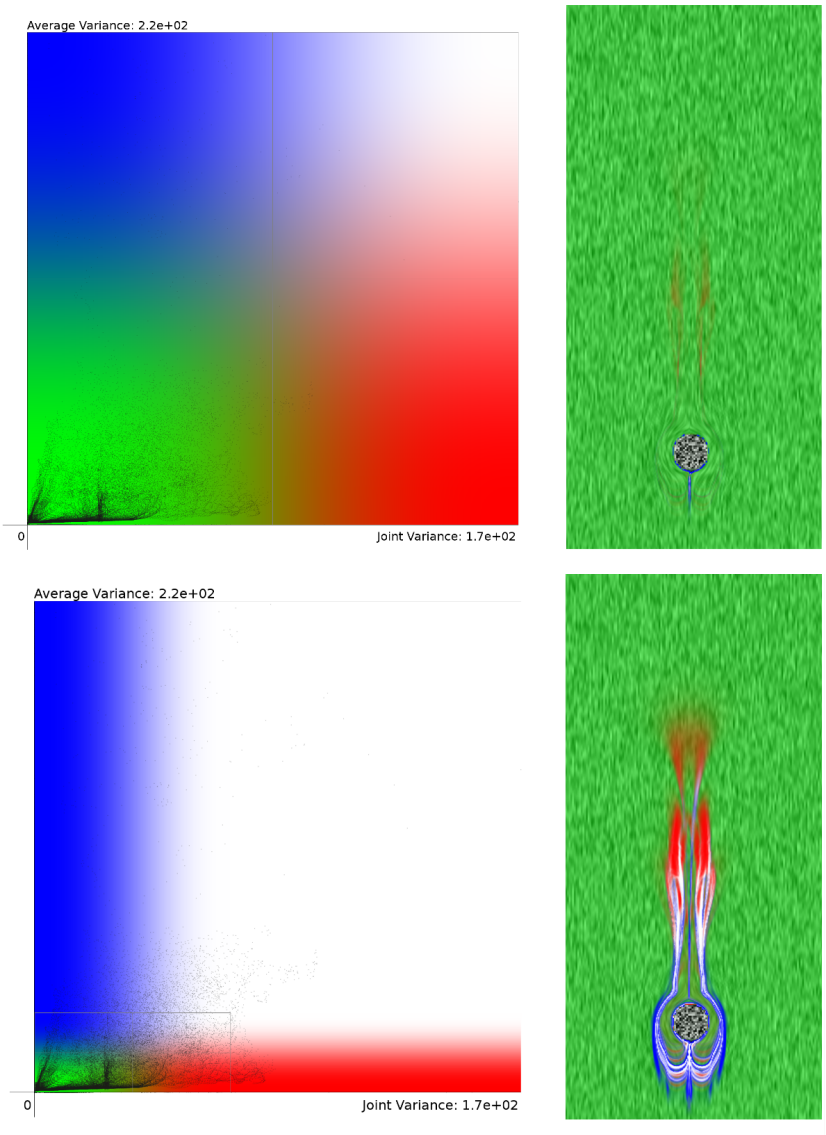

Figure 4.7: Visual comparison of transport in the 2D convection data set. Modifying the variance classification changes the notion of high and low variances. The resulting color map facilitates easy distinction of regions with similar and dissimilar transport behavior.

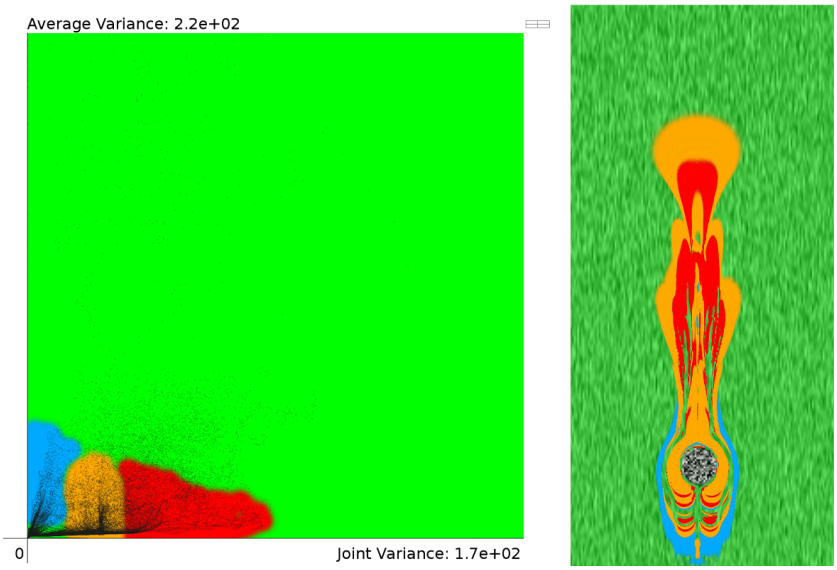

Figure 4.8: Interactive brushing in the classification space provides a direct semantic link between classification space and the ensemble. Apparent features that are visible in the scatter plot of the classification space can be painted with user-specified colors. Each spatial position in the simulation domain is then colored according to the correspondig color in the classification space.

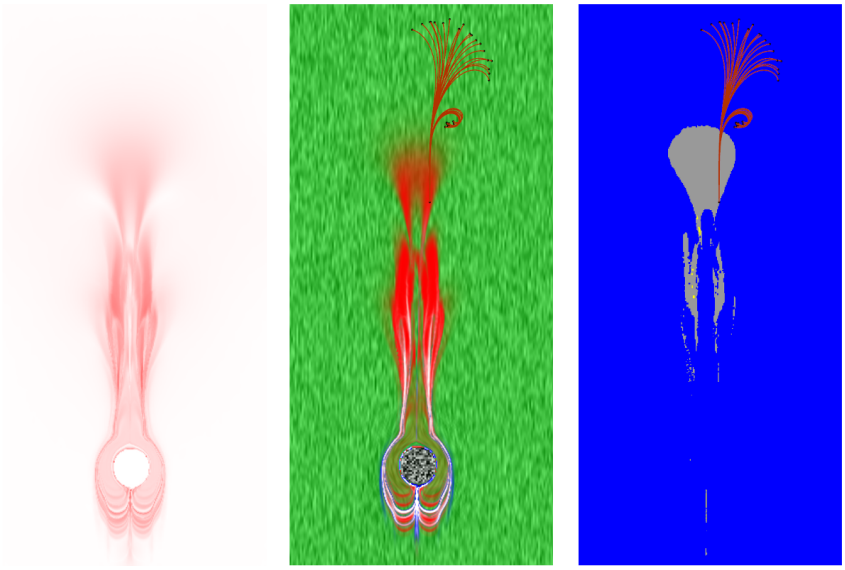

Figure 4.9: Left: The naïve visualization based exclusively on joint variance values (red shading) does not indicate comparison quality or certainty. Center: The joint classification scheme indicates regions where a comparison is not reliable without further exploration (white and blue). The visualization indicates that both regions interact in an non-deterministic fashion. Path lines seeded in red regions – where the data sets show low average individual variance and high joint variance – reveal the presence of strong transport variance, as expected. Furthermore, two distinct transport trends are visible. Right: Automatic trends analysis using MST shows that the data set consists of regions with one prevalent trend (blue) and smaller areas, where two trends are visible (gray).

Table 4.1: Timings for pre-computation of flow maps and variance values for our example data sets. The pre-computation cost depends heavily on the desired resolution and on the size of the observed time interval. Flow maps have to be computed separately for each ensemble member, and typically take up the largest portion of pre-computation time (given here as the product of the ensemble size and the average time for a single ensemble member). The column 'variances' contains the time required for computing both joint variance and individual variance, as well as the number of trends, for the entire ensemble data set.

	resolution	time steps	flow maps	variances
Convection 2D	256×512	100	$30 \cdot 23$s	15s
Stirring	336×336	150	$20 \cdot 9$s	7s
Rayleigh-Taylor	$68 \times 68 \times 68$	51	$8 \cdot 41$s	3s
Convection 3D	$64 \times 128 \times 64$	11	$10 \cdot 12$s	7s

computation and were also performed with multiple threads, but are not real time. The same holds for the computation of the minimum spanning trees for trends analysis.

After this pre-processing has been performed, the proposed interaction and exploration techniques work in real-time using standard graphics hardware.

4.5.2 Data Sets

We applied our approach to the following three different scenarios.

Convection The convection data sets are a simulation of flow around a hot, cylindrical object. Material at rest is heated around the cylinder, begins to rise, and forms a plume. By slightly perturbing the initial velocity conditions at the bottom of the domain, a two-dimensional ensemble of 30 flow fields and a three-dimensional ensemble of 10 flow fields were generated. The convection process splits the domain into two regions of upwards streaming material. A key feature to investigate is how fluid transport on the two sides interacts in the middle of the data set, which structures are formed during the convection process, and how robust they are under perturbation.

We examined a two-dimensional convection data set for a time period from $t_0 = 0$ up to $t = 5$, comprising 100 time steps in each ensemble, for a total ensemble size of 786MB. A three-dimensional data set also consists of 100 time steps with a combined size of 6GB; we considered a time period from $t_0 = 2.5$ to $t = 3$ (11 time steps).

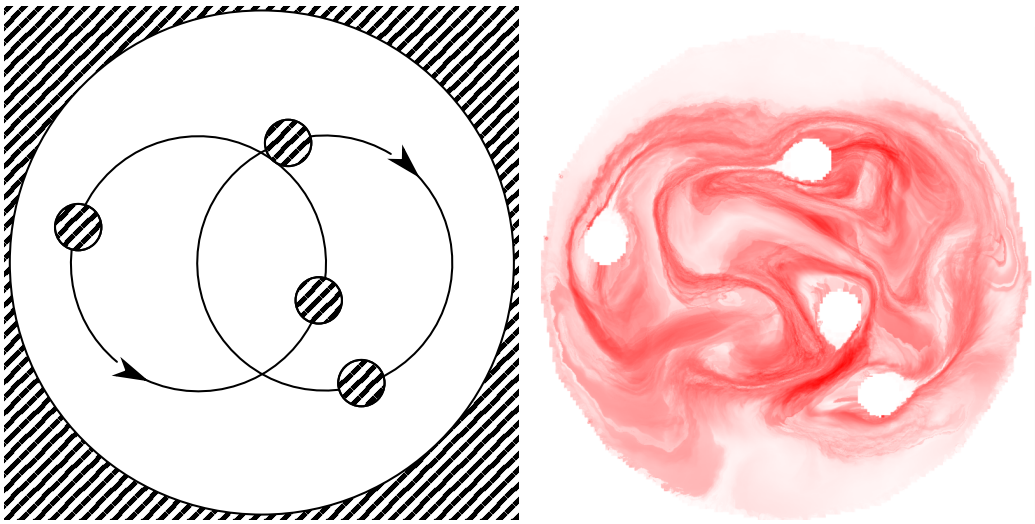

Figure 4.10: Left: The two-dimensional stirring data set features two counter-rotating mixing rods in a circular container. Right: Looking only at the naïve joint variance rendering, it is not possible to distinguish reliable disagreement from uncertain disagreement brought on by the presence of divergent flow behavior.

Industrial Stirring The stirring data set is a set of 20 two-dimensional flow fields resulting from the simulation of mixing in a stirring apparatus. The device consists of two counter-rotating pairs of mixing rods that stir a medium in a cylindrical tank (Fig. 4.10). The observed time range of the simulation from $t_0 = 5$ to $t = 10$, corresponding to 50 time steps, covers approximately one complete revolution of the stirring rods. The ensemble was generated by slightly varying the viscosity of the fluid to investigate mixing quality of the device for a range of different fluids, and totals 646MB. The key question for this data set regards the effectiveness of the stirring process. An ensemble visualization is expected to be able to identify regions where the mixing quality is high or low throughout the ensemble.

Rayleigh-Taylor Instability The Rayleigh-Taylor instability is the flow resulting from the mixing of a heavy fluid placed on top of a fluid with lower density. As gravity displaces the heavy fluid into the light fluid, unstable/chaotic behavior occurs. We use an ensemble of 8 three-dimensional simulation runs created for slightly varying density values, with each member comprised of 50 time steps and a total data size of 1.4GB. The chaotic transport behavior present in the data set suggests a highly unstable process, challenging the expressiveness of similarity-based ensemble visualization techniques. We visualized the data set in the time interval from $t_0 = 0$ to $t = 1$.

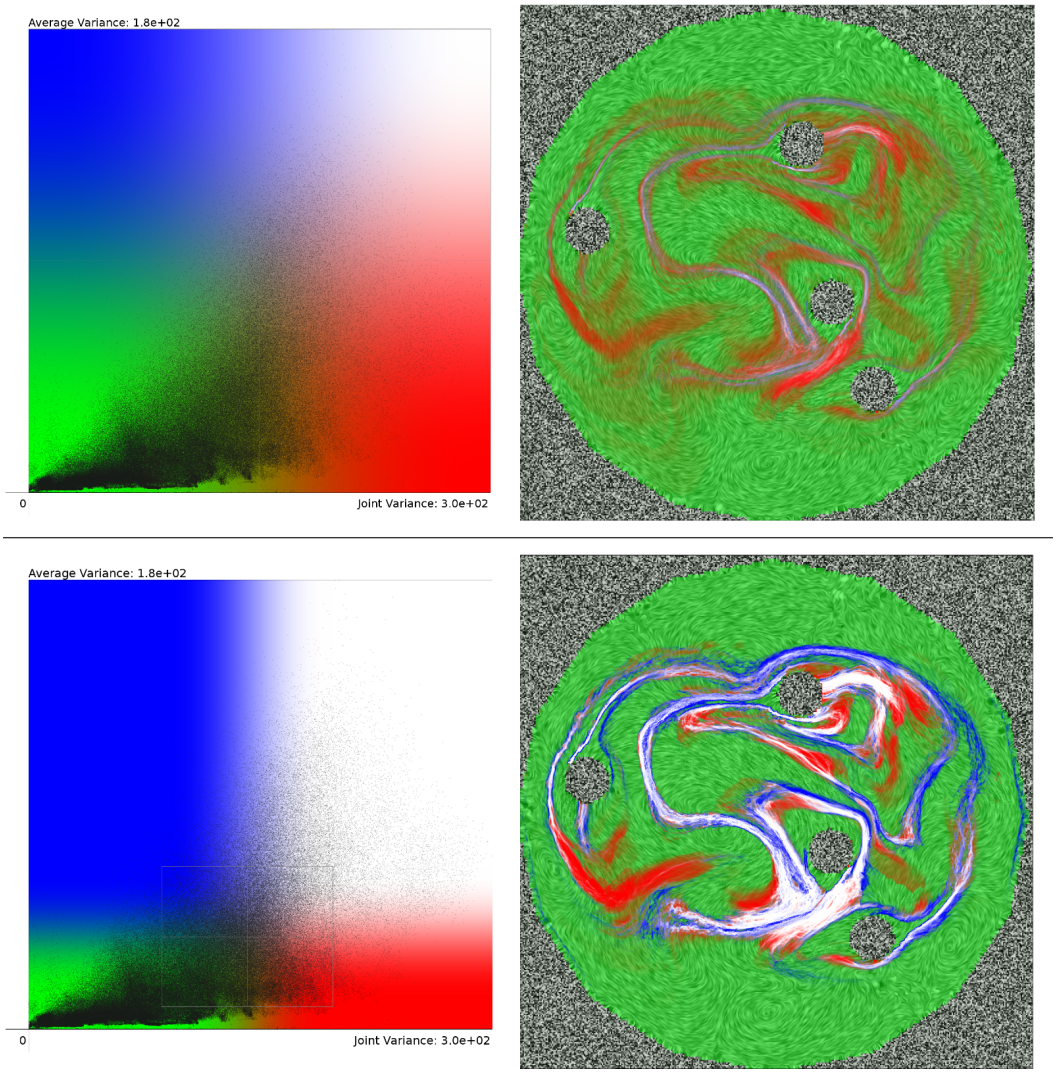

Figure 4.11: Visual analysis of an ensemble generated by varying fluid viscosities in a stirring apparatus simulation. Outer regions and especially the upper and lower parts of the data set reveal very similar and, therefore, deterministic transport behavior with low variances, indicating little to no fluid mixing. Varying the classification space color map allows to distinguish between reliable disagreement (red) and uncertain disagreement (white).

