Visualization of Two-Phase Flow Dynamics: Techniques for Droplet Interactions, Interfaces, and Material Transport

Von der Fakultät Informatik, Elektrotechnik und
Informationstechnik und dem Stuttgart Research Centre for
Simulation Technology der Universität Stuttgart
zur Erlangung der Würde eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

Vorgelegt von

Grzegorz Karol Karch

aus Świerklaniec, Polen

Hauptberichter:	Prof. Dr. T. Ertl
Mitberichter:	Prof. Dr. H. Theisel
Tag der mündlichen Prüfung:	1. September 2017

Visualisierungsinstitut
der Universität Stuttgart

2017

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

API	application programming interface
CFD	computational fluid dynamics
CFL	Courant-Friedrichs-Lewy
CPU	central processing unit
CT	computed tomography
CUDA	Compute Unified Device Architecture developed by NVIDIA
DFT	discrete Fourier transform
DNS	direct numerical simulation
FS3D	Free Surface 3D
FTLE	finite-time Lyapunov exponent
FVM	finite volume method
GLSL	OpenGL shading language
GPU	graphics processing unit
LCS	Lagrangian coherent structures
LIC	line integral convolution
MAC	marker and cell
MIMD	multiple instruction, multiple data
MPI	Message Passing Interface
ODE	ordinary differential equation
OpenGL	Open Graphics Library
PCA	principal component analysis
PDE	partial differential equation
PLIC	piecewise linear interface calculation
RAM	random access memory
SDK	software development kit
VOF	volume of fluid
WENO	weighted essentially non-oscillatory

ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude to my supervisor Thomas Ertl whose fantastic lectures kindled my interest in scientific visualization, and whose open-mindedness kept me motivated throughout my doctoral studies. I am very thankful to Holger Theisel who agreed to review my thesis and co-chaired the defense. His fascinating work on flow visualization inspired my research in many ways. I would also like to thank Filip Sadlo who often gave me invaluable advice and taught me to keep a close eye on the quality and soundness of my research.

Many people helped me get this thesis done: Oliver Fernandes, Valentin Bruder, Alex Panagiotidis, Michael Krone, Marcel Hlawatsch, Markus Huber, Alex Straub, and Sebastian Boblest. I am deeply indebted to them for proof-reading this work, and for fruitful discussions on how to make the thesis better. I would like to express my gratitude to my co-authors: Filip Sadlo, Daniel Weiskopf, Fabian Beck, Bernhard Weigand, Harald Songoro, E. Gjonaj, Thomas. Weiland, Chuck Hansen, Guo-Shi Li, and Claus-Dieter Munz. In particular, I thank my colleagues at the Institue of Aerospace dynamics: Moritz Ertl, Philipp Rauschenberger, Kathrin Schulte, Martin Reitzle and Karin Schlottke for rewarding collaboration and inspiring discussions in our projects. I would like to thank Daniel Weiskopf for keeping lively atmosphere at VISUS and for inspiring conversations. I shared the office with Valentin, Alex, and David with whom I always enjoyed great atmosphere at work. Kelly Gaither hosted me at Texas Advanced Computing Center in Austin, Texas, where I could expand my research interests in a wonderful working environment.

Of course, PhD is not only about research. I had a great time playing billiards with Alex, Marcel and Kathrin Scharnowski, I enjoyed sport activities with Markus Üffinger and Sebastian Boblest, and watching s-f movies with Kuno, Dominik, Qi and Michael Stoll. Special thanks go to Kuno for his super-spicy chilis which made the pizzas burn like hell. Many thanks to electronic music fans Markus and Steffen and especially Mar Tonmeister who introduced me to the new music landscapes. Special thanks also go to Kavi Ramasamy with whom I enjoyed weekly lunches and always a great company. I will miss our whiskey tasting with Oli, Alex Panagiotidis, Christoph, Michael Krone, and Tobi Rau, as well as sushi extravaganza outings with Lena, Guido, Christoph, Michael, Alex and Fabian.

And, last but certainly not least, I would like to thank my parents for their enduring support, their faith in me and for helping me get so far.

I conducted my research as part of the Collaborative Research Center SFB-TRR 75 founded by the German Research Foundation (DFG).

ABSTRACT

Computational visualization allows scientists and engineers to better understand simulation data and gain insights into the studied natural processes. Particularly in the field of computational fluid dynamics, interactive visual presentation is essential in the investigation of physical phenomena related to gases and liquids. To ensure effective analysis, flow visualization techniques must adapt to the advancements in the field of fluid dynamics that benefits substantially from the growing computational power of both commodity desktops and supercomputers on the one hand, and steadily expanding knowledge about fluid physics on the other. A prominent example of these advances can be found in the research of two-phase flow with liquid droplets and jets, where high performance computation and sophisticated algorithms for phase tracking enable well resolved and physically accurate simulations of liquid dynamics.

Yet, the field of two-phase flow has remained largely unexplored in visualization research so far, leaving the scientists and engineers with a number of challenges when analyzing the data. These include the difficulty in tracking and investigating topological events in large droplet groups, high complexity of droplet dynamics due to the involved interfaces, and a limited choice of high quality interactive methods for the analysis of related transport phenomena. It is therefore the aim of this thesis to address these challenges by providing a multi-scale approach for the visual investigation of two-phase flow, with the focus on the analysis of droplet interaction, fluid interfaces, and material transport.

To address the problem of analyzing highly complex two-phase flow simulations with droplet groups and jets, a linked-view approach with three-dimensional and abstract space-time graph representation of droplet dynamics is proposed. The interactive brushing and linking allows for general exploration of topological events as well as detailed inspection of dynamics in terms of oscillations and rotations of droplets. Another approach further examines the separation of liquid phases by segmenting liquid volumes according to their topological changes in future time. For visualization, boundary surfaces of these volume segments are extracted that reveal intricate details of droplet topology dynamics. Additionally, within this framework, visualization of advected particles corresponding to arbitrarily selected segment provides useful insights into the spatio-temporal evolution of the segment.

The analysis of interfaces is necessary to understand the interplay of interface dynamics and the dynamics of droplet interactions. A commonly used technique for interface tracking in the volume of fluid-based simulations is the piecewise linear approximation which, although accurate, can affect the quality of the simulation results. To study the influence of the interface reconstruction on the phase tracking procedure, a visualization method is presented that extracts the interfaces by means of the first-order Taylor approximation, and provides several derived quantities that help assess the simulation results in relation to the interface reconstruction quality. The liquid interface is further investigated from the physical standpoint with an approach based on quantities derived from velocity and surface tension gradients. The developed method supports examination of surface tension forces and their impact on the interface instability, as well as detailed analysis of interface deformation characteristics. A line of research important for engineering applications is the analysis of electric fields on droplet interfaces. It is, however, complicated by higher-order elements used in the simulations to preserve field discontinuities. A visualization method has been developed that correctly visualizes these discontinuities at material boundaries. Additionally, the employed space-time representation of the droplet-insulator contact line reveals characteristics of electric field dynamics.

The dynamics of droplets are often examined assuming single-phase flow, for instance when the internal material transport is of interest. From the visualization perspective, this allows for adaption of traditional vector field visualization techniques to the investigation of the studied phenomena. As one such concept, dye based visualization is proposed that extends the transport analysis to advection-diffusion problems, therefore revealing true transport behavior. The employed high quality advection preserves fine details of the dye, while the implementation on graphics processing units ensures interactive visualization. Several streamline-based concepts are applied in space-time representation of 2D unsteady flow. By interpreting time as the third spatial dimension, many 3D streamline-based visualization techniques can be applied to investigate 2D unsteady flow. The introduced vortex core ribbons support the examination of vortical flow behavior by revealing rotation near the core lines. For the study of topological structures, a method has been developed that extracts separatrices implicitly as boundaries of regions with different flow behavior, and therefore avoids potentially complicated explicit extraction of various topological structures.

All proposed techniques constitute a novel multi-scale approach for visual analysis of two-phase flow. The analysis of droplet interactions is addressed with visualization of the phenomena leading to breakups and with detailed visual inspection of these breakups. On the interface level, techniques for the interface analysis give insights into the simulation quality, mechanisms behind topology changes, as well as the behavior of electrically charged droplets. Further down the scale, the dye-based visualization, streamline-based concepts for space-time analysis, and the implicit extraction of flow topology allow for the investigation of droplet internal transport as well as general single-phase flow scenarios. The applicability of the proposed methods extends, in a varying degree, beyond the use in two-phase flow. Their usability is demonstrated on data from simulations based on Navier-Stokes equations that exemplify practical problems in the research of fluid dynamics.

German Abstract - Zusammenfassung

Die numerische Visualisierung ermöglicht Wissenschaftlern und Ingenieuren, Simulationsergebnisse besser zu verstehen und Einblicke in Naturprozesse zu gewinnen. Insbesondere ist die visuelle Darstellung von Ergebnissen numerischer Strömungsmechanik für die Untersuchung physikalischer Phänomene bei Gasen und Flüssigkeiten äußerst wichtig. Die numerische Strömungsmechanik profitiert einerseits von wachsender Rechenleistung handelsüblicher Desktops und Supercomputer, andererseits von den neuen Entwicklungen in der Strömungsforschung. Um eine effektive Analyse von Strömungen zu gewährleisten, müssen sich die Visualisierungstechniken kontinuierlich den Fortschritten in der Strömungsmechanik anpassen. Ein bemerkenswertes Beispiel hierfür ist die Forschung in der Zweiphasenströmung, in der Hochleistungsrechner und effiziente Algorithmen zur Phasenverfolgung hochaufgelöste und physikalisch genaue Simulationen der Flüssigkeitsdynamik ermöglichen.

Dennoch ist die Zweiphasenströmung seitens der Visualisierung weitgehend unerforscht geblieben. Insbesondere sehen sich Wissenschaftler und Ingenieure mit verschiedenen Problemen konfrontiert, die mit angepassten Visualisierungstechniken vermieden werden können. Zu den Problemen zählen beispielweise die Verfolgung und Untersuchung der topologischen Ereignisse in Tropfengruppen, hohe Komplexität der Tropfendynamik und die begrenzte Auswahl an interaktiven Methoden zur Untersuchung der Transportphänomene. Demzufolge ist das Ziel dieser Dissertation, die Entwicklung eines Ansatzes zur visuellen Analyse von Zweiphasenströmung auf mehreren Skalen mit dem Fokus auf Interaktionen zwischen den Tropfen, Dynamik der Oberfläche und Materialtransport.

Um die Analyse hochkomplexer Simulationsdaten der Zweiphasenströmung zu behandeln, wird eine auf Linked-View-Verfahren basierte Visualisierungstechnik präsentiert, in der die Tropfen sowohl in einer 3D Darstellung als auch in einer abstrakten Graph-Repräsentation visualisiert werden. Der interaktive Brushing-and-Linking-Ansatz ermöglicht eine globale Exploration der topologischen Ereignisse sowie eine detaillierte Inspektion der Dynamik im Hinblick auf die Oszillation und Rotation der Tropfen. Eine andere Technik zeigt die Aufteilung des Tropfenvolumens im zeitlichen Verlauf. Somit ermöglicht diese Methode eine ausführliche Untersuchung der Topologiedynamik mit Hilfe einer statischen Visualisierung. Dafür werden Grenzflächen erzeugt, die das ursprüngliche Volumen des Tropfens hinsichtlich der sich entwickelnden Zerfallskomponenten aufzeigen. Zusätzlich werden die zur Verfolgung der Tropfen benutzten Partikel visualisiert, um Einblicke in die Dynamik der Separation zu gewähren.

Die Analyse der Oberfläche ist notwendig, um die Wechselwirkung zwischen der Oberflächendynamik und der Dynamik der Tropfeninteraktion besser zu verstehen. Eine häufig angewendete Technik zur Verfolgung der Phasengrenzen im Volume-of-Fluid-Verfahren ist die zellenweise planare Approximation. Obwohl diese einen guten Kompromiss zwischen Genauigkeit und Performanz bietet, kann die Approximation die Qualität der Simulationsergebnisse erheblich beeinflussen. Es wird deshalb eine Visualisierungsmethode präsentiert, die die Oberfläche mit Hilfe der Taylor-Approximation erster Ordnung extrahiert und unter anderem darauf basierte Größen bereitstellt, die die Relation zwischen der Simulationsapproximation und Qualität der Ergebnisse zeigt. Die Tropfenoberfläche wird weiterhin mit einer Visualisierungsmethode analysiert, die von den Geschwindigkeits- und Oberflächenspannungsgradienten abgeleitete Größen verwendet. Die entwickelte Methode unterstützt die Untersuchung der Deformation der Oberfläche sowie die Untersuchung der Oberflächenspannung und deren Auswirkung auf die Oberflächenstabilität. Eine wichtige Forschungsrichtung in der Zweiphasenströmung ist die Analyse elektrischer Felder auf der Tropfenoberfläche. Die in der Simulation angewendeten Elemente höherer Ordnung ermöglichen physikalische Diskontinuitäten, die für die visuelle Analyse eine gesonderte Behandlung benötigen. Im Zuge dessen wird eine Methode präsentiert, welche die Diskontinuitäten visuell korrekt darstellt und zusätzlich eine Raum-Zeit-Darstellung anwendet, um Einblicke in die Phänomene an der Kontaktlinie zwischen den Tropfen und dem untersuchten Isolator zu gewähren.

Die Tropfendynamik wird oft mit der Annahme einer Einphasenströmung analysiert, beispielsweise für die Untersuchung der internen Strömung des Tropfens. Dies ermöglicht eine Anpassung und Verwendung traditioneller Visualisierungsmethoden für Vektorfelder. Eine solche Technik ist die „Dye-Advection", die in dieser Dissertation nicht nur zur Analyse der Advektion, sondern auch zur Untersuchung der Diffusion verwendet wird. Die eingesetzte hochqualitative Rekonstruktion des virtuellen Pigments bewahrt feine Details, während die Implementierung auf der Grafikkarte eine interaktive Visualisierung ermöglicht. Überdies werden einige auf Stromlinien basierende Konzepte in Raum-Zeit-Darstellung angewendet, in der die Zeit als die dritte Raumachse interpretiert wird. Demzufolge können diese Methoden zur Analyse der zeitabhängigen zweidimensionalen Strömung verwendet werden. Die eingeführten „Vortex Core Ribbons" unterstützen die Analyse der rotierenden Strömung um die Wirbelkernlinien. Für die Analyse der topologischen Strukturen wurde eine Methode entwickelt, die die Separatrizen implizit als Ränder einer Segmentierung des Vektorfeldes extrahiert. Damit wird eine möglicherweise komplexe direkte Extraktion der Separatrizen vermieden.

Die präsentierten Visualisierungsmethoden bilden ein neuartiges Multiskalen-Verfahren zur visuellen Analyse von Zweiphasenströmungen. Die Tropfeninteraktionen werden mit Hilfe einer Visualisierung dargestellt, die sich auf die Ursache des Tropfenzerfalls und deren Ablauf konzentriert. Für die Untersuchung der Oberfläche zeigen die vorgeschlagenen Techniken die Qualität der Ergebnisse hinsichtlich der Oberflächenrekonstruktion, die Mechanismen hinter den topologischen Ereignissen, als auch die Dynamik der elektrisch geladenen Tropfen auf. Andererseits werden unter Annahme der Einphasenströmung neue Techniken basierend auf Dye-Advection, Stromlinien-basierte Konzepte, sowie Verfahren zur Extraktion der Topologie untersucht, um einen besseren Einblick in den Materialtransport zu gewinnen. Die Anwendung dieser Methoden wird in dieser Dissertation auf Daten demonstriert, die durch Simulation, basierend auf Navier-Stokes-Gleichungen, erzeugt wurden.

INTRODUCTION | 1

Computational fluid dynamics (CFD) is a branch of fluid mechanics that focuses on the analysis and development of numerical methods for solving fluid dynamic problems [65, 66, 201]. Typically, these methods find solutions to the Navier-Stokes equations that describe the motion of fluids. Initially, during the 1950s and 1960s, CFD was employed to problems of general importance. These included weather modeling and material transport in ocean currents as well as the related natural convection phenomena. The spectrum of CFD applications has since then grown significantly, mainly due to the development of new numerical solvers as well as a rapid increase in computational power of supercomputers and commodity desktop computers. The applications of CFD now range from analysis of vehicle aerodynamics, through investigation of droplet splashing in combustion engines, to food processing technology.

Advancements in the field of fluid dynamics would not be possible without flow visualization that allows scientists and engineers to understand the simulated processes and consequently gain new insights into the observed phenomena [27]. There are many challenges related to the visual investigation of flow phenomena—such as handling large data or extraction of relevant information from complex flow simulations—and flow visualization has been a subject of intense research since the mid-1980s [28, 68, 99]. These challenges include the extraction of important flow features from the ever-increasing data volumes and adjusting to advancements in the algorithms used in CFD. Therefore, research in both flow simulation and flow visualization go hand in hand since effective evaluation of computational fluid dynamics simulations depends on suitable visualization techniques, and conversely, flow visualization needs to constantly adapt to the improvements on the simulation side in order to deliver analysis tools that can support the progress in flow simulation. Yet, flow visualization has so far focused on aspects mainly related to single-phase fluid dynamics, although in the CFD field, two-phase flow has already become an established research area. Two-phase flow is a simultaneous flow of gas and liquid, where one phase is typically dispersed in the other and separated from the surrounding phase by an interface. While there exist methods that provide accurate interface reconstruction for the visualization of two-phase simulation data, there are many aspects still not investigated in the scientific visualization community, such as dynamic behavior of liquid droplets, which is the focus of this thesis.

1.1 Motivation and Research Goal

Due to their ubiquity, droplets are the subject of intense research effort that encompasses various aspects of natural processes and engineering design. The dynamics of droplets are investigated, for instance, to better understand cloud formations, or to improve the effectiveness in various applications such as inkjet printers and combustion engines.

The visual analysis of droplets and two-phase flow dynamics in general poses many challenges. The presence of surface tension force leads to complex interface dynamics and frequent droplet breakups and merges. This dependency between small scale physics and large scale events means that different scales and therefore different processes must be considered. However, not only physical processes, but also complex simulation data is problematic. Direct visualization of interfaces in two-phase flow results in visual clutter, and therefore droplet merges and splits as well as intricate interface deformations are difficult to follow. Moreover, droplet simulations require a high spatial resolution to capture fine details of phase dynamics. As a result, large amounts of data are produced that must be effectively handled for constructive visual analysis.

In view of these challenges, the goal of this thesis was to develop visualization techniques that support effective analysis of droplet-related phenomena occurring at different scales. Together, the presented novel techniques constitute a powerful visualization approach that gives insights into complex droplet processes that could not be explored with previous techniques. The visualization methods allow for constructive analysis of the involved complex flow dynamics, droplet-specific phenomena as well as large data.

1.2 Contribution and Thesis Structure

This thesis is the outcome of the visualization research done within project Collaborative Research Center Transregio 75 (SFB-TRR 75) [166] whose focus is the investigation of droplets dynamics under extreme ambient conditions. Within this project, scientists from the University of Stuttgart, the Technical University of Darmstadt, and the German Aerospace Center collaborate in the investigation of droplet phenomena through modeling, experiments, and simulations. Due to the difficulties related to the visual exploration of dynamic processes in two-phase flow, and the deficiency of appropriate visual analysis techniques, visualization has been recognized as an integral part of the SFB-TRR 75. The scope of the challenges faced within this project requires a comprehensive visualization approach. In this thesis, three aspects have been identified as crucial in the analysis of two-phase flow dynamics: inter-droplet interactions, dynamics of droplet interfaces, and material transport in droplets (see Figure 1.1 for an illustration). The proposed visualization techniques address the specific problems at these scales and reveal relevant information to support a multi-scale analysis of two-phase flow dynamics. The following paragraphs outline the structure and contributions of this thesis.

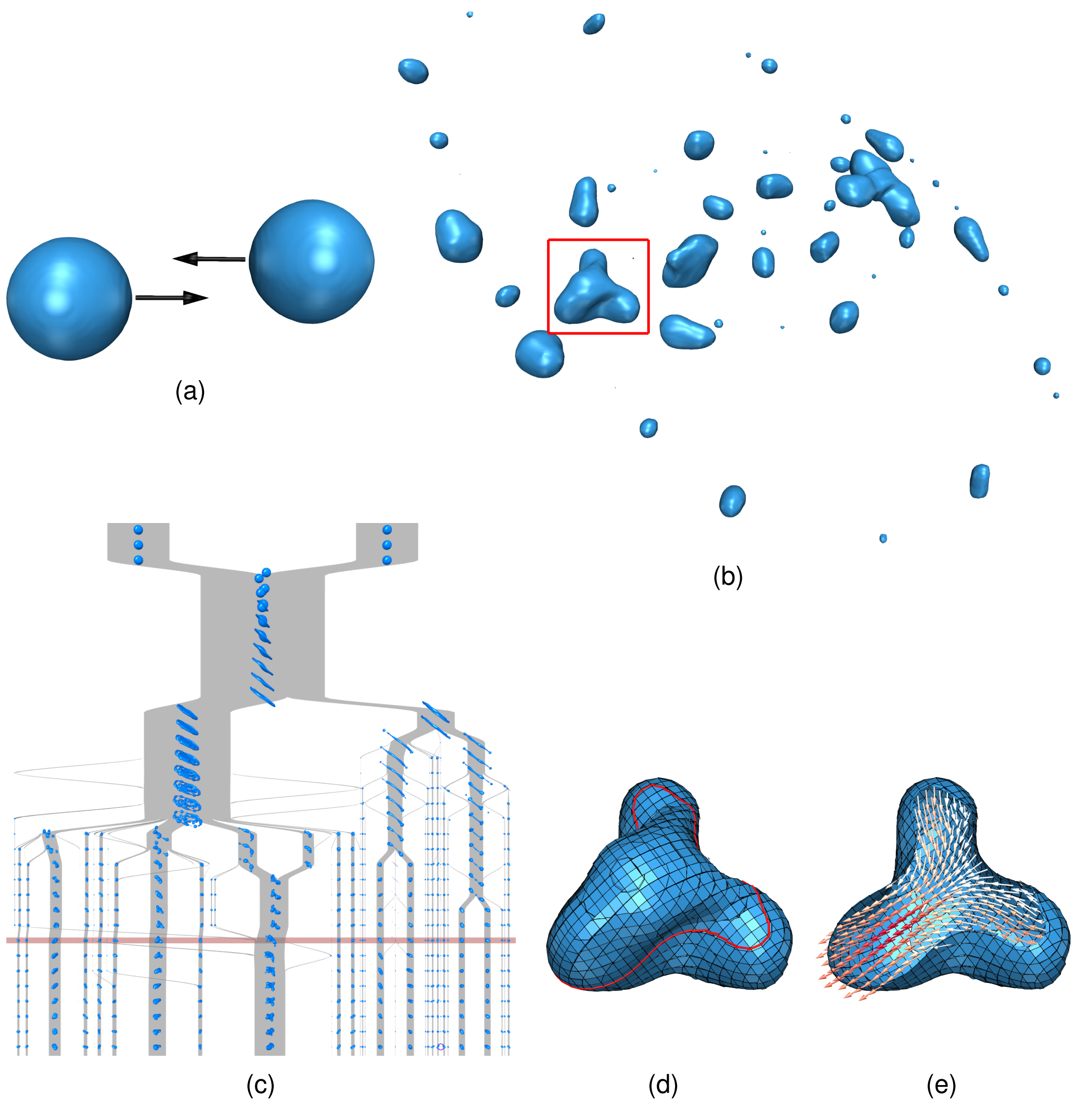


Figure 1.1: Visualization of two-phase flow. (a) In the simulation, two droplets collide. (b) After collision, a multitude of droplets are formed that undergo various topological changes. (c) The whole simulation can be analyzed with a space-time graph representation that shows various droplet interactions. (d) The liquid-gas interface is approximated with planar patches that may have some impact on the simulation quality. (e) There are a number of phenomena related to droplet dynamics for which the presence of interfaces does not have to be considered and hence, single-phase flow is assumed. This can be, for instance, droplet internal flow.

Fundamentals (Chapter 2) In Chapter 2, some basic concepts in flow simulation are provided that are utilized in later chapters. Additionally, fundamental flow visualization techniques are briefly described. These techniques are the building blocks for the visualization methods developed in this thesis. Finally, the state of the art techniques for interface visualization and feature tracking are discussed.

Droplet Interactions (Chapter 3) Droplet interactions are an essential part in the research of natural phenomena and in engineering design processes. For instance, the droplet surface area in combustion engines affects the combustion effectiveness and therefore, origins and characteristics of droplet breakup and splashing on walls are important research questions. In Chapter 3, two novel visualization techniques are presented that facilitate the investigation of droplet groups and jets, that is, fast flowing streams of liquid that typically disperse into a large number of droplets.

The first visualization technique, presented in Section 3.1, employs an abstract hierarchical graph representation of a simulation [83]. In this graph, nodes at a given layer represent droplets for the corresponding time step, whereas edges correspond to topological changes, that is, breakup and merging of drops. This abstract view is tightly coupled with a 3D domain representation in a linked-view approach. Droplet rotation and oscillation are visualized using several novel methods based on principal component analysis that support investigation of topological changes the droplets undergo. The visualization method has been developed under the supervision of Filip Sadlo.

The breakup dynamics of droplets and jets is further scrutinized using the visualization approach described in Section 3.2. Here, droplet volumes are segmented into regions that correspond to separate droplets resulting from a breakup. To reveal the temporal characteristics of the breakup, the separation areas are extracted and visualized together with the information on the separation time. Additionally, particles that are employed for tracking the droplets in time can be used for close examination of the breakup process.

Dynamics of Interfaces (Chapter 4) The dynamics of droplet interactions and the characteristics of phase interfaces are interrelated. Hence, it is important to visually inspect the interface to gain insights into the dynamics of droplets. A prerequisite for a better understanding of interface dynamics is the analysis of the interface reconstruction used in CFD solvers. A typically used reconstruction method in scientific applications is based on planar approximation of the interface [146]. While providing a good trade-off between accuracy and efficiency, it can influence the quality of the simulation results.

Therefore, in the approach presented in Section 4.1, solver interface reconstruction is analyzed using several measures, such as the size of gaps between neighboring planar patches and the curvature of the interface [85]. With the novel interpretation of the planar interface as a Taylor approximation problem, it is possible to generalize the interface to higher-order approximations. This in turn allows for more precise visual investigation of the approximation error. The visualization method has been developed under the supervision of Filip Sadlo.

The analysis of the interface deformation can help to better understand the relations between micro-scale and macro-scale phenomena in two-phase flow. For this purpose, the visualization method presented in Section 4.2 employs a metric tensor to describe interface dynamics in terms of stretching, while a novel shape-tensor-based analysis allows for detailed investigation of surface bending. To inspect the influence of the surface tension on the interface deformation, a velocity field is derived from the solver-based curvature computation. This velocity field is then used with the presented visualization of deformation. This visualization of interface stretching is partially based on the Master Thesis of Alexander Straub [168] who also implemented the visualization of interface bending.

Additionally, complex physics resulting from the presence of an electric field are investigated to help in the analysis of static discharges on wet insulators [86]. Specifically, in Section 4.3, the electric field on the contact line between droplet and insulator is visualized in a novel space-time approach, where the contact line from consecutive simulation time steps is transformed into stripes and then stacked onto each other. For visualization, volume rendering and isocontouring is used. Parts of this visualization method have been implemented by Harald Songoro who also provided the formulation for the interpolation of higher-order finite elements.

Material Transport (Chapter 5) A multitude of phenomena related to droplet dynamics processes can be examined with single-phase flow simulations. This is the case either when only internal flow is considered or when explicitly considering interfaces is not necessary to investigate the problem at hand. In those cases, traditional flow visualization techniques can be employed for visual analysis. However, in single-phase flow simulation, the dynamics of vector fields pose various challenges in the exploration of the flow physics, which are addressed from different angles in this thesis.

In Section 5.1, a novel interactive visualization using dye advection is presented that accounts for both the advection and diffusion of quantities, such as evaporating droplets [90, 89]. The finite volume approach allows for parallelization of the computation and hence interactive frame rates. The employed polynomial reconstruction of the dye substantially reduces the artificial smearing caused by repeated interpolation. This approach preserves fine details of the dye advection and thus provides accurate visual description of flow dynamics. The visualization method has been developed under the supervision of Filip Sadlo and Claus-Dieter Munz who also provided the reconstruction algorithm.

To account for the complexity of 2D unsteady flow, a space-time visualization is proposed in Section 5.2, where time is treated as the third dimension, therefore allowing for the utilization of visualization techniques based on streamlines for time-dependent flow [87, 88]. Specifically, in this representation, vortex core line extraction is employed that provides continuous representation of these features, whereas the applied vortex criteria reveal the vortical structure in the whole domain. Additionally, rotational flow around the core lines is visualized using novel streamline-based ribbons whose twist conveys the spinning motion.

The complex behavior of vector fields can be concisely described with vector field topology. To avoid potentially cumbersome extraction of the involved separatrices, a novel approach is introduced in Section 5.3 that reveals the separatrices implicitly as boundaries in the vector field segmentation [91]. This segmentation is achieved by adaptive streamline integration that captures fine topological details. Both methods presented in Section 5.2 and 5.3 have been developed under the supervision of Filip Sadlo.

Conclusion (Chapter 6) The contributions of the presented visualization techniques are recapitulated in Chapter 6. Additionally, this chapter summarizes how the the presented methods address the challenges in the visualization of two-phase flow. Finally, concluding remarks and discussion on possible future research directions are given.

Challenges in the Visualization of Two-Phase Flow

Together, the presented visualization methods enable comprehensive analysis of the droplet dynamics processes. The challenges related to the analysis of complex dynamics, droplet-specific phenomena, and large data have been approached in this thesis using various techniques, depending on the research problem and investigated data.

Complex Dynamics The existence of interfaces and the resulting surface tension force hinder the visual investigation of two-phase flow dynamics in several ways. The presence of interfaces incurs visual clutter, and highly complex drop dynamics lead to frequent and often abrupt topology changes that are problematic to follow. Therefore, in the analysis of interface reconstruction, liquid advection is visually scrutinized with the direct representation of the transported volumes. Furthermore, in the analysis of interface dynamics, the deformation is captured statically with eigenpairs of tensors describing stretching and bending. Flow dynamics is also considered in other ways: in the form of a space-time graph representation of droplets, a 3D space-time description of 2D unsteady flow, as well as using geometrical representation of temporal separation of droplets. Topology extraction is also employed to significantly reduce the data to features relevant for the analysis.

Droplet-Specific Phenomena To gain insight into the droplet related processes, specially tailored visualization techniques are often required due to the specific characteristics of the two-phase flow phenomena and the applied solvers. For detailed investigation of droplet dynamics, dimensionless quantities are adapted and new quantities based on droplet rotation and oscillation are introduced. The surface tension force causes highly nonlinear flow characteristics that are difficult to track numerically. To improve the particle-based tracking necessary for visualization of droplet topology changes, a specially designed trajectory corrector scheme has been developed. Reimplementation of solver curvature computation method allows for reliable estimation of the surface tension effects on interface deformation and breakups. For the simulation of droplets in the presence of electric fields, higher-order edge-based elements are necessary to preserve field discontinuities. These elements are efficiently handled in the visualization using specialized sampling.

Large Data The ever-increasing computational power enables higher resolutions in the physical domains, thus allowing for more accurate computations. The high resolution is a prerequisite for reliable simulation of flow physics, however, the resulting large amounts of data incur additional effort during visual investigation. In this thesis, different approaches are presented to handle this problem. Level-of-detail and clustering are used to allow for constructive analysis of a vast number of droplets occurring in jet simulations. The analysis of large data is also managed at programming level. Domain parallelization across computing nodes is employed to allow processing of large datasets that would be impossible to achieve on a single workstation. On the other hand, for effective and interactive visualization of dynamic data, a graphics processing unit (GPU) implementation is employed.

Overall, this thesis provides scientists and engineers with visualization techniques for the analysis of complex droplet dynamics. This is achieved using a multi-scale approach for the visualization of droplet interactions, interfaces, and material transport. The thesis addresses the problems of complex flow dynamics and droplet specific phenomena with specially tailored techniques. For effective visualization of large data, different variants of visualization techniques and parallelism are exploited.

Fundamentals and State of the Art | 2

This chapter is intended as an introduction to some basic concepts that are later employed in the description of the developed visualization techniques. All visualization techniques described in this thesis are related to the category of flow visualization. While some of these techniques provide the visualization for the underlying vector field directly, the majority of the proposed methods combine the vector field and a scalar field representing the fluid phase to provide meaningful visualizations of two-phase flow. The mathematical description of vector fields and flow characteristics is given in Section 2.1. Section 2.1.1 briefly introduces selected discretization approaches used for the numerical solution of the Navier-Stokes equations, including the volume of fluid method for two-phase flow. Later, in Section 2.2, essential visualization techniques, such as integral lines, are introduced, and a state of the art in interface reconstruction and feature tracking is provided—both visualization concepts are relevant to the techniques presented in Chapter 3 and 4.

2.1 Fluid Flow

In this section, selected concepts in fluid dynamics and simulation are provided to give necessary background for the visualization methods presented in later chapters. For a comprehensive description of the fundamental concepts in fluid dynamics, the reader is referred to the books by Anderson [7] and Kundu [94].

In this thesis, most of the presented visualization techniques have been devised for the analysis of time-dependent three-dimensional fluid flow. In the simulation data, the vector field **u** represents the fluid velocity at a given point **x** and instant of time t:

$$\mathbf{u}(\mathbf{x},t) = \begin{pmatrix} u(x,y,z,t) \\ v(x,y,z,t) \\ w(x,y,z,t) \end{pmatrix}. \tag{2.1}$$

The Jacobian matrix $\mathbf{J_u} = \nabla\mathbf{u}$ describes the local characteristics of the vector field and is essential in the computation and identification of many vector field properties:

$$\mathbf{J_u} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{pmatrix} . \tag{2.2}$$

Vorticity is a vector defining the axis and the magnitude of local rotation of the fluid:

$$\nabla \times \mathbf{u} = \left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}, \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}, \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) . \tag{2.3}$$

Divergence, on the other hand, is a scalar quantity that quantifies the local volume expansion of a fluid element as induced by the underlying velocity:

$$\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} . \tag{2.4}$$

In this work, the analyzed two-phase flow problems are divergence free (i.e., $\nabla \cdot \mathbf{u} = 0$), whereas some of the single phase problems in Chapter 5 involve buoyant flow and hence are divergent.

Flow Description

Fluid flow problems are commonly approached by numerically solving the Navier-Stokes equations, a set of partial differential equations (PDEs). The incompressible fluid flow can be described by the momentum equation:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla \cdot \nabla \mathbf{u} + \mathbf{g} , \tag{2.5}$$

and the continuity equation:

$$\frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho = 0 , \tag{2.6}$$

with fluid density ρ, pressure p, kinematic viscosity ν, and external force $\mathbf{g}$ which typically represents the gravitational acceleration. In general, the description of the density in fluid flow is given by the substantial derivative:

$$\frac{Dq}{Dt} = \frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho . \tag{2.7}$$

The term on the left side is the temporal change of density as the mass is transported by the flow. The first term on the right side is the temporal change of ρ at a fixed position while the second term is the change in ρ caused by advection. Therefore, the substantial derivative relates the Lagrangian and Eulerian reference frames. For incompressible flows, $D\rho/Dt = 0$ and the substantial derivative reproduces the continuity equation (Equation 2.6).

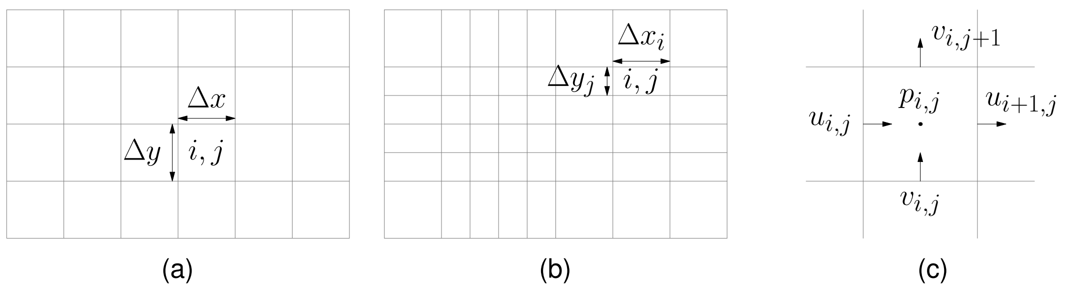


Figure 2.1: Spatial discretization of a simulation domain. Examples of uniform (a) and rectilinear (b) grids. The latter allows for better accuracy at regions of interest, e.g., at the jet core. (c) Marker and cell data representation with velocity components stored on the cell faces and scalar quantities located in the cell centers.

2.1.1 Flow Simulation

All the example datasets investigated with the presented visualization techniques were produced by CFD simulations that numerically solve the Navier-Stokes equations. While there are many CFD solvers that model turbulence to allow for computation of complex flow configurations, a popular class of solvers employed in scientific applications is the direct numerical simulation (DNS) that resolves all temporal and spatial scales [42]. This approach, however, entails substantial computational costs and therefore the simulations are typically run on supercomputers.

The investigated datasets are discretized either on uniform or rectilinear grids. A uniform grid is a type of structured grid with constant cell size defined over the whole simulation domain, as illustrated in Figure 2.1(a). A rectilinear grid represents a more flexible structured grid type, which is still simple to implement. It employs varying cell size along each dimension and therefore provides better accuracy in regions of interest, see Figure 2.1(b). It is important to note that both grid types lend themselves well for parallelization on GPUs and distributed systems, since they allow coherent memory mapping on GPU, as well as domain partitioning that can be readily achieved by determining data intervals on each spatial axis.

The computed vector and scalar data are most commonly associated with either grid nodes or grid cells. For node-based data, the data between sample points is typically reconstructed using bilinear (in 2D) or trilinear (in 3D) interpolation. In cell-based data representation, it is assumed that the data value is constant over the whole cell volume. Another data representation is the marker and cell (MAC) grid [66] where the scalar quantities (such as pressure and volume fraction) are stored on the cell centers, while the components of vector quantities are associated with cell faces whose normal is parallel to the respective component, as illustrated in Figure 2.1(c). Such representation allows for more accurate computation of the central differences used for pressure gradients and divergence [26]. For the simulation output, however, the vector quantities are usually averaged at cell centers to facilitate post-processing. This was also the case for the datasets investigated in this work.

Simulation of Two-Phase Flow

The main focus of this work is the visual analysis of phenomena related to two-phase flow where gas and liquid phases occur simultaneously. Investigation of droplets is an active area of research [49, 177, 207], and so is the analysis of liquid jets [43, 56, 104]. The reader is referred to the work by Fuster et al. [55] for a detailed introduction to multiphase flow simulation and to Lefebvre [100] for a thorough description of liquid atomization and sprays. Two-phase flow is characterized by large density and viscosity ratios, topologically complex interfaces, and presence of surface tension force. The surface tension is explicitly expressed in the momentum equation by an additional term $\mathbf{f}_\gamma$ that acts on the liquid interface:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla \cdot \nabla \mathbf{u} + \mathbf{g} + \mathbf{f}_\gamma . \tag{2.8}$$

In the simulation context, an open research problem is to capture the complex interface topology and, at the same time, accurately compute surface tension forces. There are basically two approaches to the computation of fluid interfaces. In Lagrangian schemes, the interface is represented explicitly by the moving mesh that divides the domain into regions of different phases [31]. In this case, the cells function as control volumes that move with the flow. In the Eulerian schemes, in contrast, the fluid configuration is discretized on a fixed grid, on which the interface must be tracked. In tracking based on *level-set* method [171], the distance to the interface is computed for each cell. To avoid smearing due to repeated advection and to ensure mass conservation, the interface must be regularly reinitialized. On the other hand, in the volume of fluid (VOF) method, which was applied in the example datasets, the fluid is represented by a volume fraction defined for each cell [70]. The advantage of the VOF method over the Lagrangian schemes is that it can handle arbitrarily complex flow structures and changes in topology, since no explicit representation of the fluid interface is needed.

In the VOF method, an additional volume fraction field $f(\mathbf{x},t)$ is maintained for each cell:

$$f(\mathbf{x},t) = \begin{cases} 0 & \text{in the gas phase,} \\ ]0,1[& \text{at the interface,} \\ 1 & \text{in the liquid phase,} \end{cases} \tag{2.9}$$

and it is advected by solving the advection equation

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f = 0 . \tag{2.10}$$

Please see Figure 2.2(a) for an illustration.

Interface Reconstruction in Two-Phase Flow

Since the volume fraction used to numerically solve the Equation 2.10 is discretized on a grid, the information on the exact interface position is lost. Therefore, for the

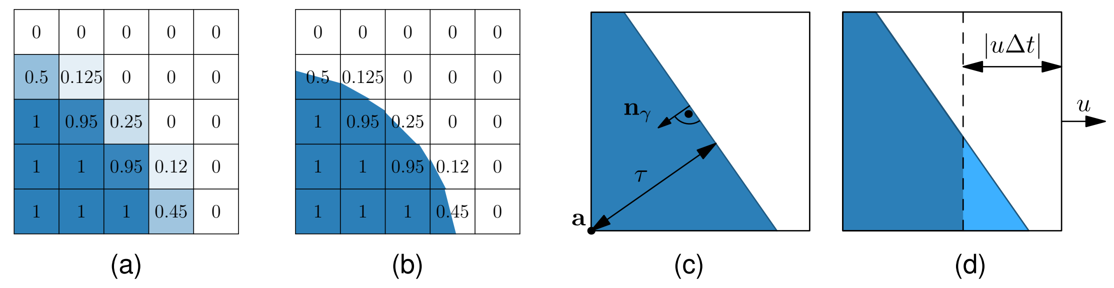


Figure 2.2: Volume tracking in two-phase flow. (a) The volume fractions of the liquid phase are defined per cell. (b) For accurate flux computation, the liquid interface is defined as a planar patch that encloses a volume fraction equal to f. (c) The interface plane is defined by its normal $\mathbf{n}_\gamma$ and distance from the attachment point $\mathbf{a}$. (d) The amount of volume advected to the neighbor cell is equal to the volume cut by the cell face translated with the velocity $-u$ over time Δt.

computation of the advection and the surface tension forces, it is approximated from the cell-constant values of f and from the information on the neighborhood. Several schemes have been proposed for the reconstruction of the interface, with piecewise constant [70, 124], stair-stepped [70], piecewise linear [208, 146] approximation, and second-order reconstruction where the fluid surface is constructed with freely arrangeable planes within a cell [9, 138]. The most widely used reconstruction in the scientific applications is the piecewise linear interface calculation (PLIC).

In the PLIC reconstruction, the interface is approximated by a plane whose normal n_γ is parallel to the gradient of f:

$$n_\gamma := -\nabla f / ||\nabla f||, \tag{2.11}$$

as demonstrated in Figure 2.2(b). The translation τ of the plane along n_γ (Figure 2.2(c)) is chosen such that the volume enclosed between the cell's boundaries and the plane equals f. For computational simplicity, this step is typically implemented by an iterative optimization. For advection in a 3D domain, dimensional splitting is employed, where the advection is performed subsequently in x-, y-, and z-direction. To determine the actual amount of f advected across the cell boundaries, the interface is cut by the downwind cell face at distance $x = \delta t u$ from the face (Figure 2.2(d)). It is apparent that a piecewise linear reconstruction is subject to C^0 and even C^{-1} discontinuities at the cell boundaries, i.e., between the PLIC patches.

The two-phase flow datasets investigated in this thesis were generated using the Free Surface 3D (FS3D) solver [60, 42] which employs the PLIC reconstruction for interface tracking and is parallelized using MPI [50] and OpenMP [129].

2.2 Flow Visualization

In this section, some visualization techniques are described that are directly related to the visualization methods presented in this work. Additionally, since the research in

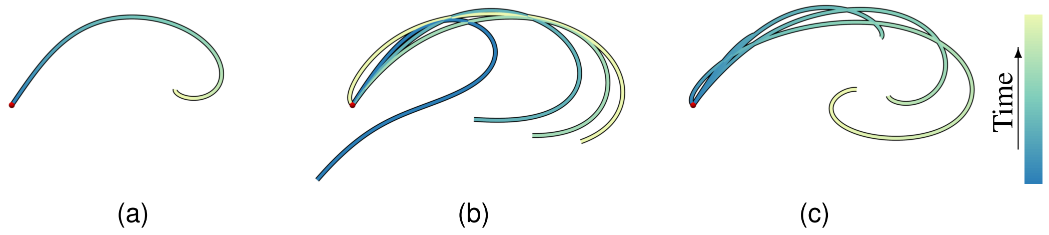


Figure 2.3: Integral lines with seed point marked red. (a) Pathline integrated over a given time interval (time for a given position color-coded as shown on the left). (b) A few streamlines seeded at different instants of time within the same time interval. (c) Snapshots of a single streakline at corresponding time instances.

interface reconstruction and feature tracking is closely related to this thesis, the state of the art is presented in Section 2.2.3 and 2.2.4, respectively. For general information on flow visualization, the reader is referred to extensive surveys on this topic [97, 117, 106].

2.2.1 Integral Lines

Integral lines are fundamental in the analysis of vector fields and are the basis for a large number of more complex flow visualization techniques. They are obtained by solving an initial value problem of an ordinary differential equation. There are three basic types of integral lines: pathlines, streamlines, and streaklines. Pathlines represent trajectories of massless particles advected by the flow. The differential and integral form of a pathline read as

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{u}(\mathbf{x}(t),t)\,, \quad \text{and} \quad \mathbf{x}(t) = \mathbf{x}_0 + \int_{t_0}^{t} \mathbf{u}(\mathbf{x}(\tau),\tau)\mathrm{d}\tau\,, \tag{2.12}$$

respectively, with initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$. Streamlines are curves which are tangent to an instantaneous vector field at every $\mathbf{x}$ and are obtained by integrating a particle path at a fixed time t_0:

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{u}(\mathbf{x}(t),t_0)\,, \quad \text{and} \quad \mathbf{x}(t) = \mathbf{x}_0 + \int_{t_0}^{t} \mathbf{u}(\mathbf{x}(\tau),t_0)\mathrm{d}\tau\,. \tag{2.13}$$

Streaklines are curves formed by a set of particles continuously released into a (time-dependent) vector field for a given time interval $\tau \in [t_0,t]$ from a fixed position $\mathbf{x}_0$. Evaluating the positions at time t provides a snapshot of the streakline parametrized by τ [8]:

$$\mathbf{x} = \mathbf{x}(\boldsymbol{\xi}(\mathbf{x}_0,\tau),t)\,, \tag{2.14}$$

with $\boldsymbol{\xi}(\mathbf{x}_0,\tau)$ representing the initial position of a particle that is located at position $\mathbf{x}$ at time t. A counterpart of a streakline in experimental visualization is a marker released into air or water from a seeding probe. It should be noted that there is a more generic definition of streaklines, the so-called generalized streaklines [196], where the seed is allowed to move over time. This concept accounts for a moving seeding probe in experimental visualization. Another useful integral line is the material line, or timeline, i.e., a curve formed by particles that were seeded at the same instant of time and move with the flow. It is worth noting that in steady vector fields, pathlines, streamlines and streaklines coincide. Please see Figure 2.3 for an illustration of these integral lines.

For visual representation, the ordinary differential equations (ODEs) of the integral lines must be solved numerically. The simplest but also the least accurate integration method is the forward Euler method which takes into account the velocity at the current position $\mathbf{p}_i$ and instance of time t_i, where $t_i = t_{i-1} + \Delta t$, to find the new position $\mathbf{p}_{i+1}$ after time step Δt:

$$\mathbf{p}_{i+1} = \mathbf{p}_i + \mathbf{u}(\mathbf{p}_i, t_i) \cdot \Delta t\,, \tag{2.15}$$

The standard method used in flow visualization is the fourth order Runge-Kutta method. It considers the changes in velocity as the particle moves with the flow using four intermediate steps:

$$\begin{aligned} k_1 &= \mathbf{u}(\mathbf{p}_i, t_i)\,, & k_2 &= \mathbf{u}(\mathbf{p}_i + \tfrac{1}{2}k_1 \cdot \Delta t, t_i + \tfrac{1}{2} \cdot \Delta t)\,, \\ k_3 &= \mathbf{u}(\mathbf{p}_i + \tfrac{1}{2}k_2 \cdot \Delta t, t_i + \tfrac{1}{2} \cdot \Delta t)\,, & k_4 &= \mathbf{u}(\mathbf{p}_i + k_3 \cdot \Delta t, t_i + \Delta t)\,, \end{aligned} \tag{2.16}$$

and the particle position is updated according to:

$$\mathbf{p}_{i+1} = \mathbf{p}_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \cdot \Delta t\,. \tag{2.17}$$

2.2.2 Features in Vector Fields

Features in vector fields are objects—usually in the form of points, lines or surfaces—that provide an abstract and meaningful representation of the flow [58]. They represent the flow behavior in a concise form that reveals the global flow structure and therefore, allow for efficient analysis of vector fields.

In the following, critical points and vortex core lines are described in detail. Other important flow features are separatrices (discussed in Section 5.3), vortex rings [135], bifurcation lines [111] and Lagrangian coherent structures [64].

Critical Points

For steady vector fields, critical points are important in the analysis of flow behavior, since the type of critical points provide general characteristics of the flow.

Critical points are isolated locations in the vector field where the velocity vanishes. The type of a critical point can be determined from the Jacobian of the vector field at that point. In 2D flow, given the Jacobian $\mathbf{J_u}$ in the form

$$\mathbf{J_u} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix}, \tag{2.18}$$

a critical point is of one of the following types, depending on the sign of the eigenvalues λ_1 and λ_2 as well as their real and imaginary parts:

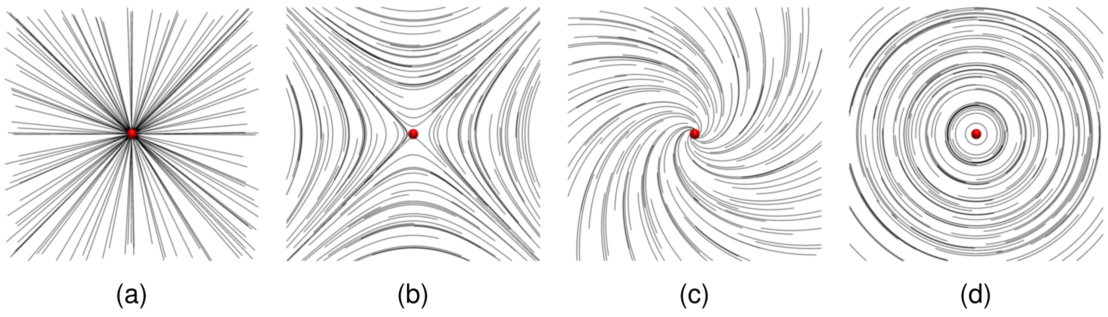


Figure 2.4: Critical points (red dots) with streamlines showing vector fields around them. Critical point types: (a) node (source or sink), (b) saddle, (c) focus (source or sink), and (d) center.

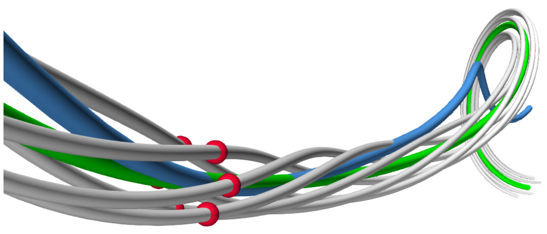


Figure 2.5: Vortex core line (blue) by Levy at al.'s criterion in a 3D steady flow. Streamline (green) started at the core line follows it for some time but then deviates from it. Streamlines (gray) started around core line (red points) go along the green one in a spiral motion as long as the green streamline follows the core line.

- Node source: $\lambda_1, \lambda_2 > 0$ and $\mathrm{Im}(\lambda_1), \mathrm{Im}(\lambda_2) = 0$
- Node sink: $\lambda_1, \lambda_2 < 0$ and $\mathrm{Im}(\lambda_1), \mathrm{Im}(\lambda_2) = 0$
- Saddle: $\lambda_1 \lambda_2 < 0$ and $\mathrm{Im}(\lambda_1), \mathrm{Im}(\lambda_2) = 0$
- Focus source: $\mathrm{Re}(\lambda_1), \mathrm{Re}(\lambda_2) > 0$ and $\mathrm{Im}(\lambda_1), \mathrm{Im}(\lambda_2) \neq 0$
- Focus sink: $\mathrm{Re}(\lambda_1), \mathrm{Re}(\lambda_2) < 0$ and $\mathrm{Im}(\lambda_1), \mathrm{Im}(\lambda_2) \neq 0$
- Center: $\mathrm{Re}(\lambda_1), \mathrm{Re}(\lambda_2) = 0$ and $\mathrm{Im}(\lambda_1), \mathrm{Im}(\lambda_2) \neq 0$

See Figure 2.4 for an illustration. It is worth noting that in case of incompressible fluid, only critical points of type center and saddle exist. In three-dimensional vector fields, there are eight types of critical point [58], however, they are not discussed further in this thesis.

Extraction of Vortices

In the analysis of 3D flow, important features are vortices, i.e., regions in fluid characterized by swirling motion of particles around an abstract curve called vortex core line. Vortices transport material at relatively long distances, and they usually exist for a long time. Additionally, they directly indicate the extent of turbulence in flow.

Since there is no general definition of a vortex, there exists a multitude of descriptions for vortex regions and vortex core lines. In steady flow, although a vortex core line must be tangent to streamlines, it is a common circumstance that it does not represent single streamlines. Instead, vortex core lines are composed of many streamline sections that successively pass through the core line [154]. In Figure 2.5 the green streamline passes through a vortex core line (blue), while the gray streamlines reveal the spinning motion around the core line. In steady flow, vortex core lines can be defined to consist locally of those streamline parts that exhibit minimum curvature (with the additional requirement of complex eigenvalues of the Jacobian). Accordingly, Sujudi and Haimes [170] defined vortex core lines to consist of those points where $\nabla \mathbf{u}$ exhibits a pair of complex eigenvalues and a real eigenvalue λ_R, and where $\mathbf{u}$ is parallel to its real eigenvector

$$(\nabla \mathbf{u})\mathbf{u} = \lambda_R \mathbf{u}\,. \tag{2.19}$$

This requirement is identical to $\mathbf{a} = \lambda_R \mathbf{u}$, i.e., the acceleration $\mathbf{a} := (\nabla \mathbf{u})\mathbf{u}$ being parallel to velocity, which requires the streamline passing through the respective point of the core line to be locally straight. A widely used vortex core line criterion is based on normalized helicity [38]. Normalized helicity h is a scalar field defined as the normalized dot product of velocity and vorticity, i.e.,

$$h := \frac{\mathbf{u} \cdot (\nabla \times \mathbf{u})}{(\|\mathbf{u}\| \, \|\nabla \times \mathbf{u}\|)}\,. \tag{2.20}$$

Vortex regions exhibit large $|h|$, whereas in non-vortical regions, e.g., in shear flow, $|h|$ is small. As described in the thesis of Roth [150], this directly leads to a vortex core line criterion defining those points as part of a core line where $\mathbf{u}$ is parallel to $(\nabla \times \mathbf{u})$. Another related vortex criterion is λ_2 [78]. It represents the medium eigenvalue of $\mathbf{S}^2 + \mathbf{\Omega}^2$, with $\mathbf{S} := (\nabla \mathbf{u} + (\nabla \mathbf{u})^\top)/2$ being the symmetric part of the Jacobian and $\mathbf{\Omega} := (\nabla \mathbf{u} - (\nabla \mathbf{u})^\top)/2$ its antisymmetric part. Vortex regions are indicated by negative values of λ_2. A further approach for obtaining vortex core line criteria is the extraction of valley lines or ridge lines from scalar vortex indicators. Sahner et al. [155] directly extracted valley lines from λ_2, while Schafhitzel et al. [160] employed isosurfaces for their topological definition.

2.2.3 Interface Reconstruction for Visualization

In this work, fluid interfaces are typically visualized using the standard marching cubes algorithm [109] that extracts an isocontour of the VOF-field. The approach is motivated by the fact that this surface extraction algorithm is commonly employed in the application domain.

Since the marching cubes algorithm operates on the node-based data, the cell-based VOF-grid must be converted to this representation before the extraction of the interface. This is done by defining each cell center $\mathbf{x}_{c_{i,j,k}} = (x_{c_i}, y_{c_j}, z_{c_k})$ as a node of a new node-based grid:

$$x_{c_i} = \frac{1}{2}(x_i + x_{i+1})\,, \quad y_{c_j} = \frac{1}{2}(y_j + y_{j+1})\,, \quad z_{c_k} = \frac{1}{2}(z_k + z_{k+1})\,, \tag{2.21}$$

where x_i, y_j and z_k are the cell coordinates along each axis of the rectilinear grid. See Figure 2.6 for an illustration.

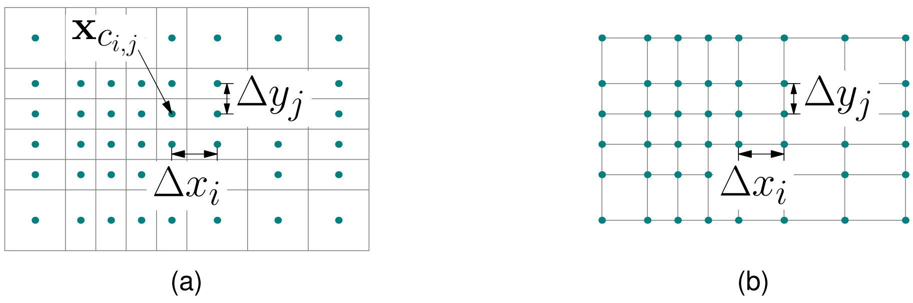


Figure 2.6: Conversion of data representation in rectilinear grids. (a) In a cell-based representation, data values are constant per cell. For conversion to a node-based representation, the data is associated with cell centers $c_{i,j}$ (blue dots). (b) In a node-based representation, data values are stored at grid nodes, and the grid is smaller than the cell-based configuration by half of the cell size on each grid side.

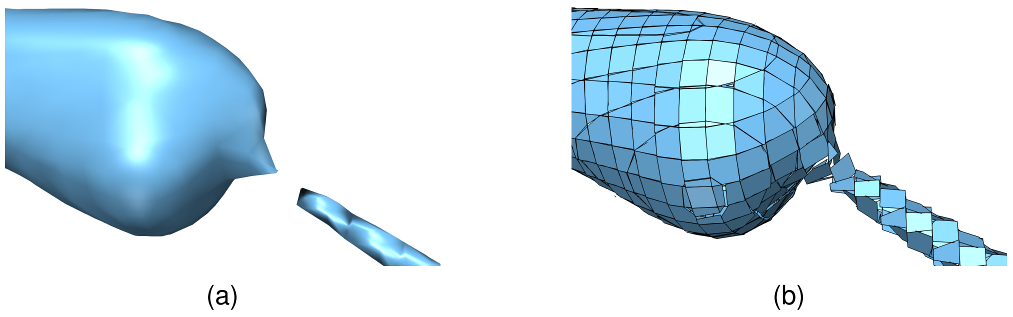


Figure 2.7: Interface reconstruction in two-phase flow. (a) Isocontour extraction with marching cubes algorithm. The algorithm gives smooth closed interfaces, but occasionally, as in this figure, it falsely separates the liquid volumes. (b) Although PLIC reconstruction is not smooth, the parts are connected, better reflecting the reconstruction used in the solver.

The isocontour extracted with the marching cubes algorithm does not necessarily have the same topology as the interface in the underlying solver (Figure 2.7). Therefore, in visualization methods introduced in this thesis for which the topology of phase components is important for the analysis, extraction of the PLIC patches for the visualization is used instead. For the generation of the PLIC patches for rendering, a custom algorithm based on marching cubes is employed. In this algorithm, the patches are extracted per cell such that the resulting planar isocontour has a normal parallel to $\mathbf{n}_\gamma$ and the volume enclosed by it corresponds to the f value in the cell (cf. Figure 2.2(c)). The details of this algorithm, which is also a contribution of this thesis, are given in Section 4.1.

Material interface reconstruction is a challenging topic in visualization, especially with respect to volume preservation in multi-material configurations, e.g., in multiphase flow, where more liquid components are considered. Bonnell et al. [20] described an algorithm that can reconstruct arbitrarily many interfaces within a single cell. However, the volume fractions enclosed within the reconstructed interfaces can deviate from the original ones. Meredith and Childs [118] developed a more accurate and smooth representation of interfaces with correct connectivity. The smoothness was also addressed by

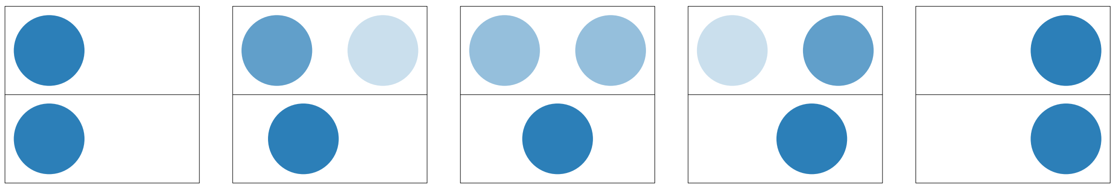


Figure 2.8: Interpolation of inclusion in Eulerian (top) and Lagrangian (bottom) frame. Five consecutive time steps shown.

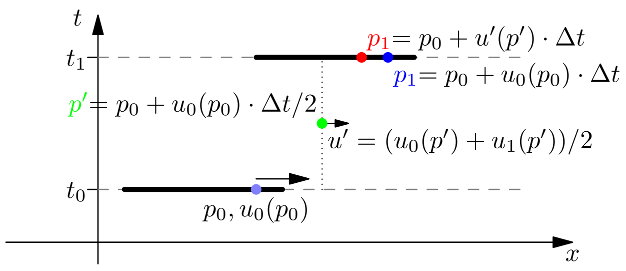
Anderson et al. [6] where smoothing and volumetric forces are applied to obtain high quality surfaces. In another work [5] they produced continuous interfaces across cell boundaries for time-varying and static data in arbitrary dimension with bounded error. Obermaier et al. [125] analyzed the stability of reconstructed interfaces by comparing with time surfaces. For algebraic surfaces, Mann [114] improved the convergence of A-patches that are used to accurately tessellate surface representation using triangle meshes. This method, however, is only suitable for implicit surfaces (of polynomial form). On the other hand, Wojtan et al. [204] developed a mesh-based surface tracking that preserves thin liquid layers. They use convex hulls of liquid in cells with complex surfaces to ensure consistent topology. As in the case of marching cubes, this approach does not preserve the fluid volume.

Interfaces are crucial for the analysis of two-phase flow. However, it is the interfaces' inherent influence on the fluid dynamics, and not its visual representation that is the focus of this thesis. Therefore, development of smooth interface reconstruction as well as reconstruction that can handle multiple fluids are not the main goals of this work, and are not pursued in the presented techniques.

The presented visualization methods related to two-phase flow are applicable to simulations of liquid dispersed in gas phase (i.e., droplets), and liquid phase dispersed in another liquid phase. Therefore, for brevity, the dispersed phases are interchangeably referred to as *inclusions* or *phase components*.

2.2.4 Inclusion Tracking

The visualization techniques presented in Chapter 3 need to track time-dependent evolution of droplets, either to detect and visualize droplet splits and merges or to provide detailed segmentation of the droplet volumes into regions that separate in the course of time. This necessitates finding correspondences between the inclusions at consecutive time steps, and in scientific visualization, the correspondence problem is approached using feature tracking methods. Different techniques have been proposed so far that track features in single-phase flow. Reinders et. al [144] employed attribute correspondence using different criteria to match features, e.g., turbulent vortex structures. Sauer et al. [159] utilized particle and volume data from the simulation runs to track features over longer time intervals. The correspondence problem has also been addressed for applications in computer graphics. Stam [167] proposed a method for the computation of



t
t_1
$p_1 = p_0 + u'(p') \cdot \Delta t$
$p_1 = p_0 + u_0(p_0) \cdot \Delta t$
$p' = p_0 + u_0(p_0) \cdot \Delta t/2$
$u' = (u_0(p') + u_1(p'))/2$
t_0
$p_0, u_0(p_0)$
x

Figure 2.9: Particle advection in two-phase flow. In case of low temporal resolution of the discretized velocity field, higher-order integration schemes can lead to erroneous results when tracking inclusions (black lines), since they interpolate the vector field in time (at green dot). This can cause interpolation between, e.g., fast liquid and slow air, and result in inconsistent tracking (red dot). Explicit Euler integration gives consistent result in this case (blue dot).

interface velocities to properly translate surfaces. Bojsen-Hansen et al. [18] developed a method for tracking surfaces undergoing topology changes, without prior information on the underlying physics. While these methods operate on surfaces, others find correspondences in general cases, such as image processing and material reflections properties. In the work by Bonneel et al. [19], Lagrangian transport was used to correctly interpolate the displacement between two known corresponding states. Solomon et al. [165] optimized the transportation in terms of Wasserstein distances, which allows for efficient shape transformation. Please see Section 3.1.1 and 3.2.1 for works on feature tracking more closely related to the presented visualization methods.

A problem inherent to the Eulerian representation of phases in two-phase flow simulations is that temporal interpolation does not provide physically correct results. This is illustrated in Figure 2.8, where on the top, temporal interpolation on a Eulerian grid is applied to obtain the transition between consecutive time steps. As can be seen, the initial left inclusion fades out while the one from the next simulation step fades in. However, there should be a movement between these two states, as shown in the bottom. Therefore, temporal interpolation of the f-field would not be an appropriate method for tracking. This problem implies that the correspondence should be found using a Lagrangian approach, which is in fact the method used in this thesis. While the details are given in the respective sections, it should be mentioned here that in the presented methods, particle advection is employed to find inclusion correspondence. In the first method, which requires the identification of features at consecutive time steps as well as possible events, explicit Euler integration provides a good trade-off between accuracy and computational complexity. Due to low temporal resolution in the investigated datasets, higher-order methods involved temporal interpolation and hence did not provide better results. This is because the involved temporal interpolation can result in incorrectly sampling the velocity in the gas phase, as illustrated in Figure 2.9. In the second method, which requires more accurate determination of correspondence (i.e., volumetric contributions), 4-th order Runge-Kutta provided better results due to better temporal resolution of the data used.

Visualization of Interactions in Droplet Groups | 3

Visual analysis of large groups of droplets and jets is a challenging task due to constant topology changes and visual clutter caused by fluid interfaces. Moreover, for large datasets, the extraction of important information is hindered by the substantial amount of data. In this chapter, two visualization techniques are presented that enable the analysis of such data, both in terms of the dynamic processes leading to topology changes and detailed investigation of the spatio-temporal evolution of these changes. Together, these two methods provide a novel approach for the investigation of droplet interactions that leads to better understanding of droplet groups and their dynamics.

The first method enables the investigation of complex two-phase flow phenomena that lead to coalescence and breakup of droplets [83]. It adapts dimensionless quantities for a localized investigation of phase instability and breakup. Additionally, with the employed principal component analysis of droplets, the method provides detailed inspection of breakup dynamics with emphasis on oscillation and its interplay with rotational motion. For an effective interactive representation of the overall dynamics, a space-time graph representation of droplets is combined with traditional 3D visualization in a highly interactive linked-view approach.

While the first method mainly concentrates on the analysis of the processes that induce topology changes, the second method focuses on the topology changes themselves by providing a spatio-temporal visualization of liquid separation. In this method, liquid volumes at some reference time step are segmented into regions that separate into distinct components in later time steps. It employs particle-based tracking to find volumetric correspondences between inclusions at two different instants of time. The segmentation is visualized by mesh boundaries generated around each volume segment of an inclusion at the initial time that correspond to the separated inclusions at the later time. The approach is complemented with spatio-temporal separation surfaces that convey the temporal evolution of the partitioning. For phase-consistent particle trajectories, a multi-stage corrector method is introduced.

3.1 Visual Analysis of Inclusion Dynamics in Two-Phase Flow

A research question central to two-phase flow problems is how and why topological changes of the phase components occur. The dynamics of breakup and coalescence are subject to several physical mechanisms, most important flow instability, centrifugal forces, oscillation, and surface tension.

In the presented technique, these mechanisms are visualized with specially tailored techniques that focus on the deformation and rotating motion of inclusions, topological changes of the interfaces, as well as the interplay of these different dynamics. Flow instabilities are visualized with a derived droplet-localized version of the Reynolds number. The analysis of angular momentum conveys rotational motion which is an important factor in the dynamics of droplets and the ligaments they typically originate from. Since rotation is often superimposed with oscillation and translation, a visual representation of combined rotation and oscillation based on principal component analysis (PCA) is provided in the form of ribbons. The related co-rotating frames of reference visualize droplets temporally, irrespective of their rotation and translation. Both the ribbons and the co-rotating camera allow for a detailed analysis of the oscillation of rotating phase components. Furthermore, analysis of inclusion oscillations by means of the Fourier transform is introduced, and a three-dimensional representation of their spectra is provided, which conveys both the frequency components and phases of oscillation. To enable effective interaction and integrative analysis, the techniques are integrated in a linked view approach, consisting of a 3D view and a 2D graph view. [1]

3.1.1 Related Work

A closely related field of research is feature tracking. Post et al. [141] provided a survey on this topic. Different approaches have been proposed so far. Reinders et al. [144] introduced a method where feature correspondence in successive frames is detected to analyze the evolution of features. Sauer et al. [159] combined particle and volume data to track features. These, however, operate in single-phase flow where interpolation in time and thus advanced particle tracing can be employed. The presented two-phase flow configuration, however, necessitates development of a special variant for tracking inclusions, as discussed in the Fundamentals (Chapter 2). Further research on feature tracking concentrates on clustering methods, including the work by Ozer et al. [130], where user-defined feature characteristics are used to determine feature groups.

A related work is the visual analytics of streamlines and pathlines using a graph representation that clusters field lines for better visual exploration [110]. Also related is the visualization of mixing processes and instabilities by Laney et al. [96]. However, both methods address single-phase flow, and thus do not need to consider the interface between phases. Similarly, merge trees on space-time isovolumes and adaptive thresh-

[1] Parts of this section have been published in: [83]

olds to track features in combustion simulations were proposed [195]. However, neither temporal interpolation (as discussed below) nor parameter thresholding can be applied in two-phase flow.

Brushing and linking is commonly used to enable abstraction of scientific data. Doleisch and Hauser [40] used brushing, based on non-discrete degree of interest functions in parametric views to select regions of interest in 3D simulation domains. Bremer et al. [24] proposed a hierarchy-based approach that alleviates the dependency on predefined thresholds. This approach was later extended and embedded into a tool with interactively linked views [25]. Gu and Wang [63] computed time-dependent state transition probabilities for volumetric data and visualized a 3D view of the volume together with a 2D graph representation of the transitions. They also used brushing and linking to connect the two views. Similarly, Jänicke and Scheuermann [82] constructed finite-state machines encoding the evolution of flow, and depicted these along with the main flow visualization. Grottel et al. [62] visualized the evolution of molecular clusters on a timeline as an addition to a 3D representation of the molecules to monitor the quality of the clustering. Preston et al. [142] developed an interactive visualization system for cosmology data based on the linked view approach.

Topological methods for vector field visualization [68, 139] do not overlap with this work, since here the focus is on the analysis of two-phase flow in terms of the *geometric* topology of *interfaces*.

In the presented technique, space-time connectivity is presented as a graph and visualized as a node-link diagram to convey changes in geometric topology of interfaces. Graph representation is a commonly employed approach to convey temporal evolution of features, e.g., in the analysis of combustion simulations [195] or combustion experiments [148], and the features can be abstracted by glyphs [144]. For graph layouting, a specialized variant of a hierarchical graph layout—the Sugiyama layout [169]—was employed. In general, branching (and merging) link structures that encode the flow or transition of quantities are known as flow maps [137] or Sankey diagrams [147]; these general concepts, however, do neither include layers nor encode time-dependent data. An overview of time-oriented data visualization is given by Aigner et al. [3], whereas a survey on graph-based visualization of scientific data is provided by Wang [184].

3.1.2 Phase Tracking in Two-Phase Flow

To visualize topology changes of phase components, they must be tracked over time. First, volumetric connected components of one of the phases (here the liquid phase) are determined by region growing, separately in each time step. Face connectivity is considered for $f_i > 0 \wedge f_j > 0$ (f_i being the f-value of cell c_i). To track the resulting components C_k over time, a volumetric approach is followed based on particle advection, similar to the one proposed by Sauer et al. [159]. Here, however, temporal interpolation of the vector field should be avoided due to the two-phase property of the flow and relatively low temporal resolution of the simulation output, as discussed in Chapter 2. Specifically, higher-order integration schemes (such as 4th-order Runge-Kutta) involve

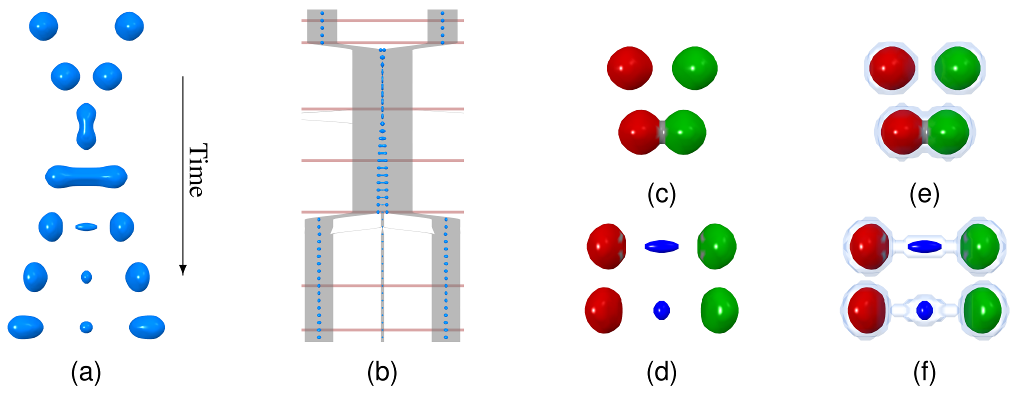


Figure 3.1: *Centric Collision* dataset consisting of two colliding droplets, with time increasing downward. (a) Spatial representation of the interface, at selected time steps. (b) Corresponding space-time graph G. Selected time steps marked with red lines. (c), (e) Tracked phase components before and after merge, and (d), (f) before and after split, using (c), (d) single isosurfaces, and (e), (f) additional isosurface for conveying connectivity. © 2017 IEEE.

interpolation in time and hence, intermediate integration steps might sample the velocity field outside the inclusion (i.e., in the gas phase), leading to incorrect advection. Therefore, a particle seeded at simulation time step t_l at the center $\mathbf{x}_i$ of each cell c_i in C_k is advected to the next time step t_{l+1} by an explicit Euler step

$$\mathbf{x}(t_{l+1}) = \mathbf{x}_i + \Delta t \mathbf{u}(\mathbf{x}_i, t_l) \,, \tag{3.1}$$

with $\Delta t = t_{l+1} - t_l$, and tested if cell c_j that contains $\mathbf{x}(t_{l+1})$ exhibits $f_j > 0$. If not, the particle does not contribute correspondence information. If, on the other hand, $f_j > 0$, then the correspondence is verified in reverse direction, i.e.,

$$\mathbf{x}(t_l) = \mathbf{x}(t_{l+1}) - \Delta t \mathbf{u}(\mathbf{x}(t_{l+1}), t_{l+1}) \tag{3.2}$$

is computed and it is tested if $\mathbf{x}(t_l)$ is located in a cell c_m where $f_m > 0$. If this is the case and $c_m \in C_k$, the respective correspondence is stored, otherwise it is rejected. For symmetry, the process of Equations 3.1 and 3.2 is carried out also in reverse direction, with particles started from the cell centers. Due to the limited accuracy of the explicit Euler integration scheme, some of the small components typically cannot be tracked. This is caused by the fact that in such cases, a significant number of particles lies in the interface cells, where the velocity of the gas and liquid phases is interpolated. In practice, however, those droplets are negligible due to their limited influence on the overall simulation.

3.1.3 Linked-View Visualization

Visualization of two-phase flow dynamics is a complex problem, since the involved interface poses an additional challenge in the identification of flow characteristics. The introduced set of tightly interrelated building blocks is aimed at tackling this problem by extracting and visualizing each of the potential flow processes.

The core of the visualization technique is a linked 3D spatial view and a 2D graph view. Figure 3.1 presents this basic approach. In Figure 3.1(a), 3D views at chosen simulation time steps are stacked, with time increasing downward. In Figure 3.1(b), corresponding time steps are marked in the inclusion connectivity graph, denoted G.

Several quantities are visualized to enable comprehensive analysis of inclusion dynamics. An important factor in droplet dynamics is rotation because of the centrifugal force involved, which can eventually lead to breakup and increased momentum during collision. The rotation is visualized by PCA-based ribbons, whose twist reveals rotation, and by mapping drop momentum to the drop interface. PCA is also used for spectral analysis of deformation. Namely, the deformation frequencies can be mapped on the edges of the space-time graph, and the amplitude of the strongest frequency in the spectrum can be color-coded on the interface of the respective phase component. For a detailed inspection of the spectrum of a phase component, oscillation glyphs visualize each frequency in the spectrum by time-varying ellipsoids. Since it is particularly difficult to observe deformation when it is superimposed by rotation, a virtual camera that "rotates with the phase component" is achieved with the help of the eigenvectors of the PCA which determine its frame of reference. Additionally, a localized version of the Reynolds number for the analysis of phase instabilities, as well as an approach for determining splash/non-splash characteristics of inclusions are presented. To provide a space-time overview of the topological changes, and as a basis for visualizing derived quantities, the connectivity graph is displayed as a node-link diagram, with each time instance of a phase component represented by a node, and each correspondence over time by a link. For the analysis of large datasets, node clustering is employed in the graph view. The following sections describe these approaches in detail.

Phase Interface Visualization

Visualization of the interface in two-phase flow is typically accomplished in the application domains by isosurface extraction at isolevel $c_f = 0.5$ (Figure 3.1(a)). This has the advantage that continuous surfaces are obtained, but it involves severe shortcomings regarding volume determination and interface topology: an isolevel $c \ll 1$ would convey the correct topology in terms of the discretization of the f-field, but would give a too large volume. To account for the somewhat contradicting requirements, a twofold approach is followed where isosurfaces at $c_f = 0.5$ support interpretation methodology from the application domains (Figure 3.1(c) and (d)), and auxiliary transparent isosurfaces at isolevel $c_f \ll 1$ (here, $c_f = 10^{-6}$) convey the connected components defined by the f-field (Figure 3.1(e) and (f)). This approach provides a good approximation of the bounds the phase interface lies within.

A straightforward approach for analyzing the events between adjacent time steps is to assign each space-time component an individual color and visualize them using the isosurface of the f-field (Figure 3.1(c)–(e) and (d)–(f)). This gives a direct picture of the

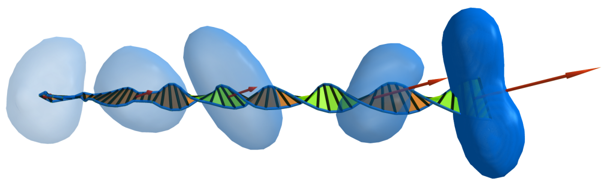


Figure 3.2: PCA ribbon (green: front, orange: back) for a droplet displayed at different time steps (increasing opacity). The PCA ribbon reveals initial oscillation (left) and rotation (right), shown with angular momentum vector **L** by red arrow glyphs. © 2017 IEEE.

split/merge processes. Regions that cannot be tracked, i.e., where the advected particles are located in regions with $f = 0$, are colored gray on the isosurface.

Visualization of Rotation

The problem of visualizing rotation of inclusions in two-phase flow is approached by several means. To support traditional physics-based reasoning, the angular momentum $\mathbf{L}$ is evaluated for each component C_k as

$$\mathbf{L}_{C_k} = \sum_{c_i \in C_k} f(\mathbf{x}_i) \cdot (\mathbf{x}_i - \mathbf{x}_{C_k}) \times (\mathbf{u}(\mathbf{x}_i) - \mathbf{u}_{C_k}), \tag{3.3}$$

with $\mathbf{x}_{C_k} = \sum_{c_i \in C_k} f(\mathbf{x}_i) \cdot \mathbf{x}_i$ the centroid of C_k, $\mathbf{u}_{C_k} = \sum_{c_i \in C_k} f(\mathbf{x}_i) \cdot \mathbf{u}(\mathbf{x}_i) / \sum_{c_i \in C_k} f(\mathbf{x}_i)$ the velocity of $\mathbf{x}_{C_k}$, and $\mathbf{x}_i$ the center of cell c_i. For visualizing rotation (and deformation, described below) the phase component C_k must not include splits or mergers. Hence, G is split at these events and the remaining component sequences, which do not exhibit splits or mergers, represent phase components C_k for the respective time intervals. Figure 3.9(a) shows phase components at an instant of time, color-coded with the magnitude of angular momentum. This provides direct notion on how much rotational motion is comprised by the individual components. To provide the rotation axis and a more quantitative representation in general, an arrow glyph representing $\mathbf{L}$, centered at $\mathbf{x}_{C_k}$ (Figure 3.2) can be displayed.

To convey dynamics of phase components, a PCA-based abstraction, termed PCA ribbons, is introduced. As compared to traditional arrow glyphs, the PCA ribbons avoid visual clutter and therefore allow for easier interpretation. The PCA ribbons are constructed as follows. For each time step, a principal component analysis is performed for a phase component C_k of the distribution $f(\mathbf{x}_i) \cdot (\mathbf{x}_i - \mathbf{x}_{C_k})$, with $c_i \in C_k$. From the resulting PCA eigenvectors $\boldsymbol{\varepsilon}_{k,j}$ and corresponding eigenvalues $\lambda_{k,j}$, with $j \in \{1,2,3\}$, frames of reference aligned with the shapes are derived which rotate with the phase components. From the three eigenvectors, one that ensures consistency between simulation time steps is chosen. To this end, the most different PCA eigenvector is determined from the others, i.e., assuming $\lambda_{k,1} \geq \lambda_{k,2} \geq \lambda_{k,3}$, the main PCA axis (or axis of "rotational symmetry") is the eigenvector $\boldsymbol{\varepsilon}_{k,m}$ corresponding to $\lambda_{k,m}$ with

$$m = \begin{cases} 1 & \text{if } \lambda_1 - \lambda_2 > \lambda_2 - \lambda_3\,, \\ 3 & \text{otherwise.} \end{cases} \tag{3.4}$$

Thus, in the time-dependent context, $\boldsymbol{\varepsilon}_{k,m}$ typically is the direction of strongest oscillation of a phase component. By simply visualizing $\lambda_{k,m} \cdot \boldsymbol{\varepsilon}_{k,m}$ with a straight line segment centered at $\mathbf{x}_{C_k}$ for all time steps and connecting them by a ribbon (i.e., meshing temporally adjacent line segments into minimum-twisted quads, with different front and back color), a concise representation of oscillation and rotation is obtained (Figure 3.2). Please note that, for brevity, the subscript m for $\boldsymbol{\varepsilon}_{k,m}$ is dropped in the remaining text.

One remaining issue is that in cases where two or more PCA eigenvalues are similar, the respective eigenvector directions become unstable (and even undefined, if $\lambda_1 = \lambda_2$ and/or $\lambda_2 = \lambda_3$). To avoid such outliers in $\boldsymbol{\varepsilon}_k$, and hence in the PCA ribbon representation, these unstable cases must be suppressed. This is achieved by obtaining more certain information from neighboring time steps in such configurations. To this end, first, for each $\boldsymbol{\varepsilon}_k(t_l)$ at time t_l a quality measure $q_l = \max(\lambda_1(t_l) - \lambda_2(t_l), \lambda_2(t_l) - \lambda_3(t_l))/\lambda_1(t_l)$ is derived, and then employed to obtain $\overline{\boldsymbol{\varepsilon}}_k$, the stabilized equivalent to $\boldsymbol{\varepsilon}_k$:

$$\overline{\boldsymbol{\varepsilon}}_k(t_l) = q_l \cdot \boldsymbol{\varepsilon}_k(t_l) + (1 - q_l) \sum_{i=l-1}^{l+1} q_i \boldsymbol{\varepsilon}_k(t_i) / \sum_{i=l-1}^{l+1} q_i \, . \tag{3.5}$$

Thus, $\overline{\boldsymbol{\varepsilon}}_k$ is used to construct the PCA ribbons, instead of $\boldsymbol{\varepsilon}_k$. One limitation of the approach is that for very fast rotating droplets it can happen that the droplet rotation between two consecutive time steps is larger than a quarter of revolution, leading to an underestimation of the rotation. As with tracking issues, this problem stems from insufficient time resolution of a dataset. One could, for instance, examine the vector field to infer the rotation direction. However, the limited temporal resolution would not ensure that the actual rotation would be captured. Favorably, in cases where the method assumes incorrect rotation, it is discernible as a discontinuity in the ribbon. Another limitation is that for slowly moving rotating inclusions the resulting ribbon is short and therefore self-occlusion can arise. Such cases, however, are relatively seldom, since they can occur only under certain conditions (e.g., when two droplets with similar linear momentum collide off-center).

Visualization of Deformation

To isolate the deformation from rotation, the co-rotating camera is applied which is transformed according to the rotation of the inclusion. Specifically, for a tracked component C_k, $\overline{\boldsymbol{\varepsilon}}_k(t_l)$ and $\overline{\boldsymbol{\varepsilon}}_k(t_{l+1})$ are taken from adjacent time steps t_l and t_{l+1}, and the rotation that transforms $\overline{\boldsymbol{\varepsilon}}_k(t_l)$ into $\overline{\boldsymbol{\varepsilon}}_k(t_{l+1})$ is computed. This transformation is obtained from the cross product of these eigenvectors which is interpreted as a rotation vector $\mathbf{r}$. A rotation matrix is constructed that rotates in the opposite direction by the determined angle. Finally, this matrix is multiplied with the model-view matrix to compensate that rotation, i.e., to obtain a co-rotating camera. In this mode, the camera can be navigated as usual, with the only difference that the phase component is not rotating in the resulting time-dependent exploration. Figure 3.3(a)–(e) and (f)–(j) gives a comparison between a standard view and a co-rotating view of a rotating droplet over time.

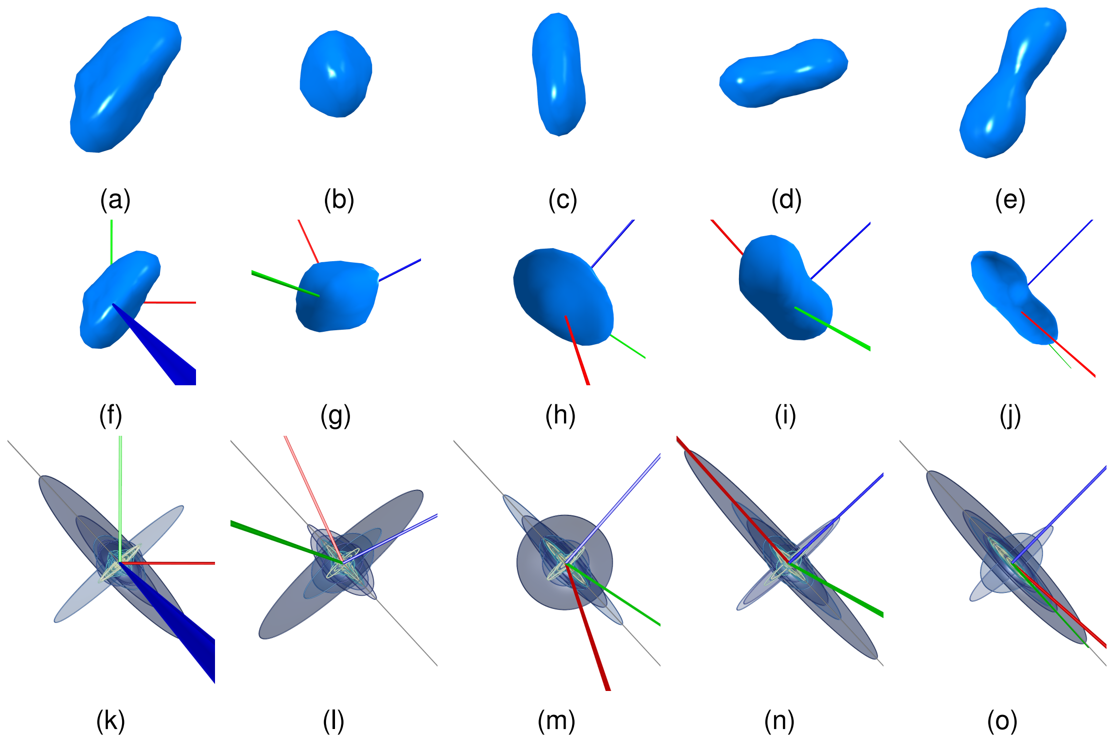


Figure 3.3: Selected phase component of *Peripheral Collision* dataset, time steps from Figure 3.2. (a)–(e) Droplet in standard view. (f)–(j) Co-rotating camera reveals droplet oscillation. (k)–(o) Respective oscillation glyphs show oscillation spectrum, including amplitude and phase. Gray thin axis shows PCA eigenvector $\overline{\boldsymbol{\varepsilon}}_k$. © 2017 IEEE.

Since deformation of phase components typically exhibits complex dynamics, advanced methods are necessary to extract meaningful information. Therefore, oscillation is analyzed primarily in terms of PCA, i.e., in direction of the PCA vector $\overline{\boldsymbol{\varepsilon}}_k$ which is the direction in which droplets typically oscillate with the largest amplitude. To separate oscillation from other types of deformation, spectra of deformation in direction of $\overline{\boldsymbol{\varepsilon}}_k$ are considered.

Since λ_k represents the "oscillating" half axis of the approximating PCA ellipsoid, spectral analysis of the oscillation can give more insight into the droplet deformation. Hence, $\lambda_k(t_l)$ with $l = m, \ldots, n$ is transformed from the time domain to the frequency domain using the discrete Fourier transform (DFT) [53], resulting in a discrete spectrum $\Lambda_k(\nu_a)$ with frequencies ν_a with $a = m, \ldots, n$. Visualization of $\Lambda_k(\nu_a)$ is achieved by mapping frequency bands of $\Lambda_k(\nu_a)$ to vertical bands on the edges of the graph G, with frequency increasing to the right. This is demonstrated in Figure 3.8(b)–(d). For overview, the amplitude $\Lambda_k(\nu_{\max})$ of the strongest frequency $\nu_{\max}$ in the spectrum is mapped to color on the interface of the respective phase component (Figure 3.9(b)). For a detailed inspection of deformation dynamics, an oscillation glyph is introduced, which consists of one ellipsoid for each frequency ν_a in the spectrum of a phase component. The lengths of the ellipsoid axes are scaled by $\Lambda_k(\nu_a)$ and the aspect ratio of the two

Algorithm 1 Construction of oscillation glyphs.

```
Input: $C_k$, eigenvectors $\lambda_k(t_l)$, $l = m, \dots, n$
Output: Oscillation glyph at time step $t_i$ for $C_k$
  $t_m, t_n \leftarrow$ first and last occurrence of $C_k$
  for each frequency $\nu_a \in \mathrm{DFT}(\lambda_k(t_l))$ do
      $\mathbf{a} \leftarrow (1,0,0) \cdot \Lambda_k(\nu_a)$
      $s \leftarrow \cos(\phi_k^i(\nu_a)) + 1 + \sigma$
      $\mathbf{b} \leftarrow (0,1,0) \cdot \Lambda_k(\nu_a) \cdot s$
      $\mathbf{c} \leftarrow (0,0,1) \cdot \Lambda_k(\nu_a) / s$
      $\mathbf{r} \leftarrow (1,0,0) \times \overline{\boldsymbol{\varepsilon}}_k(t_m) / \|\overline{\boldsymbol{\varepsilon}}_k(t_m)\|$
      $R_{\boldsymbol{\varepsilon}} \leftarrow$ rotation matrix about $\mathbf{r}$ by angle $\alpha = -\arcsin(\|\mathbf{r}\|)$
      $\mathbf{a} \leftarrow R_{\boldsymbol{\varepsilon}}\mathbf{a}$, $\mathbf{b} \leftarrow R_{\boldsymbol{\varepsilon}}\mathbf{b}$, $\mathbf{c} \leftarrow R_{\boldsymbol{\varepsilon}}\mathbf{c}$
      draw half-ellipsoid with semi-principal axes $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$
  end for
```

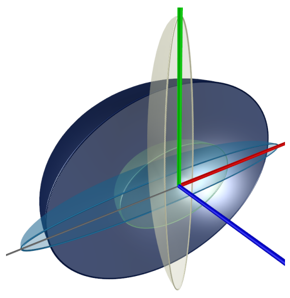


Figure 3.4: Example oscillation glyph with four ellipsoids corresponding to four frequencies ν_a. For each ellipsoid, the axis aligned with its depth is scaled by the oscillation amplitude, and the lengths of the two other axes are determined from the cosine of the oscillation phase.

smaller axes is determined from the cosine of phase $\phi_k^i(\nu_a) = \phi_k(\nu_a) + 2\pi\nu_a(t_i - t_m)$, with $m \leq i \leq n$.