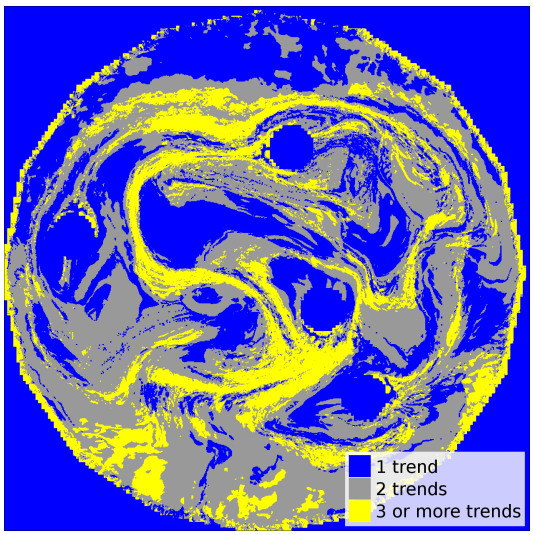

Figure 4.12: A trends analysis allows for segmentation of the data set into regions that show one, two, or more different transport behaviors. Interactive path line seeding reveals that trends in this data set are generally not caused by a grouping of different ensemble runs, but by outlier behavior present in the data set.

4.5.3 Results and Analysis

We begin by analyzing the two-dimensional convection data set. Figure 4.7 shows an application of our techniques to that ensemble. The naïve joint variance-based visualization (bottom left) seems reminiscent of typical FTLE-based visualizations, which can be expected given the conceptual similarity. However, while FTLE is used to visualize single flow fields, our method incorporates the entire ensemble consisting of 30 flow fields.

Looking at the classification space plot, we observe that the majority of the data set's points are concentrated in the lower left corner with, compared to the maximal values, low average individual variance and low joint variance. Initially, the color map spans the entire classification domain, mapping blue and red to their axes' respective maximal values. This results in most points of the data set being classified as 'low variance'. Directly below the cylinder, a small region of very high individual variances can be identified where the flow is separated either towards the left or the right side of the cylinder. Above the heating cylinder, two red ridges are faintly visible, indicating strong differences between the ensemble members' transport behavior. These features become more pronounced when the color map is modified to lower the thresholds for high individual and joint variance.

As indicated, our classification improves expressiveness over a naïve joint variance-based visualization, by including regions where unstable trajectories do not allow for reliable comparisons. Exploration in classification space and the ensemble domain facilitate goal driven comparison of the underlying transport behaviors. By applying our techniques we

find that the lateral regions of the data set correspond to extremely similar flow behavior and only the middle section above the heating cylinder contributes to transport variance in the ensemble. Interestingly, regions with high variance show some correlation with our trends analysis. As a consequence, two distinct trends are visible, where flow fields in the ensemble disagree reliably. This is an interesting observation, as it indicates that the oscillation patterns forming behind the convection cylinder are to a large extent the result of a combination of two distinct behaviors caused by slightly uneven heating of the two lateral sides.

In the lateral regions of strong agreement (green), the ensemble does not carry additional information as compared to a single flow field. Since the region is green for a wide choice of classification thresholds, ensemble generation could be optimized by avoiding the repeated simulation of this region in every run of the ensemble.

Figure 4.11 applies the same techniques to the stirring simulation ensemble. Similar to the convection data set, the highest concentration of points seems to be found near the lower left corner of the classification space. In general, however, the plots seems to be more spread out, with a diagonal, wedge-like structure fanned out towards the center. With the initial color map, several areas of high joint variance can already be identified in the ensemble domain.

Because green regions indicate areas where average variance and joint variance are low, these parts of the ensemble represent significantly less mixing than the rest of the data set. There are two clearly visible such regions at the top and bottom of the data set, where all data sets show similar low variance behavior and little to no mixing occurs. For the design of a stirring apparatus, such regions are to be avoided to improve overall mixing performance. Our observations strongly suggest modifications to the design.

Trends analysis reveals that a number of different trends are present in the data set. While the variance-based visualization and the trends visualization do not allow to identify individual ensemble members that cause outliers or trends, this can be achieved by seeding path lines at points of interest. In the example, one of the two detected trends consists of only a single outlier, while the remaining ensemble members appear to behave consistently. A domain expert could now examine if the parameter setting of the single outlier differs from the remaining ensemble. Furthermore, identifying the presence of distinct trends is of immense importance whenever ensemble simulations are used to make predictions about the behavior of natural phenomena. We note that path line-based trends analysis for large numbers of ensemble runs and in 3D requires additional research to avoid visual clutter.

In the 3D Rayleigh-Taylor instability ensemble (Fig. 4.6, left), high values of both individual and joint variance confirm the expectation that transport behavior near the mixing zone in

the center varies greatly across the ensemble. Chaotic behavior leads to incomparable flow and non-deterministic structures, as expected. These regions are successfully identified by our method. With our visual analysis techniques the stochastic process of chaotic transport in Rayleigh-Taylor flow can be separated into two distinct regions: incomparable, non-deterministic flow at the interface and similar deterministic behavior at the upper and lower regions.

Results generated with the three-dimensional version of the convection data set are presented in Figure 4.6 (right). The system setup is similar to the 2D case, with the exception that the diameter of the heating cylinder varies along the depth axis. The resulting transport behavior analysis reveals very similar structures to the 2D field. An interesting observation is how the flow structure revealed by our joint classification strategy resembles a union of individual and joint variance structures.

4.6 Discussion

Ensemble flow data sets are generated by application scientists to compensate for sensitivity with regard to input parameters, numerical errors, and inaccurate inital conditions. An area of particular interest is the material transport behavior, which is described by fluid parcel trajectories. Manual, pairwise comparisons of such trajectories in ensemble data sets is tedious at best, and completely infeasible for realistic settings featuring many simulation runs or large spatial domains. Moreover, we have shown that naïve approaches based solely on the spread of trajectory end points are thwarted by divergent flow behavior.

The methods described in this chapter extend comparative visual analysis of ensemble data to time-varying flow fields. More specifically, we have presented variance-based comparison techniques for Lagrangian fluid transport in 2D and 3D vector field ensembles. To reduce the limitations of exclusively variance-based analysis, we have combined the proposed technique with automatic trend detection and interactive exploration capabilities. Our experiments have shown that both deterministic and incomparable transport behaviors in ensembles can be identified reliably.

The presented, trajectory-based techniques allow application scientists to quickly and reliably identify regions of agreement and disagreement regarding the material transport behavior in flow field ensembles. The methods can be adapted to application-specific requirements in several ways, for example, by defining advanced trajectory similarity metrics, or by including directional information of divergence during a similarity evaluation.

In the future, we plan to investigate how the choice of simulation parameters relates to joint and individual variance values and to the number of trends as assessed by our trend

detection method. Of particular interest is the question if specific trends are associated with distinct regions in the simulation parameter space.

5 Visualization of Uncertain Multi-Phase Fluid Simulation Data

The applications that were discussed in this thesis up to this point involve uncertainty of flow simulation data either in the form of errors stemming from a lossy representation of simulation output (Chapter 3), or as an ensemble spanning a range of possible outcomes given simulation inaccuracies and incomplete knowledge of "actual" initial conditions (Chapter 4). In both cases, individual simulations calculate an approximate solution for a single, precise scenario which may or may not be captured. In contrast, this chapter discusses an application where the output of simulations can be seen as uncertain in a different way, because the simulation model itself is probabilistic.

Techniques from computational science have over the past decade replaced and augmented physical experiments in the design and optimization of engineering problems. While enabling unprecedented possibilities for insight into these problems, such numerical simulation techniques are subject to limitations in turn. In particular, for some problems, accurate simulation does not always result in data modalities that are comparable to those derived physically. In these cases, it is difficult to directly compare numerical and physical experiment, and to transfer an often rich body of domain knowledge obtained from physical experiment to numerical simulation output.

In this chapter, we consider the simulation of bubble columns, in which gaseous bubbles or liquid droplets interact with a liquid or gaseous medium. The surrounding medium is referred to as the continuous phase, whereas the bubbles make up the dispersed phase. Depending on the media employed, such bubble columns are subject to a design process that is aimed at optimizing, e.g., flow patterns to yield sufficient absorption of a gaseous solvent in a liquid medium. From the point of view of computational modeling, so called multi-phase fluid simulations are used in the simulation of flow in bubble columns. A core concept is to not model bubble formation directly — this would require prohibitively fine resolution levels — but rather model it as a stochastic process that is averaged over the cells of a computational grid.

For Method of Moments (MoM) solvers [HK64], the positions of bubbles are probabilistic. Instead of modeling bubbles explicitly, the configuration of the dispersed phase is described

by scalar fields of moments and volume fractions. Each location in the domain is assigned a population density that defines how many bubbles can be expected per unit of volume. A common approach for visualizing such datasets is to represent the moments via color coding (Fig. 5.9). Such a visualization makes it difficult to put the sizes of bubbles in relation to physical experiments (Fig. 5.10).

In this chapter, we present a visualization approach that generates a set of bubble positions by sampling the density field. Each bubble is shaped using a geometric bubble model, which is adapted, in a physically plausible manner, according to local flow characteristics. To maximize coherence between time steps while complying with the density field, the set of bubbles is advected by the flow field and updated over time. Visualized using traditional visualization tools or with photorealistic rendering, the results provide an intuitive graphical representation that allows direct comparison with physical experiments.

In particular, after providing background on Method of Moments simulations and reviewing related work in Sections 5.1 and 5.2, we make the following contributions[1]:

- We provide a parametric model for the graphical representation of bubble shapes, based on local moment information and flow parameters in Section 5.3.
- We describe a sampling approach to construct and update a specific set of bubbles to approximate a bubble distribution from moment information in Section 5.4.

Furthermore, we discuss implementation details (Section 5.5) and provide an in-depth case study to illustrate the utility of our approach (Section 5.6), before concluding on the presented work in Section 5.7.

Even though the presentation of our method is focused on a specific setup where gaseous bubbles are dispersed in a liquid medium, the described techniques can be applied to any multi-phase fluid simulation where a MoM solver is used to track populations of bubbles or droplets.

5.1 Background: Method of Moments CFD Simulation

In Computational Fluid Dynamics, multiphase flow simulations are used to study the interaction of different liquid or gaseous media. For settings that feature a continuous phase and a dispersed phase — for example gas bubbles in a liquid medium — with a non-negligible volume fraction of the dispersed phase, Eulerian-Eulerian methods are used [WW13]. With these approaches, the dispersed phase is not represented through individual bubbles or

[1] The results described in this chapter were published in [Hla+16b; Hum+17b; Hum+17a; Hla+18].

droplets, but instead the different phases are modeled as interpenetrating continua, represented by volume fractions [Rus03].

To compute the evolution of the bubble population over time, enhanced derivatives of the Method of Moments (MoM) [HK64] — including the Quadrature Method of Moments (QMoM) [McG97], Direct Quadrature of Moments (DQMoM) [MF05], One Primary and One Secondary Particle Method (OPOSPM) [Dru+10; Hla+16a], and more — are seeing increased use. These methods are based on the moments of the bubble size distribution. The r-th moment of this distribution is defined as

$$M_r(\mathbf{x},t) = \int_0^\infty n(L,\mathbf{x},t)L^r dL, \quad \text{[Mar+03]} \tag{5.1}$$

where the number density function $n(L,\mathbf{x},t)$ describes the number of bubbles with the characteristic length L. In most applications, only the first four moments are considered. The zeroth moment describes the total number of bubbles, the first moment the total length, the second moment the total area and the third one the total volume.

The simulation data that were used in this work were created using an OPOSPM solver, which offers reduced computation times. Here, only the change of the zeroth moment $M_0(\mathbf{x},t)$ is directly tracked [Hla+16b]. $M_0(\mathbf{x},t)$ defines, for each location and time, the expected local value of the number of bubbles per volume. The overall volume of the dispersed phase is represented by the third moment $M_3(\mathbf{x},t)$, which is obtained through its relation to the volume fraction of the dispersed phase:

$$\alpha_{\text{disp}}(\mathbf{x},t) = \frac{\pi}{6} M_3(\mathbf{x},t) \quad \text{[Dru+10; Pet+13]} \tag{5.2}$$

From these two moments, the volume-based diameter

$$d_{30}(\mathbf{x},t) = \sqrt[3]{\frac{M_3(\mathbf{x},t)}{M_0(\mathbf{x},t)}} \quad \text{[Dru+10]} \tag{5.3}$$

can be calculated, which defines the diameter of a hypothetical, spherical bubble at the current position.

In addition to the moments, the velocities of the dispersed phase $\mathbf{u}_{\text{disp}}(\mathbf{x},t)$ and of the continuous phase $\mathbf{u}_{\text{cont}}(\mathbf{x},t)$ are available and can be written as simulation output. The two velocity fields are not equal, because bubbles of the dispersed phase move through the continuous phase while the continuous medium moves around them. Small features like the flow around individual bubbles are not captured; instead, the velocity field provides the general motion of continuous and dispersed matter in the simulation.

5.1.1 Tasks and Requirements

To maximize the usefulness to their work, the domain scientists formulated a number of required features and properties for the proposed visualization:

Size and Density Both bubble size and bubble density should be indicated visually.

Comparability Direct visual comparisons between simulation data and physical experiments should be possible.

Temporal Coherence The visualization should be coherent over time, making it possible to track individual bubbles. If animation is used, it should provide a visual indication of bubble motion.

Non-Intrusiveness No non-standard or unfamiliar new user interfaces should be necessary; ideally, the proposed visualization should integrate well with the domain scientists' usual workflow that involves commonly used and readily available visualization software packages (e.g., VisIt [Chi+12] or ParaView [Aya15]).

5.2 Related Work

Adding to the related work already discussed in Chapter 2 and Section 5.1, this section lists further works that are relevant to this chapter.

Droplets and Bubbles Rohith et al. [Roh+11] presented a visualization tool for particle collision events. Individual collisions can be picked and displayed with an isosurface-based contextual visualization of nearby flow. Hlawitschka et al. [Hla+11b] visualized multi-phase fluid simulation data by placing spherical glyphs based on the droplet density field.

Geometric Material Models Easwaran et al. [Eas+16] introduced an automatic measurement method for fiber thickness in scanning electron microscopy images. They utilize renderings of synthetic fiber data sets to validate their automatic measurement approach.

5.3 A Parametric Bubble Model

The Eulerian multi-phase MoM-based simulations featured in this chapter do not consider the shape of bubbles explicitly. Instead, the computed moments describe – for each location in the domain – more general properties such as overall characteristic length, volume, and surface area of the bubble population. A straight-forward approach would be, asuming a set of bubble locations is known, to represent all bubbles by spheres and scale them according to the volume-based diameter $d_{30}(\mathbf{x},t)$, which can be derived from the moments. However, as can be seen in Fig. 5.10, bubbles are typically not spherical or even ellipsoidal. They can exhibit a range of shapes, including slightly flattened spheres, ellipsoids, and truncated ellipsoids, which can not be explained directly using the information contained in the moments.

The shapes of droplets or bubbles in liquids have been studied in the past, for example by Clift et al. [CGW78] and Bhaga et al. [BW81], who compiled the results from a large number of physical experiments. For the special case of bubbles or droplets rising or falling freely in (approximately) infinite media, the shapes were described in terms of two flow characteristic numbers, the Eötvös number Eo and the Reynolds number Re. Eo characterizes the ratio between gravitational forces and surface tension, while Re describes the ratio of intertial and viscous forces. The two quantities are defined in terms of the gravitational constant g, the diameter of a volume-equivalent sphere d_e, density difference between the dispersed and the continuous phase $\Delta\rho$, the surface tension σ, the density of the continuous phase ρ, the velocity of the bubble relative to the continuous phase U, and the viscosity of the continuous phase μ:

$$\mathrm{Eo} = g\Delta\rho d_e^2/\sigma, \tag{5.4}$$

$$\mathrm{Re} = \rho d_e u/\mu. \tag{5.5}$$

Bubble shapes were categorized with labels (examples: "spherical", "ellipsoidal" or "spherical cap") and charted in diagrams of Eo and Re [CGW78, p. 27], [BW81, p. 70].