Algorithm 1 shows the construction of the oscillation glyph for a given C_k at a given time step t_i. To ensure that the scaling of axes **a** and **b** is always positive (and hence the axes do not change directions), the scaling factor s is offset by $1 + \sigma$ where $\sigma \ll 1$. Additionally, the ellipsoids are transformed with a rotation matrix $R_{\boldsymbol{\varepsilon}}$ that rotates the axes by angle $\alpha = -\arcsin(\|\mathbf{r}\|)$ around a vector $\mathbf{r} = (1,0,0) \times \overline{\boldsymbol{\varepsilon}}_k(t_m)/\|\overline{\boldsymbol{\varepsilon}}_k(t_m)\|$ such that **a** is aligned with the initial eigenvector $\overline{\boldsymbol{\varepsilon}}_k(t_m)$ to ensure consistency with the co-rotating camera. The resulting time-dependent glyph, demonstrated in Figure 3.3(k)–(o), can be continuously investigated at sub-time step resolution with the co-rotating camera. See also Figure 3.4 for a more detailed illustration of the oscillation glyph.

Derived Quantities

Droplet-Localized Reynolds Number The Reynolds number is generally defined as $\mathrm{Re} = \|\mathbf{u}\| \cdot d/\nu$, with diameter d of the structure under investigation and kinematic viscosity ν of the fluid. It is widely used to judge the overall characteristics of a flow, e.g., whether the flow is rather laminar or turbulent. In two-phase flow, the Reynolds number characterizes flow instability, e.g., the Reynolds number of the gaseous surrounding is used to determine whether a given droplet will disintegrate. In such configurations, d represents the diameter of the droplet. Therefore, to determine Re, the

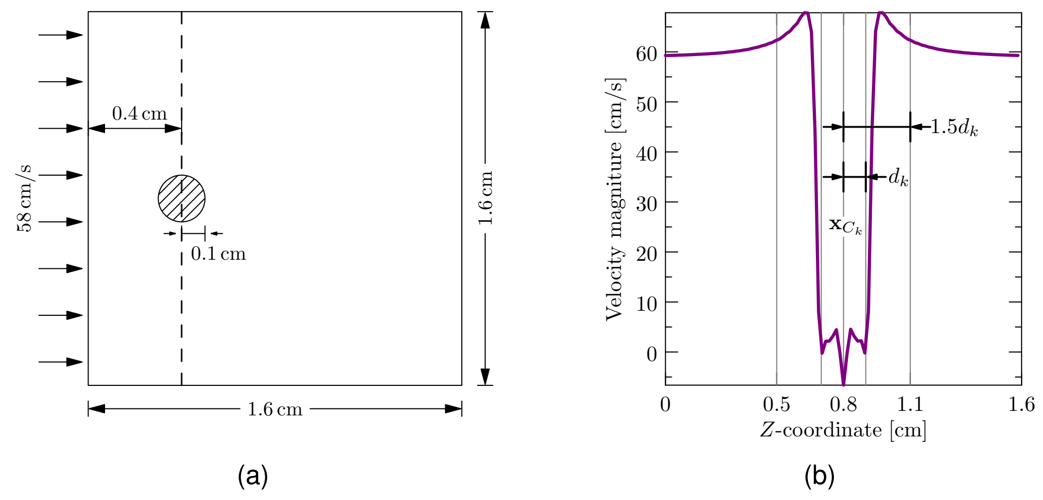


Figure 3.5: Investigation of proper stream velocity for flow with an inclusion. (a) Example configuration for the analysis of velocity around a component (hatched). (b) Velocity profile for the dashed line in (a). Taking the velocity at distance $1.5d_k$ from $\mathbf{x}_{C_k}$ (outer vertical lines), where d_k is the diameter of a sphere with equal volume, provides a good approximation to stream velocity. © 2017 IEEE.

diameter of a sphere with droplet equivalent volume is computed. That is, for the volume $V_k = \sum_{c_i \in C_k} f_i V_i$ of the liquid phase component C_k (with V_i being the volume of cell c_i), the diameter equals $d_k = 2\sqrt[3]{3V_k/(4\pi)}$. The kinematic viscosity of the liquid phase is used to compute Re with d_k.

One open problem is, however, how $\|\mathbf{u}\|$ should be obtained. In traditional fluid mechanics problems, $\|\mathbf{u}\|$ is taken to be the velocity of the free stream. The conducted studies on different two-phase flow configurations with mutual interference led to a conclusion that $\|\mathbf{u}\|$ should be evaluated on a plane normal to $\mathbf{u}_{C_k}$ at distance $1.5d_k$ from $\mathbf{x}_{C_k}$. Figure 3.5(a) shows the simulation setup for the test on the influence of the liquid phase on the gas velocity, and Figure 3.5(b) shows the resulting velocity profile at the dashed line. Therefore, $\mathbf{u}$ is evaluated on a circle centered at $\mathbf{x}_{C_k}$ with normal $\mathbf{u}_{C_k}$, employing $n = \max(16, \pi\hat{d}_k)$ samples, with

$$\hat{d}_k = d_k / \max_{c_i \in C_k} \sqrt[3]{V_i} \tag{3.6}$$

and averaging the samples, providing the droplet-localized Reynolds number

$$\mathrm{Re}_{C_k} = \sum_{\mathbf{x} \in circle} \|\mathbf{u}(\mathbf{x}) - \mathbf{u}_{C_k}\| \cdot d_k/(n\nu)\,. \tag{3.7}$$

Splash/Non-Splash Criterion Regarding a droplet impacting onto a surface, its dynamics can be described in relation to the splashing boundary which predicts whether the droplet will completely deposit on the surface or splash. The splashing boundary is typically shown in regime maps [183] as a function of the Reynolds number and the

Ohnesorge number defined as $\mathrm{Oh} = \mu/\sqrt{\rho\sigma d}$, where ρ is the liquid density, and σ is the surface tension. Up to now, a splashing boundary can only be obtained from experiments where free velocity for a single droplet is available for Reynolds number computation. Splashing phenomena are hard to predict numerically even with highly-resolved droplets. The resolution of the droplets resulting from a large jet breakup simulation, as examined below, is not yet able to provide necessary accuracy. Hence, it is reasonable to calculate the dimensionless numbers and to apply the predictive correlations to the droplets from jet breakup simulations in order to gain insights into the development of the droplets after primary breakup. Following the splash/non-splash criterion determined by Vander Wal et al. [183], the Ohnesorge number is computed with density ρ and surface tension σ provided with the simulation run, and the diameter d_k of the droplet. Droplets with the corresponding point $(\mathrm{Re}_{\mathrm{C_k}}, \mathrm{Oh})$ lying above the curve $\mathrm{Oh} = 63/\mathrm{Re}_{\mathrm{C_k}}{}^{1.17}$ belong to the splash regime and are color-coded green, whereas those below the curve are in non-splash regime and hence black, as demonstrated in Figure 3.14(a).

Interface Curvature Surface tension participates in stabilizing phase components, but also in causing breakup once they have sufficiently deformed. Surface tension is computed in two-phase flow solvers from the curvature of the interface. The mean curvature is evaluated directly from the f-field by the following expression:

$$\kappa = \frac{1}{2}\left(\lambda_a(A) + \lambda_b(A)\right), \tag{3.8}$$

with $\lambda_a(A)$ and $\lambda_b(A)$ being the eigenvalues corresponding to the eigenvectors tangent to the interface, and $A = \nabla(\nabla f/\|\nabla f\|)$, using central differences. These eigenvalues are found by excluding the third eigenvalue whose corresponding eigenvector is most parallel to ∇f. Figure 3.9(c) gives an example for κ color-coded on the interface. Such visualization supports the interpretation of the role of surface tension in breakup and coalescence of two-phase flow.

Area to Volume Ratio The area to volume ratio provides a notion of compactness, and is particularly useful for visually identifying ligaments (long liquid structures that typically break up rapidly into droplets). For each phase component C_k, it is computed as

$$s_k = \frac{A_k}{V_k}\frac{r_k}{3}, \tag{3.9}$$

where A_k is the area computed from the mesh extracted with the marching cubes algorithm. The scaling $r_k/3$, where r_k is the radius of a sphere with volume V_k, is used to eliminate the dependency on the volume of the phase component. Thus, for spherical components, $s_k = 1$.

3.1.4 Space-Time Graph Representation

The space-time graph visualization employs a layered graph layout, where consecutive layers represent successive time steps. By default, the graph is oriented with time increasing from top to bottom, as shown in Figure 3.6(a), with a horizontal red indicator of the currently chosen time step.

In simulations with confined containers, the total volume of both phases stays constant, i.e., all inclusions typically represent a partition of unity. This motivates the utilization of Sankey diagrams [147] for the representation of inclusion volumes in the layered graphs, i.e., by mapping the volume of the phase components to edge width. Figure 3.6(b) shows an example of the basic graph layout. Furthermore, a close integration of the 3D and space-time graph views is achieved by integrating the 3D inclusion interface at each respective node of the 2D graph (Figure 3.6(c)).

The developed graph layout is inspired by the Sugiyama layout [169]. Specifically, the barycentric method is applied to reduce edge crossings between consecutive layers. The computation of the final horizontal positions, however, has been modified. That is, the spacing between nodes is determined from the already computed nodes to the left and volume of the currently evaluated node. Hence, the horizontal space consumed by a component scales linearly with its volume. This, together with edge width adjustment, greatly enhances the visualization of volumetric structure. Figure 3.6(a) and (b) shows the impact of volume consideration.

Droplets that cross the domain boundary are indicated by attaching a violet circle glyph at the respective node of the layered graph (Figure 3.6(c)). Birth events, on the other hand, are marked with a yellow circle glyph. Components that appear and vanish in the same time step are indicated with an orange death glyph. Both, contact with domain boundaries and events of birth and death, are also visualized in the 3D view by mapping these colors to the interface.

The problem with the volume-based approach is that in case of a death event, all nodes to the right of the corresponding node shift to the left, resulting in skewed edges and therefore less readable layout. To remedy this problem, ghost nodes are inserted for disappearing components, with volume corresponding to those nodes. This ensures that the total volume is preserved and the edges in subsequent time steps remain straight if the topology does not change.

Node Clustering in Parametric Visualization

For the analysis of large datasets, such as jet simulations (Section 3.1.5), node clustering is employed in the graph view. Here, agglomerative complete-link clustering is used, as described in [149], where initially each node constitutes a cluster and two clusters are grouped if the distance of all pairs of elements from both clusters is smaller than a predefined threshold. The threshold is a global quantity and is defined as the median of the quantity for which the distance is computed. The clustering is performed for each time step separately. Figure 3.6(d) demonstrates the approach. To differenti-

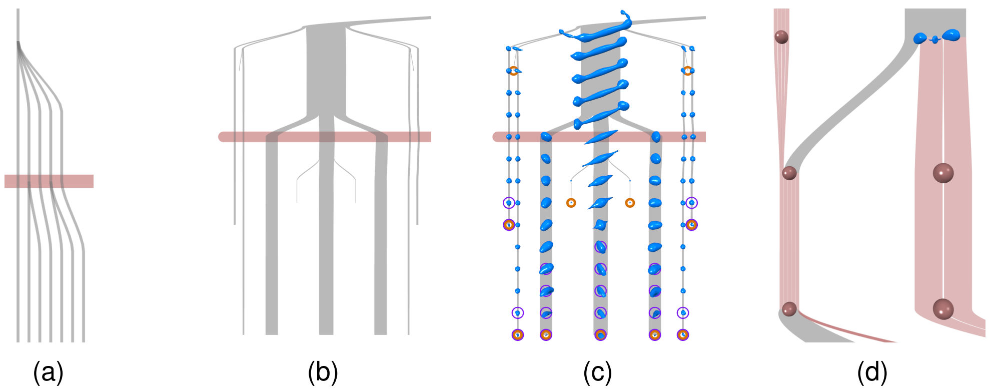


Figure 3.6: Space-time graph representation. (a) Basic graph layout with the selected time step marked with a red stripe. (b) Edge width and spacing determined from droplet volume. (c) Interactive 3D phase components as graph nodes. Death and birth events, and boundary components are marked with orange, yellow, and purple circles, respectively. (d) Node clustering (red edges) for large datasets, with edges split according to volume contribution of each droplet within the cluster. © 2017 IEEE.

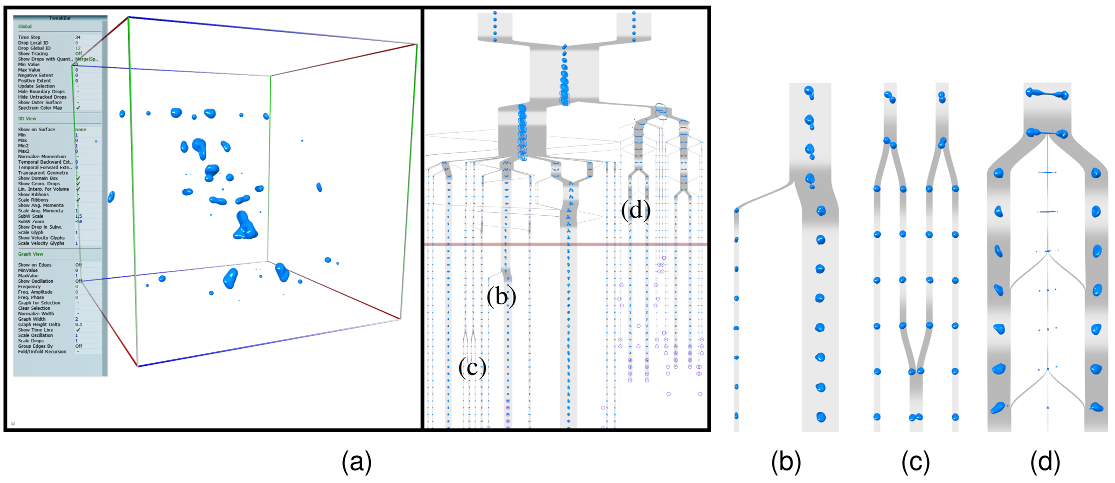


Figure 3.7: Overview of the visualization tool. (a) A screenshot of the visualization of gas-liquid dynamics. Branches undergoing merge or split events are highlighted (dark gray) to facilitate comparison. (b)–(d) Selected droplets from the parametric view for closer examination. © 2017 IEEE.

ate between nodes representing clusters from single droplets, the clusters are indicated with red spheres. For consistent cluster representation, edges that have a common start node and end at a cluster node are additionally clustered. Volume distribution among droplets represented by the cluster is shown on the cluster edges by vertical separating lines, where the edges are divided according to the volume contributions of the clustered droplets. For detailed investigation, the visualization supports a level-of detail approach by allowing the user to unfold (and fold back) single clusters, such that all nodes represented by a given cluster are visible, together with their connectivity (Figure 3.12).

Brushing and Linking

All components of the presented technique are tightly coupled. In Figure 3.7(a), the overall visualization is shown. It is split into the 3D view on the left and the space-time graph view on the right. The user can select the quantities and visualization components described in this section in the toolbox on the left. For comparative visualization, it is possible to highlight graph nodes that satisfy a user-selected condition. For instance, in the same figure, branches that undergo split or merge events are gray, whereas other branches are faded. Additionally, the user can specify, e.g., the minimum number of branches for these events, or select other conditions, such as any combination of merge/split, drop velocity magnitude, or volume.

The brushing and linking methodology employed in the presented visualization framework is tailored toward the analysis of droplet dynamics and their connectivity changes. It provides several modes of interaction to facilitate both the exploration of the space-time domain as well as the analysis of individual drops and their history.

The fluid interface geometry in the graph representation (Figure 3.6(c)) is interactive, i.e., when the dataset is rotated in the 3D view, all instances rotate simultaneously also in the 2D graph around their respective centroid. The analysis of connectivity changes is facilitated by color-coding the droplet surfaces according to their relation to drops from adjacent time steps. As illustrated in Figures 3.1(c)-(f), different colors are assigned to each drop after split (before merge) events. Each vertex on the surface of a drop before split (after merge) obtains the color of the corresponding drop found during tracking.

Users can also select droplets with a mouse click on the graph to display them in the 3D view. Moving to different time steps allows them to analyze the drop history. This provides a detailed analysis of the selected drop in the 3D view in the context of the space-time representation of the whole simulation. The visualization also supports brushing, i.e., the interactive selection of droplets in the 2D and 3D views which leads to the construction of a new graph with individual layouting, where all children and ancestors of the selected drop are included. For instance, Figure 3.10(b) and (h) was achieved by selecting a drop in the *Peripheral Collision* dataset. In the 2D and 3D representations, the user can select the number i of previous and the number j of subsequent time steps to visualize, i.e., for each selected drop k that resides at time step t_k, the temporal history is visualized within the time step interval $[t_k - i, t_k + j]$. This is achieved by color coding the geometries (and transparency in 3D). For example, in Figure 3.10(h) and (g), the geometry of the past time steps is colored purple, while in Figure 3.10(a) and (b), future time steps are selected (i.e., $j > 0$), with drops color transitioning to yellow.

3.1.5 Results

The utility of the presented approach is demonstrated using three CFD datasets. Two datasets are collision simulations of well-resolved liquid inclusions in gaseous surrounding, and one is a jet simulation where single droplets are not well resolved. However, as will be shown, the visualization approach still provides useful insights in such cases.

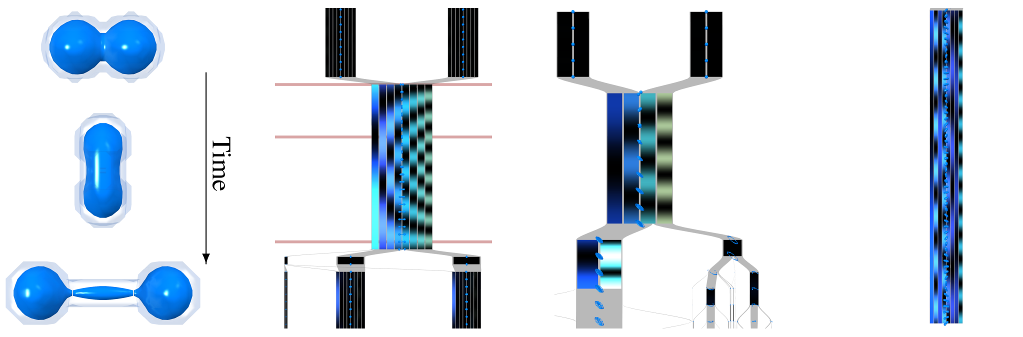


Figure 3.8: (a), (b) Centric Collision dataset. (a) Three selected time steps of the collision with (b) frequency spectrum visualized on the graph. The frequency spectrum forms an arch-like pattern. (c) Corresponding visualization for the *Peripheral Collision* dataset with the initial collision and (d) selected droplet from Figure 3.3. © 2017 IEEE.

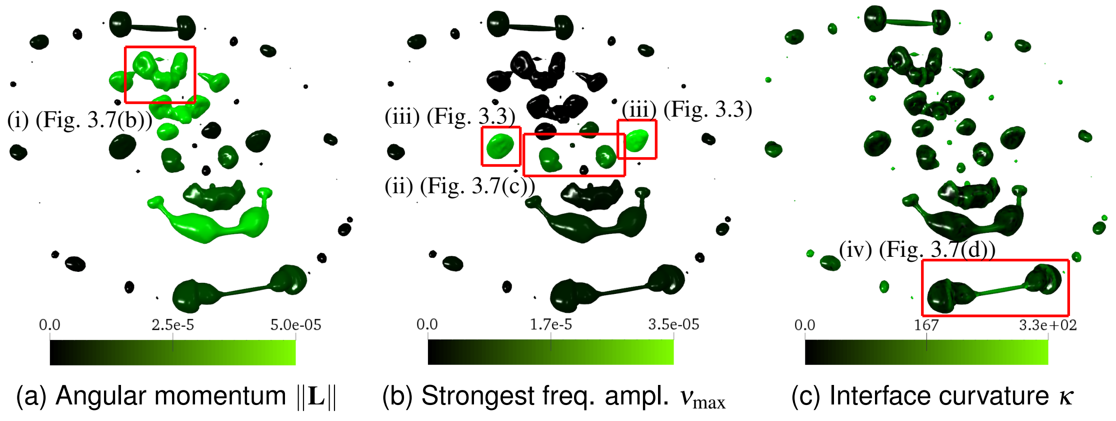


Figure 3.9: *Peripheral Collision* dataset (time step 23) with different quantities mapped on phase interface. (a) Angular momentum $||\mathbf{L}||$ $[\mathrm{g\,cm^2/s}]$ reveals droplets subject to rotation. (b) Oscillation of droplets is indicated by strongest frequency amplitude $v_{\max}$ $[\mathrm{cm}]$. (c) Interface curvature κ $[1/\mathrm{cm}]$ reveals causes for oscillation and potential breakup. © 2017 IEEE.

Centric Collision Dataset This dataset is a CFD simulation of two equally-sized colliding droplets, on a regular grid with $128 \times 64 \times 64$ cells resolution and 101 time steps. Figure 3.1(a) gives an overview on the overall dynamics. The collision leads to the formation of three droplets—two droplets that oscillate as they move away from the center, and one in the middle that remains at its position throughout the simulation.

The collision is investigated in Figure 3.8(a) and (b) by means of its 3D representation and its frequency spectrum visualized on the space-time graph. The spectrum in Figure 3.8(b) reveals an interesting arch-like pattern, with all frequencies at peak just before the breakup. This pattern was found to be typical for major changes in inclusion shape, characterized by an orderly deformation, that leads to breakup.

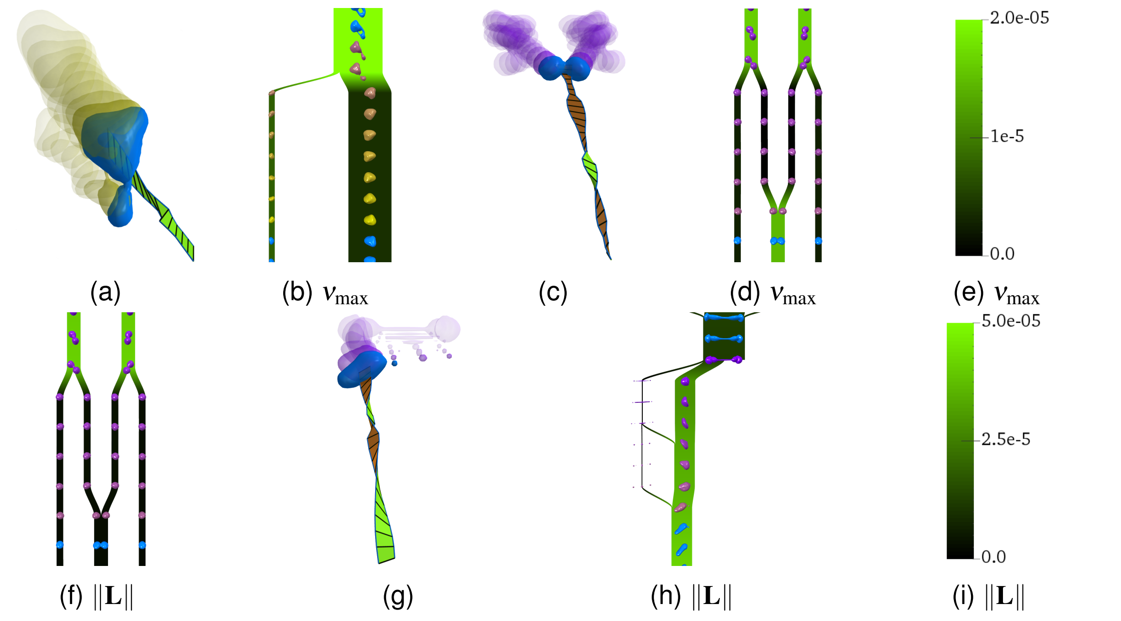


Figure 3.10: Selected components in *Peripheral Collision* dataset. (a) PCA ribbon for an oscillating droplet before breakup and (b) corresponding graph with color-coded strongest frequency amplitude (color legend in (e)). The oscillation decreases after breakup. (d) For the droplets in (c), the strongest frequency amplitude also decreases after initial breakup and increases again after coalescence. (c) PCA ribbons for a droplet after collision and (g) after breakup. (f), (h) Corresponding graphs with angular momentum color-coded on the edges (color legend in (i)). In (c), the angular momenta of the colliding droplets annihilate, whereas in (g), breakup leads to increased rotational motion. © 2017 IEEE.

Peripheral Collision The second dataset is a CFD simulation of two peripherally colliding droplets, on a regular grid with $256 \times 256 \times 256$ cells and 79 time steps.

The visualization session starts with the general overview of the dataset (Figure 3.7), where the data is displayed in the 3D view and in the 2D graph view. In the latter, with the help of the phase components on the graph nodes, the topology of drop dynamics can be readily seen—the peripheral collision of two equally-sized droplets leads to the formation of a disk-shaped droplet that breaks into many smaller droplets. In this view, several intriguing configurations were observed. Namely, in Figure 3.7(b), a drop splits into two after a considerable amount of time. There is also a case exhibiting symmetry where two droplets split simultaneously and then two of the new drops merge into one in Figure 3.7(c). A drop split in Figure 3.7(d) with a resulting ligament indicates high influence of surface tension and the continuous split of secondary droplets.

To gain knowledge on the mechanisms behind these processes, several physical quantities at earlier simulation step (time step 23, also selected in Figure 3.7(a)) are analyzed. These quantities are mapped on the drop surfaces in the 3D view, as shown in Figure 3.9, where the selected drops from Figure 3.7 are marked with red boxes. In Figure 3.9(a), the angular momentum amplitude $\|\mathbf{L}_{C_k}\|$ is high for strongly deformed droplets that break up in the following time steps. Particularly, the droplet in the box

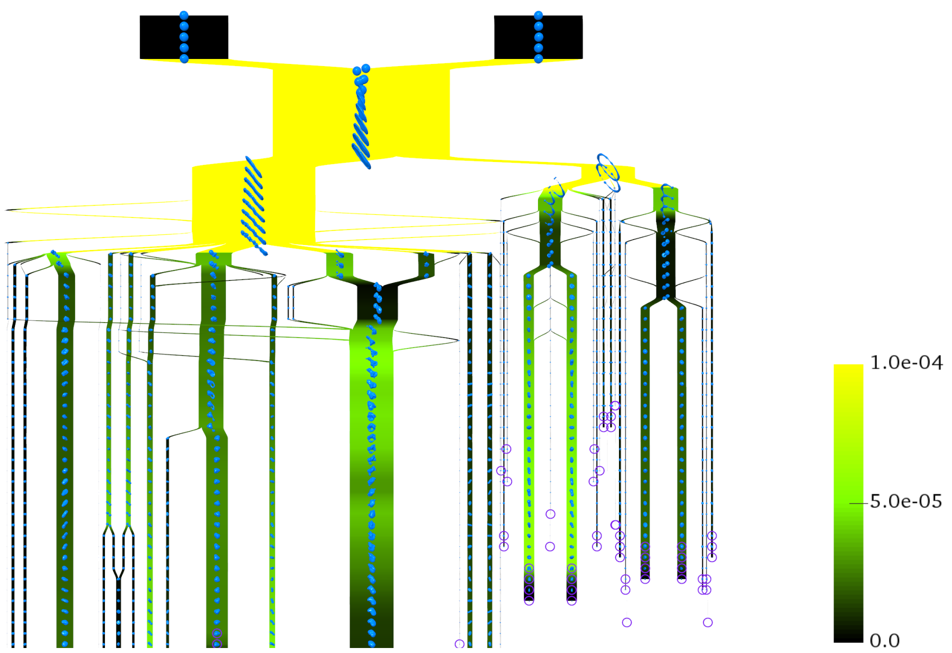


Figure 3.11: Space-time graph view for the *Peripheral Collision* dataset with angular momentum [g cm^2/s] color-coded on edges, downscaled to enhance values on the lower part. For many droplets, the angular momentum increases after breakup. © 2017 IEEE.

(i) has a high value. The strongest frequency amplitude in Figure 3.9(b) is high for the droplets that do not break up immediately but rather start to rotate. Therefore, these droplets—the drops in the box (ii) as well as the two droplets with the maximum value in boxes (iii)—are further investigated with suitable visualization techniques. The mean curvature of the interface (Figure 3.9(c)) reveals potential breakup points on the surface of the droplets. If the breakup occurs, these points are pulled by the surface tension force, which in turn can result in oscillation. As can be seen, the droplet in the box (iv) has high curvature in the middle.

To further inspect the origins of the particular droplet behavior, in Figure 3.10(a)–(h), the angular momentum and strongest frequency amplitude are analyzed thoroughly for the selected droplets. The droplet from Figure 3.7(b) (before breakup) is shown in Figure 3.10(a) with transparent yellow surface for future time steps, and the corresponding graph with strongest frequency amplitude in Figure 3.10(b). The frequency amplitude decreases substantially after the breakup, which indicates that both separated parts contributed opposing energy to the oscillation before. Amplitude decrease can be also seen after the breakup shown in Figure 3.10(d) (case (c) in Figure 3.7), however, it increases again after the merge of the two resulting droplets. Therefore, the dynamics of the merged droplets are analyzed in Figure 3.10(c) by means of a PCA ribbon to see the resulting oscillation. The droplet in blue is shown just after the merge from Figure 3.10(d) of two droplets that in turn broke up from strongly rotating drops, as shown with the transparent purple (past) time projection. Interestingly, the PCA ribbon reveals initial oscillation of the droplet, followed by rotation. In Figure 3.10(f), the magnitude of the angular momentum is mapped on the graph for the selected droplets. As can be seen, the angular momenta of the two rotating droplets cancel out due to the merge. However, the ribbon in Figure 3.10(c) indicates rotational motion in later time steps. This can be explained by Figure 3.10(d), where the strongest frequency amplitude increases considerably after the collision. This oscillation is in turn transformed into rotational motion of the droplet in Figure 3.10(c). A similar behavior can be observed for the droplet in Figure 3.10(g), where the separation of the droplets causes initial oscillation which then transforms into rotation.

In Figure 3.9(b), the two droplets with the largest amplitude value (boxes) deserve more attention, and therefore their spectral characteristics were analyzed with oscillation glyphs, as demonstrated in Figure 3.3(k)–(o). It can be seen in figure (k) that there are two rather strong frequency components in direction of the PCA eigenvector $\overline{\boldsymbol{\varepsilon}}_k$ (gray axis), but all other components are perpendicular to it, which reflects that the droplet is at its maximum contraction. In the time step in Figure 3.3(l), the distribution is similar, but more components are now pointing in direction of $\overline{\boldsymbol{\varepsilon}}_k$. During the time steps shown in Figure 3.3(m)–(n), the oscillation components get stronger in direction of $\overline{\boldsymbol{\varepsilon}}_k$, until almost all are aligned in $\overline{\boldsymbol{\varepsilon}}_k$ (figure (o)), indicating maximum elongation and alignment of these oscillation components in $\overline{\boldsymbol{\varepsilon}}_k$-direction. The spectral analysis for the droplet, shown in Figure 3.8(d), shows no correlation between frequencies (as opposed, e.g., to 3.8(c)). This results in a damping effect of the frequencies and could explain why this droplet does not split further, as would be expected due to centrifugal forces.

The cases with increasing angular momentum for some droplets after split or merge events warranted some closer investigation as to whether this phenomenon occurs more often in the dataset. In fact, the graph shown in Figure 3.11 shows that this behavior can be observed with almost all rotating phase components. This can also be seen, for example, by the temporally increasing length of the angular momentum arrow glyphs in Figure 3.2. This seemed to violate preservation of energy, i.e., preservation of angular momentum. However, when analyzing the PCA ribbon, it was found that, right before the split, the droplet is not oscillating anymore and is strongly elongated. This observation led to the conclusion that angular momentum is increasing over time for droplets that oscillate and rotate at the same time, because the initial oscillatory energy is transformed into angular momentum, i.e., the oscillating component in radial direction competes with centrifugal forces and thus is transformed to angular momentum.

Jet Dataset The last analyzed dataset is a CFD simulation of a polymer-water solution injected into air with a pressure of 30 bar with a dual nozzle (with higher stream velocity in the outer ring), discretized on a regular grid with $1024 \times 512 \times 512$ cells and 61 time steps. In this dataset, the focus is on the general characteristics due to the immense number of droplets and topological changes. The goal is to find general processes and features of the flow. For this purpose, the graph representation with clustered quantities should provide insights. Here, the spatio-temporal distribution of droplets, as well as

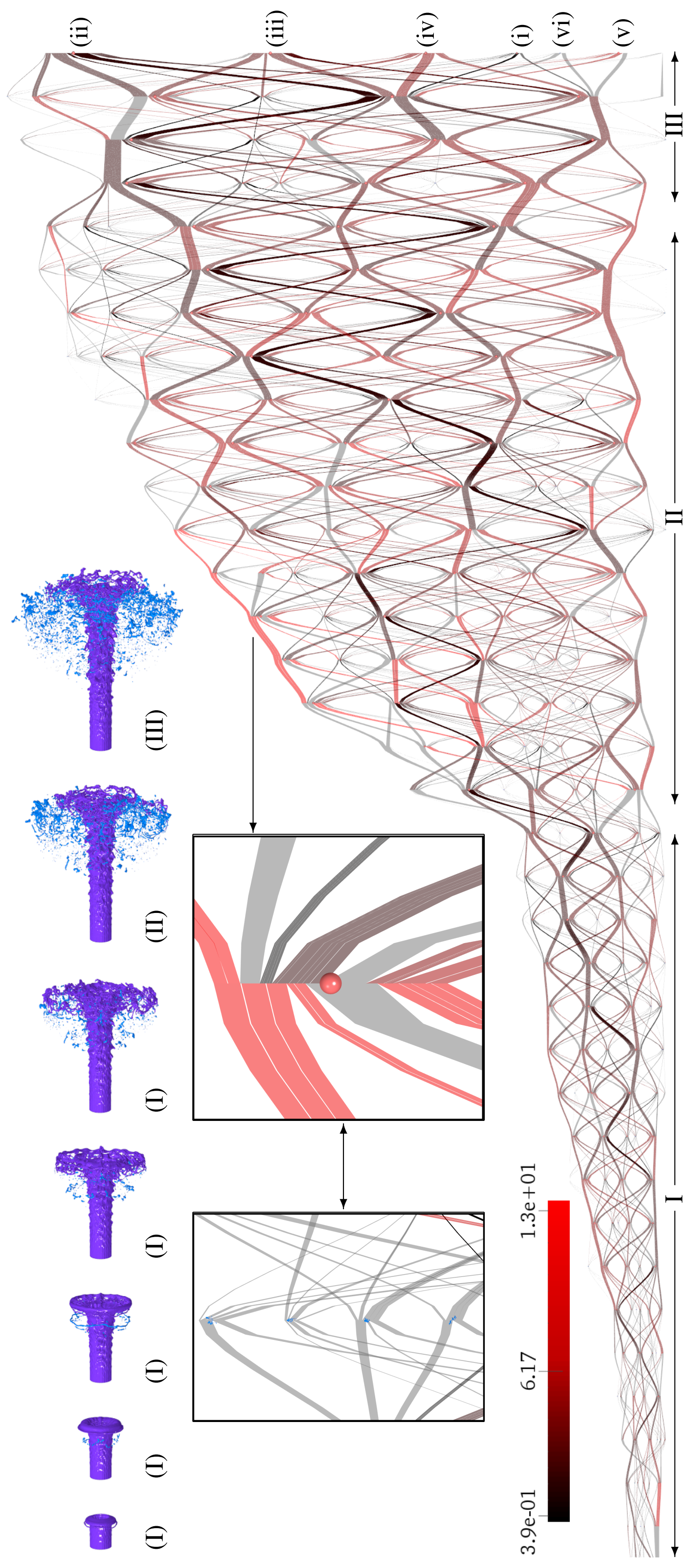


Figure 3.12: Jet dataset with selected time steps shown. (Top) Jet (purple) is injected from left, and due to its high velocity at the nozzle, it splatters into various-sized droplets (blue). (Bottom) Graph shows the space-time evolution of the jet stream. Edge clusters represent droplets with similar area to volume ratio. The box in the middle shows an enlarged node with volume distribution indicated on the clustered edges. A node cluster can be unfolded (left box) and folded back to reveal the actual edge connectivity and drop geometry. Here, highly deformed ligaments with large area to volume ratio are visible. © 2017 IEEE.

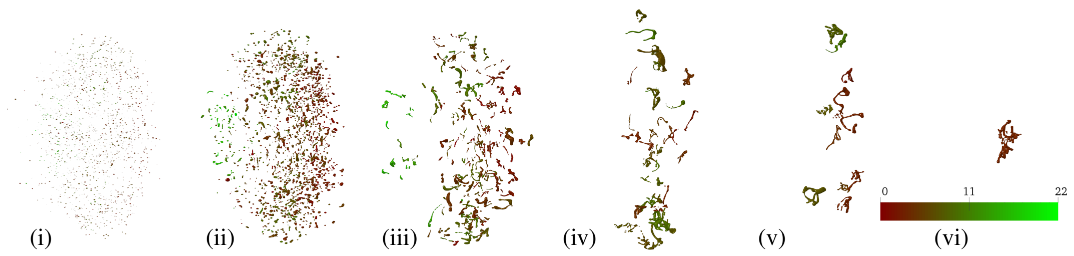


Figure 3.13: For selected clusters at the last time step in Figure 3.12, drops are color-coded with the number of breakups in their history. © 2017 IEEE.

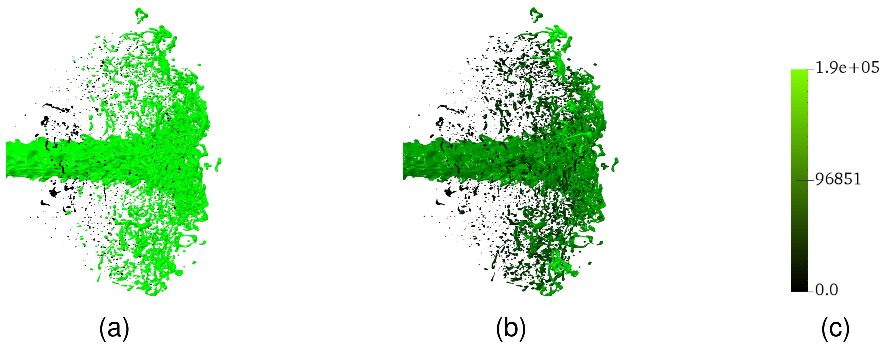


Figure 3.14: *Jet* dataset with (a) splash/non-splash criterion (green/black) color-coded on phase interface, and (b) droplet-localized Reynolds number Re_{C_K}(color legend in (c)). © 2017 IEEE.

instabilities that possibly lead to further breakup are analyzed.

Figure 3.12 provides the respective visualization. In the top, the jet development is shown from left to right, with the front of the stream spreading outward and disintegrating into a multitude of droplets. The jet core is colored purple and the droplets that separated from it are marked blue. Using the clustering of the droplets by the area-to-volume ratio s shown in the graph in the bottom of the figure, the evolution and distribution of droplets of different forms can be inspected. In the graph, clusters with high average values, i.e., consisting of more elongated drops, are indicated by a more saturated red color. Interestingly, such clusters occur throughout the simulation time. As can be seen in the zoomed parts, the ligaments are highly deformed, and—as indicated by complex connectivity—undergo many breakup and merge events, the latter possibly due to the high density of the droplets.

Three characteristic stages were observed in this dataset, as indicated on the left side of Figure 3.12. In the first stage (I), only a small number of droplets detach from the jet tip. In the second stage (II), the jet becomes unstable and starts to atomize. This is reflected in the increase of clusters in the graph, particularly with high deformation, indicating elongated ligaments breaking off the jet core. In the next stage (III), these ligaments in turn break up into smaller droplets. This can be again observed in the graph, where the number of clusters with high deformation is decreasing again, while the total number of clusters stays about the same.

To further investigate the formation and distribution of droplets, at the last time step, each cluster in the graph was selected and the number of breakups each drop within a cluster underwent was color-coded (dark red to bright green), as shown in Figure 3.13. Such a decomposition gives some insight into drop evolution. Small drops are distributed uniformly, whereas ligaments are situated mostly at the front of the jet. The number of breakups is larger directly behind the jet front, both for spherical and slightly elongated droplets. For ligaments, however, such regularity cannot be observed. For instance, in the clusters (iv) and (v) in Figure 3.13, ligaments with different number of breakups are mixed.

In Figure 3.14(b), the droplet-localized Reynolds number Re_C is visualized on the jet interface. As can be seen, long ligaments exhibit higher values, indicating their instability and therefore possible breakup. The drops further away from the disintegrating front, on the contrary, have smaller Reynolds number. An explanation is that the free stream velocity is reduced in this region. As described in Section 3.1.3, with the droplet-localized dimensionless numbers Re and Oh, the splashing behavior of the droplets can also be investigated, as demonstrated in Figure 3.14(a). The distribution of non-splash droplets roughly correlates with low Reynolds number (especially for the drops at the back), although a prevailing number of droplets in the splashing regime can be clearly seen. This visualization provides valuable information on droplet dynamics in jet simulations, where the splashing characteristic is often desired for efficiency of combustion engines. The opposite case, i.e., the deposition of droplets forming a thin liquid film on combustion walls, reduces the effective fuel combustion rate and could be likewise analyzed in the presented framework.

Performance

To analyze the tracking performance of the explicit Euler scheme, it has been compared with 4th-order Runge-Kutta integration. The results in Table 3.1 show that both methods perform similarly, except for the Jet dataset, where the explicit Euler method can track more droplets. Overall, the table shows that, as compared to the total number of drops (i.e., accumulated over all time steps), the number of the erroneous events is relatively small, except for the Jet, where the number of the untracked drops is considerable. However, as mentioned earlier, these are very small droplets that have little influence on the observed phenomena.

Table 3.1: Tracking results for 4th-order Runge-Kutta (with 10 substeps) and explicit Euler integration scheme. Sum of births and deaths, number of untracked drops (where birth is immediately followed by death event), and total number of drops (summed over all simulation steps).

	Births + Deaths		Untracked		Total #Drops
Dataset	RK4	Euler	RK4	Euler	
Centric	3	3	0	0	243
Peripheral	10	11	6	6	1 989
Jet	2 219	2 202	8 476	8 299	76 821

Table 3.2 provides computation times for the components of the visualization approach, as well as rendering performance. The computation time largely depends on the complexity of the dataset—a larger number of components increases tracking time, as the advected particles must be tested against more components. However, since the analyzed quantities do not require any parametrization, they can be precomputed offline, stored to disk, and loaded on demand for immediate analysis. Rendering performance mainly depends on the geometric complexity of the extracted surfaces, as well as the number of components, since all of them are visualized on the graph at the same time. The visualization of the Jet dataset runs with relatively low frame rate, however, it did not hinder the analysis of the dataset. Component tracking takes also considerable computation time due to costly particle advection, but it can be also precomputed offline.

Table 3.2: Computation times (in seconds) for graph layouting, tracking, quantities, and rendering performance (in frames per second) for the analyzed datasets. Measurements taken on Intel i7 3.6 GHz (single process).

Dataset	Graph [s]	Tracking [s]	Quantities [s]	Render [fps]
Centric	< 0.01	0.43	1.45	20
Peripheral	0.02	13.5	60	19
Jet	0.03	16 464	1 645	8.3

3.2 Visualization of Droplet Separation

In the visualization context, droplets and ligaments in two-phase flow can be interpreted as features. This allows for application of feature-based visualization methods where visual data representation is reduced to important characteristics, typically physical quantities, flow topology, and quantities derived from generic scalar fields [141]. While such visualization reveals the overall feature topology dynamics (e.g., split and merge events), detailed spatial information on volumetric partitioning of features is difficult to obtain with existing techniques. Thus, a visualization approach is proposed that can reveal the separation dynamics within the inclusions in two-phase flow. It is based on the extraction of boundaries around regions within an inclusion that correspond to inclusions developing from the original one in the course of time. Additionally, the temporal information of the separation is conveyed by means of separation surfaces that encode the time at which the separation occurred.

The visualization technique expands upon traditional feature visualization in several ways. First, it allows for static visualization of dynamic processes, and therefore reduces visual clutter. Second, it combines advantages of standard feature tracking methods and finite-time Lyapunov exponent (FTLE)-based methods, as it allows for a detailed inspection of the separation of inclusions. Third, it is applicable to a broad class of multiphase flow, including two-phase flow (liquid in gas surrounding) and multi-component flow (liquid-liquid or multi-component liquid in gaseous surrounding).

Typically, only a fraction of the simulation time steps is stored for post-processing, and therefore, a corrector scheme for particle integration is proposed to ensure phase-consistent particle trajectories within the available data. [2]

3.2.1 Related Work

The field of research which is closely related to this technique is feature tracking, already discussed in Section 3.1.1. The presented method differs from these approaches, in that it focuses on the spatio-temporal aspect of feature representation, where the topology is directly conveyed in the feature volume. In Chapter 2, challenges related to tracking volumes in two-phase flow are considered.

For the analysis of the combustion process in engine simulation, Garth et al. [57] proposed a set of visualization methods that operate on time-varying unstructured grids. A recent work by Sauer et al. [159] utilized particle and volume data from the simulation runs to track features over longer time intervals. Here, particles are inserted after the simulation run, which gives a better control over the particle density and therefore the detail level, however, necessitates some corrector schemes for phase-consistent advection in two-phase flow.

Delocalized quantities are employed in unsteady flows to statically investigate the dynamics of scalar fields [164], where quantities are averaged over time along integral

[2] Parts of this section have been published in: [84]

curves. The presented concept can be viewed as a special case of this, where only the values at trajectory end points are stored at the seed point. The visualization of delocalized quantities can be improved using uncertainty information from the FTLE field [152]. For the topological analysis in general unsteady flow, FTLE has become a standard visualization method [64]. Sanderson [157] proposed an alternative inspired by Lyapunov exponents that is based on traveled particle distance instead of particle divergence.

The dynamics of fluid flow is often represented by surfaces or volumes. In the work by van Wijk [197], stream surfaces were obtained by extracting isosurfaces from a scalar field. These are generated by advecting streamlines backward up to the boundary with predefined scalar values and resampling those values along the streamlines. Becker et al. [14] used flow volumes in unsteady flow to reveal the flow dynamics around regions of interest. An implicit version of the flow volumes by Xue et al. [205] allows for more detailed inspection of the flow.

3.2.2 Visualization Method

The technique employs particle advection to determine the volumetric correlation of inclusions between a reference time step t_0 and the target time step t_F. Usually, only a certain fraction of time steps of a simulation are stored for later analysis. Using these data with reduced time resolution can lead to errors in the estimation of trajectories, especially in multiphase flow, where particles potentially leave the initially assigned phase for this reason. This problem could be avoided if particles were advected during the simulation, which is increasingly popular, as reported by Sauer et al. [159]. Additionally, particles could be densely populated using the method proposed by Agranovsky et al. [2] to capture more details of the topology of droplet dynamics. In the investigated simulation datasets, however, particle data was not provided. Nevertheless, to still ensure robust particle advection in terms of phase consistency in multiphase flows, a three-stage corrector method is introduced which utilizes the f-field in a corrector step during particle advection to ensure that particles remain in the respective phase throughout integration.

For the purpose of the following discussion, an inclusion M is defined as a region with the volume fraction f exceeding a threshold τ: $M = \{\mathbf{x} : f(\mathbf{x},t) > \tau\}$. Here, τ is set to 0. Inclusion separation is illustrated in Figure 3.15(a), where an initial inclusion $M_1(t_0)$ splits within the time interval $]t_0, t_F[$, resulting in three inclusions $M_1(t_F)$, $M_2(t_F)$, and $M_3(t_F)$.

3.2.3 Separation-Based Inclusion Segmentation

The aim of the visualization technique is to capture how phase components develop topologically in the course of time. For instance, if an inclusion splits into two new ones, it is of interest how the volume of the initial inclusion is divided among them, or, in other words, what the volumetric contributions of the new inclusions in the initial one

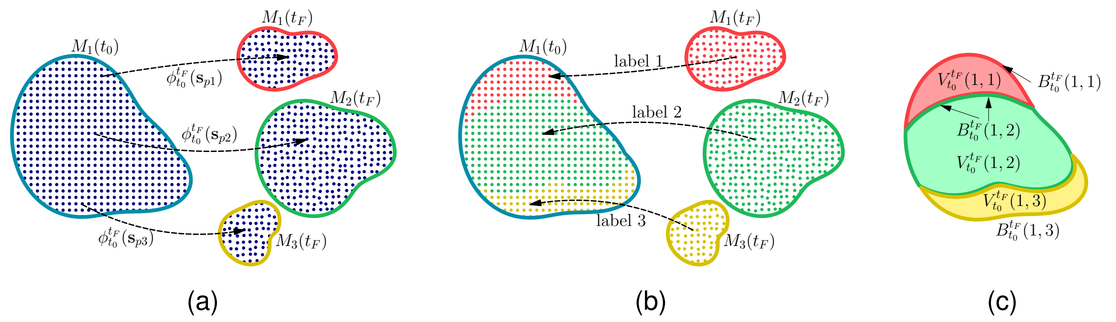


Figure 3.15: Illustration of the separation boundaries for a simple case with one inclusion at time t_0, and three inclusions at $t_F = t_2$. (a) Particles are seeded at time step t_0 inside the inclusion and advected to t_F, resulting in a flow map $\phi_{t_0}^{t_F}$ (black arrows). At t_F, the inclusions have been labeled red, green, and yellow. (b) Inclusion labels are assigned to the particles that are inside a given inclusion. These labels are then transferred to the seed points (arrows). (c) For each label, the corresponding volume V is identified (red, green, and yellow areas), and the boundary B is extracted (corresponding darker curves).

are. To accomplish this goal, the spatio-temporal correspondence between the inclusion $M_i(t_0)$ at time t_0, and inclusions $M_j(t_F)$ at some other time t_F are determined. That is, for each $M_j(t_F)$, its volumetric contribution within $M_i(t_0)$ is defined as

$$V_{t_0}^{t_F}(i,j) = \{\mathbf{x} : \mathbf{x} \in M_i(t_0) \wedge \phi_{t_0}^{t_F}(\mathbf{x}) \in M_j(t_F)\}, \tag{3.10}$$

where $\phi_{t_0}^{t_F}(\mathbf{x})$ maps the initial position $\mathbf{x}$ at time t_0 to its position at time t_F, as it is advected by the flow. The quantity $\Delta t = t_F - t_0$ represents the computation time interval. For the visualization, the closed boundary of the volume V is extracted:

$$B_{t_0}^{t_F}(i,j) = \partial V_{t_0}^{t_F}(i,j). \tag{3.11}$$

The method allows t_F to be earlier in time than t_0, in which case $\Delta t < 0$. In Figure 3.15, the technique is illustrated for a simple case where an inclusion splits into three.

Volumetric contributions $V_{t_0}^{t_F}(i,j)$ can be composed of disconnected segments, either due to a merge followed by a split of the initial inclusions, or due to disjoined volume segments inside the initial inclusion that together form a separate inclusion. As it will be shown in Section 3.2.5, the visualization allows one to readily identify and analyze these cases.

3.2.4 Temporal Separation Surfaces

To support temporal analysis of inclusion separation, the technique is complemented with the extraction of temporal separation surfaces, denoted S. While the above method finds the volumetric correspondences in the initial time t_0 to the inclusions at target time t_F, here, the separation surfaces are constructed within the whole interval $]t_0, t_F]$ to reveal the temporal information on successive separations. They divide the volumetric contributions in the initial inclusion as the corresponding inclusions split into new ones.

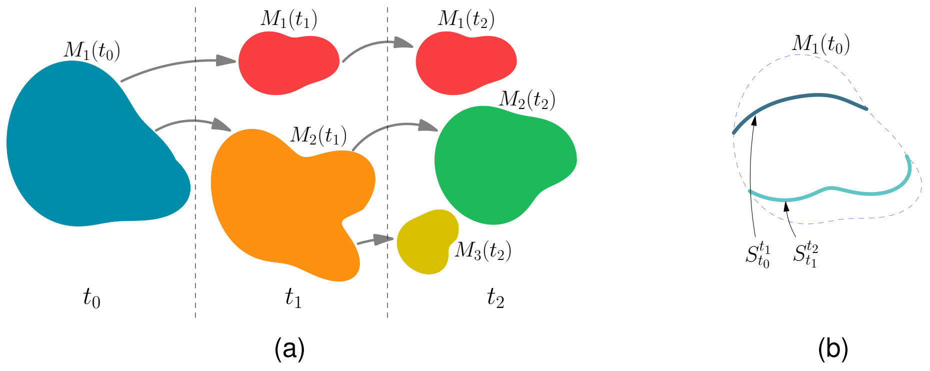


Figure 3.16: Visualization of separation surfaces. (a) There is one phase component at time t_0, which splits into the red and the orange phase components at the intermediate time step t_1, and the orange one splits again at $t_2 = t_F$ into the green and yellow inclusions. (b) Temporal separation surfaces $S_{t_0}^{t_1}$ and $S_{t_1}^{t_2}$ partition the initial inclusion according to the volumetric correspondences of the newly created inclusions at time t_1 and t_2, respectively.

This is illustrated in Figure 3.16(a), where the initial inclusion $M_1(t_0)$ has split into two at time t_1, and the resulting inclusion $M_2(t_1)$ has further separated into new phase components $M_2(t_2)$ and $M_3(t_2)$ at time t_2. In Figure 3.16(b), both split events are illustrated by the separation surfaces $S_{t_0}^{t_1}$ and $S_{t_1}^{t_2}$ that divide the volume according to the volumetric contributions of the inclusion resulting from the separation.

To create the separation surfaces and determine the time at which they occur, changes in contributions are detected within small time intervals. Specifically, for each time increment $\delta t = t_2 - t_1$, where $\delta t \ll \Delta t$, contributions $V_{t_0}^{t_2}(i,k)$ in each inclusion $M_i(t_0)$ are computed and compared with the previous contributions $V_{t_0}^{t_1}(i,j)$. If the number of contributions $V_{t_0}^{t_2}(i,k)$ for which $V_{t_0}^{t_1}(i,j) \cap V_{t_0}^{t_2}(i,k) \neq \emptyset$ is greater than one, a separation has occurred in the given interval inside the volume contribution $V_{t_0}^{t_1}(i,j)$, and the separation surface is generated for this time interval in $M_i(t_0)$ where different $V_{t_0}^{t_2}(i,k)$ adjoin.

Theoretically, the temporal evolution of inclusion separation could be accomplished by repeatedly extracting the closed boundaries $B_{t_0}^{t_k}(i,j)$ for varying k. This would, however, lead to repeated construction of overlapping boundaries that would be difficult to analyze. The separation surfaces S provide an open structure that complements the extracted closed boundaries B.

3.2.5 Numerical Approach

The method operates on both the Eulerian frame, in which the simulation data is defined, and the Lagrangian frame, where the inserted particles represent the inclusion volume and are used to determine the volumetric correspondences.

The visualization framework works as follows. The input to the approach are a time series of a vector field $\mathbf{u}$ for particle advection and a scalar field f for inclusion definition. The initial time t_0 is chosen, at which particles are seeded within $M_i(t_0)$. These particles are then advected up to time t_F, whereby integration is performed between

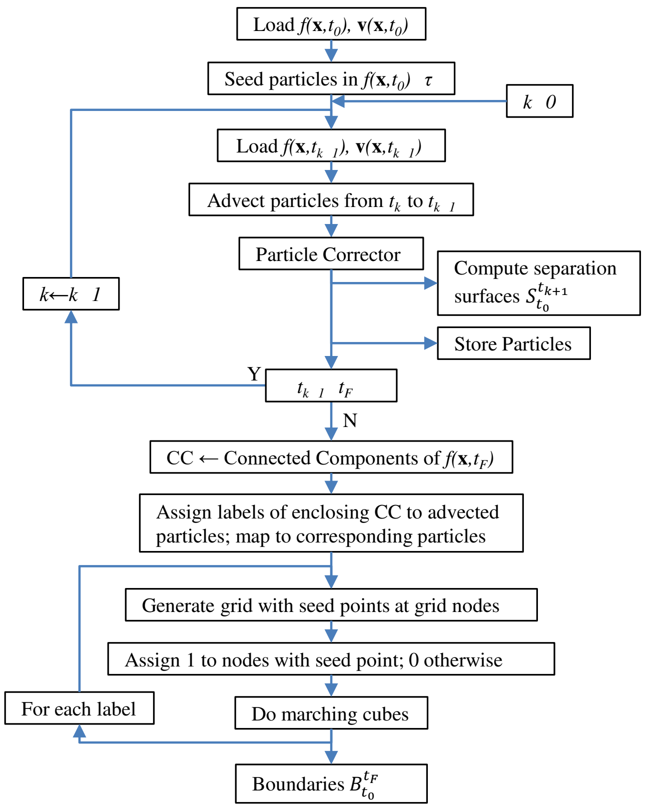

Load $f(\mathbf{x},t_0)$, $\mathbf{v}(\mathbf{x},t_0)$
Seed particles in $f(\mathbf{x},t_0)$ τ
k 0
Load $f(\mathbf{x},t_{k\ 1})$, $\mathbf{v}(\mathbf{x},t_{k\ 1})$
Advect particles from t_k to $t_{k\ 1}$
Particle Corrector
Compute separation surfaces $S_{t_0}^{t_{k+1}}$
Store Particles
$k \leftarrow k$ 1
Y
$t_{k\ 1}$ t_F
N
CC ← Connected Components of $f(\mathbf{x},t_F)$
Assign labels of enclosing CC to advected particles; map to corresponding particles
Generate grid with seed points at grid nodes
Assign 1 to nodes with seed point; 0 otherwise
For each label
Do marching cubes
Boundaries $B_{t_0}^{t_F}$

Figure 3.17: Overview diagram of the numerical approach. The upper loop represents advection performed between consecutive simulation time steps and includes storage of advected particles used for visualization. Separation surfaces S reveal temporal changes in inclusion segmentation and therefore are computed between consecutive simulation steps. The lower loop is performed to obtain boundary B for each label, i.e., for each inclusion at target time t_F.

consecutive simulation steps, and so is the construction of separation surfaces. At time t_F, the connected components that define the inclusions $M_j(t_F)$ are determined, and for each advected particle, the surrounding inclusion is found. The inclusion labels are then mapped to the seed points, and finally, boundaries are extracted for each label within $M_i(t_0)$. In Figure 3.17, the algorithm is presented as a diagram. In the following sections, each step is described in detail.

Particle Seeding and Advection

At time t_0, particles are seeded at $\mathbf{s}_p$ if $f(\mathbf{s}_p, t_0) > \tau$. This is illustrated in Figure 3.18, where τ is represented by the PLIC interface that divides the cell into liquid and gaseous phase, and seed points are located only at positions $\mathbf{s}_p$ enclosed by the PLIC patch. The seed position $\mathbf{s}_p$ and the maximum number of seeds per cell n_c are controlled by a global refinement parameter r in the form $n_c = (2^d)^r$, where $d = 3$ is the data dimension. For $r = 0$, the seeds are positioned at the centers of the simulation cells. For $r > 0$, the cells are recursively divided r times into equally sized subcells, and the particles are positioned at the centers of these subcells.

To determine if $\mathbf{s}_p$ lies within the phase of interest, the seed position relative to the PLIC patch is computed. To this end, the patch orientation and position are determined. To find the PLIC normal $\mathbf{n}$, the gradient of the f-field at the cell center $\mathbf{x}_c$ is calculated and normalized. Next, the attachment point $\mathbf{a}$—defined as the most distant cell node

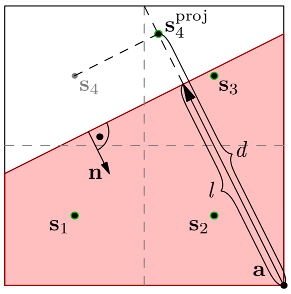


Figure 3.18: Particle seeding with PLIC interface reconstruction. Seed positions $\mathbf{s}_p$ are determined in each cell with $f > 0$ by the refinement parameter r. Here $r = 1$, and $\mathbf{s}_1$–$\mathbf{s}_4$ are at the centers of four subcells. For each $\mathbf{s}$, the $\mathbf{n}$-projected distance d to the attachment point $\mathbf{a}$ is compared with the PLIC patch translation l to determine if particle p is seeded at $\mathbf{s}_p$. $\mathbf{s}_4$ will not be seeded in this case.

from the PLIC patch in the direction of $\mathbf{n}$—is determined. The PLIC patch position l, i.e., its distance from $\mathbf{a}$, is obtained iteratively by ensuring that the volume enclosed by the patch equals the value f in the given cell. Then, l is compared with the distance d from $\mathbf{s}$ projected onto the normal. If $d < l$, a particle is positioned at $\mathbf{s}_p$. In the figure, the distance $d > l$ for $\mathbf{s}_{m+3}$ implies wrong phase, and hence, no particle will be seeded there.

To compute the flow map $\phi_{t_0}^{t_F}(\mathbf{s}_p)$, the particles p are set at the seed points and advected up to time t_F (Figure 3.15(a)) using the standard fourth-order Runge-Kutta scheme performed between consecutive simulation time steps in the interval $[t_0, t_F]$, whereby linear interpolation in time is used. The seed points $\mathbf{s}_p$ are stored for later processing to determine the correspondence between inclusions at times t_0 and t_F.

Inclusion Labeling

To extract the inclusions at time t_F, a grid with the same structure as the input simulation grid is created, where cells c with $f(\mathbf{x}_c, t) > \tau$ are set to 1 and the others to 0. With this bit mask, connected components that correspond to the inclusions $M_j(t_F)$ can be found by region growing, whereby two cells c_m and c_n are considered connected if they are face neighbors. Each inclusion is then identified by a label $j \in \mathbb{N}$. The connected components are stored in a label field, which again structurally corresponds to the simulation grid. This greatly simplifies particle assignment, as described in the next section. All grid cells without connected components are marked with an invalid label (i.e., -1). In Figure 3.15(a), the labeled inclusions are marked red, green, and yellow. Inclusion labeling can be performed independently of the advection step, since determination of inclusion labels only requires the scalar field $f(\mathbf{x}, t_F)$.

Particle Label Assignment

In the next step, for each advected particle p at time t_F, the surrounding inclusion M_j is found (Figure 3.15(b)) and its label j assigned to the particle. If the particle does not lie inside any inclusion, a non-valid label (i.e., -1) is assigned to it. On rectilinear grids, the bounding cell can be easily found by iterating over the node coordinates x_i of the grid until $x_i < x_p < x_{i+1}$, with $x_i \in \{x, y, z\}$. The inclusion label j is also mapped to the stored seed points $\mathbf{s}_p$ corresponding to the advected particles (shown by dashed arrows in Figure 3.15(b)). Additionally, to visualize the temporal evolution of the inclusions, the intermediate particle positions (i.e., for $t_0 < t_i < t_F$) are stored during the computation of the visualization, and the computed labels are assigned to them at this stage.

Boundary Extraction

The labeled seed points allow the extraction of the volume boundaries $B_{t_0}^{t_F}$ within the inclusion $M_i(t_0)$ that correspond to the inclusions $M_j(t_F)$, according to Equation 3.11. For this purpose, a bounding box is computed around all seed points $\mathbf{s}_p$ with the given label j and a rectilinear grid is generated within the bounding box such that the seed points coincide with the grid nodes. At each grid node, value 1 is set if a seed point is actually located at this position, and 0 otherwise. This way, the boundaries B can be extracted using the standard marching cubes algorithm. Since the grid node positions must correspond with the seed points, the size and number of cells is implicitly controlled by the refinement parameter r. In Figure 3.15(c), the boundaries B are marked by darker curves that bound the volumes V.

The disconnected inclusion segments (Section 3.2.2) can be visually detected either directly from the color of the boundaries (each inclusion maps to a different color label of the volume regions, and disconnected volume regions belonging to the same inclusion have the same color) or using the particles from intermediate time steps that also carry the inclusion label information.

Extraction of Separation Surfaces

To extract the separation surfaces S described in Section 3.2.4, inclusion labeling and particle label assignment are performed after each advection step to obtain the current inclusion segmentation. The resulting seed labels are stored for two subsequent time steps t_k and t_{k+1}. For each label at t_k, the corresponding seeds and their labels at t_{k+1} are analyzed. More than one unique label at t_{k+1} indicates that the segment has split in the current time interval, and therefore, separation surfaces are extracted by performing a modified marching cubes algorithm for each 2-combination of the labels. In the algorithm, node values are set to "$+$" and "$-$" which correspond to either of the labels, and the surface passes through the edge centers between "$+$" and "$-$" nodes. To ensure open surfaces, the outer nodes (i.e., not belonging to any of the two labels) are marked with an invalid negative value. In the final step of marching cubes, the edges of each triangle are tested for whether they lie on edge with the outer node, and discarded in this case. Finally, the surfaces and the corresponding time stamps t_{k+1} are stored for visualization.

Phase-Consistent Trajectories in Multiphase Flow

Since typically only a fraction of simulation time steps is saved for analysis and visualization (due to potentially high storage demands of large simulations), and because two-phase flow is highly nonlinear (due to, among others, surface tension forces), there is usually not enough information to ensure phase consistency of particles advected with

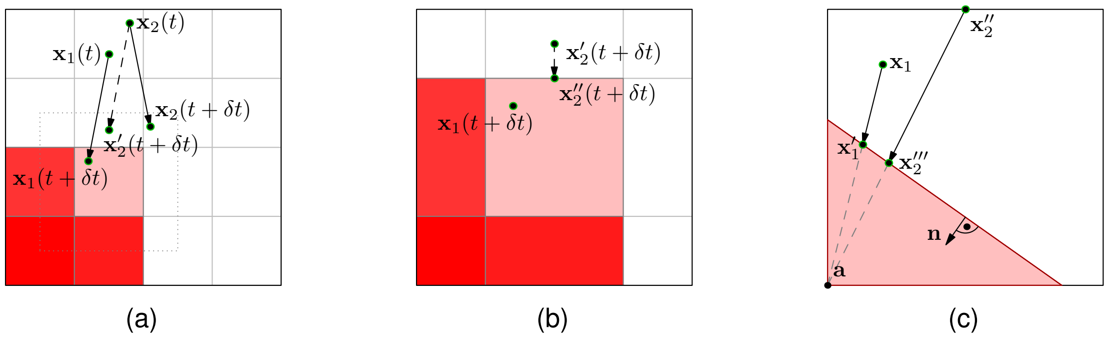

$\mathbf{x}_2(t)$
$\mathbf{x}_1(t)$
$\mathbf{x}_2(t+\delta t)$
$\mathbf{x}_2'(t+\delta t)$
$\mathbf{x}_1(t+\delta t)$
(a)
$\mathbf{x}_2'(t+\delta t)$
$\mathbf{x}_2''(t+\delta t)$
$\mathbf{x}_1(t+\delta t)$
(b)
$\mathbf{x}_2''$
$\mathbf{x}_1$
$\mathbf{x}_1'$
$\mathbf{x}_2'''$
$\mathbf{n}$
$\mathbf{a}$
(c)

Figure 3.19: Particle corrector method in multiphase flows with PLIC interface reconstruction. (a) After each advection step, it is checked whether some particles left the original phase ($f > 0$), and if so, the position is computed with the displacement vector of the nearest phase-consistent particle. (b) (Dotted rectangle from (a)) If it is still not in cell with $f > 0$, the particle is moved in the direction of the nearest such cell up to its boundary. (c) Afterwards, the particles outside the volume enclosed by PLIC patch are projected onto the PLIC plane in the direction of $\mathbf{a}$.

the standard integration methods, i.e., the particles may stray off to the other phase during advection. However, assuming that physically-based phase transitions do not occur in the simulation, the particles must stay in the same phase throughout advection. To meet this requirement, and to provide plausible results with the available information, a three-stage approach is introduced that utilizes the scalar field f to correct positions of stray particles during advection.

Particle Corrector

After each integration step between t_k and t_{k+1}, a three-stage procedure is applied to each particle that left the assigned phase in order to translate it back to its original phase. In the first stage, the neighborhood of the particle is searched for valid particles (i.e., particles that remain in the assigned phase at t_{k+1}). That is, in the $3 \times 3 \times 3$ cell neighborhood at time t_k (i.e., before the current advection step) the nearest neighbor $\mathbf{x}_p$ (if any) is chosen. Its displacement vector $\mathrm{d}\mathbf{x}_p = \mathbf{x}_p(t_{k+1}) - \mathbf{x}_p(t_k)$ is applied to the stray particle, i.e., $\mathbf{x}'(t_{k+1}) = \mathbf{x}(t_k) + \mathrm{d}\mathbf{x}_p$ (Figure 3.19(a)). Since this does not guarantee that the particle will be back in the original phase, in the second stage, the nearest cell with $f > 0$ is searched (using a loop over neighboring cells), and the particle is translated to the boundary of this cell along the line connecting the particle with the cell center, as shown in Figure 3.19(b), where the new particle positions are denoted $\mathbf{x}_p''$. Afterwards, in the third stage, for each particle in the interface cell, the position of the PLIC patch is calculated at the current simulation time step. Then, it is tested if the particle position lies within the volume enclosed by the PLIC patch. If not, the particle is translated along the line $\mathbf{x}_p - \mathbf{a}$ to the patch (Figure 3.19(c)).

3.2.6 Implementation

The visualization method has been implemented as a plugin in the ParaView visualization framework [10]. To improve the visual representation of the boundaries B and separation surfaces S, a smoothing filter is applied to the meshes extracted by the marching cubes algorithm. Although some fine details might be lost in the process, a smoothed representation improves perception, and hence provides a reasonable trade-off. For the visualization of the labeled inclusions in multiphase flow datasets, the labels were resampled on the geometric representation of the PLIC interfaces [85]. To visualize temporal evolution of the inclusions, the intermediate particle positions are stored after every n-th simulation time step ($n = 8$ in the datasets), see Figure 3.20. For the second stage of the particle corrector scheme (translation to a cell boundary), the line-box intersection algorithm proposed by Kay and Kajiya [92] was adopted.

Parallelization

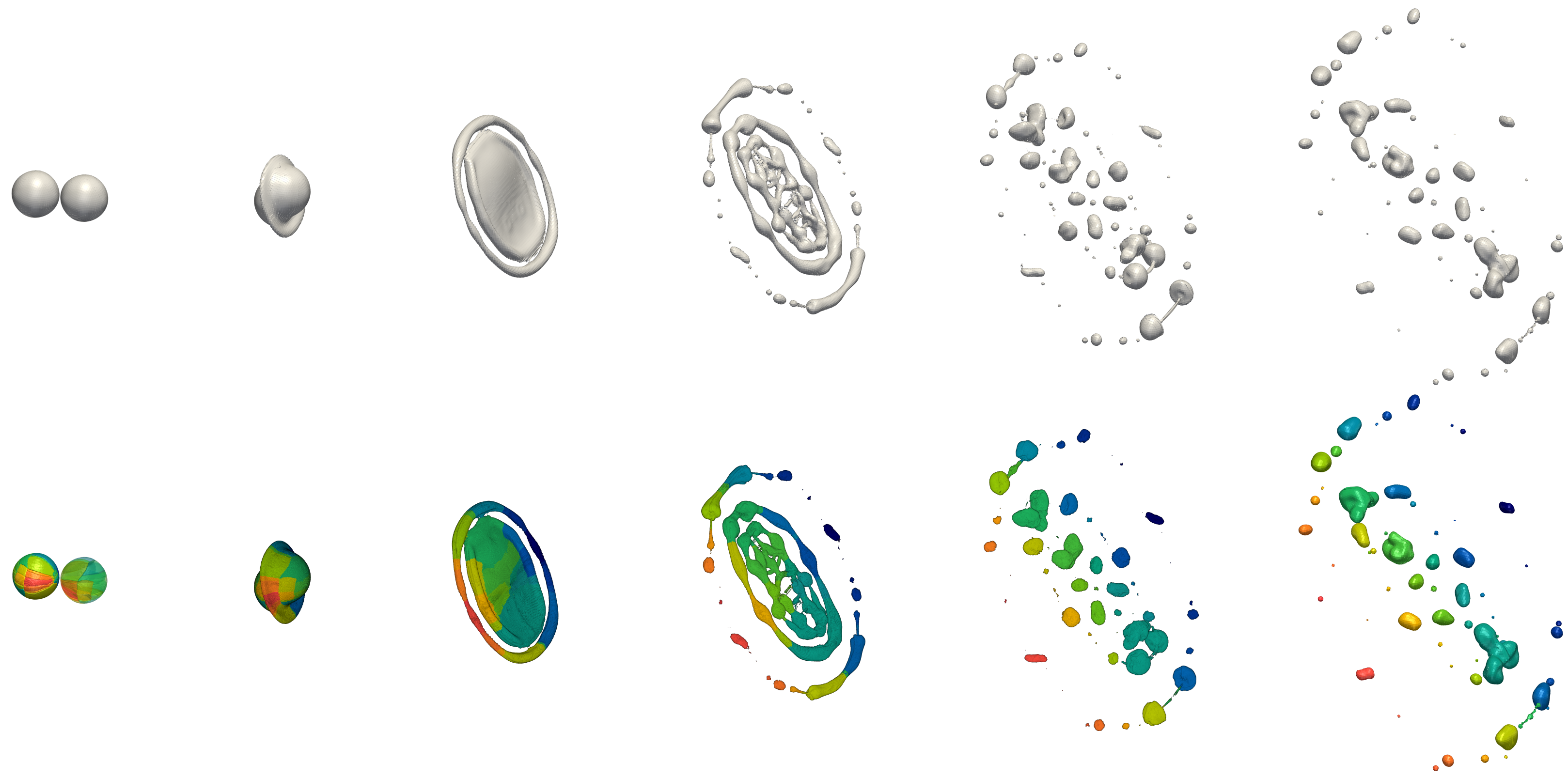
For large datasets, parallelization is necessary due to large memory requirements of the simulation data. For instance, in the *Non-Newtonian Jet* simulation (Section 3.2.7), each simulation time step consists of over 20 GiB, which could not be processed on a regular desktop computer. Hence, in this method, data parallelism (i.e., the approach adopted by ParaView) is employed, where the domain is split among several processes (possibly running on different machines), and each process works on a preassigned subdomain. In this approach, explicit communication is necessary for data exchange.

Due to the global nature of particle advection, those particles that leave a subdomain must be sent to the processes managing the subdomains the particles have entered. To avoid cases where the Runge-Kutta substeps are computed across neighboring subdomains, ghost cells around each subdomain are stored. Another global algorithm employed in this work is connected component labeling. The current implementation is based on the method proposed by Harrison et al. [67]. Additionally, the transfer of particle labels to their respective seed points must be handled explicitly. That is, for each particle, its ID within the original subdomain as well as the process ID responsible for that subdomain is stored. This information is then used to transfer the labels to the correct processes and to save the label at the correct position in the label array.

In addition to the process-level parallelization, thread-level parallelism provided by the OpenMP [129] parallel for-loops is also employed for particle advection to speed up iteration over particles.

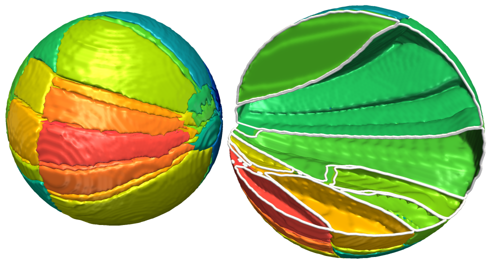
3.2.7 Results

The utility of the presented method is demonstrated on two datasets from direct numerical simulations of incompressible two-phase flow, where liquid and gas phases occur simultaneously. For the first dataset (*Peripheral Collision*), the refinement parameter is set to $r = 2$, whereas for the second (*Non-Newtonian Jet*), $r = 0$.

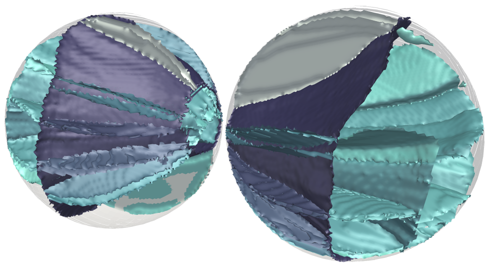


(a) $0.478\,\mathrm{ms}$ (b) $0.719\,\mathrm{ms}$ (c) $1.31\,\mathrm{ms}$ (d) $1.93\,\mathrm{ms}$ (e) $2.43\,\mathrm{ms}$ (f) $2.85\,\mathrm{ms}$

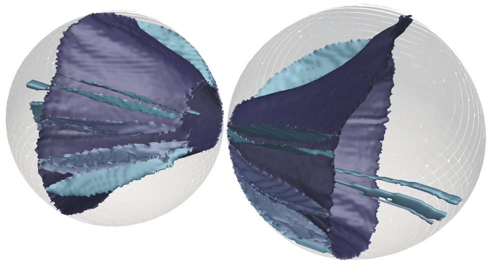
Figure 3.20: Visualization of separation in *Peripheral Collision* dataset with two peripherally colliding drops. The top row shows the extracted liquid interface for selected simulation time steps. After collision, the drops initially form a disk that disintegrates into many differently shaped droplets. In the bottom row on the left, volumetric contributions of each droplet from the last shown simulation time step are visualized, colored according to the labels of these droplets (shown on the right). The visualization reveals the topology of the drop disintegration spatially. For the four time steps in the middle, particles with colors corresponding to the drop labels are visualized, which allows for spatio-temporal tracking of the droplets.



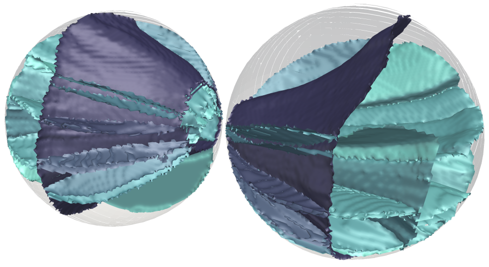
(a) $t_F = 2.660\,\mathrm{ms}$



(b) $t_F = 2.660\,\mathrm{ms}$



(c) $t_F = 1.925\,\mathrm{ms}$



(d) $t_F = 2.431\,\mathrm{ms}$

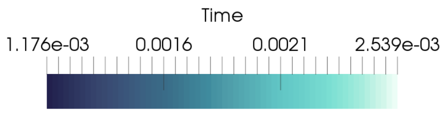


Figure 3.21: Visualization of droplet separation in *Peripheral Collision* dataset. (a) The section cutout (marked with white curve) of the boundaries B from Figure 3.20 exposes the inner topology structure—almost all developed drops radially expand from the collision center. (b) The separation surfaces reveal segmentation in the temporal context. Dark blue surfaces indicate the early separation of disk shaped and ring shaped droplets. Afterwards, both parts separate in similar time further, as indicated by turquoise color. (c), (d) Visualization of separation surfaces for the earlier time steps shows the evolution of the separation.

Peripheral Collision In the first dataset, two water droplets collide peripherally. This dataset consists of 256^3 cells, which cover a domain size of $1\,\mathrm{cm}^3$, and 461 time steps. In the top row of Figure 3.20, selected simulation time steps (from t_0 to t_F) are shown. The inclusions are investigated from time $t_0 = 0.478\,\mathrm{ms}$ (i.e., 60th time step, just before the collision) up to time $t_F = 2.85\,\mathrm{ms}$ (i.e., 461st time step, with small droplets resulting from the collision). The two merged drops form a flat shape that splits into an inner disk and an outer ring. Finally, both structures separate into small droplets.

The visualization in the bottom row reveals the volumetric correspondences between the volumes of the two initial drops and the drops at the final time step. Many small droplets are formed from narrow sections that extend radially from the collision center. These small drops, however, form only on the sides, whereas the volume sections at the bottom and top contribute to relatively large drops.

In Figure 3.21(a), a section of the right drop from Figure 3.20 has been cut out to reveal the inner structure of the volume contributions. Interestingly, almost all droplets that develop after the collision are formed from volumes that are close to the collision center and propagate outward to the back side. This structure bears some similarity to

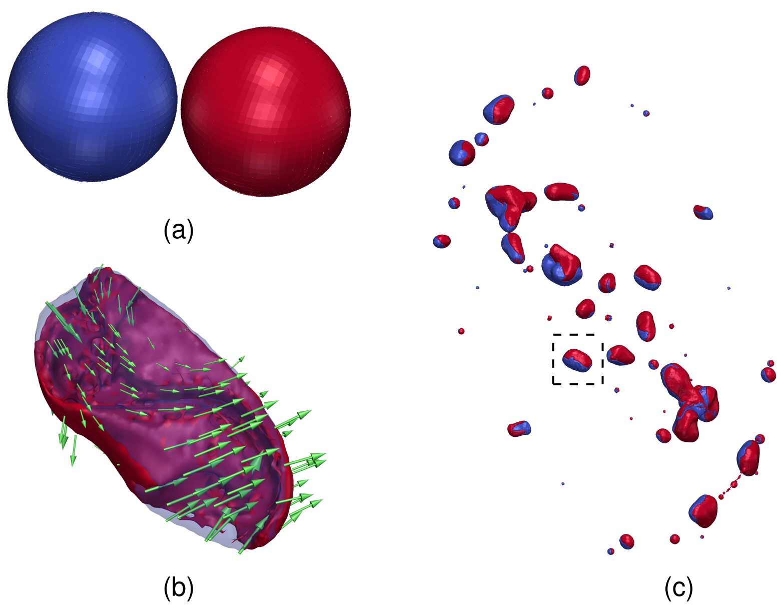


Figure 3.22: Separation of droplets computed backward in time for the *Peripheral Collision* dataset. (a) Drops at t_F are labeled (blue and red). (c) The volumetric contributions are visualized at t_0. (b) For the selected rotating drop (marked with box in (c)), the technique reveals S-shaped structure within the drop volume.

cracks in a solid material that propagate from the impact point. In Figure 3.21(b)–(d), separation surfaces S are shown. Here, the breakup of the inner disk and outer ring is apparent in the form of cone-like structures in both droplets. The surfaces within the cones have similar colors, indicating simultaneous separation of multiple droplets.

For this dataset, the method has been additionally applied in reverse time direction (i.e., by setting $t_0 = 2.85\,\mathrm{ms}$ and $t_F = 0.478\,\mathrm{ms}$) to reveal how exactly the initial two droplets contributed to the later smaller droplets. The result is shown in Figure 3.22. In Figure 3.22(a), the two droplets at time $t_F = 0.478\,\mathrm{ms}$ are visualized by the PLIC interface. They are colored by the connected component labels, and their contributions in the droplets at time t_0 are visualized in Figure 3.22(c). As can be seen, the volume from the red drop dominates on the upper right part of the droplets at later physical time. Figure 3.22(b) shows a selected droplet that rotates after separation from the disk-shaped drop. The boundary from the blue droplet is transparent to reveal the inner structure of the drop. The interface between the two boundaries B forms an S-shape, and there is distinguishable symmetry of the two volumes—it clearly shows the interplay between distribution of volumes originating from different inclusions and rotational motion of the resulting inclusion.

Non-Newtonian Jet The second dataset is a simulation of the injection of a jet made up of a non-Newtonian shear thinning aqueous solution of Praestol 2500 (0.3% weight) into air at $p_{\mathrm{atm}} = 1\,\mathrm{bar}$. The jet is introduced into a rectangular computational domain

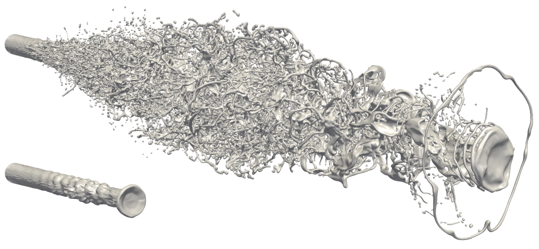


Figure 3.23: Selected time steps in the *Jet* dataset. On the bottom, the jet at time step $t_0 = 1.60$ms is about to break up into a multitude of droplets and ligaments. On the top, a moderately developed jet at $t_F = 3.97$ms is shown.

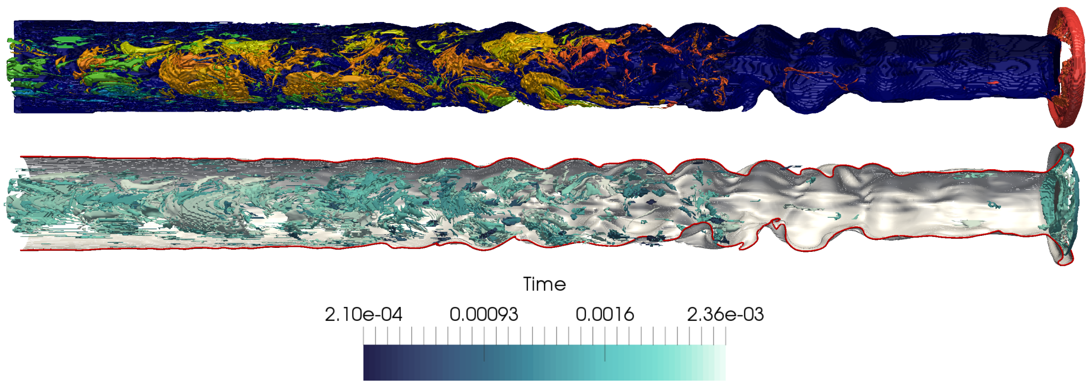


Figure 3.24: Separation boundaries B (top) and temporal separation surfaces S (bottom) in the *Non-Newtonian Jet* dataset, with front part cut out to reveal the inner structure. The jet disintegrates mostly in the rear part and at the very front, which bends back. Small dark blue separation surfaces indicate that small droplets disintegrate early.

with a diameter of $\mathrm{D} = 1.2$cm and with the velocity profile from a short nozzle with a mean velocity of $u = 75$m/s, and therefore, a Reynolds number $\mathrm{Re} = 19\,000$. The domain has a size of $42\mathrm{D}$ in the direction of the injection and of $10\mathrm{D}$ in each of the directions of the injection plane. The domain is discretized over $2688 \times 512 \times 512$ cells. The temporal development of the jet is shown in Figure 3.23. At $t_0 = 1.60$ms, the jet has started to interact with the surrounding gas, resulting in bending back of the jet tip and the development of surface waves. Due to the high Reynolds number, the jet becomes unstable very quickly, and at $t_F = 3.97$ms, the core has disintegrated substantially and has mostly broken up into ligaments and droplets—the atomization has begun.

Figure 3.24 shows the boundaries B and separation surfaces S at t_0. Small polygons have been removed for clear overview. Apparently, in the investigated time interval, the front of the jet does not undergo strong disintegration (except for the tip), whereas in the rear, many segments imply the origins of elongated droplets that develop later. The separation surfaces at the bottom of the figure are uniformly distributed, although small droplets detach earlier, as indicated by small dark blue surfaces.

In Figure 3.25, the temporal evolution of one selected ligament (colored) is shown with the entire jet (gray) in the background, from t_0 at the top to t_F at the bottom. The liquid mass, which will later form the ligament, is initially distributed over one third of the jet length. As the liquid mass starts to surface, it is decelerated due to the interaction with the surrounding atmosphere. In the meantime, the rest of the mass, which origi-

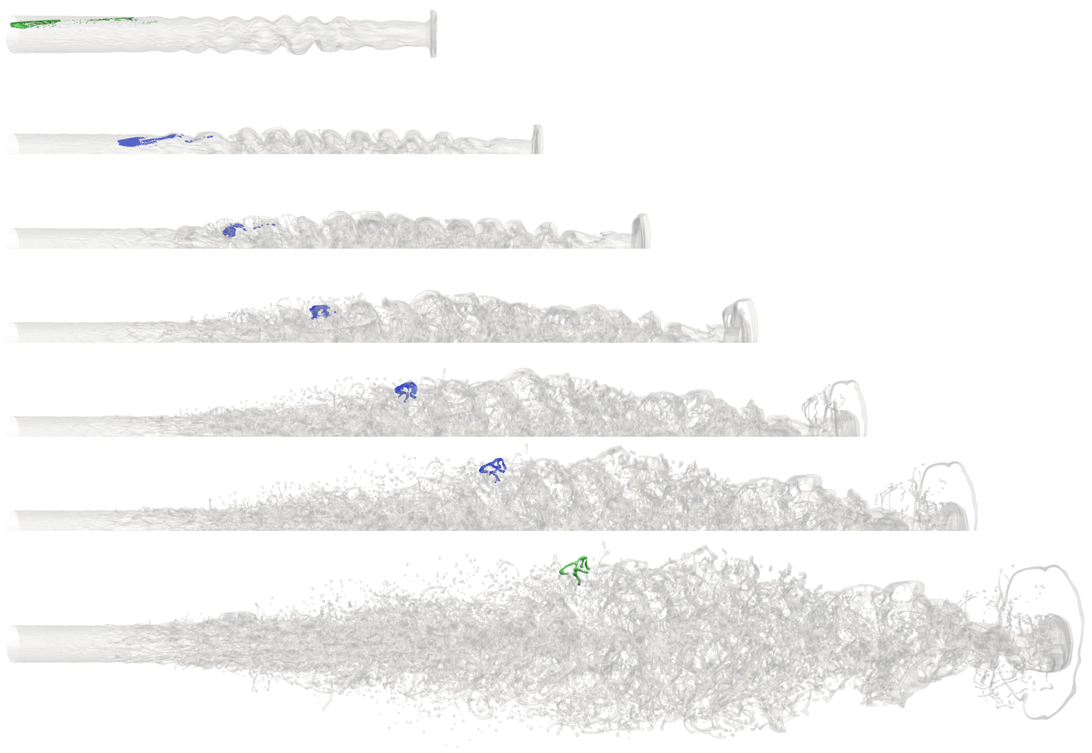
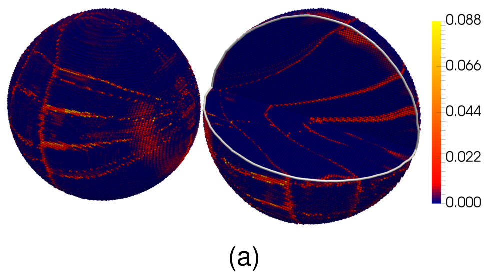


Figure 3.25: Selected volumetric contributions (green) in the *Non-Newtonian Jet* dataset at t_0 in the top figure and the corresponding ligament at t_F in the bottom figure. In the middle figures, the temporal evolution of particles (blue) for the selected contribution is shown at intermediate time steps. The volumetric contribution curls into the final ligament as it is pushed away from the jet core by the surrounding air.

nated in the central part of the jet, has approached the surfaced part, and by the third time step, the bulk of the liquid mass is located within the first surface wave, while a few parts are located in the next wave. From the underlying jet, one can see that the waves are already starting to deform strongly from the shape of a periodic wave. At the next time step, almost the whole liquid mass has merged and moved even closer toward the surface, where it has reached its most compact form. In the next time step, the surface wave has disintegrated and the liquid mass has become a complex branching ligament. With time, the ligament moves further away from the core axis, interacting more strongly with the atmosphere. Finally, the first droplet breaking off the ligament in the final time step can be observed.

The proposed visualization method allows to make interesting observations about the jet simulation. Contrary to the seemingly radial expansion in Figure 3.23, the flow field inside the core is creating the analyzed inclusion from liquid mass scattered throughout the core, as revealed in Figure 3.25. Furthermore, this figure is an excellent representation of the different steps of the primary jet break-up showing the formation of surface waves, the deformation of the waves, the rupturing of the waves into ligaments, and the stretching and break-up of the ligaments into droplets.



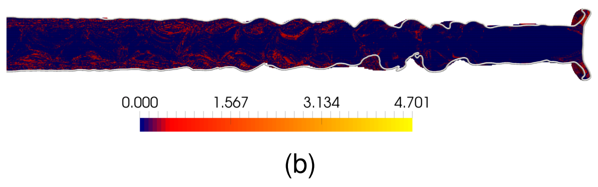


Figure 3.26: Visualization of the accumulated displacement for the investigated multiphase flow datasets. Cross-sections are shown as white lines. The displacement values ε_p of the integrated particles are mapped on the corresponding seed points. Both in the *Peripheral Collision* (a) and *Non-Newtonian Jet* (b) datasets, larger values correspond to the boundary regions, which are most unstable.

The presented method allows for a simple and intuitive way to analyze the spatial and temporal development of the primary break-up of liquid jets. With this method, an insight into the movement patterns of selected inclusions of the break-up process could be gained which would otherwise have been hard and unlikely to obtain.