We utilize these results to provide a more intuitive visualization and to facilitate comparison of simulation data to physical experiments by adapting the shape of bubbles in our visualization.

We restrict the bubble shape to stable shapes that are rotationally symmetric and define them based on the Eötvös number Eo, the Reynolds number Re, the volume-based diameter d_{30}, and the direction of the relative velocity $\mathbf{u}_{\mathrm{rel}} = \mathbf{u}_{\mathrm{disp}} - \mathbf{u}_{\mathrm{cont}}$.

The bubble geometry is obtained by rotating one half of a B-spline boundary curve representing the bubble's cross section.

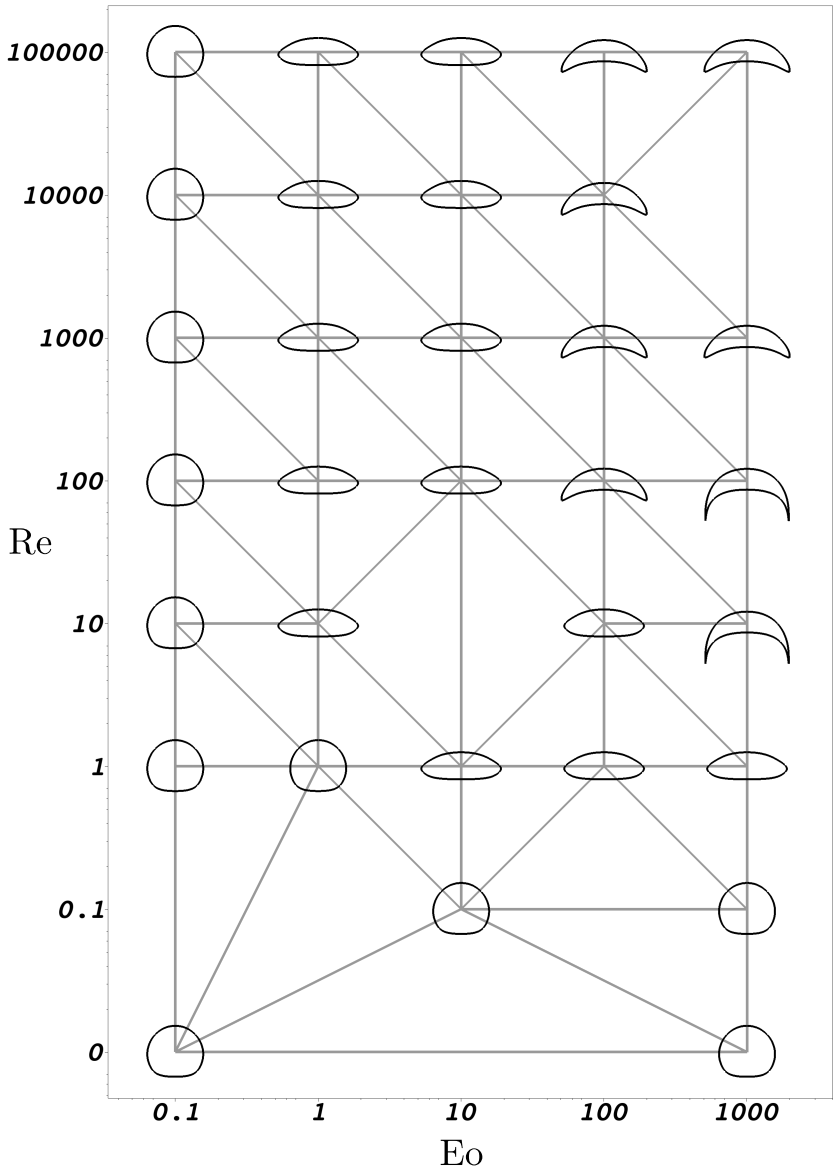

Figure 5.1: Bubble shapes are determined based on pre-defined representatives which are arranged in the diagram of the Eötvös number and the Reynolds number.

Bubble Shape Representatives The control polygon of the cubic B-Spline curve that is used to construct the bubble geometry is defined based on a pre-designed set of templates. Roughly following the shape classifications in [CGW78] and [BW81], we created a number of template bubbles for a set of combinations of Eo and Re (Fig. 5.1). These representatives can easily be modified for different measurement data or in case of more specific requirements. To find the control polygon for a new bubble, the three closest templates in the Eo-Re space are selected. Each vertex of the control polygon is calculated using barycentric interpolation of the corresponding vertex in the three templates (Fig. 5.2).

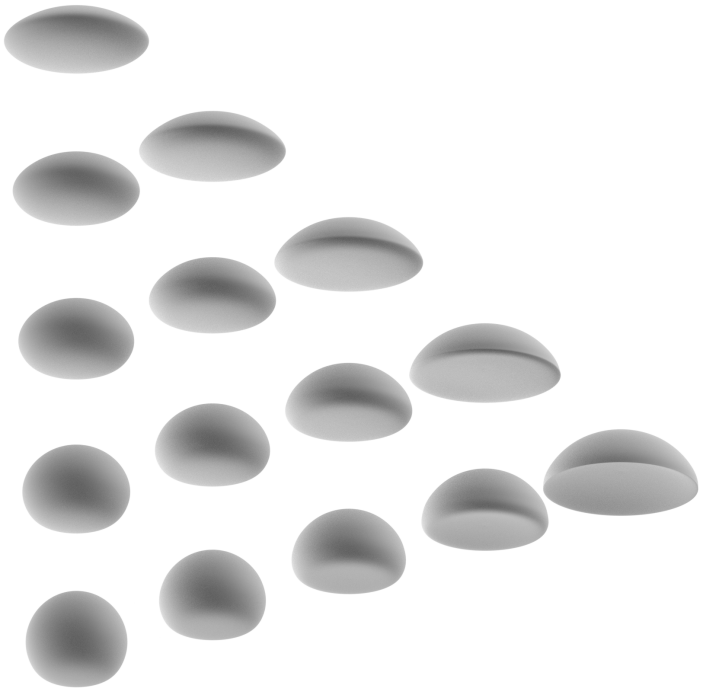

Figure 5.2: Intermediate shapes between three arbitrary bubble representatives can be obtained using barycentic interpolation.

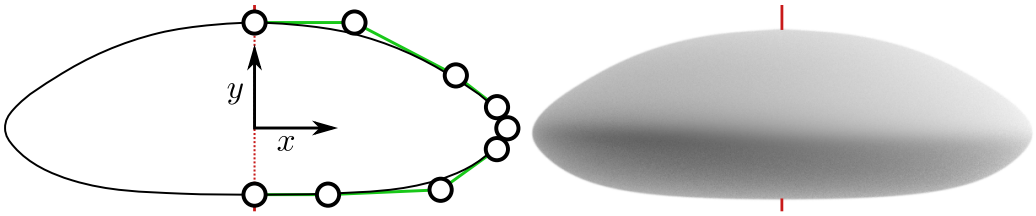

Figure 5.3: Left: The outline of each bubble is defined by a B-Spline curve in the *xy* plane. Right: The three-dimensional bubble is obtained by rotating the outline curve around the *y*-axis (red).

Bubble Geometry The boundary curve is modeled as a cubic B-Spline curve in the *xy*-plane, with the first and last point placed on the *y*-axis. The knot vector is chosen in a way that the endpoints of the curve coincide with the endpoints of the control polygon by repeating the first and last knot three times. For our examples, we used nine control points per curve, resulting in the knot vector $\{0,0,0,1,2,3,4,5,6,6,6\}$ [2]. The B-Spline curve is then discretized by applying the de Boor algorithm for consecutive parameter values, resulting in a piecewise linear outline curve. A triangle mesh is produced by incrementally rotating the curve around the *y*-axis. Successive line segments are connected with triangle strips. This triangle mesh already represents the basic shape of the bubble.

[2] A comprehensive guide to B-Spline curves, knot vectors, and related algorithms can be found in Farin [Far02].

Adjustment To adjust the size of the bubble, it has to be scaled in such a way that its volume equals the volume of a sphere of diameter d_{30}. This is achieved by calculating the volume of the triangle mesh and then multiplying each vertex with a scale factor s:

$$V_{\text{target}} = \frac{4}{3}\pi(d_{30}/2)^3 \tag{5.6}$$

$$s = \sqrt[3]{V_{\text{mesh}}/V_{\text{target}}} \tag{5.7}$$

Finally, the resized triangle mesh is rotated to align with the velocity $\mathbf{u}_{\text{rel}}$ of the dispersed phase in relation to the continuous phase.

5.4 Bubble Distribution: Sampling and Updating

The zeroth moment $M_0(\mathbf{x},t)$ defines the expected number of bubbles per volume unit at the location $\mathbf{x}$ and time t. We use a sampling approach to obtain a set of bubble positions from this density field, which can be interpreted as a probability density function. When multiple time steps are considered, a naïve approach would be to calculate the sampling separately for each time step. However, such an approach would result in bubbles suddenly appearing and disappearing, with no apparent coherence between consecutive time steps. Instead, to enable visual tracking of individual bubbles in our approach, bubbles are advected along the continuous phase velocity field as time progresses. To handle in- and outflow as well as coalescence and separation events, the distribution is then updated to reflect the density field in each time step.

This section first describes how the density field can be sampled to obtain an initial set of positions, before explaining how the bubble set is updated over time.

5.4.1 Filtering

For numerical reasons, the zeroth moment $M_0(\mathbf{x},t)$ as produced by the simulation is always greater than zero, even though large areas of the domain may not contain any bubbles at all. In such areas, the volume fraction of the dispersed phase, $\alpha_{\text{disp}}(\mathbf{x})$, is very close to zero. At the same time, bubbles cannot exist in the gas-filled area above the liquid surface. This area has a very large dispersed volume fraction of close to one. If M_0 were used directly, this would result in the formation of many extremely small bubbles in areas that should only contain liquid, or very large bubbles in areas that should contain only gas. Both areas should not contain any bubbles at all. To avoid this problem, only cells that satisfy a threshold

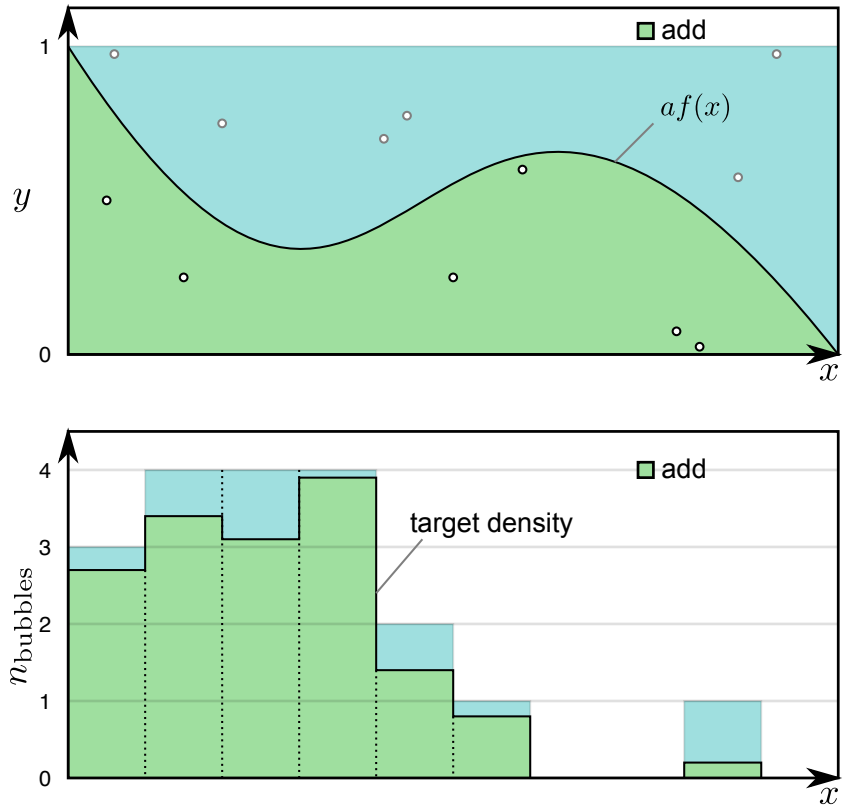

Figure 5.4: Initial sampling using rejection sampling; one-dimensional examples. Top: In the basic rejection sampling algorithm, samples (x,y) are drawn uniformly from $[x_{\min},x_{\max}] \times [0,1]$ and kept if $y \leq af(x)$ (green); otherwise they are rejected (cyan). Bottom: Rejection sampling is performed separately for each cell, minimizing the number of rejected samples. The y-axis in the illustration represents n_{required} (eq. 5.9).

$\alpha_{\min} \leq \alpha_{\text{disp}} \leq \alpha_{\max}$ are considered for bubble generation. In all other cells, the density is forcibly set to zero:

$$M_{0,\text{filtered}}(\mathbf{x},t) = \begin{cases} M_0(\mathbf{x},t), & \text{if } \alpha_{\min} \leq \alpha_{\text{disp}}(\mathbf{x},t) \leq \alpha_{\max} \\ 0, & \text{otherwise} \end{cases} \tag{5.8}$$

For the sake of brevity, use of M_0 always refers to $M_{0,\text{filtered}}$ in the remainder of this chapter.

5.4.2 Initial Sampling

The density field $M_0(\mathbf{x})$ defines, for each location $\mathbf{x}$ within the domain, how many bubbles per volume should be present. Since M_0 is constant within each mesh cell, the exact number of bubbles required within each cell i can be calculated:

$$n_{\text{required}}(\text{cell}_i) = M_0(\text{cell}_i) \cdot \text{volume}(\text{cell}_i) \tag{5.9}$$

This number is generally not integer, which means that a set of bubbles with the correct number of bubbles within each cell cannot be achieved in practice; instead, the number of bubbles should on average satisfy the density requirement.

In a naïve approach, a set of bubble positions could be obtained by rounding and choosing, e.g., $\lfloor n_{\text{required}}(\text{cell}_i) \rfloor$ random positions within the cell. However, the overall density of the resulting set of bubbles would always be smaller than the required density M_0; a problem that is exacerbated by small cell sizes or low densities. If, for instance, the density field were to mandate slightly less than one bubble per cell, the naïve algorithm would never produce any bubble locations.

Instead of simply counting the expected number of bubbles per cell, we decide to interpret the density function like the probability density function of a stochastic distribution. A straight-forward method for drawing samples from any given probability density function $f(x)$ is rejection sampling [Von51; Liu08]: A factor a is chosen such that $af(x) \leq 1$. Then, two random numbers $x, y \in [0,1]$ are chosen uniformly. x is kept as a sample if $y \leq af(x)$ and rejected, otherwise. An illustration of this algorithm can be found in Figure 5.4 (top).

In the following, we describe how this basic approach can be adapted to bubble density fields.

Starting with an empty set of bubble locations, we first produce an initial sample using a rejection sampling approach. To avoid having to reject too many samples, we treat each cell separately (Fig. 5.4, bottom). In contrast to the classic rejection sampling algorithm, which can produce an arbitrary number of samples, a specific density has to be achieved. For cell_i, we first define the number of attempts as follows:

$$n_{\text{attempts}(\text{cell}_i)} = \lceil n_{\text{required}(\text{cell}_i)} \rceil \tag{5.10}$$

We then use the following criterion for acceptance:

$$y \leq \frac{n_{\text{required}}(\text{cell}_i)}{n_{\text{attempts}}(\text{cell}_i)} \tag{5.11}$$

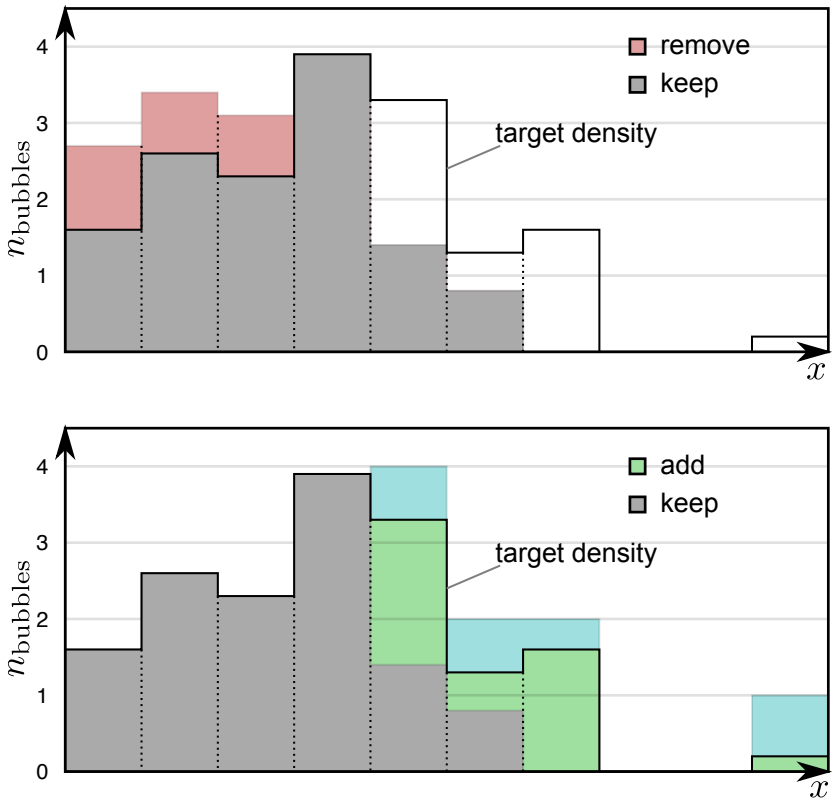

Figure 5.5: Bubble set update; one-dimensional examples. Top: An existing bubble set is updated using the measured density (enclosing the red and gray area) and the target density (enclosing the gray and white area). Superfluous bubbles are removed by calculating a new y for each bubble in cells where the measured density exceeds the target density. If the particle falls into the red area, it is removed from the set. Bottom: Finally, missing bubbles are added in areas where the target density is greater than the measured density. The approach is identical to the initial sampling, with the difference that the green acceptance area is defined by the difference between measured and target density.

5.4.3 Updating the Bubble Positions

As time progresses in the simulation, bubbles move through the domain in accordance with buoyancy and the velocity of the surrounding liquid medium. The simulation models this behavior via a separate, time-varying velocity field $\mathbf{u}_{\text{disp}}(\mathbf{x},t)$ for the dispersed phase. In part due to this motion, a set of bubble positions that reflects the density field at the time t_{i-1} generally does not reflect the density at the next time step t_i. The motion can be accounted for by moving the bubbles in accordance with the dispersed velocity field. For each bubble positioned at $\mathbf{x}(t_{i-1})$, the differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{u}_{\text{disp}}(\mathbf{x}(t),t) \quad t \in [t_{i-1} \dots t_i] \tag{5.12}$$

is solved using a numerical integration method such as the Dormand-Prince (4)5 scheme [DP80]. The resulting set of new bubble positions should then match the new density field more closely.

However, aside from motion, there are more causes for change in the density field. Coalescence and separation cause bubbles in a given location to merge or split up. Even more noticable are changes brought on by in- and outflow — new bubbles introduced into the domain at inlets as well as bubbles leaving the domain through outlets — and of bubbles bursting at the transitional area between the liquid medium and the gas-filled area above the surface of the liquid.

To incorporate these changes, after the bubbles have been moved along their trajectories, the bubble set is updated to reflect the new density field. This is achieved in a two-step approach that is based on the rejection sampling algorithm used for the initial sampling.

Using an estimate of the current density field of the bubble set, it is straight-forward to identify cells with too low or too high density by looking at the difference to the target density field. Similar to equation 5.9, the measured density field can be converted to a number of bubbles for each cell i:

$$n_{\text{measured}}(\text{cell}_i) = \text{measured\_density}(\text{cell}_i) \cdot \text{volume}(\text{cell}_i) \tag{5.13}$$

In the first step of the update, the rejection sampling algorithm is used to filter the existing bubble set. Instead of drawing new samples $(\mathbf{x},y)$, the existing bubble positions are used, while each bubble is assigned a random value $y \in [0,1]$. The bubble is then kept in the set only if the acceptance criterion

$$y \leq \frac{n_{\text{required}}(\text{cell}_i)}{n_{\text{measured}}(\text{cell}_i)} \tag{5.14}$$

is fulfilled; otherwise, it is removed. This approach is illustrated by Figure 5.5 (top).

In the second step (Fig. 5.5, bottom), new bubbles are added in cells where the measured density is lower than the target density. Again, the rejection sampling algorithm is applied. For each affected cell, a number of samples according to Equation 5.10 is drawn. The acceptance criterion for adding these new bubbles to the set is slightly modified to account for the bubbles that are already present:

$$\frac{n_{\text{measured}}(\text{cell}_i)}{n_{\text{attempts}}(\text{cell}_i)} < y \leq \frac{n_{\text{required}}(\text{cell}_i)}{n_{\text{attempts}}(\text{cell}_i)} \tag{5.15}$$

The resulting, updated set should reflect the new density field.

5.4.4 Measuring the Bubble Density

The quality of the updated bubble set clearly depends on the method used to measure the density field of the bubble set. A naïve approach would be to simply count, for each cell, the number of bubbles that are located within the cell. The problem with such an approach is analogous to the problem discussed in the beginning of this section: For a set of bubbles that was drawn from a density field which calls for only few bubbles in some areas, the number of bubbles contained within cells in these areas will be either much higher or much lower than the required number n_{required}, even if the overall density in the neighborhood is correct.

This problem is not unique to bubble density fields. In statistics, methods for kernel density estimation are used to estimate the probability density function of a distribution based on a given set of samples [Ros56]. For a set of samples $[x_1 \ldots x_n]$, the probability density function $f(x)$ is estimated as follows:

$$f_n(x) = \frac{1}{n}\sum_{i=1}^{n} w(x - x_i), \tag{5.16}$$

where w is a kernel with the property $\int_{-\infty}^{\infty} w(u)du = 1$. We can adapt this approach to measure the local density of a set of positions $[x_1 \ldots x_n]$:

$$\text{density\_measured}(\mathbf{x}) = \sum_{i=1}^{n} w(\mathbf{x} - \mathbf{x}_i) \tag{5.17}$$

In this case, the three-dimensional kernel w has to fulfill the requirement

$$\iiint_{\mathbb{R}^3} w(\mathbf{v})d\mathbf{v} = 1. \tag{5.18}$$

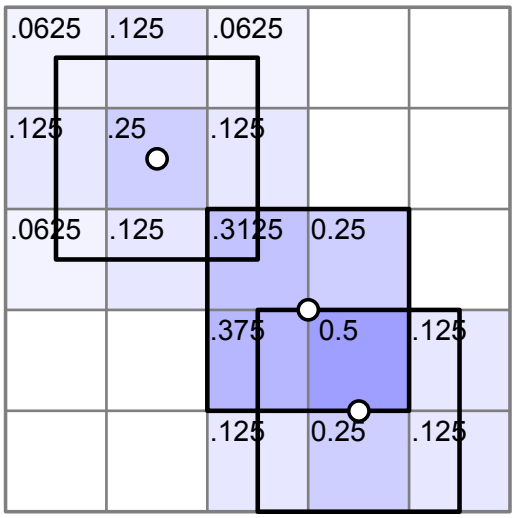

Figure 5.6: Measuring the density of a given point set, 2D example. Three points (depicted as white dots) are arranged in a mesh with a cell width of 1. A box kernel of width $h = 2$ is used to estimate the density (illustrated by black squares). Integrating the sum of kernels over each cell is straight forward: For each point and cell, the density value is proportional to the ratio of overlapping area and kernel volume. When, in this example, an entire cell is covered by a kernel, the contribution of the corresponding point to the measured density of the cell is 0.25.

The density for an entire cell i can then be obtained as follows:

$$\text{density\_measured}(\text{cell}_i) = \iiint_{\text{cell}_i} \text{density\_measured}(\mathbf{x}) d\mathbf{x} \tag{5.19}$$

We choose a box kernel of width h

$$w(\mathbf{v}) = \begin{cases} h^{-3} & \text{if } \mathbf{v} \in [-h/2, h/2]^3 \\ 0 & \text{otherwise,} \end{cases} \tag{5.20}$$

because it allows us to calculate the cell-based density value (Eq. 5.19) in a straight-forward manner. This is illustrated for a two-dimensional example in Figure 5.6.

Finally, the width h of the kernel has to be chosen. h is the only parameter of our method for which the optimal choice is not obvious. For the purpose of kernel density estimation, h should be chosen to minimize the Integrated Mean Square Error (IMSE) [Ros56]

$$\text{IMSE}(f_n, f) = \int_{-\infty}^{\infty} E|f_n(x) - f(x)|^2 dx, \tag{5.21}$$

which is difficult in practice since $f(x)$ is not known. In general, a smaller number of available samples mandates a larger kernel width. A possible approach would be to choose

h such that the IMSE is minimal when comparing the required density field to the measured density field of a freshly sampled set of bubble locations:

$$\text{IMSE}(\text{density\_required}, \text{density\_measured}) = \sum_i \left(\frac{\text{density\_required}(\text{cell}_i) - \text{density\_measured}(\text{cell}_i)}{\sum_j \text{density\_required}(\text{cell}_j) \cdot \text{volume}(\text{cell}_j)} \right)^2 \text{volume}(\text{cell}_i) \quad (5.22)$$

On the other hand, in the context of visualization, it may be preferable to minimize the number of changes in the bubble set, i.e., to avoid removing existing bubbles and inserting new bubbles as much as possible.

Rendering With the bubble positions known, bubble shapes can be generated. Since such a bubble set is in its essence a triangle mesh, multiple conventional methods are available to display it. We explore two possible approaches: The bubble set can be used together with an existing scalar field visualization (Fig. 5.9) using established visualization software. Alternatively, the triangle mesh can be used with a photorealistic renderer to produce images in a style similar to photographs of a physical experiment.

5.5 Implementation

Our implementation is designed for a straight-forward fit into the simulation and visualizaton workflow. We have implemented a platform-independent software, written in C++, that can run either on a workstation or in situ as the simulation progresses. The time steps produced by the simulation can be read and processed successively. Alternatively, individual time steps can be examined. An internal representation of the bubble positions is kept; each point is decorated with the local value of all variables that are present in the simulation output.

With the set of bubble positions known, triangulated bubble geometry is produced.

Traditional Visualization Our visualization approach can be combined with traditional visualization methods in multiple ways. To ensure compatiblity with commonly used visualization packages such as ParaView [Aya15] or VisIt [Chi+12], the triangulated bubbles are stored in a VTK PolyData file format [Avi+10]. Bubble sets can then be loaded and visualized with the original simulation data, enabling the domain scientist to use them together with familiar, traditional visualization methods, for example by color coding information directly on the bubbles. More advanced rendering techniques, which could provide, e.g.,

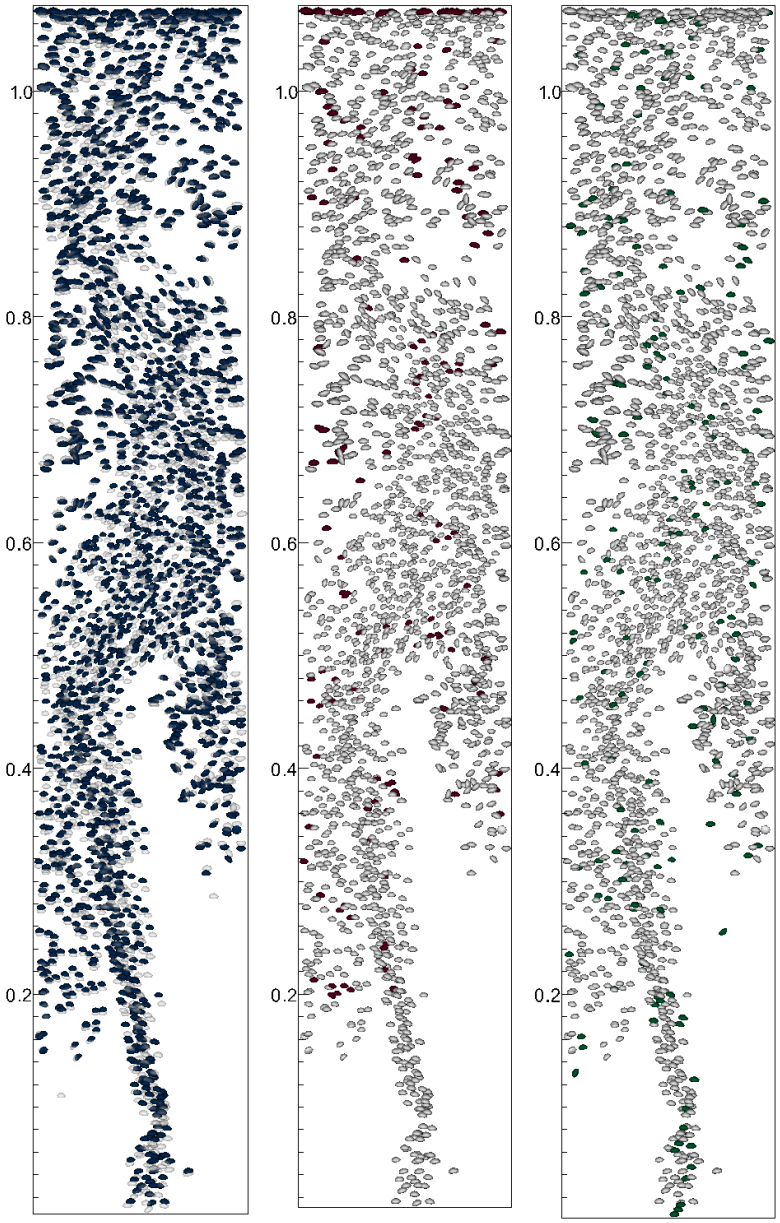

Figure 5.7: Advancing the bubbles through time. Left: Bubbles are moved according to the velocity field. Gray: old position, blue: new position. Center: Superfluous bubbles (red) are deleted. Right: New bubbles (green) are added.

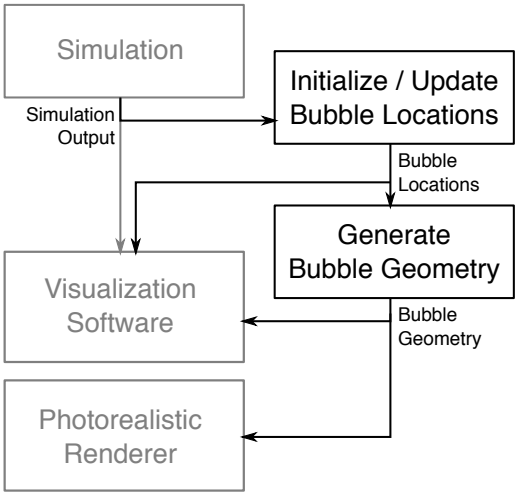

Figure 5.8: Bubbles sets generated by our sampling approach can be used as additional input for standard visualization software. Triangulated bubble geometry can be directly used with photorealistic renderers.

reflection, refraction, or global illumination, are not necessary for this traditional visualization because these effects are not directly relevant to the analysis and could even obscure information or cause distraction.

Photorealistic Rendering Alternatively, photorealistic renderings of the triangulated bubbles can be produced. We used the open source 3D modelling and rendering software Blender [Ble16] with its integrated path tracing renderer Cycles to produce images in a style similar to photographs of physical experiment. The main difference between these images and the non-photorealistic images that are produced by visualization tools is the support for optical effects such as refraction and reflection (Fig. 5.10). Optical properties of the dispersed and continuous media were chosen to match the properties of the substances that were used in both physical experiment and simulation.