Analysis of the Corrector in Multiphase Flow

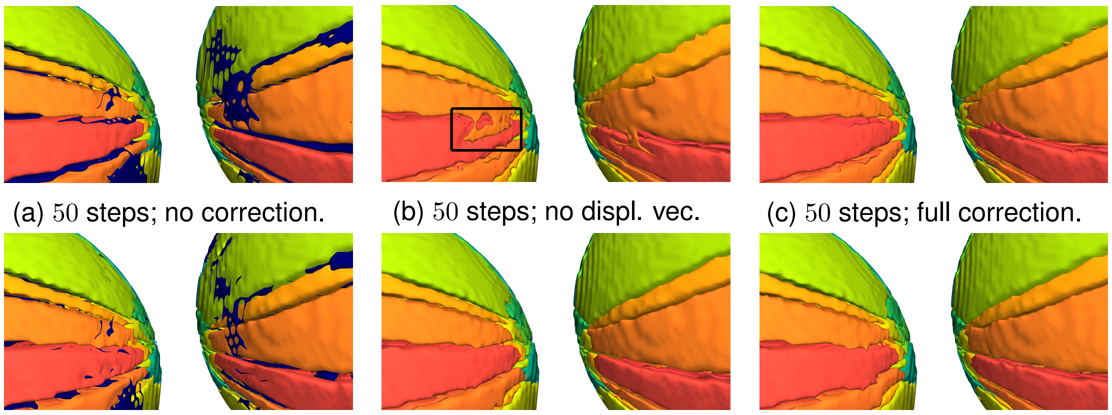
To quantify the amount of applied correction that is necessary to keep the particles within the initial phase in multiphase flow (Section 3.2.5), the introduced displacement is accumulated for each particle over all time steps i:

$$\varepsilon_p = \sum_{i \in I} \left(|\mathbf{x}'_{p,i} - \mathbf{x}_{p,i}| + |\mathbf{x}''_{p,i} - \mathbf{x}'_{p,i}| + |\mathbf{x}'''_{p,i} - \mathbf{x}''_{p,i}| \right), \tag{3.12}$$

where the first term is the translation with the displacement vector of the nearest phase-consistent particle (Figure 3.19(a)), the second term in the sum corresponds to the translation to the nearest cell with $f > 0$ (Figure 3.19(b)), and the third term is the translation to the PLIC patch in the interface cells (Figure 3.19(c)). For the analysis, the value ε_p is displayed at the seed positions $\mathbf{s}_p$ for the *Peripheral Collision* and *Non-Newtonian Jet* datasets in Figures 3.26(a), and (b), respectively. The particles that were translated most correspond with the boundary positions. This is easy to interpret, since the particles that most probably leave the liquid phase are close to the phase interface at unstable regions. Although the maximum value of the accumulated correction is relatively large (0.07 of the domain size for *Peripheral Collision* and 0.06 of the domain size for the *Non-Newtonian Jet*), it occurs only for small number of particles, and the correction for most of the particles is considerably smaller (as indicated by dark orange color in the figures).

Robustness

In Figure 3.27, the sensitivity of the particle advection to the temporal resolution of the data and different variants of the particle corrector are investigated. The selected



(a) 50 steps; no correction. (b) 50 steps; no displ. vec. (c) 50 steps; full correction.

(d) 400 steps; no correction. (e) 400 steps; no displ. vec. (f) 400 steps; full correction.

Figure 3.27: Analysis of corrector robustness for 50 (top) and 400 (bottom) time steps. (a), (d) No correction applied, with many stray particles (corresponding blue surfaces). (b), (e) Only second and third stage. (c), (f) full correction. Incorrectly assigned particles at the collision region in (b) are corrected either with (e) better temporal resolution or with (c) full correction.

droplet region from the *Peripheral Collision* dataset is particularly prone to error due to the collision, which is difficult to capture during integration. For the analysis, 50 (top row) and 400 (bottom row) simulation time steps were used, spanning the same time interval. The left column shows the resulting boundary reconstruction for particles with no correction applied, in the middle column, the particles were moved to the closest cell and consequently to the PLIC patch without the adjustment by the displacement vector. In the right column, full correction was applied. With no correction, even with high temporal resolution, there are regions containing particles that left the initial phase, as indicated by dark blue surfaces. Interestingly, for two-stage correction, the stray particles are gone, but in Figure 3.27(b), some particles at the front are incorrectly assigned. With full correction, little difference can be seen between both resolutions. This shows that the correction method can provide reliable results also for lower temporal resolution, although in the presented experiments, a higher temporal resolution was used to further increase the reliability of the results.

Performance

The computation of the visualization for the *Peripheral Collision* dataset was performed on a commodity desktop with Intel i7 3.6 GHz processor (4 cores) and 32 GiB RAM. For the *Non-Newtonian Jet* dataset, a Cray XC40 System was used with 64 nodes, each with two Intel Xeon E5 processors and 128 GiB of memory, and one process running per node. Table 3.3 shows the configurations (i.e., number of simulation time steps and the number of seeded particles) and timings (i.e., total time, average time for one advection step and extraction of S, and time for extraction of B) for the three datasets. Although both datasets have comparable numbers of particles, the computation of the *Non-Newtonian Jet* dataset is not much faster, despite the parallelization. This is due to the fact that with the data parallelism in a distributed system, the particles are not evenly distributed among the processes. Moreover, the exchange of particles across the subdomains introduces some overhead after each integration step. However, since the computation is done off-line, this performance cost is acceptable considering clear implementation design in the data-parallel approach.

Table 3.3: Computation times for the three datasets. For *Peripheral Collision* (Drops), the measurement was performed on a desktop computer, and on the Cray system for the *Non-Newtonian Jet* (Jet) dataset. “Time” is the total computation time, “Adv. Step+S” is the time for one step (done for each simulation step) of advection and extraction of S. “B” is time for extraction of boundaries (done once).

Dataset	#Steps	#Particles	Adv. Step+S	B	Total
Drops	400	1.1e6	54.7 s	5.8 s	6.8 h
Jet	237	1.7e6	29.5 s	6.1 s	3.9 h

The performance dependency on parameter r has been also investigated. As Table 3.4 shows, in case of *Peripheral Collision*, the difference between $r = 0$ and $r = 1$ is relatively small arguably due to better usage of thread-level parallelism. For $r = 2$, on the other hand, the number of particles increases the computation time substantially. For the *Non-Newtonian Jet*, with 8-fold increase in particle number, there is 8-fold increase in time for the advection step and the total time increases about 5 times, suggesting that the interprocess communication constitutes considerable time, and is less dependent on the number of particles.

Table 3.4: Computation times for different values of parameter r for *Peripheral Collision* and *Non-Newtonian Jet* datasets. Same notation as in Table 3.3.

Dataset	r	#Particles	Adv. Step+S	B	Total
Drops	0	1.8e4	2.1 s	0.3 s	0.9 h
Drops	1	1.4e5	7.9 s	0.9 s	1.5 h
Drops	2	1.1e6	54.7 s	5.8 s	6.8 h
Jet	0	1.7e6	29.5 s	6.1 s	3.9 h
Jet	1	13.6e6	234.0 s	46.2 s	18.9 h

For the visualization, a commodity desktop computer for all datasets could be used, since the data produced by the technique (i.e., the particle data and mesh for boundaries B) does not exceed 1 GiB, even for the *Non-Newtonian Jet* dataset.

Visualization of Liquid Interface Dynamics | 4

In the previous chapter, visualization techniques for droplet groups and jets were presented, where the focus was on the analysis of phenomena leading to breakup and merge events as well as on detailed investigation of liquid separation dynamics. Equally important in the analysis of two-phase flow is the study of liquid interfaces, and in this chapter, several visualization techniques are presented for this purpose that focus on various aspects of liquid interfaces and the related processes.

In scientific two-phase flow simulations, the volume of fluid method is typically used for interface tracking. Furthermore, for reconstruction, piecewise linear interface calculation is usually employed, since it provides a reasonable trade-off between accuracy and computational effort. However, since the reconstruction influences the quality of a simulation, and therefore the observed phenomena, such as droplet breakups, it is judicious to investigate the reconstruction properties visually. Consequently, a framework is presented for the visual analysis of the piecewise linear interface calculation together with its implications on the overall simulation [85]. In this framework, PLIC reconstruction is interpreted as an isosurface extraction problem from the first-order Taylor approximation of the underlying volume of fluid field. This enables error analysis and geometric representation of the reconstruction, including the fluxes involved in the simulation. At the same time, it allows for generalization of PLIC to higher-order approximation.

The observed two-phase flow dynamics and related topological changes are strongly influenced by the interplay of fluid dynamic and surface tension forces. Surface tension force has a particular importance in the small-scale phenomena, where it dominates the deformation dynamics due to the large curvature values. Combined with fluid dynamic forces, it leads to the generation of intricate interface formations and potential phase disintegration. The analysis of these processes can provide better understanding of interface instability and potential breakups. To this end, a solver-based approach for curvature computation is implemented, and from this, the surface tension together with the respective velocity field are derived to investigate the effect of surface tension on interface dynamics. For the analysis of interface deformation, induced by both the surface tension-derived and the simulation-based velocity fields, an approach is presented to compute interface stretching and bending based on metric and shape tensors, respectively. For the visual investigation, the eigenpairs of these tensors are used that define the direction and strength of the respective deformation type.

In two-phase flow, an important research direction is the investigation of water droplets on the surface of a high voltage insulator. These droplets can lead to static discharges that have detrimental effect on insulators. For the numerical analysis of the resulting electric field problems, finite element-based electric field simulations are commonly used, where the employed edge-conforming representations of the electric field ensure the desired field discontinuity at material boundaries. However, a major drawback so far has been the lack of appropriate visualization techniques, necessitating a resampling step with all the involved drawbacks, including artifacts in the form of imposed continuity across material boundaries. Here, a visualization framework is presented for the discrete field data resulting from such simulations on quadratic tetrahedral grids [86]. It evaluates edge-conforming data by means of vector basis functions and provides different visualization approaches for the investigation of the electric field at material boundaries.

4.1 Visualization of Solver Interface Reconstruction

In this section, a technique is presented that allows for a detailed visual analysis of two-phase flow simulations in terms of interface modeling. As indicated in the Fundamentals (Chapter 2), in CFD, two-phase flow simulation is achieved by advecting an additional scalar field that represents cell-wise material distribution. A widely used approach for tracking phase interfaces is PLIC due to Youngs [208, 209] which balances tracking accuracy and algorithmic simplicity (and therefore computational cost). However, since the tracking step must be repeated in each simulation step, and PLIC produces a linear approximation of the interface, it is important to investigate the influence of the interface reconstruction on the simulation outcome.

The primary focus of the visualization technique presented in this section is therefore to provide a set of simulation-oriented visual analysis tools that allow scientists to evaluate and assess the quality of simulation runs and to gain better understanding of the numerical processes that affect the results. Specifically, by identifying PLIC reconstruction as an isosurface extraction problem from the first-order Taylor approximation of the VOF field, a versatile framework is obtained. On the one hand, it allows to derive error bounds on the implicit approximation of the VOF-field, on the other hand, it provides several geometry-based error measures with respect to the shape of the reconstruction and the discontinuities at cell boundaries. It also provides geometric representations of the volume enclosed by PLIC as well as the flux of the VOF field. Finally, it generalizes PLIC reconstruction to higher-order approximation which allows for more rigorous investigation of the interface reconstruction. [1]

4.1.1 Related Work

Many visualization methods have been proposed for material interface reconstruction. The related work in this topic can be found in the Fundamentals (Chapter 2). Some of these works visualize 3D PLIC patches for comparison with their smooth interface reconstruction techniques, they do not, however, concentrate on the PLIC interface reconstruction quality.

Interface reconstruction for multiple materials was analyzed by Prilepov et al. [143] who constructed volume fraction function and employed isovalue fields to represent the material interfaces. Li et al. [105] proposed a technique for multiphase interface tracking which combines implicit and explicit contouring schemes and can track complex topology changes. The interface reconstruction was also investigated for the boundary extraction in computed tomography (CT) volumes with multiple materials [128]. However, while these works concentrate on appropriate visualization and extraction of the resulting features, here, the focus is on visualization of interface reconstruction with respect to the simulation process. Hence, the presented technique does not aim at providing a smooth representation.

[1] Parts of this section have been published in: [85]

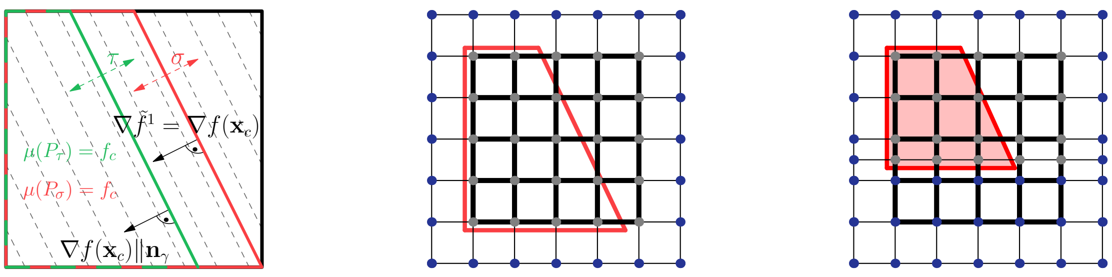

τ
σ
$\nabla\tilde{f}^1 = \nabla f(\mathbf{x}_c)$
$\mu(P_\tau) = f_c$
$\mu(P_\sigma) = f_c$
$\nabla f(\mathbf{x}_c) \| \mathbf{n}_\gamma$
(a)
(b)
(c)

Figure 4.1: (a) PLIC as isosurface extraction problem in $\tilde{f}^1$. First-order approximation $\tilde{f}^1$ exhibits planar isosurfaces (dashed lines) perpendicular to $\nabla f(\mathbf{x}_c)$ (red), and so does PLIC by construction (green). Hence, instead of adjusting τ such that the volume $\mu(P_\tau)$ of the enclosed polyhedron P_τ equals f_c, one can adjust isolevel σ accordingly. (b) and (c) Discretization (gray nodes) of $\tilde{f}^k$ for $k > 1$ within a simulation cell (bold grid). (b) An isosurface (red polygon) representing the PLIC polyhedron P is obtained by setting all boundary nodes (blue) of the supersampling grid to `-FLT_MAX`. (c) Cells are split at "reversely advected" cell face to generate flux volume (pale red), $\tilde{f}^k$ is interpolated at the new (gray) nodes. Nodes below the face are set to `-FLT_MAX` to obtain the respective isosurface representation (red). © 2013 IEEE.

Only few visualization methods have been developed that investigate the behavior of solvers with interfaces. Obermaier et al. [125] analyzed the stability of reconstructed interfaces by comparison with time surfaces. More recently, Fernandes et al. [47] examined the interrelation of coupled fluid and structure solvers in an in-situ visualization framework.

As the reconstruction of PLIC patches involves gradient estimation, a related work is by Hossain et al. [75] who presented gradient estimation methods for field data on regular lattices. These methods were further improved by Alim et. al [4] where storage overhead was reduced.

4.1.2 PLIC Reconstruction for Visualization

The PLIC patches are planar and perpendicular to the gradient of $f(\mathbf{x})$, and in the simulation, their position is defined by distance τ from a cell node $\mathbf{a}$ that would be swept last if the patch were translated along the gradient direction (Figure 2.2(c)). This distance is chosen to ensure that the volume V_P of a polyhedron P enclosing the cell's liquid side equals f_c. Central to the presented visualization technique is the observation that the properties of PLIC planarity and perpendicularity to $\nabla f(\mathbf{x})$ can also be realized by isosurfaces extraction of the first-order Taylor approximation of f centered around the cell center $\mathbf{x}_c$, with $f(\mathbf{x}_c) = f_c$:

$$\tilde{f}^1(\mathbf{x}_c + \mathbf{h}) := f(\mathbf{x}_c) + (\nabla f(\mathbf{x}_c)) \cdot \mathbf{h}, \tag{4.1}$$

where $\mathbf{x}_c = (x_1, x_2, x_3)^\top$, $\mathbf{h} = (h_1, h_2, h_3)^\top \in \mathbb{R}^3$. Hence, the PLIC patches can be represented and obtained by marching cubes isosurface extraction from $\tilde{f}^1$, separately within each cell. In this formulation, instead of finding τ such that the corresponding polyhedron volume $V(P_\tau)$ equals f_c, an isolevel σ of $\tilde{f}^1$ is selected that ensures $V(P_\sigma) = f_c$, as illustrated in Figure 4.1(a). In other words, the polyhedron P, and consequently its volume V_P, is parameterized by isosurface σ, instead of translation τ.

The above formulation allows to obtain higher-order approximations of the interface by isosurface extraction from higher-order Taylor approximations $\tilde{f}^k$ of f. This requires $\tilde{f}^k$, which in the form of multivariate Taylor approximation of f, centered at the cell center $\mathbf{x}_c$ reads

$$\tilde{f}^k(\mathbf{x}_c + \mathbf{h}) := \sum_{|\boldsymbol{\alpha}| \le k} \frac{\partial^{\boldsymbol{\alpha}} f(\mathbf{x}_c)}{\boldsymbol{\alpha}!} \mathbf{h}^{\boldsymbol{\alpha}}$$

with $\boldsymbol{\alpha} \in \mathbb{N}_0^3$, using *multi-index notation*. For $k = 2$ this results in the second-order Taylor approximation

$$\begin{aligned}\tilde{f}^2(\mathbf{x}+\mathbf{h}) = f(\mathbf{x}) & \\ &+ \left.\frac{\partial f(\mathbf{x})}{\partial x_1} h_1 + \frac{\partial f(\mathbf{x})}{\partial x_2} h_2 + \frac{\partial f(\mathbf{x})}{\partial x_3} h_3 \right\} 1^{st} \\ &\left.\begin{aligned} &+ \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} h_1 h_2 + \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_3} h_1 h_3 + \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_3} h_2 h_3 \\ &+ \frac{\partial^2 f(\mathbf{x})}{2\partial x_1^2} h_1^2 + \frac{\partial^2 f(\mathbf{x})}{2\partial x_2^2} h_2^2 + \frac{\partial^2 f(\mathbf{x})}{2\partial x_3^2} h_3^2 \end{aligned}\right\} 2^{nd}\end{aligned} \tag{4.2}$$

with the respective zeroth, first, and second-order terms. The investigations in this section are constrained to a maximum of order two, although any order is straightforward to use within the framework.

4.1.3 Discretization

To extract the PLIC patches as isosurfaces of $\tilde{f}^k$ for $k = 1$ (i.e., up to 1st-order terms in Equation 4.2), the computation of ∇f in each cell is required. To ensure consistency with the simulation process, the respective gradient estimation process is employed. The gradient of cell-centered quantities such as f_c is obtained by computing the partial derivatives using finite differencing at the centers of the cell faces between the respective cells, e.g., the partial derivative in x-direction is computed at the center of the yz-face. Then, the partial derivatives at the centers of 2×2 co-planar faces (in this case yz-faces) are averaged and stored at the node that is shared between those faces. Repeating this procedure in y- and z-direction provides the node-based gradient. Finally, it is interpolated to the cell center. After computing $\tilde{f}^k$ at cell nodes, the isosurface extraction algorithm is applied, which readily produces the PLIC patches.

One remaining step in the computation of PLIC patches is finding the isovalue σ which produces the right offset along $\nabla f(\mathbf{x}_c)$ (Figure 4.1(a)). To this end, the volume $V(P_\sigma)$ of the PLIC polyhedron (Section 4.1.2) must be determined. Since the approach should address generic cases with $k \geq 1$, a possible procedure would be to apply triangulation to close the isosurface along the respective face parts of the original cell. This would be, however, a demanding task with respect to computational cost and implementation complexity, particularly for $k > 1$. Instead, the marching cubes algorithm is also applied for this task. An auxiliary uniform grid G_c of 3^3 cells is constructed, with the original cell as the inner cell whose node values were obtained from $\tilde{f}^k$. The values on the outer nodes N_{out}, on the other hand, are set to a very large negative value γ (e.g., $\gamma =$ `-FLT_MAX`). This way, any isosurface is tightly closed along the face regions of the original cell, providing the triangulated representation of P_σ corresponding to isolevel σ. The inaccuracy introduced by the additional volume corresponding to the faces external to the original cell is negligible (in the order of numerical accuracy), because, as γ dominates the $\tilde{f}^k$-values, the additional parts of the isosurface are located very close to the faces of the original cell. However, since this is only the case if the partial derivatives of f are reasonably bounded, i.e., below the order of γ, the modulus of each of the values within G_c is tested if it exceeds $-\gamma \cdot 10^{-3}$ and the user is warned in such a case (this, however, never occurred in the investigated datasets). The volume can be obtained from the resulting triangulation as $V(P_\sigma) = \sum_{t \in P_\sigma} \mathbf{v}_1 \cdot (\mathbf{v}_2 \times \mathbf{v}_3)/6$, with $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ being the three vertices of triangle t.

With the developed method for the volume computation from the polyhedron enclosed by the PLIC interface, it is now possible to determine the isolevel σ such that $V(P_\sigma) = f_c$. The problem is addressed by iterative optimization, i.e., the bisection method is applied to find σ. Please see the original paper [85] for the details on the algorithm.

Higher-Order Interface Approximation The above method for the generation of the first-order PLIC interface already provides a basis for the analysis of the reconstruction, as discussed in Section 4.1.4. However, for higher-order interface approximation, which is used for comparison and visualization of the approximation error with respect to the curvature of the interface, additional steps are necessary to obtain the geometry representing the higher-order approximation.

For $k = 2$, beyond the gradient, second-order partial derivatives must be computed. To ensure consistency with the simulation code also with this respect, a finite difference scheme based on the simulation of surface tension is employed. However, the curvature estimation code in the solver computes $\nabla(\nabla f / \|\nabla f\|)$ and hence, cannot be directly utilized. Instead, the Hessian $\nabla(\nabla f)$ using the same scheme was implemented for the visualization. Specifically, it is computed from the node-based gradient by evaluating the finite differences between neighboring nodes along the cell edges. This provides a partial x-derivative at the center of each x-edge. To obtain the cell-centered Hessian, these partial derivatives (in fact its columns) at the four x-edges of a cell are averaged. Repeating this process in y- and z-direction yields the full Hessian.

To compute the polyhedron volume $V(P_\sigma)$ of higher-order approximations, it is necessary to subdivide each simulation cell by a factor $n > 1$ into n^3 subcells and resample

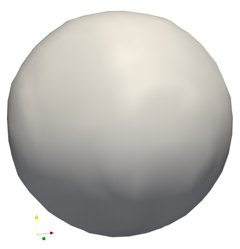


Figure 4.2: *Sphere* dataset with interface extracted using standard marching cubes algorithm. PLIC interface reconstruction and visualization measures are demonstrated for this dataset in Figure 4.3.

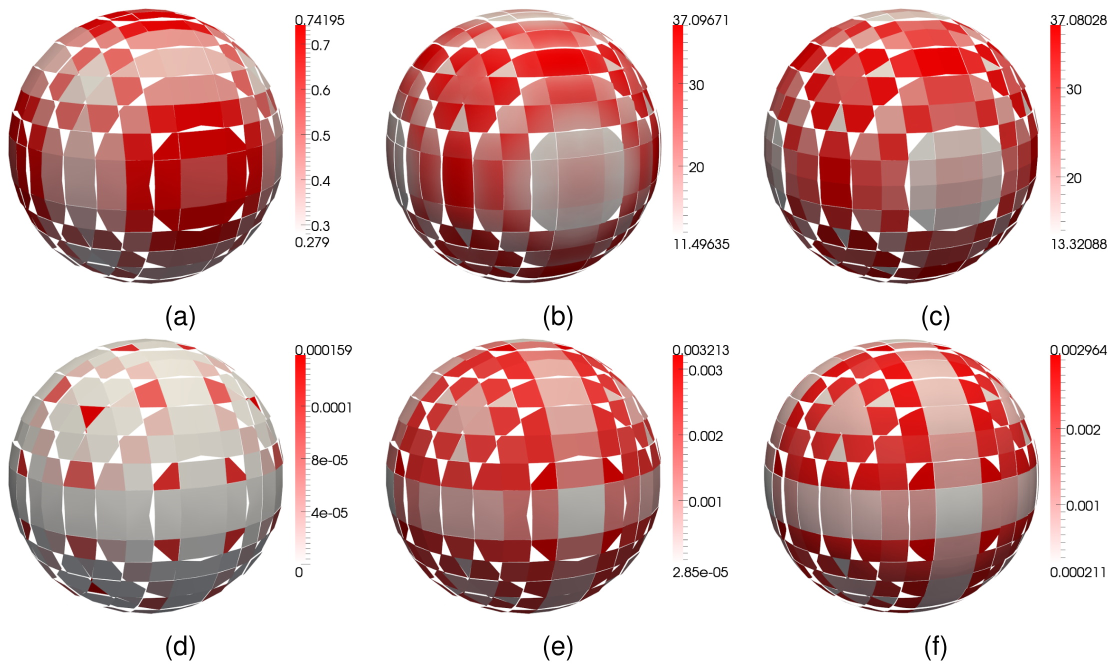


Figure 4.3: Visualization measures applied to the *Sphere* dataset. Measures: (a) bound B on approximation error with respect to f, (b) second-order PLIC patches colored with surface curvature $\kappa_{\max}$, and (c) cell-wise maximum $\hat{\kappa}_{\max}$ of (b) colored on the PLIC patches. The radius of the sphere according to the PLIC reconstruction is $0.0505\,\mathrm{m}$ (initialized as $0.05\,\mathrm{m}$), hence the curvature should be ideally $20\,\mathrm{m}^{-1}$. (d) Minimum discontinuity $\delta_{\min}$, (e) maximum discontinuity $\delta_{\max}$ on PLIC and (f) on second-order patches. While (d), (e), and (f) account for the displacement of the patches along $\nabla f(\mathbf{x})$, (a), (b), and (c) consider only partial derivatives of f and are hence not directly dependent on the displacement. © 2013 IEEE.

$\tilde{f}^k$ at the resulting nodes. Similarly to the case $k = 1$, an auxiliary subgrid G_c is constructed, with $n+2$ subcells in each dimension, and with the (now subdivided) original cell occupying the inner part, as shown in Figure 4.1(b). By setting the values on outer nodes to γ and applying the marching cubes algorithm, the triangulation of P is obtained, which is used in the bisection method (with $k = 2$) for the estimation of σ.

4.1.4 Measures for Visual Analysis of Interface Reconstruction

In this section, several visualization quantities are introduced that measure the error of the interface reconstruction.

Bound on Approximation Error The first error measure of PLIC reconstruction is the bound on the approximation error of $\tilde{f}^k$ which is calculated as follows. According to the multivariate version of Taylor's theorem, since f is $k+1$ times continuously differentiable (Section 4.1.2) within the cell volume C, there is $R_{\boldsymbol{\beta}} : \mathbb{R}^3 \to \mathbb{R}$ with $\boldsymbol{\beta} \in \mathbb{N}_0^3$ such that

$$f(\mathbf{x}_c + \mathbf{h}) = \sum_{|\boldsymbol{\alpha}| \le k} \frac{\partial^{\boldsymbol{\alpha}} f(\mathbf{x}_c)}{\boldsymbol{\alpha}!} \mathbf{h}^{\boldsymbol{\alpha}} + \sum_{|\boldsymbol{\beta}| = k+1} R_{\boldsymbol{\beta}}(\mathbf{x}) \mathbf{h}^{\boldsymbol{\beta}} .$$

From this follows

$$|R_{\boldsymbol{\beta}}(\mathbf{x})| \le \frac{|\boldsymbol{\beta}|}{\boldsymbol{\beta}!} \max_{|\boldsymbol{\gamma}| = |\boldsymbol{\beta}|} \max_{\mathbf{y} \in C} |\partial^{\boldsymbol{\gamma}} f(\mathbf{y})| =: M , \tag{4.3}$$

with $\mathbf{x} \in C$, leading to the uniform estimate that bounds the approximation error of the k^{th}-order Taylor approximation

$$f(\mathbf{x}) - \tilde{f}^k(\mathbf{x}) \le M c^{k+1} \tag{4.4}$$

within C, with c being the maximum of x-, y-, and z-extent of the cell. Hence, the upper bound B for the approximation error of f with respect to PLIC reconstruction is obtained by $k = 1$ in Equation 4.4 and inserting Equation 4.3:

$$B := \frac{2}{2} \max_{|\boldsymbol{\gamma}|=2} \max_{\mathbf{y} \in C} |\partial^{\boldsymbol{\gamma}} f(\mathbf{y})| \cdot c^2 = \max_{\mathbf{y} \in C} \|\nabla(\nabla f(\mathbf{y}))\|_{\max} \cdot c^2 , \tag{4.5}$$

with the maximum norm $\|A\|_{\max} = \max\{|a_{ij}|\}$. Thus, to obtain B, the largest modulus of the elements of the Hessian of f within C must be determined and then multiplied by the square of the largest cell extent.

As detailed in Section 4.1.3, the columns of the Hessian are bilinearly interpolated at the cell center from those at the cell edges. Hence, the absolute maximum of the elements of the Hessian within the cell is the absolute maximum of the elements of the Hessian columns determined at the cell edges, which is easily evaluated and gives the error bound B on the first-order approximation of f.

Since PLIC reconstruction corresponds to first-order approximation of f, B not only provides a conservative bound on possible inaccuracies due to the advection of the f-field, but also indicates more general discretization problems of f, e.g., aliasing with respect to resolution and orientation of the simulation grid, as observed in Figure 4.3(a) and, in particular, in Figure 4.4. B bounds the approximation error of f with respect to the PLIC reconstruction (including the offset of the PLIC patch along $\nabla f(\mathbf{x}_c)$) and can be multiplied by c to provide an estimate how far the isosurface moves along $\nabla \tilde{f}^k$ if the isovalue is varied by the value B (Figure 4.4). B is, however, not able to directly provide insight into the deviation of the planar PLIC patch from the shape of the curved isosurface of $\tilde{f}^k$. To this end, the curvature of these isosurfaces and the discontinuities between the PLIC patches are visualized, as described below.

Measurement of Isosurface Curvature Here, a measure is derived for the PLIC approximation error with respect to the shape of the interface. In compliance with the derivation of B, the PLIC reconstruction is analyzed in terms of its deviation from the second-order interface approximation. That is, the isosurface of $\tilde{f}^1$ at isolevel σ_1 is compared with the corresponding (curved) isosurface of $\tilde{f}^2$ at isolevel σ_2, with σ_1 and σ_2 obtained according to Section 4.1.3, i.e., such that $V(P_1) = V(P_2) = f_c$.

The principal curvatures of an isosurface of $\tilde{f}^k$ at point $\mathbf{x}$ are given by two nonzero eigenvalues λ_1 and λ_2 of $\nabla(\nabla\tilde{f}^k(\mathbf{x})/\|\nabla\tilde{f}^k(\mathbf{x})\|)$. The isosurface of $\tilde{f}^2$ at isolevel σ_2 is extracted, followed by the evaluation of $\kappa_{\max} := \max(|\lambda_1|, |\lambda_2|)$ at all its vertices. These values are either directly visualized on the surface (see, e.g., Figure 4.3(b)), or the maximum $\hat{\kappa}_{\max}$ over the isosurface within the original cell is determined and visualized by a uniform color on the corresponding PLIC surface patch (see, e.g., Figure 4.3(c)). The former enables detailed inspection of this second-order surface approximation, while the latter provides the maximum deviation of the PLIC patch from the second-order interface approximation in terms of curvature. If the absolute distance between the two isosurfaces is of interest, as in mesh resolution analysis (see, e.g., Figure 4.6), $\hat{\kappa}_{\max} \cdot c^2$ provides a conservative bound with respect to the cell extent c.

Measurement of C^{-1} Discontinuities While the measurement of isosurface curvature captures the deviation of the PLIC patch shape from that of a second-order approximation, it is not capable of measuring discontinuities between the patches. Regarding flux computation (see below), $C^{\geq 0}$ continuity is of lower impact than C^{-1} discontinuity because these represent gaps in the reconstructed interface. Therefore, a technique for the analysis of the discontinuities is provided.

The cell-wise discontinuity value is obtained by extracting the boundary curves of the isosurface(s) within each original cell (note that multiple isosurface parts can result in case of $k > 1$) and by measuring the minimum Euclidean distance between these boundary curves and all other boundary curves of the other cells. These distances are measured by sampling each edge of the current cell's mesh boundary polygons with r samples (in the analyzed datasets, $r = 9$) and determining for each sample the shortest distance to all boundary curves residing in other cells. These other boundary curves do not need to be supersampled because the involved point-to-segment distance can be determined exactly. From the resulting minimum distances along the current cell's boundary polygons, both their minimum $\delta_{\min}$ and their maximum $\delta_{\max}$ are taken. Both measures can be mapped to the isosurface(s) of the current cell using uniform color. The measure $\delta_{\min}$ provides a lower bound on the discontinuity of the current cell (i.e., the "best case"), while $\delta_{\max}$ indicates the upper bound (i.e., the "worst case"). Please see Figure 4.3(d)–(f).

4.1.5 Flux Volumes

Although the measures introduced in Section 4.1.4 already give insight into the implications of PLIC reconstruction in simulation codes, they do not provide a direct approach to the impact on the simulation result. Therefore, a visualization of fluxes involved in the solver advection step has been developed that completes the presented visualization framework. Here, the approach for the computation of volume fluxes used in the simulation is adopted, and it is the only part of the framework that takes into account the progress in time, however, only in the sense that the duration of the time step Δt is required. As discussed in Section 2, the advection step is performed separately in u-, v-, and w-direction, and the amount of concentration that moves to the neighboring cell is determined by computing the volume resulting from "cutting" the PLIC polyhedron P at the distance $-u\Delta t$ from the respective cell face (Figure 2.2(d)).

To obtain a geometric representation of this volume for visualization, the marching cubes approach that is used for obtaining the triangulation of P (Section 4.1.3) is utilized also in this case. The only difference is that the cells of G_c are split (by inserting new nodes) at the cut distance, i.e., where the respective cell face intersects G_c if the face is advected in reverse direction, see Figure 4.1(c). The approximation $\tilde{f}^k$ is resampled at the inserted nodes and all nodes of G_c that are located on the other side of the advected face are set to γ, with the effect that the resulting isosurface (taken at isolevel σ determined according to Section 4.1.3 for the original polyhedron) represents the flux volume, i.e., the amount of f that is advected through the respective cell face.

As flux computation is accomplished using operator splitting, the visualization of the fluxes in u-direction is straightforward. For those in v-direction, one needs to first compute the flux volumes in u-direction, update the f_c-field therefrom accordingly, repeat PLIC reconstruction on the updated f_c-field, and compute the flux volumes then in v-direction. For the flux volumes in w-direction, one needs to repeat this process once more in w-direction. Note that typically only every m^{th} step is output during simulation and hence, the spacing between simulation results is larger than the simulation step. Therefore, either every time step must be output for analysis by the technique or the step size must be provided with the simulation.

4.1.6 Results

This section demonstrates the utility of the presented visualization framework for the assessment of interface reconstruction quality. The framework was applied to several two-phase flow simulation datasets in the context of droplet dynamics. Table 4.1 provides timings of the implementation measured on a commodity desktop with Intel i7 3.6 GHz processor and $32\,\mathrm{GiB}$ RAM. The computation of the discontinuities δ_{min} and δ_{max} is clearly the most expensive step. This is mainly due to the employed basic search structure for the distance test, and this step would lend itself well to GPU acceleration.

Performance is significantly reduced for second-order patches as the number of triangles per patch increases quadratically due to the involved supersampling. However, the current implementation still allows for productive operation.

Sphere Dataset A dataset with stationary sphere (Figure 4.2) of known curvature $\kappa = 20\,\mathrm{m}^{-1}$, with 64^3 cells, provides a reference configuration with a wide spectrum

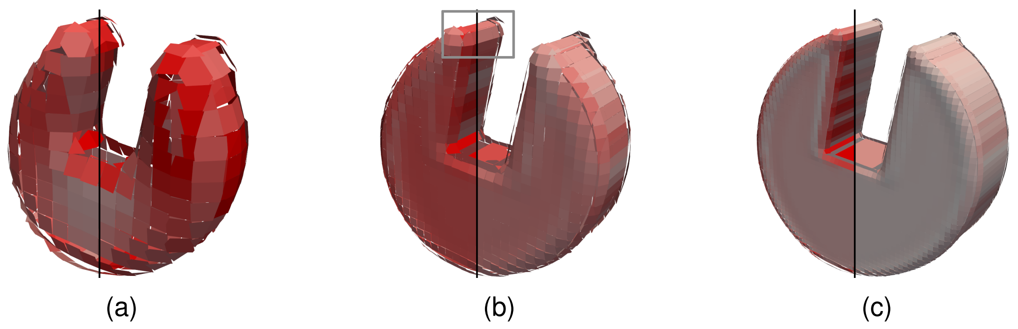


Figure 4.4: *Zalesak* dataset at time step 25 (time $0.0375\,\mathrm{s}$) and simulation grid (a) $32^2 \times 16$, (b) $64^2 \times 32$, and (c) $128^2 \times 64$, visualized by PLIC patches. The coloring by B (left halves) and $c \cdot B$ (right halves) conveys approximation problems regarding the f-field, including aliasing. The gray box depicts the region analyzed in Figure 4.5, however for the onset of the simulation. © 2013 IEEE.

of normal directions. This dataset serves for evaluation and illustration of the framework, and gives insight into the dependency of reconstruction quality on the sampling grid. All visualizations of the framework were applied to this dataset, except for the flow volumes as these require velocity data. Figure 4.3(a)–(f) provide the respective results. Interestingly, each measure captures different discretization properties of PLIC. While the discontinuity measures δ_{min} and δ_{max} exhibit low values on sphere where $(x=0) \vee (y=0) \vee (z=0)$, the approximation-based measures exhibit high (in case of B) or low (in case of κ_{max}) values at $x_{\text{min}} \vee x_{\text{max}} \vee y_{\text{min}} \vee y_{\text{max}} \vee z_{\text{min}} \vee z_{\text{max}}$ of the sphere. This complies with the flattened (low κ_{max}) parts visible in Figure 4.2 and the involved aliasing in general (large B). Hence, discontinuities are typically small at axis-aligned patches, while aliasing (B) is typically large in these cases, however, with the potential benefit of flattened interfaces and thus better approximation by PLIC. This behavior is expected in CFD simulations with PLIC-based reconstruction.

Zalesak Dataset A 3D variant of Zalesak disc is initialized in Couette flow $\mathbf{u}(\mathbf{x}) = (ay, 0, 0)^\top$, $a = 5\,\mathrm{s}^{-1}$, with the lower domain boundary at rest and the upper one moving at a velocity of $1\,\mathrm{m/s}$. The simulation was carried out at different grid resolutions and the data consists of 109 time steps which were output at the frequency of a single sim-

Table 4.1: Timings (in seconds) for the datasets from Figure 4.6 (C1: 32^3, C2: 64^3, C3: 128^3) and from Figure 4.8 (M), QUAD: second-order surface, Flux: flux volumes. Total execution time in brackets.

Data	PLIC	QUAD	Curvature	δ_{max} PLIC	Flux PLIC	Flux QUAD
C1	0.33 (0.35)	2.13 (2.16)	0.28 (0.62)	2.51 (2.85)	0.88 (1.34)	5.19 (7.53)
C2	1.26 (1.37)	10.5 (10.7)	1.22 (2.67)	18.4 (19.8)	2.93 (4.32)	27.9 (39.2)
C3	5.28 (5.74)	48.7 (49.4)	5.33 (11.4)	89.4 (95.1)	13.6 (19.8)	118 (171)
M	10.5 (12.6)	95.0 (98.0)	10.9 (24.3)	155 (167)	23.6 (35.9)	227 (327)

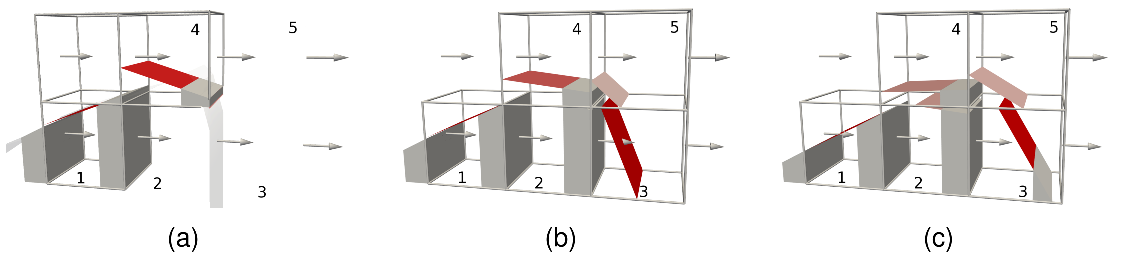


Figure 4.5: Visualization of fluid interface deterioration for the initial time evolution of *Zalesak* dataset. Interface cells shown as boxes, flux volumes as gray polyhedra, and Zalesak outline by transparent isosurface in (a). In (b), the PLIC normal in cell 2 points to the right, which is not consistent with the gas distribution in the surrounding cells. (c) This influences liquid distribution and results in the deterioration of the Zalesak pattern. © 2013 IEEE.

ulation time step $\delta t = 0.0015\,\mathrm{s}$. The Zalesak shape is traditionally used to investigate the influence of numerical diffusion and reconstruction properties in general since it features surfaces of greatly varying curvature, questions that lend themselves to analysis based on B. Figure 4.4 visualizes the dataset using PLIC patches colored with B. It is apparent that both B and the robustness $c \cdot B$ (Section 4.1.4) of the PLIC patch with respect to the isovalue σ are larger on the front face of the disc in (b) than in (a), although (b) was simulated at higher resolution. This is due to aliasing with respect to the resolution of the simulation grid, which is also visible as periodic pattern at the cutout in (c). It is apparent especially in (c) that the sharp edges at the top deteriorate during advection.

Figure 4.5 demonstrates how the framework can be used to investigate the deterioration resulting from the interface reconstruction and the involved fluxes. It illustrates a reconstruction error, and the arising flux calculation error, due to the normal vector calculation. In the initial state (a), cell 2 is filled and cell 3 is empty. The gradient in cell 3 does not point straight to the right but has an upward component due to finite differencing. Hence, the resulting PLIC patch degenerates to the red strip visible at the upper left edge of cell 3. Updating the concentrations with the flux volumes produces the interface reconstruction by the PLIC patches in (b). In this state, since f_c is now larger in cell 3, one can see the respective patch and its inclination. In the third state (c), a PLIC patch appears in cell 2 due to $f_c < 1$. However, steeper gradients in cell 3 cause a slight inclination of its PLIC normal to the right and shifting of the gas phase in the same direction inside the cell. Over time, cell 3 obtains more fluid and the overall f_c-distribution causes further inclination of the patch in cell 3, further deteriorating the Zalesak pattern. Additionally, a reconstruction problem, which can lead to "multiple fronts", has been identified using the technique in this context (Figure 4.5(c)).

Peripheral Collision Dataset This dataset is a simulation of a peripheral droplet-droplet collision. Figure 4.6 (top row) provides an overview of the temporal evolution. After initial collision, the liquid phase forms a disk which then splits into outer ring and an inner smaller disk. Both structures break up into multiple smaller droplets afterward. The simulation has been conducted at different resolutions to investigate the impact of

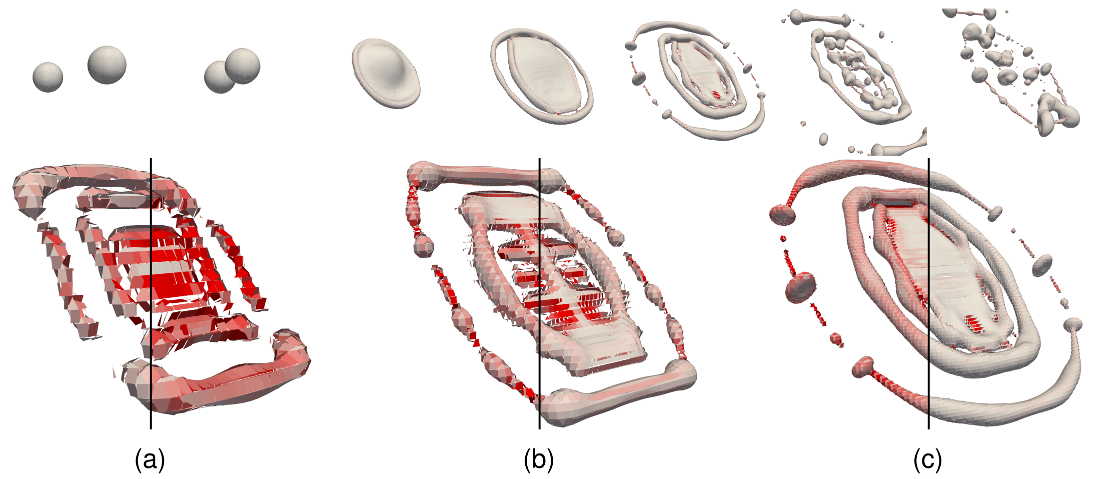


Figure 4.6: *Peripheral Collision* dataset. Visualization by PLIC patches with $\hat{\kappa}_{\max}$ (left halves) and $\hat{\kappa}_{\max} \cdot c$ (right halves), for same time step at resolution (a) 32^3, (b) 64^3, and (c) 128^3. $\hat{\kappa}_{\max} \cdot c$ provides good estimate of approximation quality. Top: Time evolution by PLIC colored with $\hat{\kappa}_{\max}$. © 2013 IEEE.