5.6 Case Study: Bubble Column

To evaluate the capability of our visualization approach, we present a case study for a multi-phase fluid simulation dataset of a bubble column. Bubble columns are used in process engineering to transfer dissolved substances between different phases. The domain scientists have developed an OPOSPM [Dru+10] multi-phase fluid simulation solver, integrated into the open source CFD software OpenFOAM [Wel+98], that will help predict the performance of bubble columns with different substances without having to conduct physical experiments

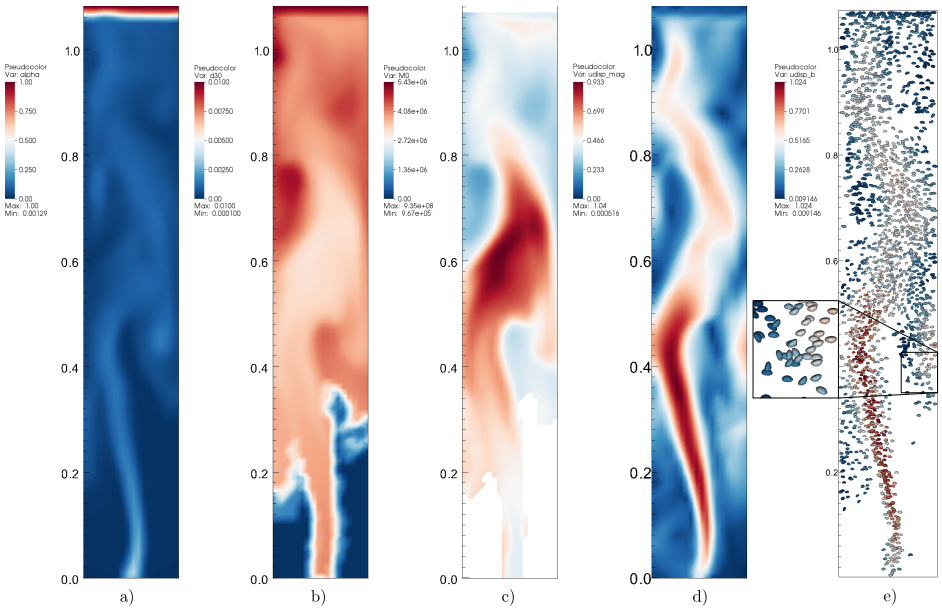

Figure 5.9: A single time step of the bubble column dataset. From left to right: a) Two-dimensional slice of the volume fraction α_{disp}. b) Two-dimensional slice of the volume-based bubble diameter d_{30}. c) Density field M_0. d) Velocity magnitude $\mathbf{u}_{\text{disp}}$ of the dispersed phase. e) Bubble set generated with our visualization approach. Color represents velocity magnitude of the dispersed phase.

for each configuration. To validate their model, they have run a simulation and also conducted a physical experiment with water as the continuous phase and with air as the dispersed phase. Bubbles of air are continuously released at the bottom of a water-filled, rectangular vessel (Fig. 5.10). The air bubbles form the dispersed phase, while the water forms the continuous phase. The results of the physical experiment have been recorded in form of photographs (Fig. 5.10, center).

Complex flow structures, which can be observed in bubble columns, require adequate visualization techniques. Domain experts in the CFD area are still using just fairly basic visualization methods, including color plots, single point analyses or commonly used criteria such as λ_2 for vortices. Especially in dispersed phase flows, the visualization of the dispersed phase as continuous is far away from the experimental observation of the domain expert.

The visualization should aid the domain expert by allowing fast comparison between experimental data and simulation. Simultaneously, it should allow for the encoding of additional feature based visualization techniques, such as the identification of interfacial contact or swirl structures.

Simulation Data The simulation uses a three-dimensional unstructured grid of 8550 hexahedral cells. Output was written every 1/60 second of simulation time, resulting in a

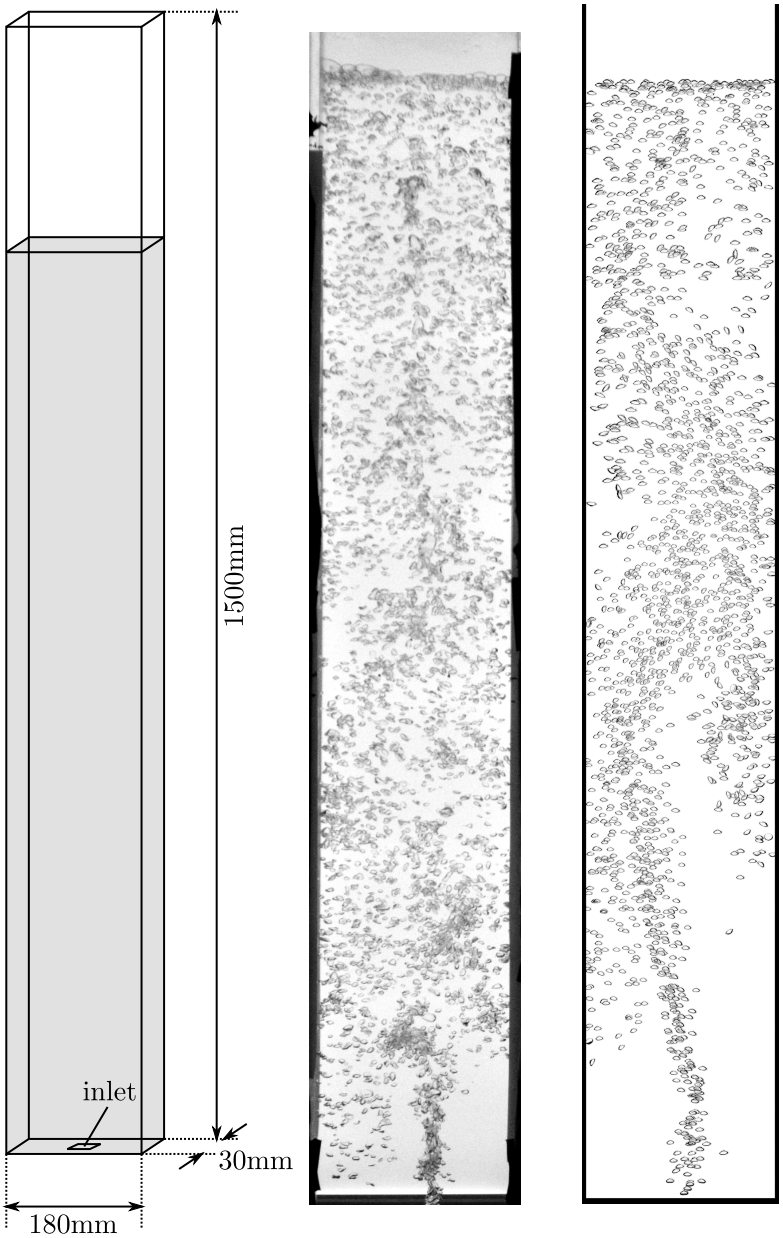

Figure 5.10: Left: The experiment featured in the case study consists of a rectangular, hexahedral vessel. In the physical experiment, the vessel is constructed from a transparent material. Center: To facilitate photographic records of the physical experiment, the white backdrop is illuminated. Right: Using path tracing software (the *Cycles* engine included with the 3D modelling package *Blender* [Ble16]), simulation data processed by our sampling approach can be rendered in a photorealistic manner.

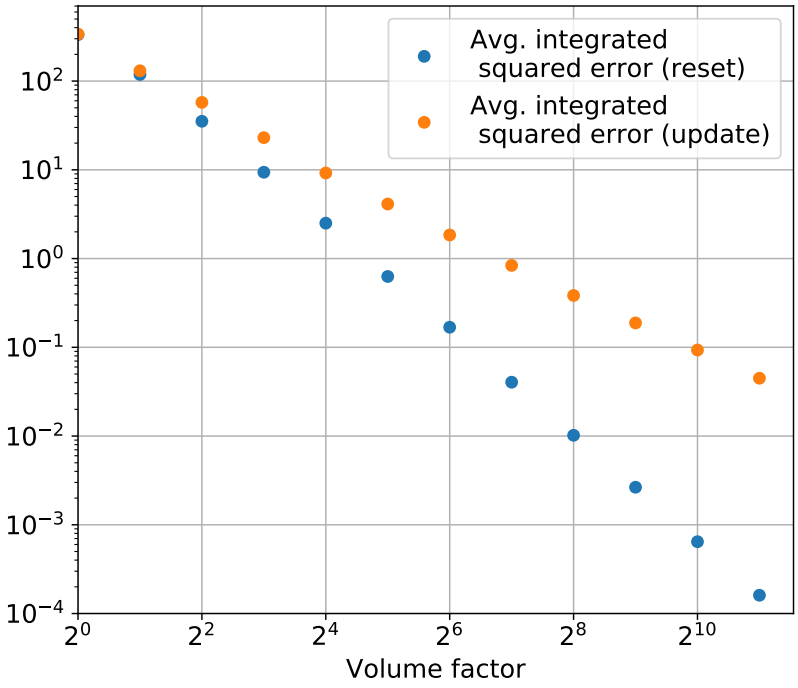

Figure 5.11: Average integrated squared error with cell volume, and therefore the number of bubbles, artificially inflated by a range of factors.

dataset consisting of 1201 time steps spanning 20 seconds. The overall size of the dataset is 5.4GB.

Traditional Visualization A common approach used by domain scientists to visualize the simulation output is to define a two-dimensional slice of the domain and then color-code a variable of interest using common visualization software. The left four columns of Fig. 5.9 show plots generated this way using VisIt [Chi+12], using a diverging blue-red colormap. Depicted are the volume fraction α_{disp} of the dispersed phase a), the volume-based bubble diameter d_{30} b), the density function M_0 filtered according to equation 5.8 c), and the velocity magnitude $\mathbf{u}_{\text{disp}}$ of the dispersed phase d).

Clearly visible at the top of the α-plot a) is the transitional area near the water surface, indicated by a jump from medium-low (mostly water) to very high (air). The plot of $M_{0,\text{filtered}}$ c) shows this as well, since the α-threshold cuts off everything above the surface.

Also visible is a column with an increased amount of air rising from the inlet at the bottom towards the center of the dataset with a slight slant to the left. The same column is clearly indicated in the velocity plot d). ,

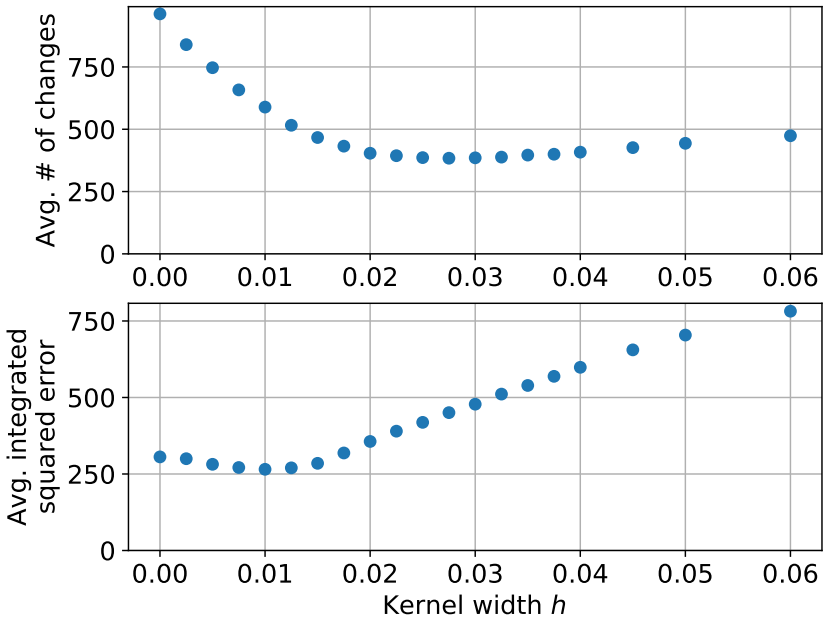

Figure 5.12: Average integrated squared error and average number of changes (bubble removals and additions) for different choices of the kernel width h.

Initial Sampling. We begin by evaluating the initial sampling step. First, an initial set of bubbles is computed for several time steps. This is achieved by removing all bubbles after each time step, providing a clean slate for the next. Then, the average of the integrated mean square error (Eq. 5.21) is calculated. Since the density field requires only up to about three bubbles per cell, some noise can be expected in the result. To deal with this issue, a number of samplings are calculated, with the cell volume artificially multiplied by a factor to increase the number of bubbles and therefore the quality of the density measurement. Figure 5.11 (blue) shows the results for factors between 1 and 2048, using a logarithmic scale for both the integrated mean square error (vertical axis) and the volume factor (horizontal). It can be seen that the integrated mean squared error converges towards zero as the number of bubbles is increased.

Updated Sampling. Next, we repeat the same measurements as for the initial sampling, but without clearing the bubble set between time steps. The bubble set is updated between each time step according to the method described in the previous section. Since ultimately, a very large number of bubbles will be used, the kernel width h is set to zero for the density measurement. Effectively, this means that the density value is determined simply by the number of bubbles within each cell. The results are also shown as orange dots in Figure 5.11. The error still approaches zero as the number of bubbles is increased. However, the slope of the curve is considerably less steep than in the case of initial sampling, indicating possible opportunities for improving the update algorithm. The practical impact of this discrepancy

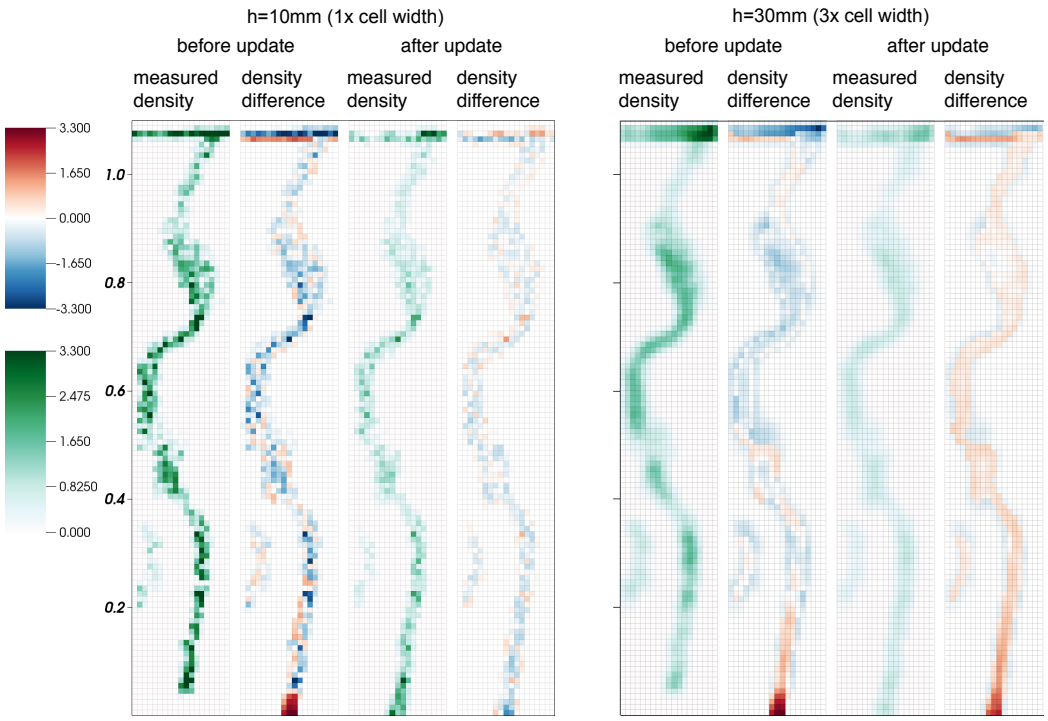

Figure 5.13: Measured densities (green) and density differences (blue-red) in a single time step of the bubble column dataset for two choices of the kernel width. The images show a two-dimensional slice through the center of the z-Axis. Red color in the difference fields indicates that the density is too low; blue indicates that the density is too high.

is not clear, because for initial sampling and updating with the original cell volumes, the average integrated squared error is nearly identical.

Kernel Width. So far, the trivial kernel width of zero has been used for the measurements. We now study how the kernel width affects the sampling quality when updating a bubble set. First, the update method described in the previous section is used to produce and update sets of bubbles, using a range of different choices for the kernel width h. The density of each bubble set is measured for every time step both before and after updating. Then, for each kernel width, the average integrated squared error of the measured density after updating compared to the required density is calculated. As can be seen in Figure 5.12 (bottom), the error measure starts out with a value of 306 at $h = 0$m and drops slightly until its minimum of 265 at $h = 0.01$m, which corresponds to a kernel size equal to the cell size. From there on, the error value increases rapidly for larger kernel sizes.

Judging by this result, the optimal kernel width for this dataset would be equal to the cell size. However, from a visualization standpoint, there is another quality measure that could be of interest. For the purpose of visualizing the bubble set over time, the bubble set should,

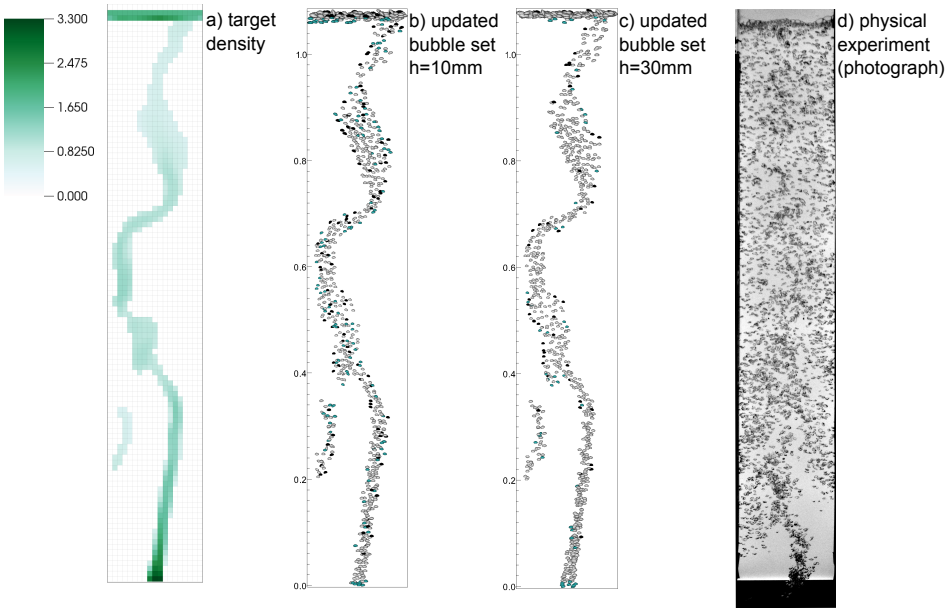

Figure 5.14: For the required density field (a), bubble glyphs [Hla+16b] illustrate bubble sets generated using two different width values (b,c). Black bubbles have been removed in the current step; the color cyan indicates that the bubble has been newly added. The glyph represention allows direct comparison to photographs of physical experiments (d).

of course, follow the density field as closely as possible, with the additional requirement that bubbles should not be removed or added more often than necessary. Without this additional requirement, it would make sense to simply compute an entirely new set of bubbles at each time step, an approach that, as we have already seen, produces a good fit with the required density field. At the same time, such an approach provides no continuity over time, making it impossible to track individual bubbles visually. Therefore, we consider an additional measure of quality: Minimizing the average number of bubbles that are added or removed in each time step improves the continuity of the bubble set over time. Figure 5.12 (top) shows the average sum of the number of added bubbles and the number or removed bubbles for each tested kernel width. The minimum of this measure does not coincide with the smallest integrated squared error; instead, the kernel widths between 0.02m and 0.03m (two and three times the cell size, respectively) provide the smallest number of changes. Since the two criteria for the kernel width are in disagreement, the tradeoff between correctness and visual coherence has to be considered.

Figure 5.13 illustrates the tradeoff between different choices of the kernel width h. The left half shows the results for $h = 0.01$m, which was shown above to provide the lowest integrated mean square error, while the right half shows results for $h = 0.03$m. It is clearly visible that for the smaller kernel width, the measured densities are subject to some noise

when compared to the target density (Fig. 5.14). This behavior is expected because of the relatively small spatial density of bubbles in relation to the kernel size. In contrast, the larger kernel width produces smoother measurements which appear blurry in comparison to the target density. For both choices, the approach fulfils its purpose of updating the bubble set. This is most apparent when looking at the area above the inlet: as the existing bubbles have moved up since the previous time step, a gap is clearly visible where more bubbles should stream in from the inlet. The gap can also be identified by the red color in the pre-update difference plots. After the update, the gap has been filled in. Overall, the smaller kernel width appears to produce a smaller density difference after the update. However, both choices produce very similar results for the final bubble visualization (Fig. 5.14). In the figure, the bubbles are colored to indicate removals (black) and new additions (cyan). Compared to the larger kernel width on the right, the smaller kernel width results in a larger number of removals, which in turn has to be compensated by a larger number of new additions. We applied our sampling-based approach to the bubble column dataset by computing a set of bubbles for each available time step using a consumer-grade desktop computer. Excluding input and output, it took 56 seconds to advance and update the bubble set through all 1201 time steps and to generate bubble geometry for all bubbles.

Fig. 5.9e shows the bubble set for the same time step as Figs. 5.9a-d. Because the bubbles are scaled according to the diameter field d_{30}, the bubble size can be assessed visually in relation to the size of the domain. The arrangement of bubbles provides an intuitive impression of the density field. The color map represents the velocity magnitude of the dispersed phase $\mathbf{u}_{\text{disp}}$. Even though the plot to the left depicts the same variable, the range of values is not the same. The reason lies in the fact that some areas, especially the lower corners, contain no bubbles. Also, the 2D plot only shows values on a slice through the center of the domain, a restriction that does not apply to bubbles.

Taking a closer look at the bubbles reveals the relative direction of motion of the bubbles in relation to the fluid, providing some additional insight of the local flow properties: In some areas, all bubbles within the neighborhood point into the same direction, which weakly indicates laminar flow. In other areas, bubbles point in many different directions despite their close proximity, hinting at the presence of vortices.

It is apparent that even in those more turbulent areas, all bubbles have more or less the same shape. This is owed to the fact that the dataset covers only a small portion of the Eötvös-Reynolds chart, resulting in exclusively spherical to ellipsoidal bubble shapes.

Fig. 5.10 shows a photograph of the physical experiment next to a photorealistic rendering of the bubble set. The rendering with reflection and refraction makes it easy to compare the shapes and sizes of bubbles in both images. Similarities and some differences are immediately apparent. The photograph contains many bubbles with a shape and size similar

to the bubbles in the rendering. However, the bubbles in the photograph exhibit a much larger range of sizes. Near the center of the vessel, clusters consisting of very large bubbles surrounded by very small bubbles can be observed traveling upward. It would not be possible to depict this behavior in the visualization, because the presence of these phenomena is not supported by the simulation data.

Aside from the size differences, bubbles, especially large ones, are often not shaped as regularly as they are in the rendering. This is at least in part due to the wakes of rising bubbles in the vicinity, collisions with other bubbles, and shear flow. These effects are not covered by the bubble shape model used for our visualization and will be explored in future work. However, since small-scale flow features are not present in the simulation data, care must be taken not to introduce additional, possibly errorneous information that is not supported by the data.

5.7 Discussion

We have presented a visualization approach for moment-based probabilistic multi-phase flow simulation data. The approach uses a parametric model for the graphical representation of bubble shapes, based on local moment information and flow parameters. A specific set of bubbles is reconstructed from moment information to approximate a bubble distribution over time. We have described how the approach can be implemented and we have provided a case study that illustrates the utility of our approach. Specifically, we have shown that we support the tasks and fulfill the requirements that were formulated by the domain scientists:

Size and Density Bubble size is conveyed directly through the parametric bubble model. By displaying specific sets of bubbles, the density can be visually assessed.

Comparability Using photorealistic rendering of bubble sets that are shaped using our parametric model allows to directly compare the simulation results to the physical experiment.

Temporal Coherence Updating the bubble sets by moving bubbles and adapting the set to the new density field ensures temporal coherence of the visualization when cycling through time steps.

Non-Intrusiveness The visualization approach works well with established visualization tools and can be used to enhance and not replace the domain scientists' visualization workflow.

We have further studied the influence of our method's only parameter, the kernel width h. Our experiments have established that this parameter serves to control a tradeoff between smoother animation quality and closer approximation of the density field. Optimizing for either of those criteria resulted in satisfactory results.

In the future, we will extend the parametric bubble model to support deformations based on shear flow and bubble collisions. We also plan to explicitly model coalescence and separation events and to support their visualization using deformations of the bubble geometry. To avoid unnecessary deletion and addition of bubbles, we plan to improve the density estimation process using Kernel Density Estimation. We will also study how force-based relaxation methods [SJ13] could be employed to adjust the bubble distribution while representing the velocity field of the dispersed phase as closely as possible. More generally, we will examine how visualization can support the investigation of reactive bubble columns and the influence of hydrodynamics to the reaction efficiency.

6 Conclusion and Future Research

This dissertation has introduced a number of methods, techniques, and applications for the visualization of flow simulation data using trajectories. Our methods are able to aid application scientists in multiple areas, including the control of simulation output parameters, the analysis of flow simulation ensemble data, and the visual validation of probabilistic, multi-phase fluid simulations.

For a Lagrangian flow field representation, the interval flow map, a definition for upper bounds of the errors that arise when constructing trajectories from the representation were proposed. Based on this theoretical definition, it was shown how the error of individual trajectories can be estimated as they are being constructed. The approach was validated by comparing the estimated errors to actual measurements for example flow data sets. Our approach will enable application scientists to evaluate the quality of trajectories which are later used for visualization. At the same time, it is now possible to define error thresholds, which can then be used to automatically steer both the temporal frequency and spatial resolution of flow data stored in the Lagrangian representation.

A new method was introduced for classifying spatial areas of agreement and disagreement regarding the transport behavior in flow simulation ensembles. The method makes use of a classification space spanned by the joint variance and the individual variance of trajectory endpoints. While joint variance handles trajectories seeded in all ensemble members, individual variance is based on small perturbations within each simulation run, thus taking into account divergent flow behavior that makes the detection of similar and dissimilar transport behavior difficult. Together with interactive visualization methods such as brushing and linking, spatial regions with a specific joint transport behavior can be identified. In addition, a rudimentary method is proposed to detect the presence of trends and outliers within the ensemble. The application of the methods was demonstrated using several example data sets. Using our approach, application scientists can quickly and reliably identify regions of agreement and disagreement, as well as the presence of trends or outliers, regarding the material transport behavior in flow field ensembles.

For flow data stemming from probabilistic, multi-phase flow simulations of bubbly flows, a new visualization method was proposed that generates bubble sets based on the numerical

description of the bubble size distribution. The shape of each individual bubble is adjusted to the local flow behavior using a new, parametric bubble model. Smooth animation is enabled by advancing bubbles along their trajectories, and updating the bubble set to account for changed distribution properties. Using examples, it was shown that the method produces satisfactory results and can be used for direct comparisons of simulation data and photographic records of real-world physical experiments. In out method, application scientists have a powerful, intuitive flow visualization method that integrates well with their traditional visualization work flow.

All methods and techniques introduced in this dissertation, though demonstrated using specific examples, are applicable to a wide range of applications in science and engineering. For example, ensemble flow simulation can be used to simulate the behavior of cutting fluids or liquid coolants. Both our error estimates for Lagrangian flow field representation and our comparative ensemble visualization can be directly applied to such settings. Simulations of manufacturing processes may also involve the formation of gas bubbles in coolants, or the motion of liquid droplets and solid shavings or burrs. Such scenarios may also be use cases for method-of-moment based simulation, where our approach for the visualization of bubbles could be applied with only minor modifications. More generally, the results of this thesis inspired a number of new research questions which will be interesting to pursue in the future.

Lagrangian Flow Field Representations. The error measures introduced in Chapter 3 were defined only for a single spatial dimension, and were then used to create error rectangles (or cuboids) by treating each spatial dimension of the data sets separately. Intuitively, it would make sense that areas of large divergence should behave anisotropically. In that case, a larger error in one direction would not necessarily increase the error in the orthogonal directions. Formulating the error measures in terms of multiple dimensions, i.e., using the Eigenvalues of Jacobian and Hessian wherever first and second spatial derivatives are required, could have an impact on the accuracy of the method and should be studied.

Further, during the evaluation of the method, it became clear that the resolution of the underlying discretization has huge influence over the accuracy of such a representation, but that in many cases, only a few localized areas of the spatial domain have a large contribution to the error. In computational fluid dynamics, it is common practice to adapt the resolution of discretizations locally, using, for example, unstructured grids or so-called Adaptive Mesh Refinement (AMR) grids. A promising direction for future research would be to study how the error estimate can be defined on such a multi-resolution representation, and then utilize it to guide the mesh generation process. Such an approach has the potential to greatly increase

the accuracy of the Lagrangian representation, while requiring only a limited amount of additional storage space.

Visualization of Flow Field Ensembles. The topic opened several avenues for future research. The ensemble visualization based on the classification space provides insight into the range of possible outcomes for the different sets of parameters and initial conditions of the ensemble. However, it is completely independent of what those different settings actually are, and, therefore, does not provide any information regarding the influence of those parameters. It could be of interest to establish a relationship between the classification space and the space of simulation parameters, and to detect if particular choices of parameters are associated with specific trends in the ensemble.

To detect the presence of trends, a minimum spanning tree method is used together with a user-specified edge length threshold. This method makes it possible, depending on the choice of the threshold, to detect the local presence and number of trends. In future work, further improvements and independence of the threshold parameter could be achieved using more advanced techniques, such as state-of-the art clustering methods, to divide the ensemble into subsets that exhibit similar behavior. These subsets of simulation runs could then locally be treated separately by the classification method.

Visualization of Probabilistic, Multi-Phase Flow Simulation Data. The presented geometric bubble model is rather simplistic, because it only incorporates the two flow properties Eötvös number and Reynolds number to determine the bubble shape. In reality, bubble shape depends on a number of other properties. Especially for larger bubbles, properties such as shear or vorticity can have strong influence on the shape and even cause time-dependent behavior such as periodic wobbling. Therefore, it would be interesting to study this influence and adjust the geometric model accordingly.

The positions and adjustment of the bubble set itself opens further questions for future research. With the proposed method, bubbles can appear or vanish at constrained but random positions to satisfy changed density requirements. In reality, the appearance and disappearance of bubbles is caused by separation and coalescence events, i.e., single bubbles can split to form multiple new bubbles, or multiple bubbles can merge, resulting in a single, new bubble. It would be interesting to research whether or not local flow properties can help determine where these coalescence and separation events are likely to occur, and then handle such events explicitly without the need to generate new bubbles or remove existing ones at random locations.

References

[Agr+14] A. Agranovsky, D. Camp, C. Garth, et al. “Improved Post Hoc Flow Analysis Via Lagrangian Representations”. In: *Large Data Analysis and Visualization (LDAV), 2014 IEEE 4th Symposium on*. Nov. 2014, pp. 67–75 (cit. on pp. iii, v, 3, 18, 30, 31).

[Arn89] V. I. Arnold. *Mathematical Methods of Classical Mechanics*. 2nd. New York, USA: Springer-Verlag, 1989 (cit. on p. 25).

[Aya15] U. Ayachit. “The ParaView guide: a parallel visualization application”. In: (2015) (cit. on pp. 82, 93).

[BGT12] 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 (Aug. 2012), pp. 1368–1380 (cit. on p. 26).

[Bat00] G. K. Batchelor. *An introduction to fluid dynamics*. 2nd. Cambridge, United Kingdom: Cambridge University Press, 2000 (cit. on p. 7).

[BW81] D. Bhaga and M. E. Weber. “Bubbles in viscous liquids: shapes, wakes and velocities”. In: *Journal of Fluid Mechanics* 105 (Apr. 1981), pp. 61–85 (cit. on pp. 83, 84).

[Ble16] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Blender Institute, Amsterdam, 2016 (cit. on pp. 95, 97).

[Bon+14] G.-P. Bonneau, H.-C. Hege, C. R. Johnson, et al. “Overview and State-of-the-Art of Uncertainty Visualization”. In: *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization*. Ed. by C. D. Hansen, M. Chen, C. R. Johnson, A. E. Kaufman, and H. Hagen. London: Springer London, 2014, pp. 3–27 (cit. on p. 27).

[Bor+10] S. Born, A. Wiebel, J. Friedrich, G. Scheuermann, and D. Bartz. “Illustrative Stream Surfaces”. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (Nov. 2010), pp. 1329–1338 (cit. on p. 23).