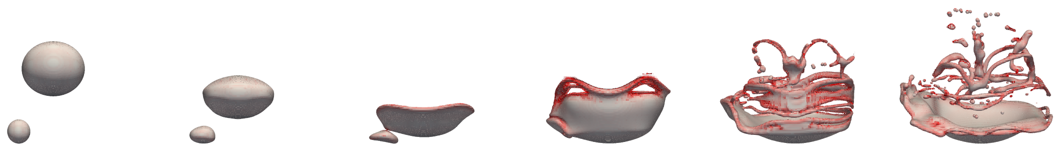
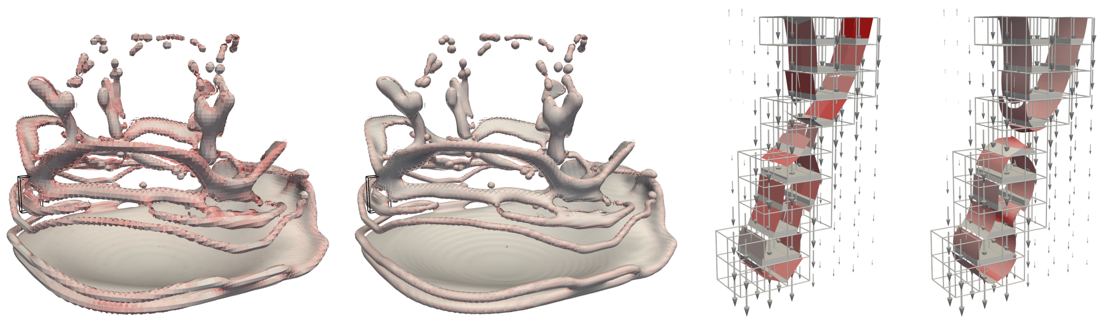


Figure 4.7: Time sequence of Merging Drops dataset. Visualization by PLIC patches colored by $\delta_{\max}$, in frame of reference moving with the drops. © 2013 IEEE.

grid resolution on the overall quality. It turned out that $\kappa_{\max} \cdot c^2$ provides a good estimate of approximation error. It is subject to future work to investigate this approach further, e.g., for adaptive mesh refinement during simulation. It is apparent that resolutions (a) and (b) are insufficient while (c) starts to exhibit physically correct behavior, consistent with the estimation $\kappa_{\max} \cdot c^2$. This can be explained by the fact that disintegration is closely related to surface tension and hence, to the curvature of the interface.

Merging Drops Dataset This case models the peripheral collision of two drops of different size discretized on a grid of $512 \times 256 \times 256$ cells, see Figure 4.7 for an overview of the time evolution. This represents a rather complex case due to droplet disintegration and formation of small ligaments that are resolved with only few cells and therefore exhibit large discontinuities in PLIC reconstruction. As visualized in Figure 4.8, the physically important breakup of the ligaments is essentially dominated by the f-fluxes. The visualization in Figure 4.8(c) indicates that the instabilities leading to breakup of sheets are suppressed by the PLIC reconstruction, i.e., that the flux out of the breakup cell would be larger if second-order interface reconstruction 4.8(d) would be used (visible at the larger flux volume at the lower side of the breakup cell).



(a) (b) (c) (d)

Figure 4.8: Merging Drops dataset. PLIC patches colored (a) by discontinuity δ_{max}, (b) by curvature κ_{max} on second-order surface reconstruction, and (c) a closeup thereof visualized using surface reconstruction and flux volumes based on first-order (PLIC) approximation, and (d) second-order approximation. As shown in (d), disconnection should occur due to larger flux volume in the lower region. In (c) PLIC, suppresses the breakup. © 2013 IEEE.

4.2 Visual Analysis of Interface Deformation

The deformation of the interface in two-phase flow is caused by the interplay of surface tension and fluid dynamic forces. Whereas the former has a stabilizing effect on the phases as it leads to a spherical shape, the latter tends to deform the phases and can cause interface breakup. Additionally, depending on the materials, the surface tension and fluid dynamic forces have different influence on the overall interface dynamics and therefore, the dynamics of phase breakups can vary significantly. Thus, the analysis of these forces is of particular importance for the study of two-phase flow phenomena, and visualization lends itself well for the investigation of their interdependence.

Visual analysis of interface deformation requires an approach that can capture its important characteristics and present them in an effective manner. To this end, a visualization technique based on metric and shape tensor analysis has been developed in this thesis that allows for efficient investigation. The *metric tensor* provides information on surface stretching and contraction, whereas the *shape tensor* indicates shape deformation in terms of bending direction and strength. The shape tensor is computed on a paraboloid that locally approximates the interface. For visualization, the shape tensor of the difference between the undeformed and the deformed interface is used to determine the bending characteristics. Both the metric tensor and the shape tensor are based on the local interface deformation induced by a velocity field. Applying the tensors on the simulation velocity field reveals the effective deformation. However, to gain more insight into the surface tension effect on the interface deformation, the tensors are additionally employed on a velocity field derived from the surface tension force. This allows for the analysis of the interplay of the fluid dynamic and surface tension forces.

For visualization, cylindrical glyphs represent the orientation of the eigenvectors for the metric and shape tensors, while the glyph color reveals respective stretching and bending strength.

4.2.1 Related Work

Surfaces and volumes have been frequently used in flow visualization for the analysis of deformations. Obermaier et al. [126] employed glyphs and stream volumes to visualize deformations induced by vector fields. A work closely related to the presented technique is the analysis of stretching and shearing by Obermaier and Joy [127] who utilized metric tensors for glyph-based visualization of deformation and to enhance integral surfaces. For the visualization of integral surfaces in divergence-free vector fields, Bartoň et al. [13] used the stretching information to adaptively find stretch-minimizing stream surfaces. Brambilla et al. [23] employed metric deformation analysis to parameterize and visually compare time surfaces from different simulation time steps. These works address single phase flow, where the surfaces can be arbitrarily placed in the vector field. In this thesis, however, the analyzed surfaces represent the fluid interfaces that undergo deformation not only caused by advection, but also by phase dynamics and surface tension. Therefore, they require a different approach for constructive analysis.

The notions of surface parameterization and related metric and shape tensors are utilized in this work for the description of deformation. A relevant introduction to the topic can be found in the work by Floater and Hormann [48]. The concept of curvature difference was utilized by Grinspun et al. [61] for the computation of discrete shells used in computer graphics and animation for modeling deforming thin objects. In her dissertation, Berres [15] provided an extensive description of surface deformation and visualization.

A problem inherent to VOF-based techniques is the generation of large surface tension forces due to artificially high curvature values resulting from the discretization on grids. Several approaches have been proposed to decrease this effect [22, 36] and the one presented by Popinet [140] offers an accurate and robust curvature approximation that combines height function computation and paraboloid-fitting. This method is used here for the visualization of the surface tension effect on interface deformation.

Curvature computation is an important topic in computer graphics and computer aided manufacturing. Rusinkiewicz [151] developed an algorithm for the estimation of curvature and its derivatives, such as curvature change along the surface, on triangle meshes. For volume rendering, Kindlmann et al. [93] proposed multi-dimensional transfer functions based on the principal curvatures to provide enhanced visualization of scalar volumetric data. Goldfeather and Interrante [59] provided a robust method for the estimation of principal directions on triangular meshes that achieves third-order approximation. A review of various curvature estimation methods can be found in the work by Mesmoudi et al. [119].

4.2.2 Measures for Visualization of Deformation

In two-phase flow, various forces act on the fluid phases that cause changes in the interface shape and potentially lead to breakups. Phase deformation typically results in a change of the interface area, which can be locally interpreted as interface stretching. Therefore, for the analysis of two-phase flow in this work, the metric tensor $\mathbf{I}_f$ is utilized. Obermaier and Joy [127] employed this tensor to analyze flow characteristics in single-phase configuration, where a virtual surface can be placed in the velocity field to investigate stretching at arbitrary regions. In this thesis, the metric tensor is employed in two-phase flow, where the surface of interest is naturally defined by the fluid interface. This poses some challenges that need to be addressed in the computation of the related flow characteristics, such as velocity gradients. The visualization of the metric tensor gives valuable insights into the local variations of the surface area for a given time interval. However, with this method, different types of interface deformation cannot be discriminated, and hence, potential breakup regions are not clearly visible. To gain a more complete picture of the interface deformation and its influence on the phase breakups, the bending characteristics of the interface are considered, which are expressed by a shape tensor.

For a smooth surface, the eigenvalues and eigenvectors of the shape tensor represent the principal curvatures and principal directions of the surface, respectively. If the surface deforms, its principal curvatures and directions change, resulting in a different shape tensor. To extract the bending characteristics, the tensor of the original surface is subtracted from the tensor corresponding to the deformed surface. The eigenvalues and eigenvectors of the resulting *shape change tensor* $\Delta\mathbf{S}$ provide the information on the bending direction and strength.

To better understand the influence of the surface tension on the deformation, a velocity field, denoted $\mathbf{u}_\gamma$, is derived from the surface tension force. This is the velocity the interface would assume to minimize the potential energy, and thus, to minimize the surface area. Therefore, a visualization thereof can provide important information on the local interface deformation and the related breakup dynamics, e.g., to what extent $\mathbf{u}_\gamma$ stabilizes the fluid structures. To extract the surface-tension-derived velocity, a state-of-the-art curvature computation method is utilized which computes the curvature by employing a height field approach and, in the case of poorly resolved phase components, it resorts to a paraboloid-fitting method to ensure robustness [140].

For visualization of deformation, i.e., stretching and bending, the eigenpairs of the tensors are represented by cylindrical glyphs whose orientation and color reveal the respective direction and strength of deformation. For graphical representation of the interface, the solver-based PLIC method has been employed. Although such reconstruction does not provide a smooth representation, it conveys the topology changes more accurately than, e.g., the standard marching cubes algorithm. Moreover, the deformation tensors and their eigenpairs are evaluated at the interface position corresponding to that computed by the solver. Therefore, it is reasonable to adhere to the solver-based interface reconstruction for visualization.

4.2.3 Surface Tension and Derived Velocity

For the computation of the surface-tension-derived velocity $\mathbf{u}_\gamma$, the Navier-Stokes equation for momentum, extended by an additional surface tension term $\mathbf{f}_\gamma$ (Equation 2.8 in Section 2), is considered. In the VOF method this term can be estimated as [51]:

$$\mathbf{f}_\gamma = \sigma\kappa\nabla f\,, \tag{4.6}$$

where σ is the surface tension coefficient that depends on the involved fluids, κ is the mean curvature of the interface in the given cell, and ∇f is the gradient of the volume fraction field f. For visualization, the contribution of the surface tension force to the total velocity at the interface can be expressed by:

$$\mathbf{u}_\gamma = \mathbf{f}_\gamma \cdot \Delta t/\rho_l\,, \tag{4.7}$$

with simulation time step Δt and density of the liquid ρ_l. Please note that this is only an estimation, since the computed velocity relates to the next simulation time step and not to the current one. Although one could consider the surface tension force in the previous time step to compute the current $\mathbf{u}_\gamma$, it would require complex interface tracking and possibly high temporal resolution of the simulation data in order to find corresponding

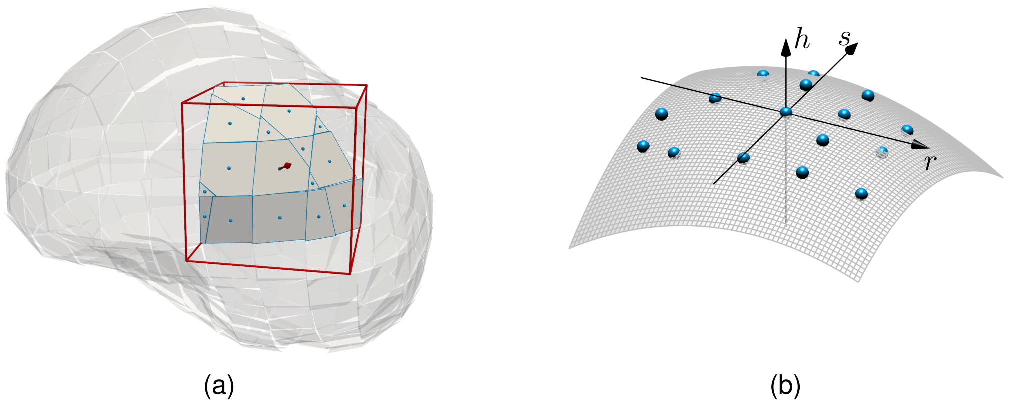


Figure 4.9: Interface approximation by paraboloid fitting. (a) Inclusion interface represented by PLIC reconstruction (transparent). Here, the paraboloid is fitted for the cell with the arrow representing the surface normal. The $3 \times 3 \times 3$ cell neighborhood (red box) is considered for approximation. (b) Paraboloid interface approximation, with blue dots corresponding to the PLIC barycenters, where the central barycenter lies at the origin of the coordinate system. Surface normal is aligned with the z-axis.

interface points at different time steps. In the presented framework, it is therefore not possible to directly compare the total velocity with the surface tension component for a given instant of time. Nevertheless, $\mathbf{u}_\gamma$ already provides useful information about the dynamics of the liquid phase with respect to the influence of surface tension.

Since the metric and shape tensors described below are applicable to both simulation velocity $\mathbf{u}$ and the surface-tension-derived velocity $\mathbf{u}_\gamma$, the notation $\mathbf{u}'$ is used to indicate either of them. The discrimination of the two quantities is postponed to the results section, where the usability of the overall approach is demonstrated.

The discrete nature of the volume fraction field causes spurious currents, i.e., artificially induced velocity on the interface caused by large curvature. This is particularly evident for curvature computation based on the Hessian of the VOF-field [17] and much effort has been put into improving the curvature computation. In this work, the method introduced by Popinet [140] is implemented for the visual analysis of the surface tension, as it considerably reduces spurious currents and has been also applied for the generation of the simulation datasets. Popinet's method combines two different approaches to provide accurate and robust curvature estimation. The first approach is based on height function computation on the volume fraction field. If the height field cannot be computed unambiguously, the algorithm falls back to the second method: paraboloid fitting. Since the latter is also employed for the computation of the shape tensor $\Delta\mathbf{S}$, algorithm details are provided in the next section. For a thorough description of the height field method, please refer to the work by Popinet [140].

Interface Approximation

In order to approximate the interface at a given cell C with a paraboloid, the barycenter $\mathbf{x}_C$ of the respective PLIC patch as well as the barycenters $\mathbf{x}_i$ of the neighboring patches in a $3 \times 3 \times 3$ stencil are considered. To obtain the paraboloid function, the coordinates of the barycenters are transformed such that $\mathbf{x}_C$ lies at the origin of the new coordinate system, and the corresponding interface normal points in the h-direction. The transformation to this coordinate system is thus achieved with $(r,s,h,1)^\top = T^{-1} \cdot (\mathbf{x}_c, 1)^\top$, using the matrix

$$T = \begin{pmatrix} x_r & x_s & x_{\mathbf{n}_\gamma} & x_c \\ y_r & y_s & y_{\mathbf{n}_\gamma} & y_c \\ z_r & z_s & z_{\mathbf{n}_\gamma} & z_c \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{4.8}$$

where $(x_{\mathbf{n}_\gamma}, y_{\mathbf{n}_\gamma}, z_{\mathbf{n}_\gamma})$ is the interface normal, (x_r, y_r, z_r) and (x_s, y_s, z_s) form an orthonormal basis with $\mathbf{n}_\gamma$, and (x_c, y_c, z_c) are the components of the central barycenter. With the positions in the new coordinate system, a paraboloid function can be computed that approximates the barycenter heights:

$$f(r,s) = a_0 r^2 + a_1 s^2 + a_2 rs + a_3 r + a_4 s + a_5 \,. \tag{4.9}$$

The approximation is found by minimizing the following equation using linear least squares:

$$F(a_j) = \sum_{1 \le i \le n} \left[h_i - f_{a_j}(r,s)\right]^2 , \tag{4.10}$$

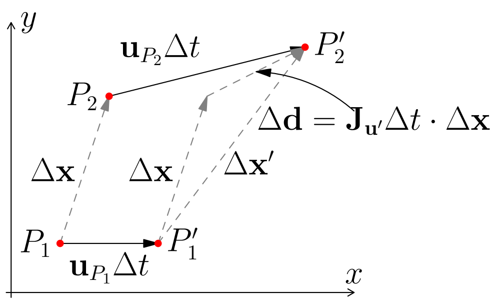
where h_i is the i-th barycenter height, n is the number of interface barycenters used for approximation, and $f_{a_j}(r,s)$ is the approximation function with coefficients a_j, $j = 1, \ldots, 6$. It is worth noting that the stationary point of the paraboloid f does not necessarily coincide with the point $\mathbf{x}_c$ and hence the coefficients $a_j \neq 0$, for $j = 3,4,5$. However, in the investigated datasets, these coefficients had marginal effect on the computed quantities and only at interfaces whose least squares approximation was of low quality (e.g., for very small droplets). Thus, dropping the coefficients a_j, $j = 3,4,5$ does not incur noticeable error in the visualization, but it does considerably simplify the following description and computation. Hence, the paraboloid function in Equation 4.9 can be simplified into the quadratic form:

$$h \approx f(r,s) = a_0 r^2 + a_1 s^2 + a_2 rs \,. \tag{4.11}$$

Figure 4.9(b) illustrates the paraboloid approximation for a fragment of the interface from a simulation, whose $3 \times 3 \times 3$ stencil is marked in Figure 4.9(a).

4.2.4 Metric Tensor for Stretching Visualization

In two-phase flow, due to the incompressible nature of the investigated liquids, interface stretching has no physical meaning as such. This is because whenever the surface area



y
x
$\mathbf{u}_{P_2}\Delta t$
P_2'
P_2
$\Delta\mathbf{d} = \mathbf{J}_{\mathbf{u}'}\Delta t \cdot \Delta\mathbf{x}$
$\Delta\mathbf{x}$
$\Delta\mathbf{x}$
$\Delta\mathbf{x}'$
P_1
P_1'
$\mathbf{u}_{P_1}\Delta t$

Figure 4.10: Illustration of the change in displacement $\Delta\mathbf{d}$ between two points P_1 and P_2 due to the velocity gradient.

increases due to deformation, some molecules that were originally located inside the fluid move to the surface, thereby contributing to the additional interface area. Nevertheless, the stretching formulation allows for an intuitive description of the interface deformation and is suitable for the analysis on the macro-scale. Although the idea to employ a metric tensor for flow visualization is not new (see [127]), it has not been applied yet to two-phase flow where it can provide useful insights into the interface dynamics. Moreover, by applying the metric tensor on $\mathbf{u}_\gamma$, the velocity induced by surface tension can be analyzed.

The interface stretching due to $\mathbf{u}'$ can be described by a metric tensor that represents the first fundamental form $\mathbf{I}_f$ from differential geometry. The metric tensor can be derived from the first order approximation of the volume deformation rate $\mathbf{J}_{\mathbf{u}'}$, i.e., the Jacobian of $\mathbf{u}'$. It can be intuitively described as follows (see also Figure 4.10). Given two points P_1 and P_2 in 3D space, their displacement vector is given by $\Delta\mathbf{x} = P_2 - P_1$. After time Δt, the change of displacement between these points due to $\mathbf{u}'$ is given by $\Delta\mathbf{d} = \mathbf{M}\cdot\Delta\mathbf{x}$, where $\mathbf{M} = \mathbf{J}_{\mathbf{u}'}\Delta t$. The new displacement vector is therefore $\Delta\mathbf{x}' = \Delta\mathbf{x} + \Delta\mathbf{d} = (\mathbf{E}+\mathbf{M})\Delta\mathbf{x}$, where $\mathbf{E}$ is an identity matrix and the matrix $\mathbf{F} = (\mathbf{E}+\mathbf{M})$ is known as a deformation gradient tensor. Subsequently, for the computation of $\mathbf{I}_f$, the points P_1 and P_2 can be constrained to the tangent space of the PLIC interface such that the displacement $\Delta\mathbf{x}$ is perpendicular to the PLIC normal $\mathbf{n}_\gamma$. This is done by right-multiplying $\mathbf{F}$ with a 3×2 matrix $\mathbf{N} = (\mathbf{r}\,|\,\mathbf{s})$, with orthonormal vectors $\mathbf{r}$ and $\mathbf{s}$ tangent to the PLIC patch. The resulting matrix $\mathbf{J}_f = \mathbf{F}\mathbf{N}$ is the Jacobian of the displacement, and, as noted by Floater and Hormann [48], can be used to construct a 2×2 matrix representing the metric tensor $\mathbf{I}_f$ as

$$\mathbf{I}_f = \mathbf{J}_f^\top \mathbf{J}_f\,. \tag{4.12}$$

The square roots of its eigenvalues $\sigma_l = \sqrt{\lambda_l}$, where $l \in \{1,2\}$, indicate contraction when $\sigma_l < 1$, and stretching when $\sigma_l > 1$. Consequently, for a displacement vector $\Delta\mathbf{x}$ parallel to an eigenvector $\hat{\boldsymbol{\varepsilon}}_l$, $\sigma_l = \|\Delta\mathbf{x}'\|/\|\Delta\mathbf{x}\|$. The stretching directions $\boldsymbol{\varepsilon}_l$ in 3D space can be obtained from the eigenvectors $\hat{\boldsymbol{\varepsilon}}_l$ of $\mathbf{I}_f$ by $\boldsymbol{\varepsilon}_l = \mathbf{N}\hat{\boldsymbol{\varepsilon}}_l$. Finally, $\boldsymbol{\varepsilon}_l$ and σ_l are used for visualization of stretching direction and strength, respectively. Please note that, since $\mathbf{I}_f$ is a symmetric tensor, it does not capture deformation due to shearing.

One remaining problem is that the velocities $\mathbf{u}'$ used to evaluate the Jacobian $\mathbf{J}_{\mathbf{u}'}$ are given at the barycenters $\mathbf{x}_i$ of the PLIC patches within the $3\times 3\times 3$ stencil (Fig-

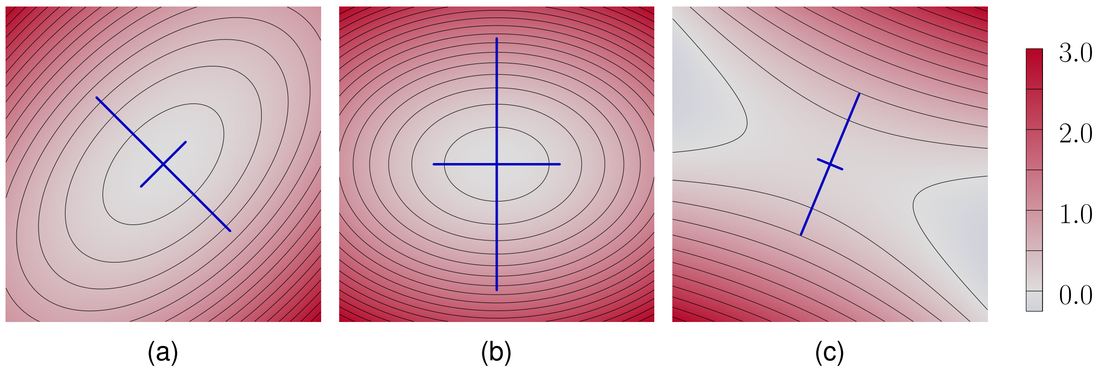


Figure 4.11: Shape change tensor $\Delta\mathbf{S}$ illustrated for two analytic functions representing undeformed and deformed interface: (a) $f(x,y) = x^2 + y^2 - xy$ and (b) $f(x,y) = x^2 + 2y^2$. Notice that the resulting principal directions $\mathbf{k}_l$ (blue lines, scaled by the corresponding curvature values) are not aligned with any of the corresponding principal directions from the undeformed and deformed interfaces.

ure 4.9(a)). To compute $\mathbf{J}_{\mathbf{u}'}$, a regression-based method is therefore employed that accounts for the scattered point positions and uses inverse distance weighting for improved accuracy [35]. The gradient of each velocity component of $\mathbf{u}' = (u', v', w')$ defines an over-constrained linear system of equations in the form

$$\mathbf{WX}\nabla u'_k = \mathbf{Wb}\,, \tag{4.13}$$

where $u'_k \in \{u', v', w'\}$ are the gradient components, X represents displacements from the central barycenters, where i-th row is equal to $(\mathbf{x}_i - \mathbf{x}_C)^\top$, $i = 0, \ldots, n-1$, and $\mathbf{b}$ is the corresponding difference in velocity components, i.e., $b_i = u'_k(\mathbf{x}_i) - u'_k(\mathbf{x}_C)$. The matrix $W = \mathrm{diag}(w_i)$ with $w_i = 1/\|\mathbf{x}_i - \mathbf{x}_c\|^2$ weighs the influence of points $\mathbf{x}_i$ by their distance from $\mathbf{x}_c$ for which the Jacobian is computed. The system can be solved with the linear least squares method.

4.2.5 Shape Tensor for Bending Visualization

While the eigenpairs of the metric tensor $\mathbf{I}_f$ provide the information on the direction and extent of interface stretching after some time interval Δt, the eigenpairs of the shape change tensor $\Delta\mathbf{S}$ give analogous details on interface bending. The shape change tensor is computed as

$$\Delta\mathbf{S} = \mathbf{S}(t + \Delta t) - \mathbf{S}(t)\,. \tag{4.14}$$

In this equation, $\mathbf{S}(t)$ represents the shape tensor of the interface approximation given by Equation 4.11 and $\mathbf{S}(t + \Delta t)$ represents its counterpart deformed by $\mathbf{u}'$ after time Δt. It is important to note that $\Delta\mathbf{S}$ does not represent any transformation matrix in an operator form, and in particular, it does not map the eigenvectors of $\mathbf{S}(t)$ to the eigenvectors of $\mathbf{S}(t + \Delta t)$. Rather, $\Delta\mathbf{S}$ describes how a surface with the shape tensor $\mathbf{S}(t)$ should be bent in order to obtain the surface whose shape tensor is $\mathbf{S}(t + \Delta t)$. This is illustrated in Figure 4.11 for two analytic surfaces. Notice that the eigenvectors of the resulting tensor do not align with any of those from the input shape tensors.

Since the elements of the tensor $\mathbf{S}$ exhibit rather complex dependency on the paraboloid surface approximation, it is preferable to express the shape change tensor $\Delta\mathbf{S}$ as a finite difference using the tensors evaluated for the given time interval Δt instead of estimating the approximation of its time derivative (which would result in a shape change tensor defined as $\Delta\mathbf{S} = \mathbf{S}_t \cdot \Delta t$). In the following, the computation of the shape tensor $\mathbf{S}$ and the related eigenpairs is described in detail, as it also applies to the shape change tensor $\Delta\mathbf{S}$.

A shape tensor is defined as $\mathbf{S} = \mathbf{I}^{-1}\mathbf{II}$, where the elements of the tensor $\mathbf{I}$ are dot products of the partial derivatives of vector $\mathbf{f} = (r, s, f(r,s))$, with $f(r,s)$ obtained from Equation 4.11:

$$\mathbf{I} = \begin{pmatrix} \mathbf{f}_r \cdot \mathbf{f}_r & \mathbf{f}_r \cdot \mathbf{f}_s \\ \mathbf{f}_s \cdot \mathbf{f}_r & \mathbf{f}_s \cdot \mathbf{f}_s \end{pmatrix}. \tag{4.15}$$

Since the shape tensor is evaluated at the origin of the coordinate system where $f_r(0,0) = f_s(0,0) = 0$, $\mathbf{I}$ reduces to an identity matrix. The elements of $\mathbf{II}$ are obtained from dot products of the second-order partial derivatives of $\mathbf{f}$ and the interface normal $\mathbf{n} = (0,0,1)$:

$$\mathbf{II} = \begin{pmatrix} \mathbf{f}_{rr} \cdot \mathbf{n} & \mathbf{f}_{rs} \cdot \mathbf{n} \\ \mathbf{f}_{rs} \cdot \mathbf{n} & \mathbf{f}_{ss} \cdot \mathbf{n} \end{pmatrix} = \begin{pmatrix} 2a_0 & a_2 \\ a_2 & 2a_1 \end{pmatrix}. \tag{4.16}$$

To compute the shape tensor $\mathbf{S}(t + \Delta t)$, particles representing the barycenters $\mathbf{x}_i$ are advected for the time interval Δt. Afterwards, paraboloid fitting is applied on the resulting positions that produces the paraboloid function $f(r,s)$ at time $t + \Delta t$. The advection, however, can lead to two issues that must be handled before paraboloid fitting. First, the central barycenter $\mathbf{x}_c$ is typically no longer located at the frame origin after the advection. Second, the interface positions represented by the barycenters within the $3 \times 3 \times 3$ stencil can rotate about an arbitrary axis during advection. To fix $\mathbf{x}_c$ at the origin of the reference frame, its velocity is subtracted from the velocities corresponding to all barycenters:

$$\hat{\mathbf{u}}'_i = \mathbf{u}'_i - \mathbf{u}'_0, \tag{4.17}$$

where $\mathbf{u}'_0$ is the velocity at the central barycenter $\mathbf{x}_0$. The rotation about the origin is then removed in two steps. In the first step, the average angular velocity is computed:

$$\boldsymbol{\omega} = \frac{1}{n} \sum_{i=1}^{n} \frac{\mathbf{x}_i \times \hat{\mathbf{u}}'_i}{||\mathbf{x}_i||^2}, \tag{4.18}$$

In the second step, the vortical component is subtracted from the velocities by

$$\tilde{\mathbf{u}}'_i = \hat{\mathbf{u}}'_i - \boldsymbol{\omega} \times \mathbf{x}_i. \tag{4.19}$$

With the computed tensors $\mathbf{S}(t)$ and $\mathbf{S}(t + \Delta t)$, the shape change tensor $\Delta\mathbf{S}$ is obtained according to Equation 4.14 and the eigenpairs $(\kappa_l, \mathbf{k}_l)$ used for visualization are calculated as follows. The principal curvatures representing the bending strength of $\Delta\mathbf{S}$ are obtained as

$$\kappa_1 = H - \sqrt{H^2 - K}\,, \quad \kappa_2 = H + \sqrt{H^2 - K}\,, \tag{4.20}$$

where K is the Gaussian curvature and H the mean curvature [133]:

$$K = \det(\mathbf{II}) = 4a_0a_1 - a_2^2\,, \quad H = \frac{1}{2}\mathrm{tr}(\mathbf{II}) = a_0 + a_1\,. \tag{4.21}$$

The eigenvectors of $\Delta\mathbf{S}$ representing the bending directions can be found by determining the tangent of an angle that either of the eigenvectors $\mathbf{k}_l$ forms with the r-axis. To compute $\mathbf{k}_1$ corresponding to κ_1, the tangent, denoted d, is determined according to the following formula:

$$d = \begin{cases} -\frac{a_2}{2a_1-\kappa_1} & \text{if } 2a_1 - \kappa_1 \neq 0 \\ -\frac{2a_0-\kappa_1}{a_2} & \text{if } a_2 \neq 0 \\ 0 & \text{otherwise}\ . \end{cases} \tag{4.22}$$

Subsequently, the principal directions are computed:

$$\mathbf{k}_1 = (1,d)\,, \quad \mathbf{k}_2 = (-d,1)\,. \tag{4.23}$$

If the paraboloid is a paraboloid of revolution, the principal directions can be arbitrarily chosen and hence d is set to zero. Finally, for visualization of the l-th eigenpair, $\mathbf{k}_l$ is used at each interface cell to orient a cylindrical glyph in the bending direction, whereas κ_l is mapped to color reflecting the bending strength.

An example of the computation of $\Delta\mathbf{S}$ is shown in Figure 4.12 for a small neighborhood on the droplet interface, marked by the red box in Figure 4.12(a). Through both the points defining the undeformed interface in Figure 4.12(b) and the points defined by their advection in Figure 4.12(c), a paraboloid is fitted. These paraboloids are shown in Figures 4.12(d) and 4.12(e), respectively. The difference of these quadratic forms, as shown in Figure 4.12(f), now reveals the principal curvatures and their respective directions.

4.2.6 Visual Representation of Deformation

For visualization, cylindrical glyphs are used to represent the orientation of the eigenvectors for the metric and shape tensors, and glyph color is employed to represent respective stretching and bending strength. The positive and negative values for bending can be interpreted as increasing and decreasing convexity, respectively. As an additional indication of stretching, the PLIC interfaces are color-coded by surface area change $\Delta A = \sigma_1\sigma_2$, with blue indicating decreased area, and red increased area.

The visualization approach has been implemented as a set of ParaView [10] filters. The simulation data is loaded as file series, separately for the volume fraction field f and velocity field $\mathbf{u}$. From f, the surface-tension-derived velocity field $\mathbf{u}_\gamma$ can be generated, which, together with $\mathbf{u}$, can be used as input for the metric tensor and shape tensor analysis.

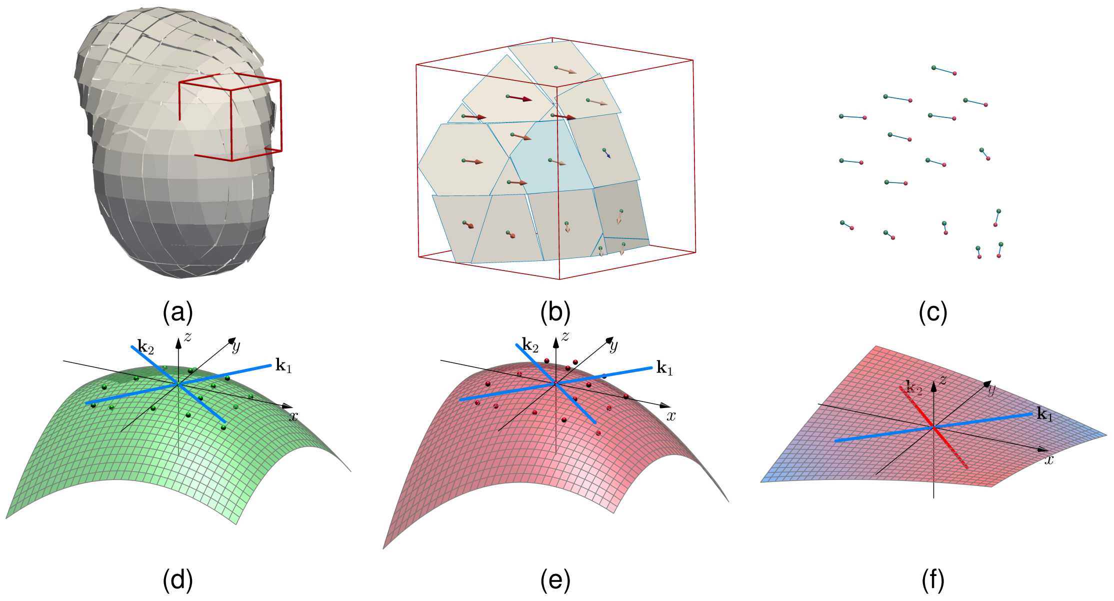


Figure 4.12: Computation of the shape change tensor $\Delta\mathbf{S}$. (a) Selected interface fragment in the red box. (b) In the close-up, the displacement vectors from the barycenters (green points) are shown, and in (c) the advected barycenter positions (red). (d), (e) Reconstructed paraboloids for the initial and advected barycenter positions (after elimination of the translation and rotation component), with corresponding principal directions $\mathbf{k}_1$ and $\mathbf{k}_2$. (f) The computed difference of the paraboloids in (e) and (d) reveals the bending direction as principal directions and their strength as the principal curvatures of the resulting paraboloid (with blue indicating bending downward and red upward).

4.2.7 Results

The utility of the visualization approach is demonstrated on two multiphase datasets. The first one is the two-phase *Peripheral Collision* dataset consisting of a water droplet collision in gaseous surrounding, and the second one, *Oil Inclusions* dataset, is a simulation of two colliding oil inclusions immersed in water surrounding. These two datasets exhibit considerably different flow characteristics due to diverse surface tension coefficients and viscosities of the surrounding phase.

Peripheral Collision Dataset The dataset consists of 256^3 cells, covering a domain size of $1\,\text{cm}^3$, and 401 time steps corresponding to the simulation time between $0.48\,\text{ms}$ and $2.85\,\text{ms}$. The average time step is therefore $\Delta t = 6\,\mu\text{s}$. The density of water droplets is equal to $\rho = 0.9982\,\text{g/cm}^3$, and the surface tension coefficient is $\sigma = 72.75\,\text{mN/m}$.

In Figure 4.13, selected simulation time steps are shown. The two merged drops form a flat shape that splits into an inner disk and an outer ring. Finally, both structures separate into small droplets. To get insight into the dynamics of the phase deformation, the interface dynamics have been analyzed with glyphs, for both the metric tensor $\mathbf{I}_f$ and shape change tensor $\Delta\mathbf{S}$, as demonstrated in Figure 4.14. In case of stretching, the orientation of red and blue glyphs in Figure 4.14(a) indicates elongation and thinning

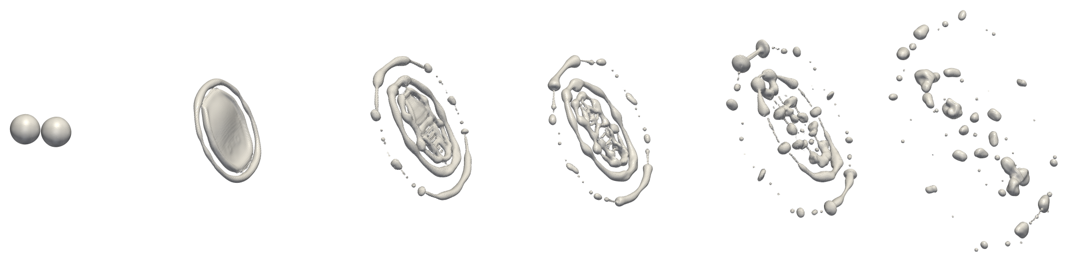


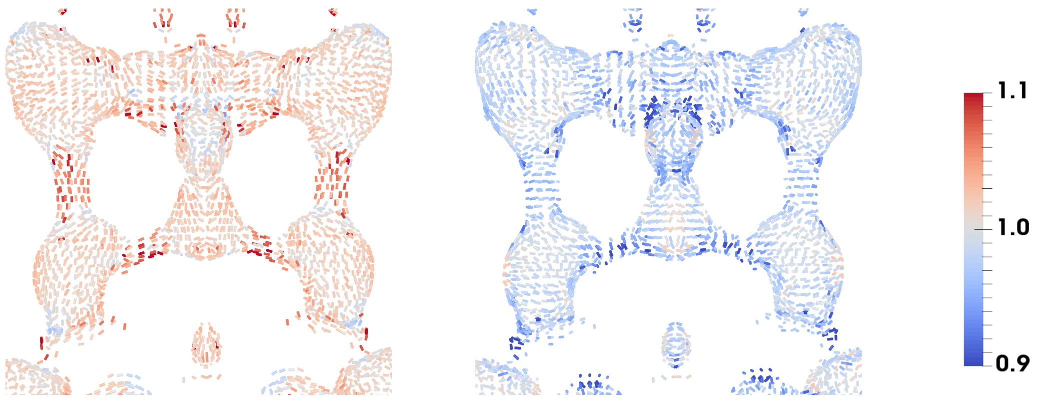
Figure 4.13: Extracted liquid interface for selected simulation time steps in case of the *Peripheral Collision* dataset (time in milliseconds). After collision, the drops initially form a disk that disintegrates into many differently shaped droplets.

of the ligaments—the red glyphs represent stretching along the ligaments, whereas the blue ones, oriented perpendicularly to the ligament axis, represent contraction. The shape change tensor $\Delta\mathbf{S}$ is shown in Figure 4.14(b) and (c). The glyphs representing the eigenpair $(\kappa_1, \mathbf{k}_1)$ have large values where the ligaments join the rounder inclusion segments (black boxes), indicating increasing convexity. This allows for clear identification of unstable regions and potential interface breakups. Indeed, in the subsequent time steps, the droplet breaks apart, as Figure 4.14(c) demonstrates.

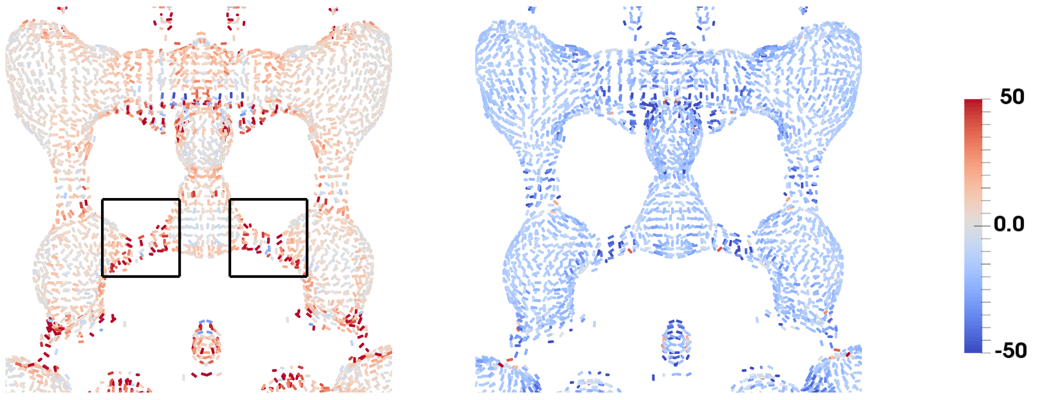
In this dataset, the metric tensor of the deformation induced by surface tension has also been investigated. In Figure 4.15, part of the disc-shaped drop is shown which disintegrates after the collision (Figure (c)). In Figures 4.15(a) and (b), surface-tension-derived velocity $\mathbf{u}_\gamma$ and total simulation velocity $\mathbf{u}$ are shown, respectively. The flat area is unaffected by the surface tension. However, in the region where breakup occurs, surface tension attempts to minimize the interface area, and hence, it separates the outer ring from the inner disk.

In Figure 4.16, the surface-tension-derived velocity $\mathbf{u}_\gamma$ is compared with the total simulation velocity $\mathbf{u}$ (respectively (a) and (d)) for a time step at which many liquid segments undergo deformation and breakup. In regions where both quantities are consistent, surface tension plays an important role in the interface deformation. This is the case in the outer regions of the satellite droplets (Figure 4.16(b) and (e)), where surface tension induces a spherical shape, as well as in the disintegrated part in the lower part of the central structure (Figure 4.16(c) and (f)). It is interesting to note that in case of the droplet in Figure 4.16(b) and (e), the deformation induced by the fluid dynamics (red surfaces in 4.16(d)) interacts with the surface tension, and, as shown by the visualization of rotation (cf. Figure 3.2), the droplet's initial oscillating motion transitions into rotation.

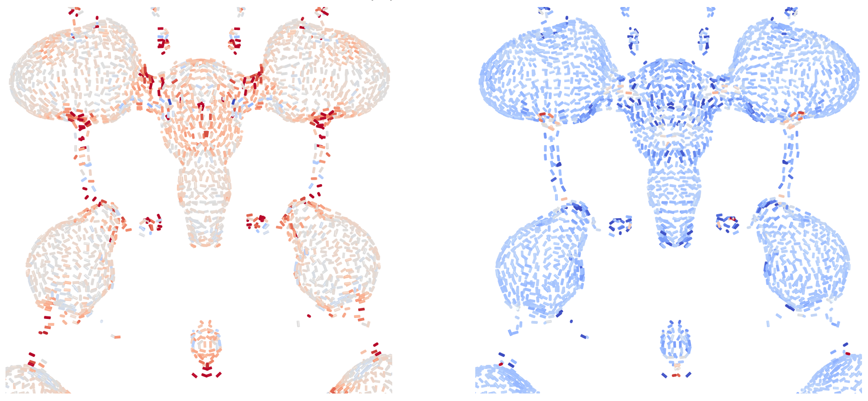
Oil Inclusions in Water The second dataset is a simulation of colliding oil inclusions in a water surrounding. The simulation domain consists of 256^3 cells covering $1\,\mathrm{cm}^3$, and 501 time steps that span the simulation time between $0\,\mathrm{ms}$ and $12.5\,\mathrm{ms}$. In Figure 4.17, selected simulation time steps are shown. The presence of liquid surrounding



(a) $2.07\,\mathrm{ms}$



(b) $2.07\,\mathrm{ms}$



(c) $2.19\,\mathrm{ms}$

Figure 4.14: Visualization of interface deformation in *Peripheral Collision* dataset at simulation time corresponding to Figure 4.13. (a) Visualization of stretching by metric tensor, with eigenpairs $(\sigma_1, \boldsymbol{\varepsilon}_1)$ on the left and $(\sigma_2, \boldsymbol{\varepsilon}_2)$ on the right. Stretching along the ligaments and contraction in the perpendicular direction, indicated by more intense red and blue colors, respectively, reveal thinning of the ligaments. (b), (c) Visualization of curvature change by shape change tensor, with $(\kappa_1, \mathbf{k}_1)$ on the left and $(\kappa_2, \mathbf{k}_2)$ on the right. The visualization of the curvature change better conveys the actual breakup as indicated in (c). Notice that the vertical ligaments highlighted by the metric tensor in (a) remain connected.

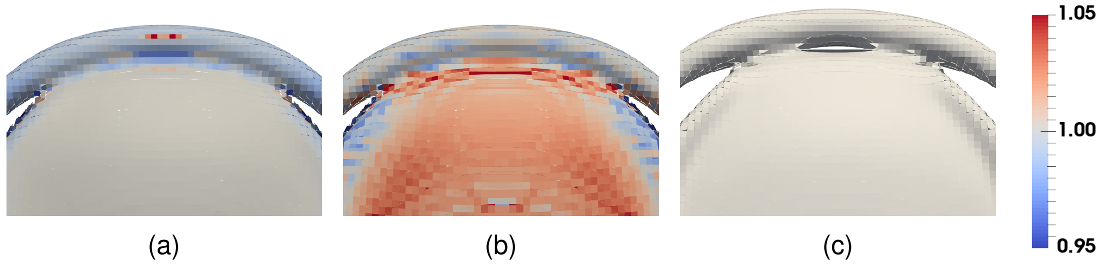


Figure 4.15: (a), (b) Interface area change ΔA in the *Peripheral Collision* dataset color-coded on PLIC interface reconstruction (color legend on the right). (a) With the visualization of ΔA computed from the surface-tension-derived velocity $\mathbf{u}_\gamma$, the region where the ring disintegrates is apparent. (b) The visualization of ΔA induced by the simulation velocity $\mathbf{u}_\gamma$, on the other hand, reveals thinning of the inner disk. (c) Detachment of the outer ring.