[BWE06] R. P. Botchen, D. Weiskopf, and T. Ertl. “Interactive visualisation of uncertainty in flow fields using texture-based techniques”. In: *12th International Symposium on Flow Visualisation*. 2006 (cit. on p. 54).

[BWE05] R. P. Botchen, D. Weiskopf, and T. Ertl. “Texture-based visualization of uncertainty in flow fields”. In: *VIS 05. IEEE Visualization, 2005.* IEEE. 2005, pp. 647–654 (cit. on p. 54).

[Bra+12] A. Brambilla, R. Carnecky, R. Peikert, I. Viola, and H. Hauser. “Illustrative Flow Visualization: State of the Art, Trends and Challenges”. In: *EG 2012* State of the Art Reports (2012). Ed. by M.-P. Cani and F. Ganovelli, pp. 75–94 (cit. on p. 21).

[BM10] S. Bruckner and T. Möller. “Result-Driven Exploration of Simulation Parameter Spaces for Visual Effects Design”. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1467–1475 (cit. on p. 56).

[Bru+01] R. Bruckschen, F. Kuester, B. Hamann, and K. I. Joy. “Real-Time Out-of-Core Visualization of Particle Traces”. In: *Proceedings of the IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics*. IEEE Press. 2001, pp. 45–50 (cit. on p. 31).

[BR10] 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 (cit. on pp. 26, 31).

[BHJ16] R. Bujack, M. Hlawitschka, and K. I. Joy. “Topology-inspired Galilean invariant vector field analysis”. In: *2016 IEEE Pacific Visualization Symposium (PacificVis)*. Apr. 2016, pp. 72–79 (cit. on pp. 25, 26).

[BJ15] R. Bujack and K. I. Joy. “Lagrangian Representations of Flow Fields with Parameter Curves”. In: *Large Data Analysis and Visualization (LDAV), 2015 IEEE 4th Symposium on*. IEEE. 2015 (cit. on pp. 30, 31, 34).

[CL93] B. Cabral and L. C. Leedom. “Imaging vector fields using line integral convolution”. In: *Proceedings of the 20th Annual Conference on Computer graphics and Interactive Techniques*. SIGGRAPH ’93. Anaheim, CA, 1993, pp. 263–270 (cit. on p. 67).

[COJ15] J. Chandler, H. Obermaier, and K. Joy. “Interpolation-Based Pathline Tracing in Particle-Based Flow Visualization”. In: *Visualization and Computer Graphics, IEEE Transactions on* 21.1 (Jan. 2015), pp. 68–80 (cit. on pp. 9, 31).

[Chi+12] H. Childs, E. Brugger, B. Whitlock, et al. “VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data”. In: *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*. Oct. 2012, pp. 357–372 (cit. on pp. 82, 93, 98).

[Chi+10] H. Childs, D. Pugmire, S. Ahern, et al. “Extreme Scaling of Production Visualization Software on Diverse Architectures”. In: *IEEE Computer Graphics and Applications (CG&A)* 30.3 (May 2010), pp. 22–31 (cit. on pp. 18, 29).

[CM93] A. J. Chorin and J. E. Marsden. *A mathematical introduction to fluid mechanics*. 3rd. New York: Springer, 1993 (cit. on p. 7).

[CGW78] R. Clift, J. R. Grace, and M. E. Weber. *Bubbles, drops, and particles*. Academic Press, 1978 (cit. on pp. 83, 84).

[CRK12] 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 (cit. on p. 26).

[Dal83] U. Dallmann. “Topological structures of three-dimensional vortex flow separation”. In: *16th Fluid and Plasmadynamics Conference*. American Institute of Aeronautics and Astronautics, 2018/01/26 1983 (cit. on p. 22).

[Dic89] R. R. Dickinson. “A Unified Approach To The Design Of Visualization Software For The Analysis Of Field Problems”. In: vol. 1083. 1989, pp. 173–180 (cit. on p. 21).

[DP80] J. R. Dormand and P. J. Prince. “A family of embedded Runge-Kutta formulae”. In: *Journal of computational and applied mathematics* 6.1 (1980), pp. 19–26 (cit. on pp. 21, 90).

[DCH88] R. A. Drebin, L. Carpenter, and P. Hanrahan. “Volume rendering”. In: *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. SIGGRAPH ’88. 1988, pp. 65–74 (cit. on p. 67).

[Dru+10] C. Drumm, M. Attarakih, M. W. Hlawitschka, and H.-J. Bart. “One-group reduced population balance model for CFD simulation of a pilot-plant extraction column”. In: *Industrial & Engineering Chemistry Research* 49.7 (2010), pp. 3442–3451 (cit. on pp. 81, 95).

[Eas+16] P. Easwaran, M. J. Lehmann, O. Wirjadi, et al. “Automatic fiber thickness measurement in SEM images validated using synthetic data”. In: *Chemical Engineering & Technology* 39.3 (2016), pp. 395–402 (cit. on pp. 27, 82).

[Far02] G. E. Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2002 (cit. on p. 85).

[Feh69] E. Fehlberg. *Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems*. Tech. rep. Huntsville, AL, USA: NASA Marshall Space Flight Center, 1969 (cit. on p. 21).

[FBW16] F. Ferstl, K. Bürger, and R. Westermann. “Streamline Variability Plots for Characterizing the Uncertainty in Vector Field Ensembles”. In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (Jan. 2016), pp. 767–776 (cit. on p. 27).

[Gar+07a] C. Garth, F. Gerhardt, X. Tricoche, and H. Hagen. “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 (cit. on p. 53).

[GJ10] C. Garth and K. I. Joy. “Fast, Memory-Efficient Cell Location in Unstructured Grids for Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (Nov. 2010), pp. 1541–1550 (cit. on p. 17).

[Gar+08] C. Garth, H. Krishnan, X. Tricoche, T. Tricoche, and K. I. Joy. “Generation of Accurate Integral Surfaces in Time-Dependent Vector Fields”. In: *IEEE Transactions on Visualization and Computer Graphics* 14.6 (Nov. 2008), pp. 1404–1411 (cit. on p. 22).

[Gar+07b] 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 (Nov. 2007), pp. 1464–1471 (cit. on p. 26).

[Gar+09] C. Garth, G.-S. Li, X. Tricoche, C. D. Hansen, and H. Hagen. “Visualization of Coherent Structures in Transient 2D Flows”. In: *Topology-Based Methods in Visualization II*. Ed. by H.-C. Hege, K. Polthier, and G. Scheuermann. Mathematics and Visualization. Springer Berlin Heidelberg, 2009, pp. 1–13 (cit. on p. 67).

[Ger98] N. Gershon. “Visualization of an Imperfect World”. In: *IEEE Computer Graphics and Applications* 18.4 (1998), pp. 43–45 (cit. on p. 27).

[GL10] S. R. Ghorpade and B. V. Limaye. *A Course in Multivariable Calculus and Analysis*. New York, USA: Springer, 2010 (cit. on p. 35).

[GM77] 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 (1977), pp. 375–389 (cit. on p. 8).

[GLM96] S. Gottschalk, M. Lin, and D. Manocha. “OBBTree: A hierarchical structure for rapid interference detection”. In: 1996, pp. 171–180 (cit. on p. 17).

[Göt+06] T. Götz, A. Klar, N. Marheineke, and R. Wegener. *A Stochastic Model for the Fiber Lay-down Process in the Nonwoven Production*. Tech. rep. 102. Fraunhofer (ITWM), 2006 (cit. on p. 27).

[GS06] H. Griethe and H. Schumann. “The Visualization of Uncertain Data: Methods and Problems”. In: *Proceedings of SimVis ’06*. SCS Publishing House, 2006, pp. 143–156 (cit. on p. 27).

[GH10] D. F. Griffiths and D. J. Higham. *Numerical Methods for Ordinary Differential Equations: Initial Value Problems*. London, UK: Springer, 2010 (cit. on p. 34).

[Gur81] M. E. Gurtin. *An Introduction to Continuum Mechanics*. New York, USA: Academic Press, 1981 (cit. on p. 25).

[Hal01] G. Haller. “Distinguished material surfaces and coherent structures in three-dimensional fluid flows”. In: *Physica D: Nonlinear Phenomena* 149.4 (2001), pp. 248–277 (cit. on pp. 10, 23, 42).

[Hal02] G. Haller. “Lagrangian Coherent Structures from Approximate Velocity Data”. In: *Physics of Fluids* 14 (2002), pp. 1851–1861 (cit. on p. 31).

[HY00] G. Haller and G. Yuan. “Lagrangian Coherent Structures and Mixing in Two-Dimensional Turbulence”. In: *Phys. D* 147.3-4 (Dec. 2000), pp. 352–370 (cit. on p. 31).

[Hal15] G. Haller. “Lagrangian Coherent Structures”. In: *Annual Review of Fluid Mechanics* 47.1 (2015), pp. 137–162 (cit. on p. 25).

[HH12] G. Hämmerlin and K.-H. Hoffmann. *Numerical Mathematics*. Springer Science & Business Media, 2012 (cit. on pp. 13, 36).

[Har02] P. Hartman. *Ordinary Differential Equations*. 2nd. Philadelphia, USA: Society for Industrial and Applied Mathematics, 2002 (cit. on p. 19).

[HN98] J. L. Hintze and R. D. Nelson. “Violin plots: a box plot-density trace synergism”. In: *The American Statistician* 52.2 (1998), pp. 181–184 (cit. on p. 27).

[Hir91] C. Hirsch. *Numerical Computation of Internal and External Flows, Volume 2: Computational Methods for Inviscid and Viscous Flows*. Wiley, 1991 (cit. on p. 8).

[HSW11] M. Hlawatsch, F. Sadlo, and D. Weiskopf. “Hierarchical Line Integration”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.8 (2011), pp. 1148–1163 (cit. on p. 31).

[Hla+11a] 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 (cit. on p. 54).

[Hla+16a] M. W. Hlawitschka, M. M. Attarakih, S. S. Alzyod, and H.-J. Bart. “CFD based extraction column design - Chances and challenges”. In: *Chinese Journal of Chemical Engineering* 24.2 (2016), pp. 259–263 (cit. on p. 81).

[Hla+11b] M. W. Hlawitschka, F. Chen, H.-J. Bart, and H. Hagen. “CFD Simulation und Verbesserte Datenauswertung einer Extraktionskolonne vom Typ Kühni”. In: *YRS 2011* (2011), p. 27 (cit. on p. 82).

[Hla+16b] M. W. Hlawitschka, J. Schäfer, M. Hummel, C. Garth, and H.-J. Bart. “Population Balance Modelling Using a Multiphase-CFD Code and Comparing Visualization”. In: *Chemie Ingenieur Technik* 88.10 (2016), pp. 1480–1491 (cit. on pp. 27, 80, 81, 101).

[Hla+18] M. W. Hlawitschka, J. Schäfer, L. Jöckel, et al. “CFD Simulation and Visualization of Reactive Bubble Columns”. In: *Journal of Chemical Engineering of Japan* 51.4 (2018), pp. 356–365 (cit. on p. 80).

[HK64] H. Hulburt and S. Katz. “Some Problems in Particle Technology - a Statistical Mechanical Formulation”. In: *Chemical Engineering Science* 19.8 (1964), pp. 555–574 (cit. on pp. 79, 81).

[Hul90] J. Hultquist. “Interactive Numerical Flow Visualization”. In: *Computing Systems in Engineering* 1.2-4 (1990), pp. 349–353 (cit. on p. 22).

[Hum+10] M. Hummel, C. Garth, B. Hamann, H. Hagen, and K. I. Joy. “IRIS: Illustrative Rendering for Integral Surfaces”. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (Nov. 2010), pp. 1319–1328 (cit. on p. 23).

[Hum+13] 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 (Dec. 2013), pp. 2743–2752 (cit. on p. 52).

[Hum+16] M. Hummel, R. Bujack, K. I. Joy, and C. Garth. “Error Estimates for Lagrangian Flow Field Representations”. In: *EuroVis 2016 - Short Papers*. Ed. by E. Bertini, N. Elmqvist, and T. Wischgoll. The Eurographics Association, 2016 (cit. on p. 30).

[Hum+17a] M. Hummel, L. Jöckel, J. Schäfer, M. W. Hlawitschka, and C. Garth. “Evaluating Sampling Strategies for Visualizing Uncertain Multi-Phase Fluid Simulation Data”. In: *Physical Modeling for Virtual Manufacturing Systems and Processes*. Vol. 869. Applied Mechanics and Materials. Trans Tech Publications, Sept. 2017, pp. 139–148 (cit. on p. 80).

[Hum+17b] M. Hummel, L. Jöckel, J. Schäfer, M. W. Hlawitschka, and C. Garth. “Visualizing Probabilistic Multi-Phase Fluid Simulation Data using a Sampling Approach”. In: *Computer Graphics Forum* 36.3 (2017), pp. 469–477 (cit. on p. 80).

[Jar+15] M. Jarema, I. Demir, J. Kehrer, and R. Westermann. “Comparative visual analysis of vector field ensembles”. In: *Visual Analytics Science and Technology (VAST), 2015 IEEE Conference on*. Oct. 2015, pp. 81–88 (cit. on p. 53).

[JEH02] B. Jobard, G. Erlebacker, and H. Hussaini. “Lagrangian-Eulerian Advection of Noise and Dye Textures for Unsteady Flow Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 8.3 (2002), pp. 211–222 (cit. on p. 31).

[JS03] C. Johnson and A. Sanderson. “A Next Step: Visualizing Errors and Uncertainty”. In: *IEEE Computer Graphics and Applications* 23.5 (2003), pp. 6–10 (cit. on p. 27).

[Kas+09] J. Kasten, C. Petz, I. Hotz, B. R. Noack, and H.-C. Hege. “Localized Finite-time Lyapunov Exponent for Unsteady Flow Analysis”. In: *Proceedings of the International Workshop on Vision, Modeling and Visualization (VMV’09)*. 2009, pp. 265–276 (cit. on p. 53).

[KD98] G. Kindlmann and J. W. Durkin. “Semi-automatic generation of transfer functions for direct volume rendering”. In: *Proceedings of the 1998 IEEE symposium on Volume visualization*. VVS ’98. Research Triangle Park, North Carolina, USA: ACM, 1998, pp. 79–86 (cit. on p. 66).

[KKH02] 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 (cit. on p. 66).

[KGJ09] H. Krishnan, C. Garth, and K. Joy. “Time and Streak Surfaces for Flow Visualization in Large Time-Varying Data Sets”. In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (Nov. 2009), pp. 1267–1274 (cit. on p. 22).

[Lar+04] R. S. Laramee, H. Hauser, H. Doleisch, et al. “The State of the Art in Flow Visualization: Dense and Texture-Based Techniques”. In: *Computer Graphics Forum* 23 (2004), p. 2004 (cit. on p. 21).

[Lar+07] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post. “Topology-Based Flow Visualization, The State of the Art”. English. In: *Topology-Based Methods in Visualization*. Ed. by H. Hauser, H. Hagen, and H. Theisel. Mathematics and Visualization. Springer Berlin Heidelberg, 2007, pp. 1–19 (cit. on p. 21).

[Liu08] J. S. Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008 (cit. on p. 88).

[Lod+96a] S. K. Lodha, A. Pang, R. E. Sheehan, and C. M. Wittenbrink. “UFLOW: Visualizing Uncertainty in Fluid Flow”. In: *Visualization’96. Proceedings.* IEEE. 1996, pp. 249–254 (cit. on p. 45).

[Lod+96b] S. K. Lodha, A. Pang, R. E. Sheehan, and C. M. Wittenbrink. “UFLOW: Visualizing uncertainty in fluid flow”. In: *Visualization’96. Proceedings.* IEEE. 1996, pp. 249–254 (cit. on p. 54).

[Löf+97] H. Löffelmann, L. Mroz, E. Gröller, and W. Purgathofer. “Stream arrows: enhancing the use of stream surfaces for the visualization of dynamical systems”. In: *The Visual Computer* 13.8 (Nov. 1997), pp. 359–369 (cit. on p. 23).

[LB98] A. Lopes and K. Brodlie. “Accuracy in 3D Particle Tracing”. In: *Mathematical Visualization: Algorithms, Applications and Numerics*. Ed. by H.-C. Hege and K. Polthier. Springer, 1998, pp. 329–341 (cit. on p. 62).