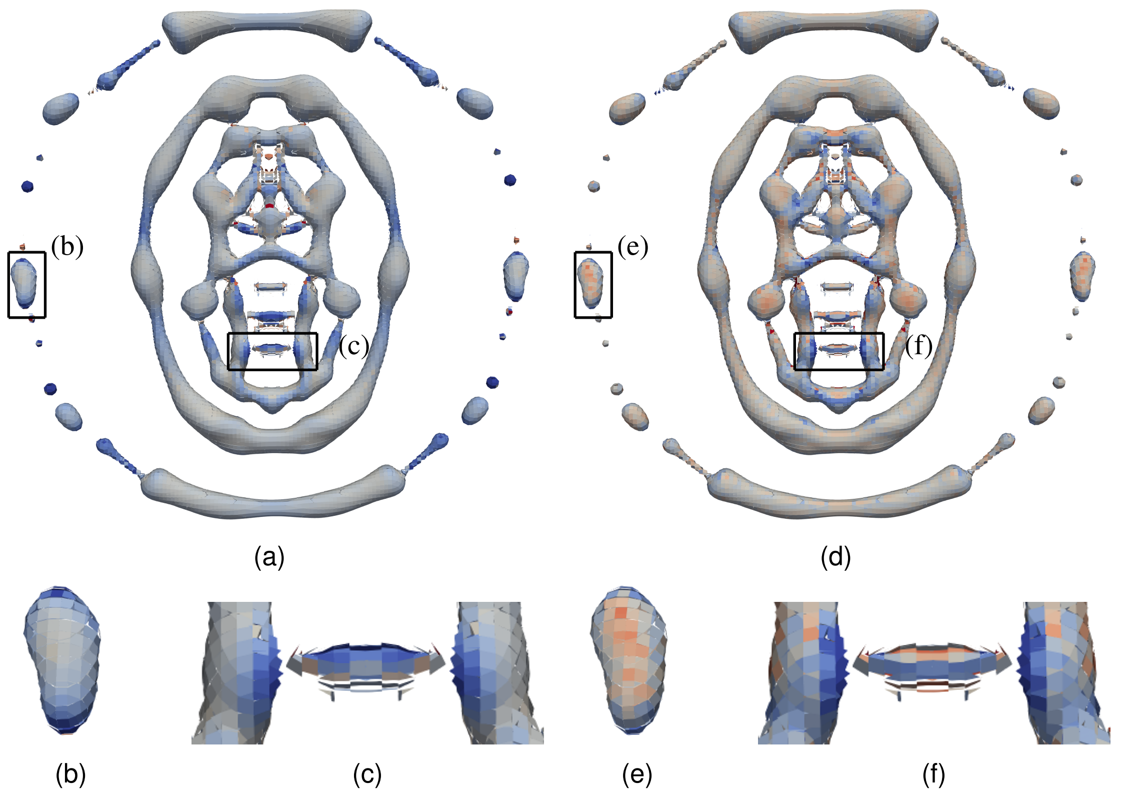


Figure 4.16: Interface area change ΔA in the *Peripheral Collision* dataset induced by (a) surface-tension-derived velocity $\mathbf{u}_\gamma$ and (d) simulation velocity $\mathbf{u}$. Deformations where both quantities have consistent values (shown respectively in (b), (e) and (c), (f)) are largely influenced by the surface tension force. (Color map as in Figure 4.15.)

changes the surface tension and dynamic forces, as compared to the first dataset—even before the collision, the initially spherical inclusions deform strongly and lose a considerable amount of momentum due to friction. In the final time steps after the collision, the inclusions quickly form spherical shapes and their velocity drops to almost zero.

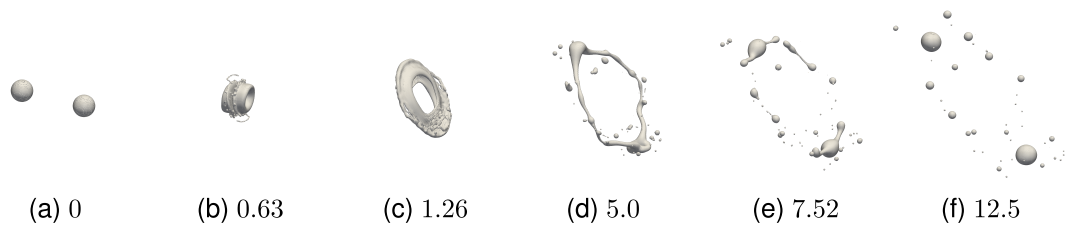


Figure 4.17: Extracted liquid interface for the simulation of *Oil Inclusions*. Time in milliseconds. Even before collision, the inclusions deform strongly due to the high viscosity of the surrounding phase that dampens the initial momentum due to friction. After the collision, the ring disintegrates and the resulting inclusions form into spherical shapes.

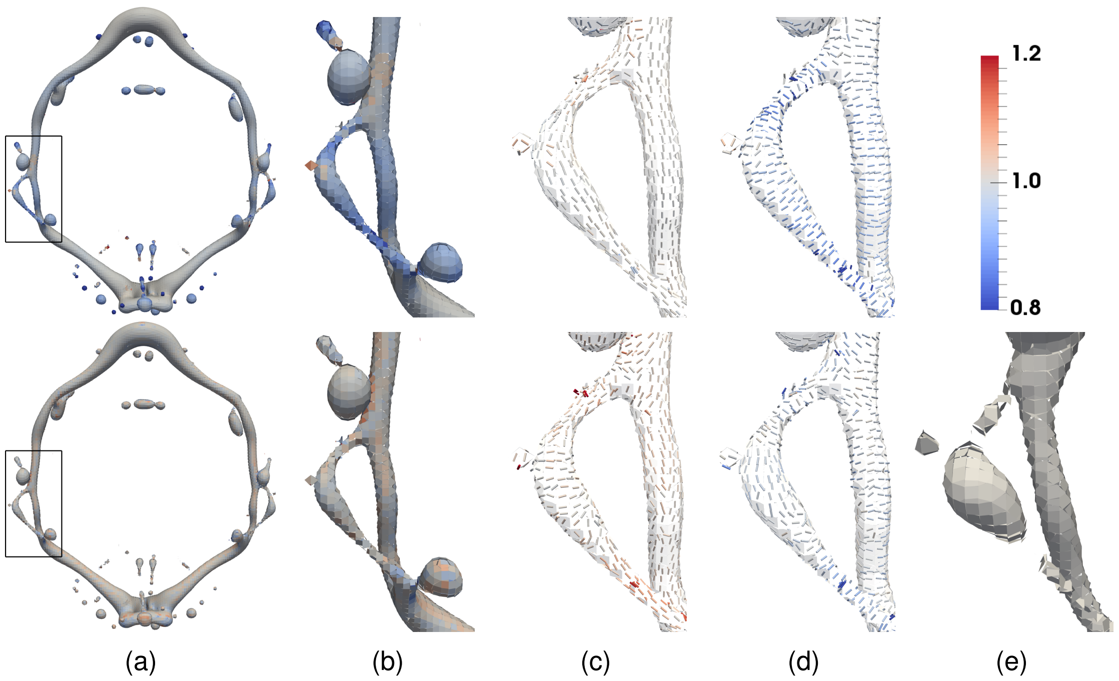


Figure 4.18: (a) Interface area change ΔA in the *Oil Inclusions* dataset induced by surface-tension-derived velocity $\mathbf{u}_\gamma$ (top row) and simulation velocity $\mathbf{u}$ (bottom row). (b) Enlarged part from (a). In the top image, the breakup region is revealed by dark blue color. (c), (d) Visualization of stretching by $(\sigma_1, \boldsymbol{\varepsilon}_1)$ and $(\sigma_2, \boldsymbol{\varepsilon}_2)$. Inspection of the velocity-induced stretching reveals that the upper part disintegrates due to the surface tension, whereas the lower part disintegrates due to the fluid dynamics. (e) Subsequent time steps show the eventual breakup.

Figure 4.18 compares the influence of $\mathbf{u}_\gamma$ (top row) and $\mathbf{u}$ (bottom row) in this dataset at simulation time 4.26ms, which is before the time step in Figure 4.17(d). In the overview in (a), where ΔA is visualized, regions with high surface tension can be identified on the outer ligaments of the ring-shaped inclusion, with the left one shown in the close-up in (b). The ligament is further zoomed in Figures 4.18(c) and (d), where glyphs representing stretching are visualized. The surface tension has the largest effect in the direction orthogonal to the ligament axis, as indicated by dark blue color. Strong

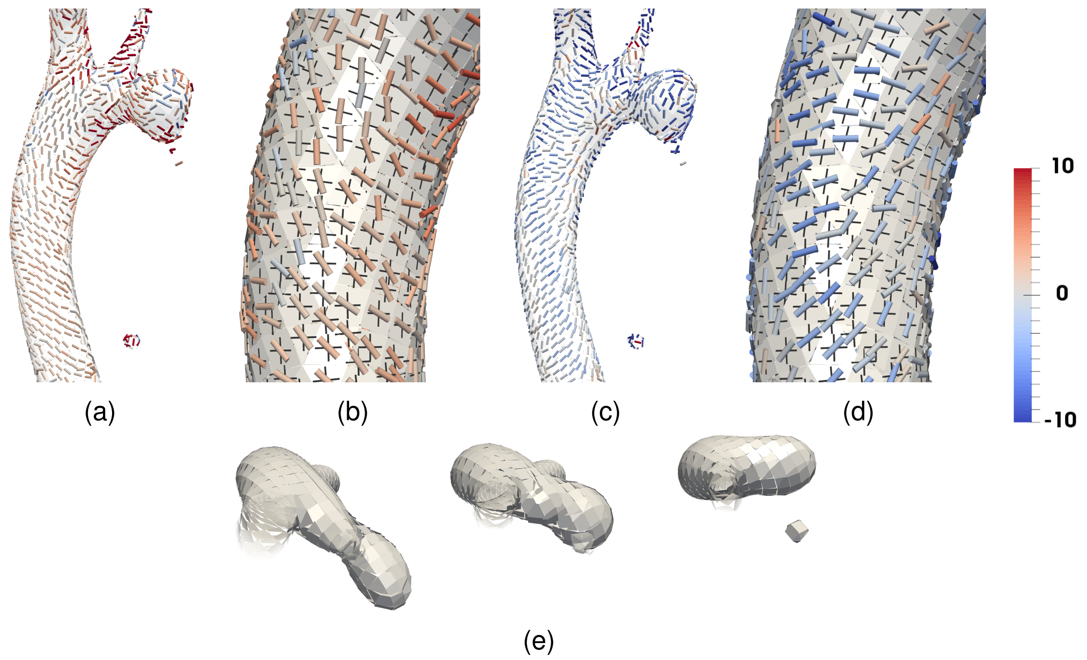


Figure 4.19: Visualization of the curvature change in the *Oil Inclusions* dataset. (a), (c) Visualization by eigenpairs $(\kappa_1, \mathbf{k}_1)$ and $(\kappa_2, \mathbf{k}_2)$. Interestingly, the glyphs representing bending direction are at an angle with the curvature directions of the interface (shown with black lines in (b) and (d)), thereby indicating torsional deformation. (e) Top view of subsequent time steps showing the rotation of the upper part of the phase.

stretching induced by the total simulation velocity in the bottom part of the ligament axis is revealed by red glyphs in Figure 4.18(c) (bottom). This shows that—contrary to the lower part that breaks up because of stretching caused by fluid dynamics—the upper part of the ligament disintegrates due to surface tension (where stretching due to $\mathbf{u}$ is comparably lower). In Figure 4.18(e) the deformation is shown in a subsequent simulation time step.

The fragment of the ring-shaped inclusion in Figure 4.19 exhibits interesting behavior, as revealed by the visualization based on shape change tensor analysis. In Figure 4.19(e), subsequent time steps are shown, where a rotation of the upper protruding part can be seen. The visualization in Figure 4.19(a) and (c) shows that the eigenvectors of the shape change tensor representing deformation are not aligned with the principal curvatures, but rather slightly oriented at a certain angle, as revealed in Figure 4.19(b) and (d). From this structure, one can infer torsional deformation of the phase, which is also indicated by the rotational movement of the upper part.

4.3 Visualizing Edge-Conforming Field Quantities in Electromagnetic Field Problems with Interfaces

The investigation of electromagnetic fields on material boundaries, such as insulator surfaces in the presence of rainwater, is of special interest in the development of structures that are exposed to strong electromagnetic fields. While the electric field at the interfaces is continuous tangential to the material surface, it commonly exhibits discontinuities in normal direction. These discontinuities pose problems in numerical simulation with finite element methods. In the traditional approach, node-based quantities impose continuity at material boundaries due to involved interpolation, resulting in undesired smooth modeling of quantities across the interface. Therefore, an edge-based representation of vector quantities was introduced where the simulated quantity is interpolated by means of vector shape functions that, among others, ensure discontinuities at interfaces.

Current visualization frameworks do not allow for correct representation of the electric field resulting from edge-based elements—due to their component-wise interpolation they do not provide accurate representation, and more important, they miss field discontinuities across the interfaces. Thus, in this section, a framework is presented that correctly visualizes the simulated field, also at material boundaries. The field is directly evaluated from the vector shape functions, taking higher-order elements (quadratic tetrahedra) into account, thus providing visualizations consistent with the simulation model. For the analysis of droplet-insulator contact line, a space-time visualization technique has been developed that reveals the time-dependent electric field characteristics around the droplet. The utility of the approach is exemplified using electro-hydrodynamic simulation of a water droplet on the surface of a high voltage insulator. [2]

4.3.1 Related Work

Edge elements were introduced into the finite element method by Nedelec [122] and Bossavit [21]. Bossavit [21] identified Whitney elements [194], which are widely used in finite element simulations, as a natural discretization method for eddy currents. The basis functions of edge elements were generalized to arbitrary order for the finite element method by Webb [186]. The characteristics of edge elements as well as their applications were discussed by Webb [185] and Mur [121]. Isoparametric elements described by Irons and Zienkiewicz [77] represent an alternative that allows for efficient and accurate higher-order computations.

In the field of visualization, several techniques have been developed to address appropriate representation of higher-order elements. Wiley et al. [200] developed a technique for direct ray casting of curved quadratic elements without prior tessellation into linear elements. For cell-based polynomial fields, isosurface extraction from higher-order finite elements was presented in Remacle et al. [145], where adaptive mesh refine-

[2] Parts of this section have been published in: [86]

ment was employed for accurate representation. A technique for isosurface extraction providing a trade-off between rendering speed and quality was suggested by Pagot et al. [132], based on a particle transport along the gradient field, and ray casting in the neighborhood of the final location of the particles. Schroeder et al. [162] addressed the complexity of higher-order basis functions from p- and hp-adaptive methods by employing an automatic tessellation technique with recursive edge-based subdivision. Direct visualization of discontinuous Galerkin simulations was presented by Üffinger et al. [181], where an adaptive sampling technique was used for high quality volume rendering, and utilization of a GPU cluster allows for interactivity. Feature extraction from discontinuous Galerkin simulations based on the parallel vectors operator was proposed by Pagot et al. [131]. A solution for the visualization of non-conforming meshes, based on point-based rendering, was developed by Zhou and Garland [210]. Isosurfaces from higher-order elements can be also visualized in a point-based manner, as proposed by Meyer et al. [120], in which case the costly inverse mapping to evaluate the basis functions is avoided. Most recent works in the field of higher-order finite element visualization include a ray casting method with pre-computation of world-element space transformation by Bock et al. [16], as well as ray casting with depth peeling due to Liu et al. [107].

For visualization of electromagnetic fields, examples include the visualization of the field in Tokamak reactors by Sanderson et al. [156], visualization of the coronal field by Machado et al. [113], and topological analysis of magnetic fields by Bachthaler et al. [12].

4.3.2 Edge Elements in Finite Element Simulations

The vector shape functions that describe the vector field in edge elements have two characteristics important for the analysis of fields exhibiting discontinuities. In the edge based-representation, neighboring elements share tangential components on the edges, and hence, the tangential field is continuous across those elements. On the other hand, the shape functions cause the tangential component to vanish at the element faces opposite to an edge, thus allowing for discontinuities in the normal component, which in turn enables correct representation of the electric field at interfaces.

In addition to the discontinuity preservation ensured by edge elements, the introduction of nonlinear elements, such as quadratic tetrahedra, enables a more accurate approximation of curved surfaces, which has particular importance for the computation of the electric field, since sharp, piecewise linear representation would artificially amplify the electric field magnitude. Consequently, the nonlinear elements significantly reduce the required number of tetrahedra for the approximation of curved boundaries. In the following, the nonlinear elements are described first to lay a basis for the description of 1st-order and 2nd-order edge-elements.

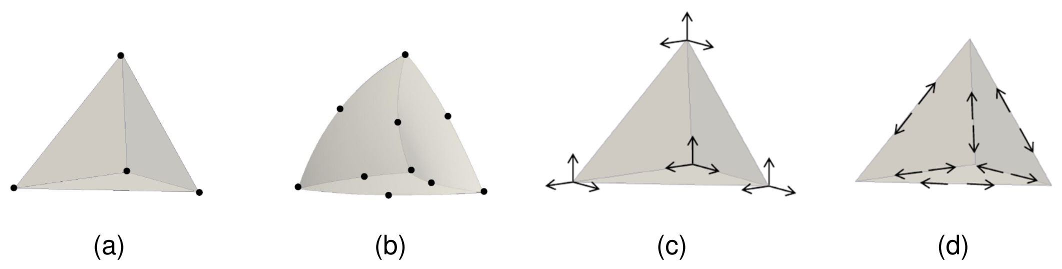


Figure 4.20: Data representation on tetrahedral elements. (a) Linear tetrahedron and (b) quadratic tetrahedron with additional mid-edge points. (c) Node-based data representation, (d) edge-based data representation.

Quadratic Tetrahedra

Quadratic tetrahedra are particularly suitable for finite element-based simulations involving electromagnetic field problems with interfaces, since they allow for curved edges and faces. Considering that the elements can adapt to relatively strong deformations, mesh refinement can be avoided for many problems. This at the same time allows for a significant reduction of the required number of elements—curved parts of the simulation domain no longer require strong refinement for accurate computation of the electric field. Quadratic tetrahedra, however, are more difficult to implement [46]. Since they have variable metric, i.e., the Jacobian determinant is not constant over a tetrahedron, point location inside cells is further complicated.

A quadratic tetrahedron is illustrated in Figure 4.20(b) (cf. Figure 4.20(a) for comparison with a linear tetrahedron). The element is defined by 10 nodes, each carrying three components of the vector field, resulting in 30 degrees of freedom. The shape functions N_l corresponding to the nodes of quadratic tetrahedra are defined as:

$$\begin{aligned}
&N_l = L_l(2L_l - 1), \quad \text{for} \quad l = 1, ..., 4 \\
&N_5 = 4L_1L_2\,, N_6 = 4L_2L_3\,, N_7 = 4L_1L_3 \\
&N_8 = 4L_1L_4\,, N_9 = 4L_2L_4\,, N_{10} = 4L_3L_4\,,
\end{aligned}$$

where (L_1, L_2, L_3, L_4) are the barycentric coordinates. The world coordinates (x, y, z) of a point inside a quadratic tetrahedron can then be obtained using:

$$x_i = \sum_{l=1}^{10} x_{i,l} N_l\,, \tag{4.24}$$

where $x_i \in \{x, y, z\}$ represents the i-th world coordinate of the sought point, and $x_{i,l}$ represent the i-th coordinate of the l-th node of the tetrahedron. Since the computation of barycentric coordinates from world coordinates is nontrivial for quadratic tetrahedra, iterative methods, such as Newton-Raphson iteration, are commonly used.

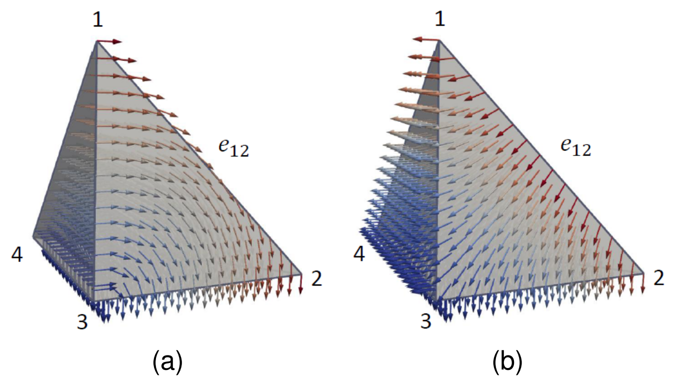


Figure 4.21: Vector basis functions on tetrahedra. (a) First order basis function $N^1_{1,2}$ on edge $e_{1,2}$ and (b) second-order basis function $N^{13}_{1,2}$ on edge $e_{1,2}$.

Edge-Conforming Data Representation

In the node-based finite element method, data is stored on the nodes of the elements. As illustrated in Figure 4.20(c), for a linear tetrahedral element there are four nodes, each carrying three components of the vector field. In the edge-based representation, on the other hand, data is attached to the edges of an element, as shown in Figure 4.20(d). Here, each edge stores two tangential components of the vector field. In both representations, there are in total twelve degrees of freedom for a linear tetrahedron. It is worth noting, however, that incomplete degrees are widely used in simulations of the electromagnetic field with interfaces in order to handle discontinuities at material boundaries. To model the electric field in the edge-based element, Whitney vector basis functions are employed. The first-order incomplete Whitney shape functions are defined as

$$\mathbf{N}^l_{i,j} = L_i \nabla L_j - L_j \nabla L_i \,, \text{ for } l = 1, ..., 6 \,, \tag{4.25}$$

where i and j are the nodes defining edge (i, j), L_i and L_j are barycentric coordinates of nodes i and j, and ∇L_i and ∇L_j are the constant gradients of the barycentric coordinates. The basis functions are then used to interpolate the electric field $\mathbf{E}$ from the tangential components c:

$$\mathbf{E} = \sum_{l=1}^{6} c_l \mathbf{N}^l \,. \tag{4.26}$$

To extend the element to the complete first order, additional edge functions are used:

$$\mathbf{N}^l_{i,j} = L_i \nabla L_j + L_j \nabla L_i \,, \text{ for } l = 7, ..., 12 \,, \tag{4.27}$$

The second-order incomplete basis functions are defined for edges:

$$\mathbf{N}^l_{i,j} = L_j(2L_i - L_j)\nabla L_i - L_i(L_i - 2L_j)\nabla L_j \,, \text{ for } l = 13, ..., 18 \,, \tag{4.28}$$

and faces:

$$\mathbf{N}^l_{i,j} = L_i L_j \nabla L_k - L_i L_k \nabla L_j \,, \text{ for } l = 19, ..., 22 \tag{4.29}$$

$$\mathbf{N}^l_{i,j} = L_j L_k \nabla L_i - L_j L_i \nabla L_k \,, \text{ for } l = 23, ..., 26 \tag{4.30}$$

$$\mathbf{N}^l_{i,j} = L_j L_k \nabla L_i + L_i L_k \nabla L_j + L_i L_j \nabla L_k \,, \text{ for } l = 27, ..., 30 \,. \tag{4.31}$$

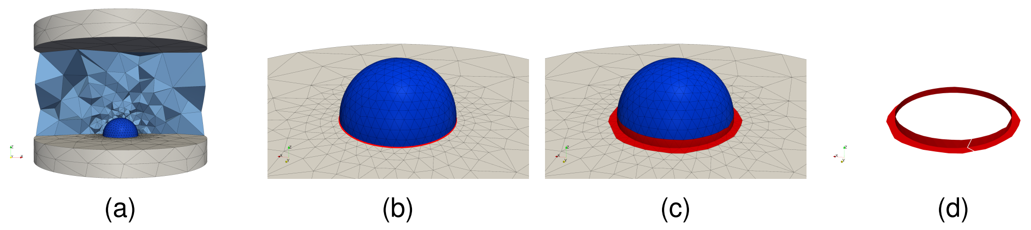


Figure 4.22: Stages for the visualization of the electric field on the contact line. (a) Simulation domain with three materials: air (light blue), water drop (dark blue), and insulator (gray). (b) The contact line (red) between drop and insulator is extracted. (c) The drop cells and insulator cells adjacent to the contact line are extracted. (d) The extracted contact region is cut before the "unrolling" stage illustrated in Figure 4.23.

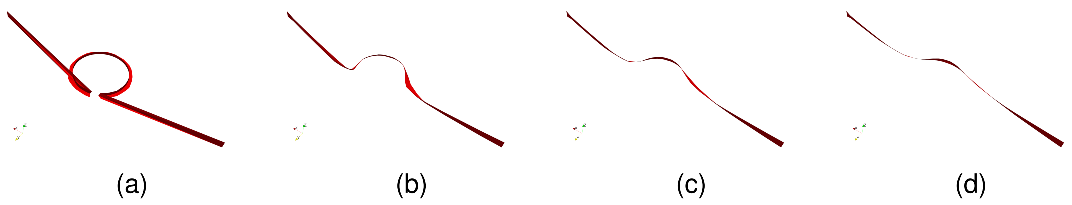


Figure 4.23: The process of Laplacian smoothing for the contact region. The cut points from each side are positioned away from each other (a). Iterative smoothing applied to the rest of the mesh points ((b) and (c)). Final rectification applied to the points, so that they lie on a plane (d).

In Figure 4.21 two edge functions are shown. Both basis functions have non-zero tangential component along the edge $(1,2)$, and zero tangential component on the faces $(1,3,4)$ and $(2,3,4)$. The edge-conforming representation has a particularly useful characteristics that the basis functions do not impose continuity more than required by the physics of the given phenomena. This means they can model discontinuities of the normal component at the interfaces while ensuring continuity of the tangential field component.

4.3.3 Visualization

Two visualization methods have been developed to analyze the electric field at the droplet interface. In the first one, the electric field on the material interface, i.e., on the interface between air and the droplet in a strong electric field, is visualized (Figure 4.22(a)). In the second one, the time-dependent electric field is analyzed in a region near the contact line between droplet and the insulator on which the droplet is located. To facilitate the analysis of this region, the droplet interface and insulator surface near the contact line are "unrolled" to create a rectangular strip. The strips from consecutive time steps are stacked, and temporal interpolation is performed to obtain a space-time representation which provides an appropriate visualization of the space-time dynamics.

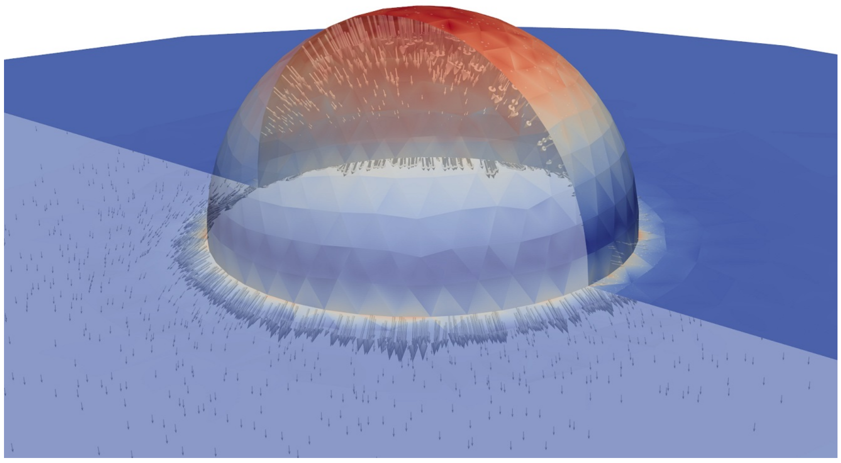


Figure 4.24: Visualization of a droplet on an insulator in the presence of a strong electric field. The insulator and droplet are made partially transparent to reveal the electric field direction (glyphs with size scaled according to the field strength). The electric field is strongest at the top of the droplet (red) as well as at the interface.

Electric Field on the Interface

For the visualization of the electric field on the air-drop interface, the quadratic tetrahedra adjacent to the interface between the two materials M_1 (air) and M_2 (water) must be found from the material information provided with the simulation data. Each tetrahedron in M_1 is checked if it faces M_2, and if this is the case, indices of both cells are stored in a list. Subsequently, a triangle mesh is constructed that represents the material interface. Please note that for the visualization, the curved tetrahedral faces from the simulation are approximated by linear triangles. Each triangle in the mesh is subdivided into smaller coplanar triangles. The middle point of each sub-triangle is computed, together with its barycentric coordinates in the original triangle. Since the coordinates are computed on tetrahedral faces, it is not necessary to employ computationally expensive point location algorithms. The shape functions are then evaluated using Equations 4.25 and 4.27–4.29, and finally the electric field is evaluated using Equation 4.26 and stored on the resulting mesh. See Figure 4.24 for a result.

Space-Time Visualization of Electric Field around Contact Line

To extract the contact line between air, water, and insulator (Figure 4.22(b)), first, two interfaces are extracted: interface I_1 between M_1 (air) and M_2 (water), and interface I_2 between M_1 and M_3(insulator). From these interfaces, only those triangles are considered that share at least one vertex with a triangle from the other interface. The resulting triangular mesh is shown in Figure 4.22(c). In the next step, the mesh is cut (Figure 4.22(d)), and the cut points are duplicated so that on each side of the cut the triangles are topologically disconnected. The cut points from one side are then positioned away from the other cut points (Figure 4.23(a)). An iterative Laplacian smoothing, illustrated in

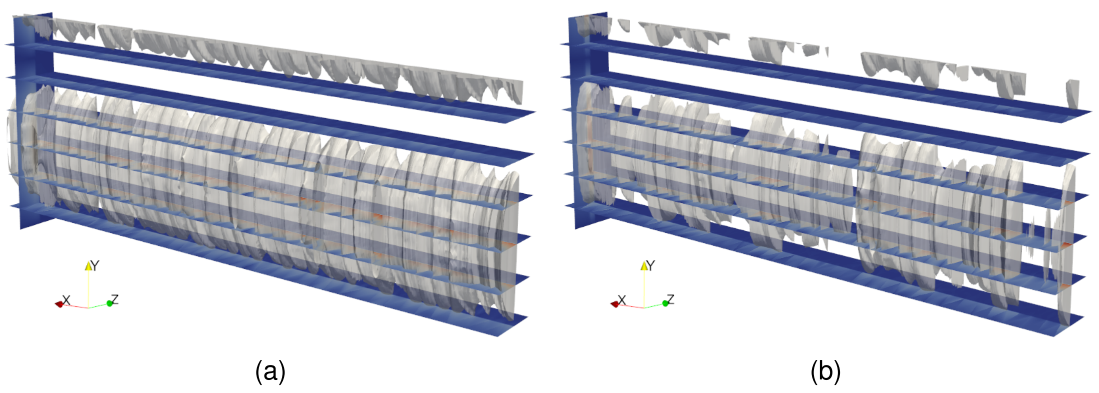


Figure 4.25: Visualization of the time-dependent electric field tangent (a) and normal (b) to the interface around the contact line, visualized with cross sections (horizontal sections) of the space-time representation. Time evolution from bottom to top (T-axis, respective time steps on the right). Air-insulator interface is at the front, drop-air interface in the back. The transparent isosurface and the vertical plane show the spatial and temporal variation of the electric field, respectively. Interestingly, the normal component varies stronger than the tangent one.

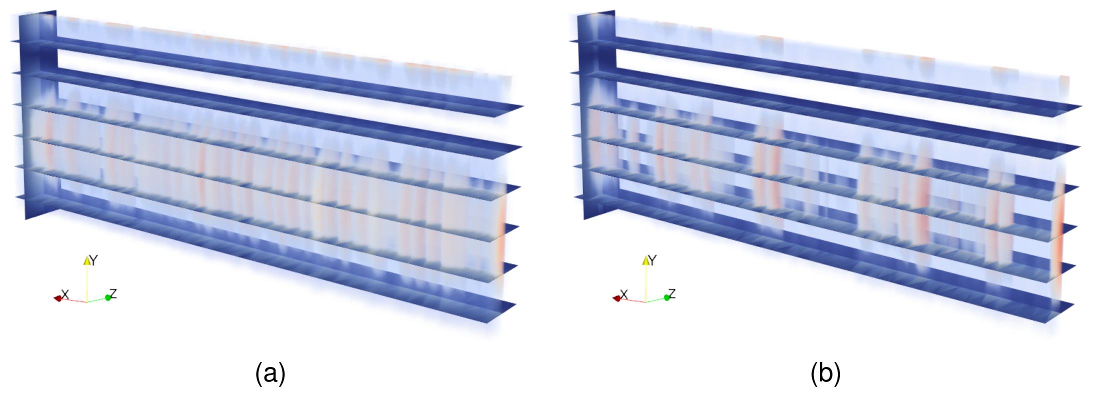


Figure 4.26: Space-time visualization of the time-dependent electric field, as in Figure 4.25. Volume rendering is shown instead of isosurfaces. The magnitude of the electric field is considerably lower for the drop-air interface (back side of the planes).

the rest of Figure 4.23, is performed on the other points of the mesh strip, placing each point on the average of direct neighbors at each iteration step. Only neighbors on the perimeter are taken into account for the points laying on it, otherwise the whole mesh would collapse into one line. After the last iteration, the strip is rectified, since the smoothing does not converge to a perfect plane after reasonable amount of iterations. The extracted contact region, shown in Figure 4.22(d), and the smoothed counterpart (Figure 4.23(d)) are subdivided into fine triangular meshes, such that for each cell in the original geometry there is a corresponding cell in the smoothed one. Finally, the data is sampled on the original subdivided mesh, as described in the previous section, and the resulting values are assigned to the corresponding points on the smoothed mesh.

To obtain a space-time representation, a 3D rectilinear grid is constructed, such that the resolution in y direction corresponds to the number of simulation time steps, while the xz-plane is aligned with the “unrolled” contact line strips. For the presented dataset, a $1024 \times 51 \times 25$ grid was used (i.e., with 51 time steps and 1024×25 samples per strip). Each smoothed triangle strip, coplanar with the grid xz-plane, is sampled at the nodes of the 3D grid at the respective (time step) position y. To reveal the temporal variation of the electric field, isosurface extraction (based on marching cubes algorithm, Figure 4.25) and volume rendering (Figure 4.26) are employed on the space-time stack.

4.3.4 Results

The visualization of electro-hydrodynamic simulations of a droplet in the presence of strong electric fields is illustrated in Figure 4.24. The simulation was obtained by a coupled fluid dynamics and electric field solver: the finite element method step determines the electric field, which is then used to compute the dynamics of the droplet with the finite volume method (FVM) according to the induced flow. The solution of the fluid dynamics stage is fed back to find the electric field in the domain. The process is repeated for each simulation step. Figure 4.24 shows one time step from the simulation. The droplet is positioned on an insulator, within a strong electric field. The electric field lines in the air (not shown in the figure) are vertical. As can be seen, the electric field is strongest at the top of the droplet. What is more interesting, however, is that the field strength on the contact line is amplified due to the sharp corner formed by the droplet-insulator contact region. Since this region is of special interest for the domain experts, the space-time visualization technique facilitates the investigation of the time-dependent electric field near the contact line.

In Figures 4.25 and 4.26, the space-time visualization of the electric field in the contact region is provided. The images on the left show the electric field tangent to the drop (insulator) surface, while the images on the right show the field normal to the interface. Six selected simulation time steps are additionally displayed, and a vertical color-coded cross section shows the time evolution of the field at one position of the contact region. The air-insulator interface is oriented to the front. Figure 4.25 shows a transparent isosurface of the magnitude of the tangent (normal) field component, while Figure 4.26 displays the magnitude by volume rendering. The visualization shows interesting characteristics of the time-dependent vector field. It exhibits periodic intervals, as indicated by the isosurface: the high magnitude regions disappear before time step 36 to appear again after time step 44. Volume rendering reveals that the tangential field has uniform distribution along the contact line, while the normal field is characterized by stronger spatial variation. It is also evident that the electric magnitude is considerably weaker on the drop boundary. The static representation of the temporal process enables insights into the overall behavior of the electric field along the drop-insulator contact region.

The most demanding part of the visualization framework, which has been implemented as a ParaView plugin [10], is the computation of tangential components from the node-based representation, which took more than 7 minutes for about 11000 simulation cells. The computation of the droplet interface took 19 seconds, while one time step of the space-time representation took about 7 seconds.

Visualization Approaches for Material Transport | 5

Visualization of single-phase flow can be beneficial in the analysis of two-phase flow as it allows for clearer interpretation of the observed physical processes, since no interfaces are present. On the one hand, flow inside liquid inclusions can be treated as single-phase, whereby fluid interfaces and therefore surface tension forces are absent. On the other hand, in the analysis of certain phenomena, the interface does not have to be taken explicitly into account. In both cases, the visualization focuses on the vector field data and concepts from the traditional flow visualization can be employed. Accordingly, in this chapter, visualization techniques for the analysis of single-phase flow are presented.

The first method presented in this chapter is dye-based visualization of advection-diffusion [90, 89]. The technique allows for interactive 3D visualization of both advection and diffusion in unsteady fluid flow by extending advection-oriented texture-based flow visualization to diffusion. The employed finite volume approach based on weighted essentially non-oscillatory (WENO) reconstruction is well parallelizable on a GPU and features low numerical diffusion at interactive frame rates. The scheme contributes to three different applications: high-quality dye advection at low numerical diffusion, physically-based dye advection accounting for diffusivity of virtual media, and visualization of advection-diffusion fluxes in physical media where the velocity field is accompanied by a concentration field.

In the dye-based visualization, the advected dye is updated in real time. An alternative approach for visualization of time-dependent flow is taken in the second method that utilizes streamline-based concepts in space-time representation [88]. Treating time as the third dimension of 2D unsteady flow enables the application of a wide variety of visualization techniques for 3D stationary vector fields. In the resulting space-time representation, 3D streamlines represent 2D pathlines of the original field. The advantages of the overall approach are demonstrated for vortex analysis and the analysis of the dynamics of material lines. The concept is applied to the extraction of vortex centers, vortex core regions, and the visualization of material line dynamics using streamsurface integration and line integral convolution in the space-time field.

Topological flow structure reveals the qualitative flow behavior using concise representation that allows for effective analysis of complex flows. The third presented visualization technique extracts vector field topology implicitly by means of adaptive sampling by streamlines and determination if the streamlines sufficiently approach a critical point [91]. It reveals regions of different flow behavior by assigning to each seed point the ID of the critical point reached by the respective streamline. The integration performed forward and backward in time provides an implicit representation of the topological skeleton as the boundaries of the obtained regions. Hence, the potentially complicated extraction of various topological structures, such as periodic orbits or boundary switch points, is avoided.

5.1 Dye-Based Visualization of Advection-Diffusion in Unsteady Flow

For the study of advective processes in experimental fluid dynamics, a commonly employed approach is the visualization of flows by markers injected into the flow stream. Typically, smoke and ink are used in the analysis of gases and liquids, respectively. Due to the visual appeal of this approach and its relatively simple interpretation, the development of a virtual counterpart has drawn much attention in the field of computational flow visualization, where it is called dye advection.

Both in simulation and visualization, major effort has been spent to avoid the involved *numerical diffusion*, i.e., artificial blurring of the quantity due to repeated interpolation during advection in discrete grids. Although intricate details tend to be removed in the effect of numerical diffusion, dye advection has the advantage that no geometric representation of dye needs to be maintained, allowing for visualization of arbitrarily complex flows.

This work contributes to the flow visualization in a multitude of ways. Since in many physical processes an important part of transport is diffusion, an approach has been developed that solves the advection-diffusion equation and hence allows for visualization of transport for quantities that undergo both advection and diffusion, as opposed to traditional dye advection which only accounts for advective transport. Additionally, to reduce the numerical diffusion, the finite volume method with WENO reconstruction scheme has been adopted. Finally, for high quality visualization at interactive frame rates, the method has been implemented on a GPU. [1]

5.1.1 Related Work

In scientific visualization, one line of research adopts a Lagrangian view on texture-based flow visualization—with line integral convolution (LIC) being the most prominent example [30]. The other area of research, which is relevant to this work, performs texture advection from frame to frame. Most of previous work in this field is based on semi-Lagrangian advection or similar schemes. For example, texture advection techniques address 2D visualization [80, 198, 192], 3D visualization [173, 193], and visualization on surfaces [98, 199, 103]. It is also possible to combine dye advection techniques with LIC, as described by Shen et al. [163]. An overview of texture-based flow visualization techniques in general, including further references to prior work, is provided by Laramee et al. [97]. A serious problem of semi-Lagrangian advection is the high level of numerical diffusion introduced by repeated resampling of the transported texture. In particular, resampling with bilinear or trilinear interpolation leads to strong blurring, which can be understood from a signal-processing perspective [191]. Therefore, higher-order reconstruction filters can reduce blurring [1]. An alternative approach adopts the concept of level-set advection to avoid blurring for advected dye [37, 190].

[1] Parts of this section have been published in: [89] and [90]

In this approach, the boundary between dye and background is modeled as an interface transported without blur.

However, none of the above methods from scientific visualization modeled diffusion explicitly. In fact, there is previous work that uses diffusion for flow visualization. For example, Sanderson et al. [158] employed reaction-diffusion for flow visualization, adapting reaction-diffusion methods for generic texture synthesis in computer graphics [178, 203]. Similarly, Markov random field texture synthesis can be adopted for flow visualization [172]. Also, anisotropic (non-physical) diffusion, known from image filtering, can be applied to flow visualization [39]. An extension to unsteady flow was provided by Bürkle et al. [29], by adapting the diffusion tensor and blending the transport diffusion evolution results started at successively incremented times. Finally, there is a large body of research on diffusion tensor visualization, mostly in medical imaging and visualization [182]. It is important to note that none of these diffusion-related papers use physical diffusion in combination with physical advection.

The most closely related work is the physically-based dye advection by Li et al. [102]. They applied the piecewise parabolic method [34] for visualization by dye advection in time-dependent 2D flow fields, providing a technique exhibiting low numerical diffusion. The technique allows for comparably large time steps by advecting each cell backward in time and sampling a parabolic reconstruction of the dye inside the resulting polygon. In the present work, several reasons motivated the choice for the weighted essentially non-oscillatory scheme [108] (WENO) instead: it represents a generalization to arbitrary order of accuracy, is based on a clear mathematical foundation, lends itself better to parallelization, and allows for the incorporation of diffusion using a finite volume formulation.

The approach is based on the WENO scheme as described by Dumbser and Käser [41], where dimensional splitting, i.e., application of the 1D scheme subsequently in the three spatial dimensions, reduces computation complexity and lends itself well for parallelization on GPUs. The implementation of active diffusion follows the method by Chou and Shu [33], with the exception that linear instead of WENO weights are used for computing the concentration gradient.

5.1.2 Dye Advection and Diffusion

For the visualization of dye in the simulation data, a one-way coupling is assumed, i.e., the advection of virtual dye has no effect on the underlying velocity field. Hence, in fluid dynamics, such simulation of transport of quantities due to prescribed velocity is called *passive advection*. It leads to a linear problem that contrasts *active advection*, the advection of velocity itself during flow simulation, resulting in a nonlinear problem that is harder to solve. This is one of the reasons why visualization by dye advection tends to be faster than the flow simulation itself. Nevertheless, numerical diffusion is a major problem also with passive advection, and substantial effort has been taken to reduce it, as, for example, in the dye advection method by Li et al. [102], typically at the cost of reduced performance.

The traditional (passive) advection equation reads

$$\frac{\partial \phi}{\partial t} + (\nabla \phi)\mathbf{u} = 0\,, \tag{5.1}$$

with the concentration $\phi = \phi(\mathbf{x}, t)$ and velocity $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$. This is the continuity equation (cf. Equation 2.6) and it states that the temporal change of dye concentration in the Eulerian frame (i.e., observed at fixed position) is only due to the movement of the dye caused by the flow.

If physical diffusion is included in the advection problem, it typically leads to the advection-diffusion equation. While only passive advection is addressed here, *diffusion* traditionally plays an active role in science and engineering, i.e., it is the concentration variation of the quantity itself that governs its diffusion. If the virtual dye takes the role of the quantity, we obtain physical dye advection accounting for diffusion of solubles in typical flow media. This approach is called here *active diffusion* to contrast it from the second variant—*passive diffusion*. Passive diffusion is introduced as a means of visualizing the mechanisms behind concentration changes due to diffusion. In this case, the flow field needs to be accompanied by a (time-dependent) concentration field for analysis by the presented technique.

Dye Active Diffusion

The active diffusion of dye follows Fick's second law, and the concentration change is dictated by the Laplacian of ϕ:

$$\frac{\partial \phi}{\partial t} = D_\phi \Delta \phi\,, \tag{5.2}$$

with the constant of diffusivity D_ϕ, and Laplacian Δ. By combining the diffusion term with the advection term, the advection-active diffusion equation is obtained:

$$\frac{\partial \phi}{\partial t} + (\nabla \phi)\mathbf{u} = D_\phi \Delta \phi\,. \tag{5.3}$$

In general flow visualization, the diffusivity D_ϕ can be chosen, e.g., from physics textbooks, to achieve physically correct interaction of the fluid and tracers used in experiments, such as smoke in the analysis of automotive design (Figure 5.2). If the velocity field resulted from a flow simulation that included diffusion, i.e., an advection-diffusion problem, and if the used diffusivity is known, it can be used as D_ϕ to obtain corresponding dye behavior. Although only isotropic diffusion is considered in this work, it is worth noticing that the finite volume approach also allows for, e.g., data-driven anisotropic diffusion. Due to the low numerical diffusion of the WENO scheme (Section 5.1.3) and its ability to provide interactive dye advection in 3D, this reduced mode already provides a useful visualization technique that mimics dye diffusion in experiments.

Dye Passive Diffusion

A large number of processes simulated with computational fluid dynamics involve diffusion, where some physical quantity, such as heat or solubles, is distributed from regions with higher concentration to regions with lower concentrations by particle collisions at the molecular level. This type of transport, i.e., the passive diffusion, follows Fick's first law, where the resulting concentration change of quantity is dictated by the gradient of the concentration field ψ:

$$\mathbf{j} = D_\psi \nabla \psi, \tag{5.4}$$

where $\mathbf{j}$ is the diffusion flux and D_ψ is the constant of diffusivity for a given quantity ψ. Hence, the visualization of transport of quantities due to diffusion requires an accompanying concentration field ψ together with its constant of diffusivity D_ψ. If passive diffusion is not combined with active diffusion ($D_\phi = 0$), as in the presented results, the equation for advection-passive diffusion is formed:

$$\frac{\partial \phi}{\partial t} + (\nabla \phi)\mathbf{u} - (\nabla \phi) D_\psi \nabla \psi = 0\,. \tag{5.5}$$

The advection-passive diffusion model can be incorporated into the traditional passive advection scheme (Equation 5.1) by combining advection flux and diffusion flux:

$$\frac{\partial \phi}{\partial t} + (\nabla \phi)(\mathbf{u} - D_\psi \nabla \psi) = 0\,. \tag{5.6}$$

Section 5.1.3 provides more details on how the approach is formulated in terms of the finite volume scheme. Interestingly, the passive diffusion term is implemented similarly to the active diffusion term there, not the advection term. Whereas the active diffusion variant builds only on the velocity field, the passive diffusion is based on the gradient of the concentration field that governs the diffusive transport. Please note that with the formulation in Equation 5.5 and 5.6, density and thermal conductivity are treated as constant values.