[Mar+03] D. L. Marchisio, J. T. Pikturna, R. O. Fox, R. D. Vigil, and A. A. Barresi. “Quadrature method of moments for population-balance equations”. In: *AIChE Journal* 49.5 (2003), pp. 1266–1276 (cit. on p. 81).

[MF05] D. Marchisio and R. Fox. “Solution of population balance equations using the direct quadrature method of moments”. In: *Journal of Aerosol Science* 36.1 (Jan. 2005), pp. 43–73 (cit. on p. 81).

[MMY06] R. A. Maronna, R. D. Martin, and V. J. Yohai. *Robust statistics: Theory and methods*. John Wiley, New York, 2006 (cit. on p. 59).

[Mat+03] O. Mattausch, T. Theußl, H. Hauser, and E. Gröller. “Strategies for Interactive Exploration of 3D Flow Using Evenly-spaced Illuminated Streamlines”. In: *Proceedings of the 19th Spring Conference on Computer Graphics*. SCCG ’03. Budmerice, Slovakia: ACM, 2003, pp. 213–222 (cit. on p. 21).

[McG97] R. McGraw. “Description of Aerosol Dynamics by the Quadrature Method of Moments”. In: *Aerosol Science and Technology* 27.2 (AUG 1997), 255–265 (cit. on p. 81).

[McL+09] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. “Over Two Decades of Integration-Based, Geometric Flow Visualization”. In: *EG 2009 - State of the Art Reports*. 2009, pp. 73–92 (cit. on pp. 18, 21).

[Mon05] J. J. Monaghan. “Smoothed Particle Hydrodynamics”. In: *Reports on Progress in Physics* 68.8 (2005), p. 1703 (cit. on pp. 8, 31).

[Nat12] National Research Council. *Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification*. Washington, DC: The National Academies Press, 2012 (cit. on p. 26).

[Obe+09] H. Obermaier, M. Hering-Bertram, J. Kuhnert, and H. Hagen. “Volume Deformations in Grid-Less Flow Simulations”. In: *Computer Graphics Forum* 28 (2009), pp. 879–886 (cit. on pp. 53, 57, 58).

[OJ12] H. Obermaier and K. I. Joy. “Function Field Analysis for the Visualization of Flow Similarity in Time-Varying Vector Fields”. In: *Advances in Visual Computing*. Vol. 7432. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 253–264 (cit. on p. 56).

[OGT11] M. Otto, T. Germer, and H. Theisel. “Uncertain topology of 3D vector fields”. In: *Pacific Visualization Symposium (PacificVis)*. 2011, pp. 67–74 (cit. on p. 53).

[Ott+10] 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 (cit. on p. 53).

[PWL97] A. T. Pang, C. M. Wittenbrink, and S. K. Lodha. “Approaches to uncertainty visualization”. In: *The Visual Computer* 13 (8 1997), pp. 370–390 (cit. on p. 27).

[Pet+13] M. Petitti, M. Vanni, D. L. Marchisio, A. Buffo, and F. Podenzani. “Simulation of coalescence, break-up and mass transfer in a gas-liquid stirred tank with {CQMOM}”. In: *Chemical Engineering Journal* 228 (2013), pp. 1182–1194 (cit. on p. 81).

[PPH12] C. Petz, K. Pöthkow, and H.-C. Hege. “Probabilistic Local Features in Uncertain Vector Fields with Spatial Correlation”. In: *Computer Graphics Forum* 31.3pt2 (2012), pp. 1045–1054 (cit. on p. 53).

[Pir+12] H. Piringer, S. Pajer, W. Berger, and H. Teichmann. “Comparative Visual Analysis of 2D Function Ensembles”. In: *Computer Graphics Forum* 31 (2012), pp. 1195–1204 (cit. on p. 53).

[Pob+11] A. Pobitzer, R. Peikert, R. Fuchs, et al. “The State of the Art in Topology-based Visualization of Unsteady Flow”. In: *Computer Graphics Forum* 30.6 (Sept. 2011), pp. 1789–1811 (cit. on p. 21).

[Pop03] S. Popinet. “Gerris: A Tree-based Adaptive Solver for the Incompressible Euler Equations in Complex Geometries”. In: *J. Comput. Phys.* 190.2 (Sept. 2003), pp. 572–600 (cit. on p. 8).

[Pot+12] K. Potter, M. Kirby, D. Xiu, and C. R. Johnson. “Interactive visualization of probability and cumulative density functions”. In: *International journal for uncertainty quantification* 2.4 (2012) (cit. on p. 27).

[Pot+10] K. Potter, J. Kniss, R. Riesenfeld, and C. R. Johnson. “Visualizing summary statistics and uncertainty”. In: *Computer Graphics Forum*. Vol. 29. 3. Wiley Online Library. 2010, pp. 823–832 (cit. on p. 27).

[Pot+09a] K. Potter, A. Wilson, P.-T. Bremer, et al. “Ensemble-Vis: A Framework for the Statistical Visualization of Ensemble Data”. In: *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*. ICDMW ’09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 233–240 (cit. on p. 53).

[Pot+09b] K. Potter, A. Wilson, P.-T. Bremer, et al. “Visualization of Uncertainty and Ensemble Data: Exploration of Climate Modeling and Weather Forecast Data with Integrated ViSUS-CDAT Systems”. In: *Journal of Physics: Conference Series*. Journal of Physics: Conference Series 180.012089 (2009), (published online) (cit. on p. 53).

[Pre+07] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. 3rd. Cambridge, United Kingdom: Cambridge university Press, 2007 (cit. on pp. 20, 21).

[Roh+11] M. Rohith, H. Parishani, O. Ayala, L.-P. Wang, and C. Kambhamettu. “CollisionExplorer: A Tool for Visualizing Droplet Collisions in a Turbulent Flow”. In: *Advances in Visual Computing: 7th International Symposium, ISVC 2011, Las Vegas, NV, USA, September 26-28, 2011. Proceedings, Part II*. Ed. by G. Bebis, R. Boyle, B. Parvin, et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 669–680 (cit. on p. 82).

[Ros56] M. Rosenblatt. “Remarks on Some Nonparametric Estimates of a Density Function”. In: *The Annals of Mathematical Statistics* 27.3 (Sept. 1956), pp. 832–837 (cit. on pp. 91, 92).

[Rus03] H. Rusche. “Computational fluid dynamics of dispersed two-phase flows at high phase fractions”. PhD thesis. Imperial College London (University of London), 2003 (cit. on p. 81).

[Sad15] F. Sadlo. “Lyapunov Time for 2D Lagrangian Visualization”. In: *Topological and Statistical Methods for Complex Data*. Springer Berlin Heidelberg, 2015, pp. 167–181 (cit. on p. 25).

[SP07] 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 (cit. on p. 26).

[SRP11] 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*. Springer Berlin Heidelberg, 2011, pp. 151–165 (cit. on p. 26).

[Sal+08a] 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 (cit. on p. 31).

[SS06] T. Salzbrunn and G. Scheuermann. “Streamline Predicates”. In: *IEEE Transactions on Visualization and Computer Graphics* 12.6 (2006), pp. 1601–1612 (cit. on p. 31).

[Sal+08b] T. Salzbrunn, H. Jänicke, T. Wischgoll, and G. Scheuermann. “The State of the Art in Flow Visualization: Partition-based Techniques”. In: *Simulation and Visualization 2008 Proceedings*. 2008 (cit. on p. 21).

[San+10] J. Sanyal, S. Zhang, J. Dyer, et al. “Noodles: A Tool for Visualization of Numerical Weather Model Ensemble Uncertainty”. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1421–1430 (cit. on p. 53).

[Sch02] M. Schatzman. *Numerical Analysis: A Mathematical Introduction*. New York, USA: Oxford University Press, 2002 (cit. on p. 34).

[Sch+12] D. Schneider, J. Fuhrmann, W. Reich, and G. Scheuermann. “A Variance Based FTLE-Like Method for Unsteady Uncertain Vector Fields”. English. In: *Topological Methods in Data Analysis and Visualization II*. Ed. by R. Peikert, H. Hauser, H. Carr, and R. Fuchs. Mathematics and Visualization. Springer Berlin Heidelberg, 2012, pp. 255–268 (cit. on p. 53).

[Sed+14] M. Sedlmair, C. Heinzl, S. Bruckner, H. Piringer, and T. Möller. “Visual Parameter Space Analysis: A Conceptual Framework”. In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (Dec. 2014), pp. 2161–2170 (cit. on p. 27).

[SLM05] 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* 212.3-4 (2005), pp. 271–304 (cit. on pp. 10, 24, 42).

[Smi+06] K. M. Smith, D. C. Banks, N. Druckman, K. Beason, and M. Y. Hussaini. “Clustered ensemble averaging: A technique for visualizing qualitative features of stochastic simulations.” In: *Journal of Computational and Theoretical Nanoscience* 3 (5 2006), pp. 752–760 (cit. on p. 53).

[Smi+09] R. L. Smith, C. Tebaldi, D. Nychka, and L. O. Mearns. “Bayesian Modeling of Uncertainty in Ensembles of Climate Models”. In: *Journal of the American Statistical Association* 104.485 (2009), pp. 97–116 (cit. on p. 51).

[SJ13] B. Spencer and M. W. Jones. “Progressive Photon Relaxation”. In: *ACM Transactions on Graphics* 32.1 (Feb. 2013), 7:1–7:11 (cit. on p. 104).

[SP10] R. Stresing and J. Peinke. “Towards a stochastic multi-point description of turbulence”. In: *New Journal of Physics* 12.10 (2010), p. 103046 (cit. on p. 27).

[Tho+04] A. L. R. Thomas, G. K. Taylor, R. B. Srygley, R. L. Nudds, and R. J. Bomphrey. “Dragonfly flight: free-flight and tethered flow visualizations reveal a diverse array of unsteady lift-generating mechanisms, controlled primarily via angle of attack”. In: *Journal of Experimental Biology* 207.24 (2004), pp. 4299–4323 (cit. on p. 18).

[Tuk77] J. W. Tukey. “Exploratory data analysis”. In: (1977) (cit. on p. 27).

[TB96] G. Turk and D. Banks. “Image-guided Streamline Placement”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’96. New York, NY, USA: ACM, 1996, pp. 453–460 (cit. on p. 21).

[Urn+06] T. Urness, V. Interrante, E. Longmire, et al. “Strategies for the visualization of multiple 2D vector fields”. In: *IEEE Transactions on Computer Graphics and Applications* 26.4 (2006), pp. 74–82 (cit. on p. 54).

[VP04] V. Verma and A. Pang. “Comparative Flow Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 10.6 (2004), pp. 609–624 (cit. on pp. 54, 62).

[Von51] J. Von Neumann. “Various Techniques Used in Connection With Random Digits”. In: *Monte Carlo Method, National Bureau of Standards Series* 12 (1951), pp. 36–38 (cit. on p. 88).

[Avi+10] L. S. Avila, U. Ayachit, S. Barré, et al. *The VTK User’s Guide*. 11th. Kitware, Inc., 2010 (cit. on pp. 11, 12, 93).

[WW13] K. E. Wardle and H. G. Weller. “Hybrid Multiphase CFD Solver for Coupled Dispersed/Segregated Flows in Liquid-Liquid Extraction”. In: *International Journal of Chemical Engineering* 2013 (2013) (cit. on p. 80).

[Wei+10] J. Wei, C. Wang, H. Yu, and K.-L. Ma. “A sketch-based interface for classifying and visualizing vector fields”. In: *Pacific Visualization Symposium (PacificVis)*. 2010, pp. 129–136 (cit. on p. 56).

[Wel+98] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. “A tensorial approach to computational continuum mechanics using object-oriented techniques”. In: *Computers in physics* 12.6 (1998), pp. 620–631 (cit. on pp. 8, 95).

[WV92] J. Wilhelms and A. Van Gelder. “Octrees for Faster Isosurface Generation”. In: *ACM Trans. Graph.* 11.3 (July 1992), pp. 201–227 (cit. on p. 17).

[WP09] A. T. Wilson and K. C. Potter. “Toward visual analysis of ensemble data sets”. In: *Proceedings of the 2009 Workshop on Ultrascale Visualization*. UltraVis ’09. Portland, Oregon, 2009, pp. 48–53 (cit. on pp. 53, 54).

[WPL96] C. M. Wittenbrink, A. T. Pang, and S. K. Lodha. “Glyphs for Visualizing Uncertainty in Vector Fields”. In: *IEEE Transactions on Visualization and Computer Graphics* 2.3 (1996), pp. 266–279 (cit. on p. 54).

[YH89] M. Yoda and L. Hesselink. “Three-Dimensional Measurement, Display, and Interpretation of Fluid Flow Datasets”. In: vol. 1083. 1989, pp. 112–117 (cit. on p. 9).

[ZTZ05] O. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. 6th. Oxford, United Kingdom: Butterworth-Heinemann, 2005 (cit. on p. 14).

[ZSH96] M. Zöckler, D. Stalling, and H.-C. Hege. "Interactive Visualization of 3D-vector Fields Using Illuminated Stream Lines". In: *Proceedings of the 7th Conference on Visualization '96*. VIS '96. San Francisco, California, USA: IEEE Computer Society Press, 1996, 107–ff. (Cit. on p. 21).

Publications

Mathias Hummel, Harald Obermaier, Christoph Garth, and Kenneth 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. Received the 2013 IEEE Scientific Visualization Best Paper Award.

Mathias Hummel, Roxana Bujack, Kenneth I. Joy, and Christoph Garth. “Error Estimates for Lagrangian Flow Field Representations“. In: *EuroVis 2016 - Short Papers*. Ed. by Enrico Bertini, Niklas Elmqvist, and Thomas Wischgoll. The Eurographics Association, 2016. Received the 2016 EuroVis Best Short Paper Award.

Mark W. Hlawitschka, Jan Schäfer, Mathias Hummel, Christoph Garth, and Hans-Jörg Bart. “Populationsbilanzmodellierung mit einem Mehrphasen-CFD-Code und vergleichende Visualisierung”. In: *Chemie Ingenieur Technik* 88.10 (2016), pp. 1480–1491

Mathias Hummel, Lisa Jöckel, Jan Schäfer, Mark Werner Hlawitschka, and Christoph Garth. “Visualizing Probabilistic Multi-Phase Fluid Simulation Data using a Sampling Approach”. In: *Computer Graphics Forum* 36.3 (2017), pp. 469–477

Mathias Hummel, Lisa Jöckel, Jan Schäfer, Mark Werner Hlawitschka, and Christoph Garth. “Evaluating Sampling Strategies for Visualizing Uncertain Multi-Phase Fluid Simulation Data”. In: *Physical Modeling for Virtual Manufacturing Systems and Processes*. Vol. 869. Applied Mechanics and Materials. Trans Tech Publications, Sept. 2017, pp. 139–148

Mark Werner Hlawitschka, Jan Schäfer, Lisa Jöckel, Mathias Hummel, Christoph Garth, and Hans-Jörg Bart. “CFD Simulation and Visualization of Reactive Bubble Columns”. In: *Journal of Chemical Engineering of Japan* 51.4 (2018), pp. 356–365

Curriculum Vitae:
Mathias Hummel

Education

2005 **Abitur.** Jürgen-Fuhlendorf-Schule Bad Bramstedt

2010 **Bachelor of Science, Computer Science.** Technische Universität Kaiserslautern
Bachelor thesis topic: *IRIS: Illustrative Rendering of Integral Surfaces*

2013 **Master of Science, Computer Science.** Technische Universität Kaiserslautern
Master thesis topic: *Towards Efficient Lagrangian Representations of Vector Fields*

Experience

2013 - 2015 **Visiting Scholar.** University of California, Davis, USA
Research focus: Ensemble Visualization and Lagrangian Methods for Flow Visualization

2014 - 2017 **Research Associate.** IRTG 2057, Technische Universität Kaiserslautern
Research focus: Trajectory-Based Analysis of Flow Simulation Data

2017 - 2018 **Developer Technology Intern.** NVIDIA Corporation, Santa Clara, USA
Topic: GPU-Accelerated Technologies for Scientific Visualization

since 2018: **Graphics Compute Engineer.** NVIDIA GmbH, Würselen, Germany