5.1.3 Finite Volume Method

As described in the previous section, the three different schemes (traditional dye advection, active diffusion, and passive diffusion) can be accomplished by solving the respective partial differential equations. A common approach in solving these equations numerically is by the finite volume method [69] where the temporal change of the quantity within a cell is modeled by its fluxes over the boundaries of the cell and the quantity itself is represented in a per-cell manner. The main advantage of this numerical scheme is that it is conservative, as the transported quantities are explicitly subtracted from the upstream and added to the downstream cell. This is especially important for the computation of diffusive transport which requires comparably small time steps. Finally, the method lends itself well for parallelization on GPUs, since the three-dimensional transport can be split into three consecutive one-dimensional integration steps.

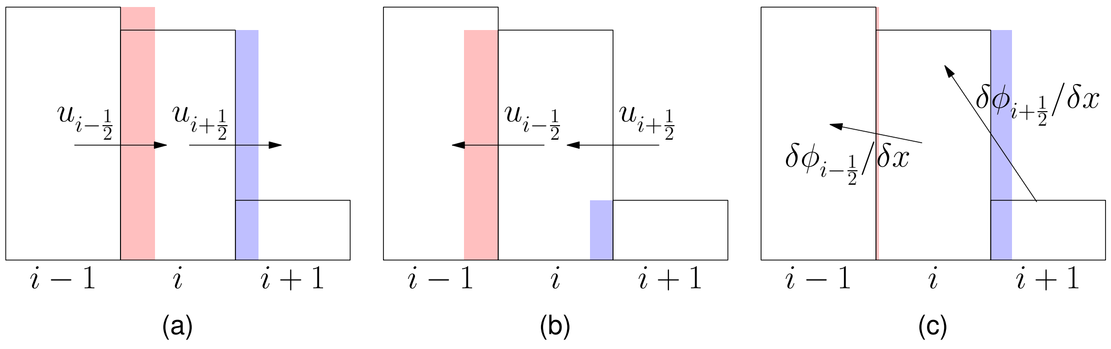


Figure 5.1: Finite volume method with (a), (b) advective flux depending on the direction of the velocity at cell faces and (c) diffusion flux depending on the gradient sign.

The finite volume approach (including WENO reconstruction) can be formulated as one-dimensional problem using dimensional splitting. Since on uniform (or rectilinear) grids cell face normals coincide with coordinate axes, one can apply the one-dimensional procedure sequentially in x-, y-, and z-direction to accomplish a time step of the 3D dye advection (see [79]). Hence, the 1D finite volume approach described below together with the 1D reconstruction described in Section 5.1.3 are applied for each direction, denoted here as x with respective velocity component u.

Advection Let $\overline{\phi}(t,x_i)$ be the amount of dye inside cell x_i at time t, and $\overline{\phi}(x_i,t+\delta t)$ the amount after time step δt. The concentration change $\delta\overline{\phi}(x_i) = \overline{\phi}(x_i,t+\delta t) - \overline{\phi}(x_i,t)$ according to Equation 5.1 is computed in the finite volume scheme as

$$\delta\overline{\phi}(x_i) = (\overline{f}_{i-\frac{1}{2}} - \overline{f}_{i+\frac{1}{2}})\delta t\,, \tag{5.7}$$

i.e., the balance over time δt between flux $\overline{f}_{i-\frac{1}{2}}$ on the left cell face and flux $\overline{f}_{i+\frac{1}{2}}$ on the right cell face:

$$\overline{f}_{i-\frac{1}{2}} = u(x_{i-\frac{1}{2}})\phi(x_{i-\frac{1}{2}})\,, \quad \overline{f}_{i+\frac{1}{2}} = u(x_{i+\frac{1}{2}})\phi(x_{i+\frac{1}{2}})\,. \tag{5.8}$$

According to Figure 5.1, the Riemann solution of $\overline{f}_{i-\frac{1}{2}}$ and $\overline{f}_{i+\frac{1}{2}}$ depends on the direction of u (u is interpolated in space and time from the simulation data). That is, the concentration is taken from the cell either on the left or on the right side of the cell faces $x_{i-\frac{1}{2}}$ and $x_{i+\frac{1}{2}}$ (see Figure 5.1):

$$\phi(x_{i-\frac{1}{2}}) = \begin{cases} \phi_{i-1} & \text{if } u(x_{i-\frac{1}{2}}) \geq 0\,, \\ \phi_i & \text{if } u(x_{i-\frac{1}{2}}) < 0\,, \end{cases} \tag{5.9}$$

$$\phi(x_{i+\frac{1}{2}}) = \begin{cases} \phi_i & \text{if } u(x_{i+\frac{1}{2}}) \geq 0\,, \\ \phi_{i+1} & \text{if } u(x_{i+\frac{1}{2}}) < 0. \end{cases} \tag{5.10}$$

Figure 5.2: Flow over a “blunt body” (front to back). Advection of dye without diffusion model (left, with a streak line for comparison). Active diffusion (right) mimics diffusion of smoke in experimental analysis. Resolution $600 \times 125 \times 121$.

For convergence, δt has to be chosen sufficiently small to avoid transport of quantities larger than the available value in the upstream cell. This is usually done by prescribing a Courant-Friedrichs-Lewy (CFL) condition value. CFL relates to the maximum velocity $u_{\max}$ in the field and cell size h by $C = u_{\max}\delta t/h$, where C is called the Courant number and $C < 1$ for convergence.

Active Diffusion Next, the active diffusion is included to obtain the finite volume formulation of Equation 5.3. Applying the vector identity $\Delta = \nabla \cdot \nabla$ to its right hand side one obtains $D_\phi \Delta\phi = D_\phi \nabla \cdot \nabla\phi$. Assuming uniform D_ϕ and applying the Gauss theorem:

$$D_\phi \int_V \nabla \cdot \nabla\phi \, \mathrm{d}V = D_\phi \int_{S=\partial V} \nabla\phi \cdot \mathbf{n} \, \mathrm{d}S,$$

the diffusion flux $D_\phi \nabla\phi$ of the virtual dye through the cell face with normal $\mathbf{n}$ is obtained. In a 1D scheme this gives rise to diffusion fluxes

$$\overline{d}_{i-\frac{1}{2}} = D_\phi \frac{\partial\phi(x_{i-\frac{1}{2}})}{\partial x}, \quad \overline{d}_{i+\frac{1}{2}} = D_\phi \frac{\partial\phi(x_{i+\frac{1}{2}})}{\partial x}. \tag{5.11}$$

Hence, including active diffusion, the concentration change in cell x_i becomes

$$\delta\overline{\phi}(x_i) = (\overline{f}_{i-\frac{1}{2}} - \overline{f}_{i+\frac{1}{2}} - \overline{d}_{i-\frac{1}{2}} + \overline{d}_{i+\frac{1}{2}})\delta t. \tag{5.12}$$

The CFL condition must be satisfied also for the active diffusion such that $C_D < 1$ with $C_D = D_\phi \nabla\phi_{\max}\delta t/h$.

Passive Diffusion Finally, passive diffusion is included in the dye transport equation (Equation 5.1) by deriving a velocity field $u = -D_\psi \partial\psi/\partial x$ for passive diffusion only, and $u = u_{\text{sim}} - D_\psi \partial\psi/\partial x$ for advection with passive diffusion, where u_{sim} is the simulation velocity field. For the computation of fluxes, Equation 5.8 is applied on the derived velocity field.

The evaluation of the necessary fluxes at the cell boundaries requires reconstruction of the cell-centered quantities. As demonstrated in Figure 5.3, the first-order scheme, which assumes uniform concentration in the donor side, leads to excessive blur. This phenomenon is known as the numerical diffusion problem and much effort has been

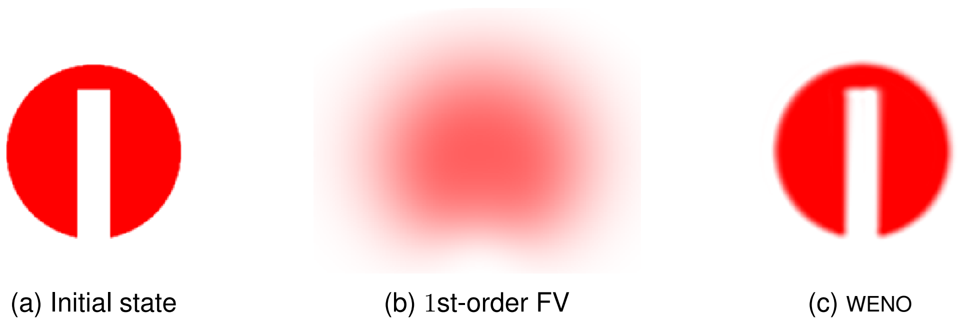


Figure 5.3: Zalesak's disk test. The WENO-based approach exhibits substantially lower numerical diffusion than the first-order upwind scheme.

put into reducing this effect. One of the developed techniques, which has been extensively used in fluid simulations, is the WENO reconstruction [41]. Since it substantially improves the quality of dye transport, and lends itself well for parallelization on GPUs, achieved by the dimensional splitting, it has also been employed in the presented technique. In the following sections, details are provided on the characteristics of numerical diffusion, as well as on the WENO procedure.

Numerical Diffusion

One disadvantage of the finite volume method—both in simulation and visualization—is that its quality highly depends on the grid resolution. The cell size effectively limits the detail level that can be captured with the finite volume approach. For instance, in flow exhibiting foliation, i.e., repeated thinning and folding of the fluid, preserving the resulting intricate details of arbitrarily many and arbitrarily finely folded sheets would require extremely fine resolution. Moreover, due to the repeated interpolation at the cell faces, a quantity undergoes numerical diffusion whose propagation speed highly depends on the cell size, and hence, finer resolution is required to minimize the effect. Undersampling and numerical diffusion are therefore omnipresent problems, typically leading to results that deviate substantially from the true physical behavior.

All three dye transport scenarios are subject to numerical diffusion. Numerical diffusion of the dye ϕ is, however, kept comparably low due to WENO reconstruction (described below). In the scenario of dye advection with active diffusion (Equation 5.3), the dye ϕ is subject to both active diffusion due to diffusivity $D_\phi > 0$ and numerical diffusion. Hence, the effective diffusion of ϕ tends to exceed that prescribed by D_ϕ. Since numerical diffusion is not quantified, one approach to judge its influence is to compare the result with dye advection using $D_\phi = 0$. The results are compared visually: small difference indicates that numerical diffusion affects (is in the order of) the one modeled by D_ϕ. Finally, in the scenario of passive diffusion (Equation 5.5), one can make use of the concentration field ψ from simulation data if there are identifiable sources therein. For the example of the *Buoyant Flow* dataset, one can obtain the region where the room is heated by applying a threshold filter to the ψ field. By continuously seeding dye in

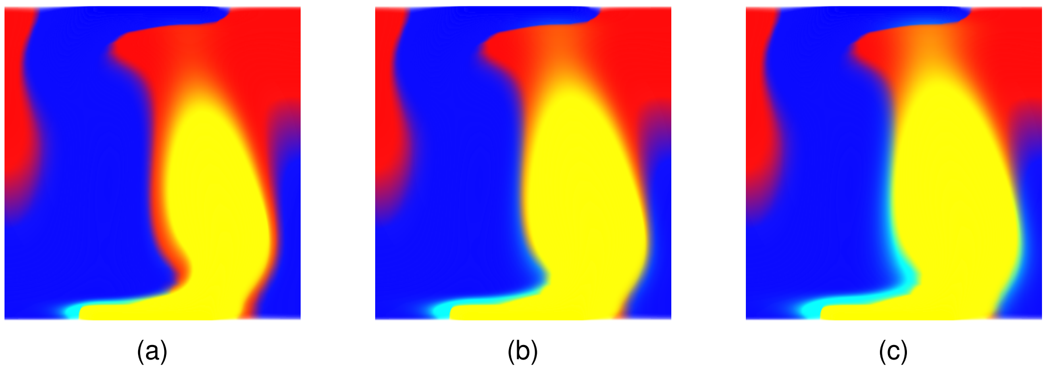


Figure 5.4: Diffusivity adjustment in the *Buoyant Flow* dataset (vertical cross section, low temperature - blue, high temperature - red). Virtual dye (green, mixed with color from temperature) seeded at hot plate at lower image border. Using (a) $D_\phi = 1.11 \cdot 10^{-5}$, (b) $D_\phi = 1.65 \cdot 10^{-5}$, and (c) (theoretical) diffusivity from simulation $D_\phi = 2.19 \cdot 10^{-5}$. In (a) the dye region is too small, in (c) too large, and in (b) the reduced diffusivity compensates numerical diffusion well.

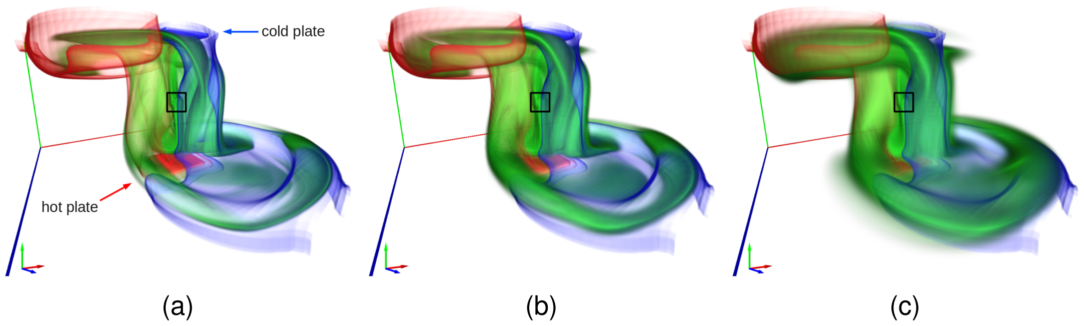


Figure 5.5: Dye advection in *Buoyant Flow* dataset. Advected dye (green, without diffusion model) seeded at center (black box). Finite volume method using WENO reconstruction at both dye resolutions (a) $244 \times 124 \times 244$ and (b) $122 \times 62 \times 122$ exhibits much lower numerical diffusion than first-order reconstruction at $244 \times 124 \times 244$ in (c).

this region and using the active diffusion model (Equation 5.3), one can adjust D_ϕ until ϕ matches the (time-dependent) ψ field, see Figure 5.4. This can compensate for inappropriate D_ψ due to both numerical diffusion in the simulation and dye advection. The obtained D_ϕ can then also be used as D_ψ in the passive diffusion model.

WENO Reconstruction

As stated in the previous section, the finite volume formulation requires the evaluation of fluxes at the cells faces, necessitating the reconstruction of ϕ between neighboring cells. As opposed to the first-order upwind scheme described in the previous section, the WENO reconstruction is a higher-order method based on polynomial reconstruction that eliminates undesired oscillations of the reconstructed polynomial and provides high quality solution to the advection problem. Because of these characteristics and also due to relatively easy parallelization, this method has been widely used in simulations.

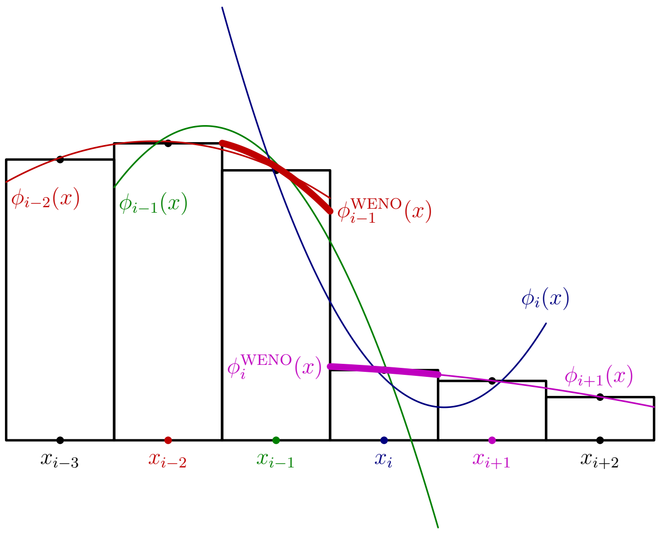
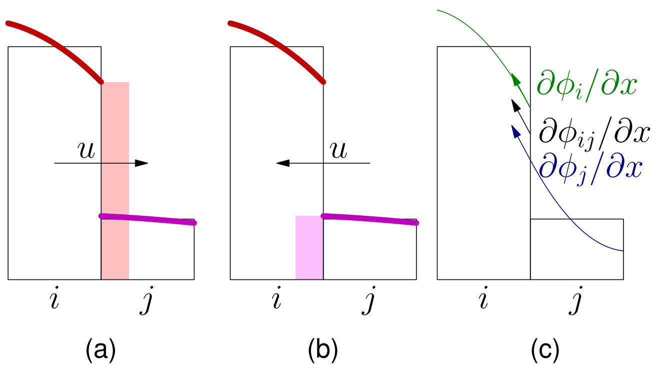


Figure 5.6: Example of 1D WENO reconstruction with quadratic polynomials. Cell centers x with cell-averaged values $\overline{\phi}$ and reconstruction polynomials $\phi(x)$. Resulting reconstruction $\phi^{\mathrm{WENO}}(x)$ for cells $i-1$ and i (bold).



u
u
$\partial\phi_i/\partial x$
$\partial\phi_{ij}/\partial x$
$\partial\phi_j/\partial x$
i
j
i
j
i
j
(a)
(b)
(c)

Figure 5.7: (a), (b) Riemann solution for fluxes. WENO reconstruction (bold) exhibits discontinuities at cell faces. Flux (transparent rectangle) is determined from advection direction u, choosing "donor" side. (c) For diffusion, gradient at faces is computed from central reconstruction polynomial (green, blue) and averaged (black) (cf. Figure 5.6).

The WENO reconstruction is described in this section in 1D, as it is applied in the dye advection technique.

Similarly to the upwind scheme (Equations 5.9 and 5.10), the flux is determined explicitly from the velocity direction at the cell face. However, the actual values $\phi(x_{i\pm\frac{1}{2}})$ are computed at the cell faces from the reconstructed polynomials ϕ^{WENO}.

$$\phi^{\mathrm{WENO}}(x_{i-\frac{1}{2}}) = \begin{cases} \phi^{\mathrm{WENO}}_{i-1}(x_{i-\frac{1}{2}}) & \text{if } u(x_{i-\frac{1}{2}}) \geq 0\,, \\ \phi^{\mathrm{WENO}}_{i}(x_{i-\frac{1}{2}}) & \text{if } u(x_{i-\frac{1}{2}}) < 0\,, \end{cases} \tag{5.13}$$

$$\phi^{\mathrm{WENO}}(x_{i+\frac{1}{2}}) = \begin{cases} \phi^{\mathrm{WENO}}_{i}(x_{i+\frac{1}{2}}) & \text{if } u(x_{i+\frac{1}{2}}) \geq 0\,, \\ \phi^{\mathrm{WENO}}_{i+1}(x_{i+\frac{1}{2}}) & \text{if } u(x_{i+\frac{1}{2}}) < 0\,. \end{cases} \tag{5.14}$$

Below, the computation of the polynomial ϕ^{WENO} is described for the example of third-order accurate reconstruction (i.e., using second-order polynomials). In the experiments demonstrated in Section 5.1.5, a good trade-off was obtained between efficiency and accuracy using fourth-order accurate WENO, i.e., cubic polynomials. The reader is referred to [41] for a thorough introduction to the topic and further details, also regarding the extension to higher degrees.

Consider a second-order reconstruction polynomial

$$\phi_k(x) = \sum_{j=0}^{2} \hat{w}_j^k x^j \,, \tag{5.15}$$

with k being the index of their central cell, and coefficients $\hat{w}_j^k$ (see Figure 5.6). The coefficients are chosen such that the integrals of ϕ_k over the cells are conservative, i.e., they are identical to the cell-centered values $\overline{\phi}_i$:

$$\int_{x_{i-\frac{3}{2}}}^{x_{i-\frac{1}{2}}} \phi_k \,\mathrm{d}x = \overline{\phi}_{i-1}\,, \quad \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \phi_k \,\mathrm{d}x = \overline{\phi}_{i}\,, \quad \int_{x_{i+\frac{1}{2}}}^{x_{i+\frac{3}{2}}} \phi_k \,\mathrm{d}x = \overline{\phi}_{i+1}\,. \tag{5.16}$$

Solving the integrals, polynomials with coefficients $\hat{w}_j^k$ are obtained

$$\begin{aligned}
\hat{w}_0^k(x_{i\text{-}\frac{1}{2}} - x_{i\text{-}\frac{3}{2}}) + \tfrac{\hat{w}_1^k}{2}(x^2_{i\text{-}\frac{1}{2}} - x^2_{i\text{-}\frac{3}{2}}) + \tfrac{\hat{w}_2^k}{3}(x^3_{i\text{-}\frac{1}{2}} - x^3_{i\text{-}\frac{3}{2}}) &= \overline{\phi}_{i-1}\\
\hat{w}_0^k(x_{i+\frac{1}{2}} - x_{i\text{-}\frac{1}{2}}) + \tfrac{\hat{w}_1^k}{2}(x^2_{i+\frac{1}{2}} - x^2_{i\text{-}\frac{1}{2}}) + \tfrac{\hat{w}_2^k}{3}(x^3_{i+\frac{1}{2}} - x^3_{i\text{-}\frac{1}{2}}) &= \overline{\phi}_{i}\\
\hat{w}_0^k(x_{i+\frac{3}{2}} - x_{i+\frac{1}{2}}) + \tfrac{\hat{w}_1^k}{2}(x^2_{i+\frac{3}{2}} - x^2_{i+\frac{1}{2}}) + \tfrac{\hat{w}_2^k}{3}(x^3_{i+\frac{3}{2}} - x^3_{i+\frac{1}{2}}) &= \overline{\phi}_{i+1}
\end{aligned} \tag{5.17}$$

that can be formulated as a matrix-vector product:

$$\mathbf{L}_k \begin{pmatrix} \hat{w}_0^k \\ \hat{w}_1^k \\ \hat{w}_2^k \end{pmatrix} = \begin{pmatrix} \overline{\phi}_{i-1} \\ \overline{\phi}_{i} \\ \overline{\phi}_{i+1} \end{pmatrix}$$

with $\mathbf{L}_k$ representing the stencil matrix:

$$\mathbf{L}_k = \begin{pmatrix}
(x_{i-\frac{1}{2}} - x_{i-\frac{3}{2}}) & \frac{1}{2}(x^2_{i-\frac{1}{2}} - x^2_{i-\frac{3}{2}}) & \frac{1}{3}(x^3_{i-\frac{1}{2}} - x^3_{i-\frac{3}{2}}) \\
(x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}) & \frac{1}{2}(x^2_{i+\frac{1}{2}} - x^2_{i-\frac{1}{2}}) & \frac{1}{3}(x^3_{i+\frac{1}{2}} - x^3_{i-\frac{1}{2}}) \\
(x_{i+\frac{3}{2}} - x_{i+\frac{1}{2}}) & \frac{1}{2}(x^2_{i+\frac{3}{2}} - x^2_{i+\frac{1}{2}}) & \frac{1}{3}(x^3_{i+\frac{3}{2}} - x^3_{i+\frac{1}{2}})
\end{pmatrix}.$$

During dye advection, the coefficients $\hat{w}_j^k$ are determined by

$$\begin{pmatrix} \hat{w}_0^k \\ \hat{w}_1^k \\ \hat{w}_2^k \end{pmatrix} = \mathbf{L}_k^{-1} \begin{pmatrix} \overline{\phi}_{i-1} \\ \overline{\phi}_{i} \\ \overline{\phi}_{i+1} \end{pmatrix}.$$

Since the stencil matrix only depends on polynomial degree and the cell index k relative to the central cell i, its inverse $\mathbf{L}_k^{-1}$ can be precomputed. The reconstruction $\phi_i^{\mathrm{WENO}}(x)$ of the concentration inside cell i is a linear combination of the polynomials $\phi_k(x)$:

$$\phi_i^{\mathrm{WENO}}(x) = \sum_{k=i-1}^{i+1} \omega_k \phi_k(x)$$

with weights $\omega_k = \tilde{\omega}_k / \sum_{j=i-1}^{i+1} \tilde{\omega}_j$ using $\tilde{\omega}_j = \lambda_j/(\varepsilon+\sigma_j)^r$. As reported in [41], values $\varepsilon = 10^{-5}$ and $r = 4$ are used, and λ_j is chosen such that the polynomial of the central cell is favored:

$$\lambda_j = \begin{cases} 10^3 & \text{if } j = i, \\ 1 & \text{otherwise.} \end{cases}$$

The oscillation indicators σ_j assign high values to less smooth polynomials and can be obtained from $\sigma_j = \hat{\mathbf{w}}^j \Sigma \hat{\mathbf{w}}^j$, where the elements of matrix Σ in the third-order WENO example are given by

$$\Sigma_{mn} = \sum_{r=1}^{2} \int_V \frac{\partial^r x^m}{\partial x^r} \frac{\partial^r x^n}{\partial x^r} \, \mathrm{d}x.$$

The matrix Σ can also be precomputed, e.g., using a computer algebra system, since, after transformation to a reference space, it neither depends on the mesh nor on the problem. In Figure 5.6, the resulting WENO reconstruction polynomials for the two cells in the center are marked with bold lines.

For the computation of the diffusion fluxes $\overline{d}_{i+\frac{1}{2}}$ and $\overline{d}_{i-\frac{1}{2}}$, the concentration gradient at position $x_{i-\frac{1}{2}}$ (and respectively $x_{i+\frac{1}{2}}$) is computed by averaging the gradients of the central polynomials of cells $i-1$ and i (i and $i+1$, respectively):

$$\frac{\partial \phi(x_{i-\frac{1}{2}})}{\partial x} = \frac{1}{2}\left(\frac{\partial \phi_{k=i-1}(x_{i-\frac{1}{2}})}{\partial x} + \frac{\partial \phi_{k=i}(x_{i-\frac{1}{2}})}{\partial x} \right),$$

$$\frac{\partial \phi(x_{i+\frac{1}{2}})}{\partial x} = \frac{1}{2}\left(\frac{\partial \phi_{k=i}(x_{i+\frac{1}{2}})}{\partial x} + \frac{\partial \phi_{k=i+1}(x_{i+\frac{1}{2}})}{\partial x} \right).$$

In the example with the second-order polynomial, this results in

$$\frac{\partial \phi(x_{i-\frac{1}{2}})}{\partial x} = \frac{1}{2}\left(\hat{w}_1^{i-1} + 2\hat{w}_2^{i-1} x_{i-\frac{1}{2}} + \hat{w}_1^{i} + 2\hat{w}_2^{i} x_{i-\frac{1}{2}} \right)$$

and

$$\frac{\partial \phi(x_{i+\frac{1}{2}})}{\partial x} = \frac{1}{2}\left(\hat{w}_1^{i} + 2\hat{w}_2^{i} x_{i+\frac{1}{2}} + \hat{w}_1^{i+1} + 2\hat{w}_2^{i+1} x_{i+\frac{1}{2}} \right),$$

on the left and right cell face, respectively. See Figure 5.7(c) for an illustration.

To simplify implementation, the computation of $\phi^{\text{WENO}}(x \pm \frac{1}{2})$ is done in a reference space, where $\delta x = 2$ and $x = 0$. Thus, the evaluation of the polynomial at the cell faces where $x = \pm 1$ reduces to $\phi_k(x) = \sum_k w_k \cdot (\pm 1)^k$.

Prediction Steps

With the computed concentration $\phi^{\text{WENO}}(x_{i+\frac{1}{2}})$, the advective flux on the right cell face could be obtained by Equation 5.8, with time-averaged $u_{\text{avg}} = [u_{\text{sim}}(x_{i+\frac{1}{2}}, t) + u_{\text{sim}}(x_{i+\frac{1}{2}}, t+\delta t)]/2$. Including passive diffusion would yield the velocity $u = u_{\text{avg}} -$

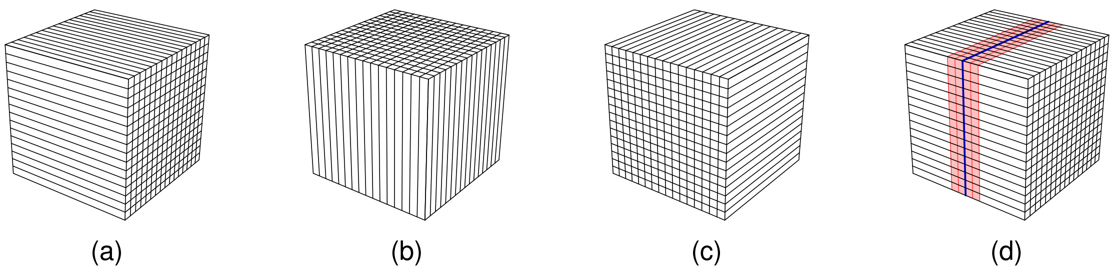


Figure 5.8: (a)–(c) Dimensional splitting applied in the finite volume method reduces the dye transport to 1D problem, subsequently in x-, y-, and z-direction. (d) Blocks that do not fit into GPU shared memory are divided (dark blue line, for x-direction), and ghost cells (light red) are stored for the computation of the reconstruction polynomials.

$D_\psi[\partial\psi(x_{i+\frac{1}{2}},t)/\partial x + \partial\psi(x_{i+\frac{1}{2}},t+\delta t)/\partial x]/2$. An analogous procedure could be done for the left cell face. However, to further improve the accuracy of the finite volume scheme, *prediction steps* are employed instead that account for changing concentration and the resulting polynomial reconstruction within the time interval δt. The underlying idea is to obtain a better accuracy for the fluxes $\overline{f}$ and $\overline{d}$ within $]t, t+\delta t]$ by generating predictions using integration. For this purpose, an approximation $\tilde{\overline{\phi}}_t$ for $\partial\phi/\partial t$ is obtained from Equation 5.3 by moving the advection term to the right side [41]:

$$\tilde{\overline{\phi}}_t := \partial\phi/\partial t = D_\phi \Delta\phi - (\nabla\phi)\mathbf{u}\,.$$

To compute the temporal change of concentration, $\tilde{\overline{\phi}}_i(\tau+\delta t/n) = \tilde{\overline{\phi}}_i(\tau) + \tilde{\overline{\phi}}_t \delta t/n$ is evaluated in parallel for all cells. In the 1D case, $\tilde{\overline{\phi}}_t = D_\phi \partial^2\tilde{\phi}_i/\partial x^2 - \partial\tilde{\phi}_i/\partial x u$. The procedure is repeated n times ($n = 1$ in the experiments demonstrated in Section 5.1.5) and each time WENO reconstruction is applied to $\tilde{\overline{\phi}}_i(\tau)$ to obtain $\tilde{\phi}_i^{\text{WENO}}(\tau)$. In each step, $\overline{f}_{i+\frac{1}{2}}(\tau)$ and $\overline{f}_{i-\frac{1}{2}}(\tau)$ as well as $\overline{d}_{i+\frac{1}{2}}(\tau)$ and $\overline{d}_{i-\frac{1}{2}}(\tau)$ are computed from $\tilde{\phi}_i^{\text{WENO}}(\tau)$. During prediction steps, these fluxes are accumulated and finally used in Equation 5.12 to obtain $\overline{\phi}_i(t+\delta t) = \overline{\phi}_i(t) + \delta\overline{\phi}_i$.

5.1.4 Implementation

The finite volume method with WENO reconstruction scheme greatly benefits from the GPU parallelization due to spatial and temporal locality of the algorithm. Moreover, the dimensional splitting reduces the dye transport computation to a much simpler 1D problem, while data streaming ensures memory efficiency.

In the implementation, the CUDA API has been employed to compute advection-diffusion (with single precision arithmetic). Each GPU thread block loads an array of data from the device memory into its shared memory along the current axis of dimensional splitting (Figure 5.8(a)–(c)). The number of thread blocks equals $b_d = r((d+1) \bmod 3) \times r((d+2) \bmod 3) \times \lceil r(d)/s \rceil$, where $r(d)$ is the resolution in dimension d, and s is the size of a thread block, limited by GPU shared memory size. If the

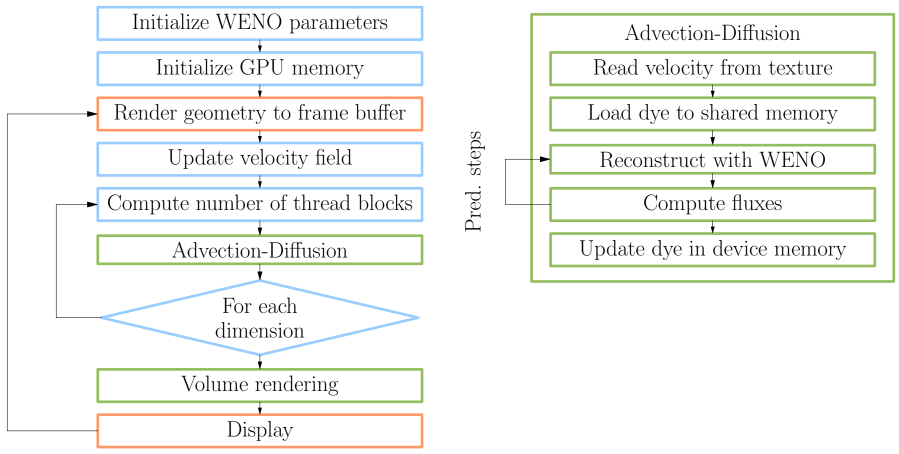


Figure 5.9: Advection-diffusion procedure. Operations in blue blocks are performed on CPU, OpenGL is used in orange blocks, green blocks are done in CUDA on GPU.

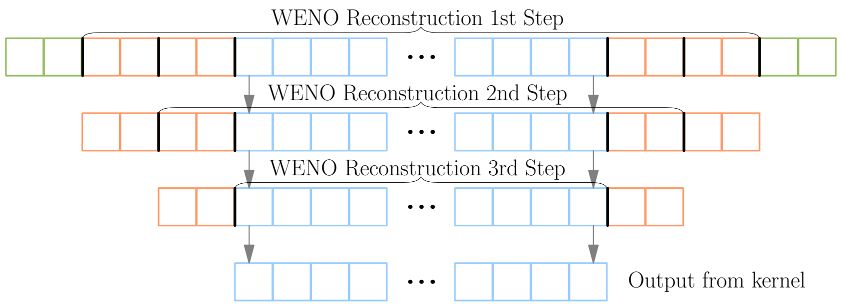

WENO Reconstruction 1st Step
WENO Reconstruction 2nd Step
WENO Reconstruction 3rd Step
Output from kernel

Figure 5.10: Domain splitting for dimensions of the dye grid exceeding GPU shared memory. Number of prediction steps and order of WENO reconstruction define the number of ghost cells.

resolution $r(d)$ is too large, the arrays must be divided into overlapping subsets (Figure 5.8(d)) which are processed separately. The overlaps (ghost cells) are necessary to perform WENO reconstruction, and their size depends on the WENO order and the number of prediction steps. In Figure 5.10, an example array is shown for three prediction steps and second-order polynomial. For optimization, processing of a thread block is skipped if no dye is present in the processed cells. This is determined using parallel reduction (i.e., summation of array elements). Please refer to the Appendix for details on the GPU architecture relevant to the implementation.

Figure 5.9 describes the overall procedure. There are several input parameters that are defined before the interactive advection, such as the order of the WENO polynomials, and the resolution of the advection grid. Additional geometry objects can be loaded and rendered together with the ray-casted volume. Device (GPU) memory is allocated for the dye, the two consecutive time steps of simulation data, and the OpenGL buffers. The input data (i.e., vector field for advection and scalar field for passive advection and/or visualization) are streamed in each loop. The WENO reconstruction is carried out, and the advection and diffusion fluxes are computed. Finally, the dye concentration is

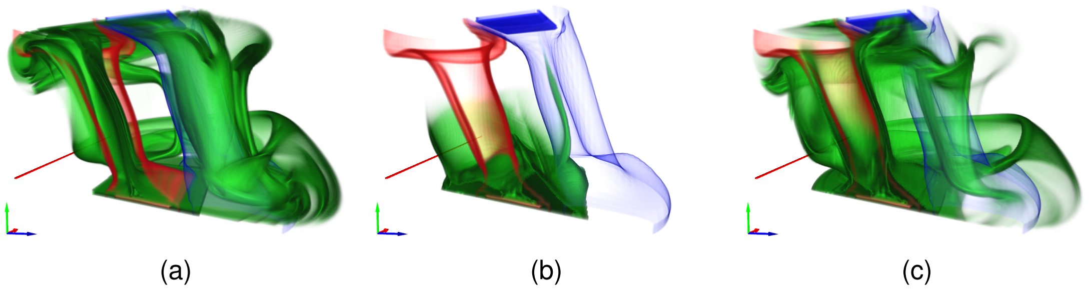


Figure 5.11: Buoyant unsteady flow example (with clipping) by green dye seeded at the hot (red) plate. Pure advection (a), only diffusion flux (b) (passive diffusion), and both advective and diffusion flux (c) (passive advection-diffusion). In (b), diffusion flux transports the dye outward hot air (red) and toward cold air (blue). In (c), the dye reveals the true transport of heat: it is diffusing toward the cold plume and partially leaving at the cold (blue) plate before it is advected downward by the cold plume.

updated in device memory. To avoid fluctuations in physical time during interaction, respective corrections are applied to the dye advection time step δt.

For volume rendering of dye and simulation scalar fields, ray casting based on the example in the NVIDIA's CUDA SDK [123] is utilized, with additional support for volume lighting, ray-casted isosurfaces [101] of the ψ field, and geometric objects.

5.1.5 Results

In this section, three CFD examples are shown that demonstrate the quality and usefulness of the advection-diffusion visualization. Additionally, performance with respect to numerical diffusion, mass conservation, and speed is evaluated. All examples were run on a GeForce GTX 960 ($4\,\mathrm{GiB}$) and were obtained with fourth-order WENO, i.e., using third-order polynomials.

Figure 5.3 demonstrates the quality of the visualization technique with WENO reconstruction on the *Zalesak* dataset. As can be seen, numerical diffusion is substantially reduced. For performance details, please refer to the Table 5.1.

Buoyant Flow Dataset The first CFD dataset, a buoyant flow inside a closed container, was simulated on a uniform grid with $61 \times 31 \times 61$ cells and 2000 time steps, spanning 50 seconds. Heat takes in this case the role of the diffusing quantity. In Figure 5.5, the dataset is shown. The heating plate at bottom (red, $75\,^\circ\mathrm{C}$) and cooling one at the top (blue, $5\,^\circ\mathrm{C}$) induce a time-dependent circular flow behavior. For context, two temperature isosurfaces, one at $38\,^\circ\mathrm{C}$ (blue) and one at $42\,^\circ\mathrm{C}$ (red), are rendered. No diffusion model was used for the dye in this figure, it therefore reveals the mixing behavior, i.e., the stretching of the fluid into sheets, leading to foliation.

The presented visualization technique is particularly useful for the buoyant flow driven by a heat gradient because such flow exhibits both advection and diffusion of heat. The dye advection in Figure 5.11(a) shows advection only (Equation 5.1): the dye

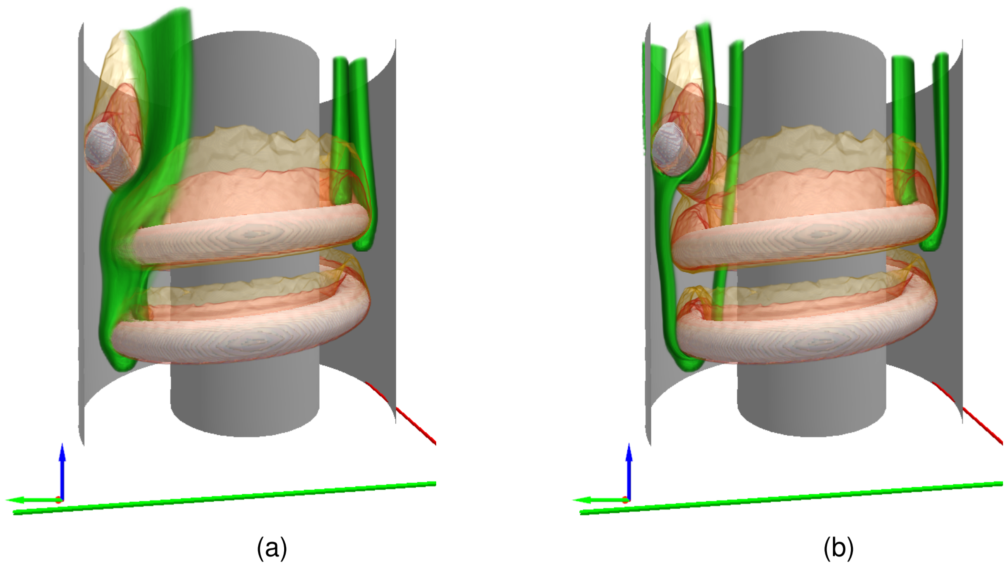


Figure 5.12: *Heating Coil* dataset. Air flow from bottom to top, dye (green) seeded at two points at lower side of coil, and isosurfaces of temperature (red). (a) Dye advection by advection only vs. (b) dye advection by advection-diffusion of heat. It is apparent that heat is repelling the dye from the coil and transporting it to the cooled walls in (b).

follows the hot plume upward and thereafter it is advected downward by the cold plume. Figure 5.11(b) shows only passive diffusion (Equation 5.5 with $\mathbf{u} = \mathbf{0}$): in this case the dye, representing heat, follows the heat gradient and therefore flows from the hot plate through the cold plume into the cold plate at the top of the container. Figure 5.11(c) shows the superposition of the two, the true advection-diffusion of heat: similarly to Figure 5.11(a), the dye is advected upward, part of it, however, is caught by the cold plate instead of being advected downward again by the cold plume.

Heating Coil Dataset The second example, a flow around a heating coil, is quasi-stationary and was simulated on an unstructured tetrahedral grid comprising 93227 cells. The dataset was converted to a uniform grid, where interpolation weights for each node

Table 5.1: Performance for *Buoyant Flow* dataset at different configurations (no lighting and no isosurfaces). 1) Single dye at resolution $122 \times 62 \times 122$ with early rejection (worst case in brackets), and 2) without. 3) Two independent dyes, and 4) one dye at $244 \times 124 \times 244$.

Config.	Avg. FPS		Render [ms]	Dye comp. [ms]	
Conf.1)	55.3	(45.9)	6.67	10.5	(15.9)
Conf.2)	34.1	(33.9)	4.96	23.2	(23.4)
Conf.3)	37.0	(26.4)	7.04	20.2	(30.3)
Conf.4)	17.0	(13.1)	5.96	53.4	(69.6)

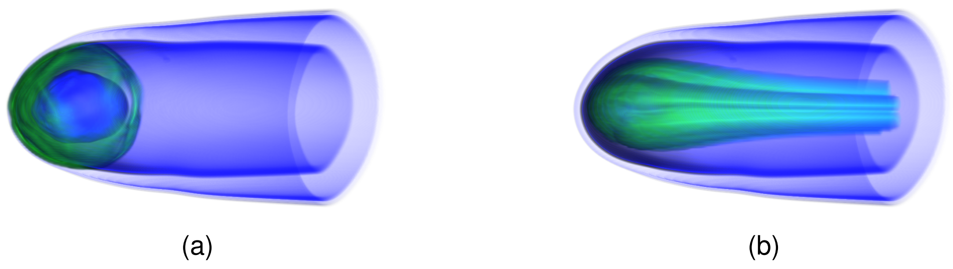


Figure 5.13: *Evaporating Drop* dataset. (a) Transport outward the drop is revealed by passive diffusion only. (b) However, due to the high drop speed, advection dominates the transport behavior.

were precomputed and used for interpolating the simulation data ($\mathbf{u}$, ψ) before transferring to the GPU, avoiding expensive point location at each time step.

Also in this dataset, heat is the diffusive quantity for which the visualization of advection-diffusion has been employed. The coil is located between two cylindrical walls: an inner and an outer one, both cooled. Air flows from bottom to top and is heated by the coil on its way up. The visualization of advection only (Equation 5.1) and advection and passive diffusion (Equation 5.5) reveal different transport behavior, shown in Figure 5.12. One can easily see how heat is transported by advection-diffusion to the cooled walls.

Evaporating Drop Dataset The last CFD example is the quasi-stationary simulation of an evaporating drop, conducted on a uniform grid with resolution $192 \times 128 \times 128$ cells, using a finite volume-based direct numerical simulation employing the volume-of-fluid method for tracking of different phases [161]. In this case, vapor takes the role of the diffusing quantity, therefore the advection-diffusion of vapor is analyzed. In Figure 5.13, the main air flow direction is from the left to the right. Isosurfaces show vapor concentration 0.0001 and 0.005 (blue), and the drop is located to the left. Figure 5.13(b) (where $\mathbf{u} = \mathbf{u}_{\text{sim}} + \mathbf{u}_{\Psi}$) shows advection-diffusion, with dye (green) seeded continuously at the drop. In this case, advection dominates diffusion—the passive advection-diffusion visualization (Equation 5.5) is indistinguishable from that of advection only. Nevertheless, passive diffusion only in Figure 5.13(a) (where $\mathbf{u} = \mathbf{u}_{\Psi}$, and dye is seeded once) reveals that diffusion is strongest at the upstream front of the drop.

5.2 Streamline-Based Concepts for Analysis of 2D Flow in Space-Time

In flow visualization, an important and well-established approach is to operate in the Lagrangian reference frame where techniques based on integral curves are used. This is complementary to the Eulerian-based techniques, an example of which is the FVM-based dye visualization presented in the previous section.

Regarding feature-based flow visualization, streamlines are the basis for many visualization techniques, however only for steady flow. Therefore, there is a shift away from streamline-related concepts to those that take time dependence explicitly into account, and the concepts in space-time representation bridge these two classes of flow visualization. Particularly in the case of 2D unsteady flow, time can be treated as the third dimension, making it possible to apply a wide variety of visualization techniques for 3D stationary vector fields. In the resulting space-time representation, 3D streamlines represent 2D pathlines of the original field. Therefore, concepts based on space-time streamlines are Galilean-invariant and time-aware. Galilean invariance is a beneficial property of visualization techniques, and it is crucial for the analysis of time-dependent flow and when analyzing flow in configurations where no natural frame of reference is given. Considering these characteristics of space-time streamlines, the application of several 3D streamline-based concepts for the analysis of 2D unsteady flow is investigated in this section, resulting in techniques for visualization and feature extraction that are Galilean-invariant and explicitly take time dependence into account. [2]

5.2.1 Related Work

Computational visualization of time-dependent flow by means of integral curves is an established research area. Recent strategies in this field concentrate on accurate interactive placement of individual curves [73] and adaptive interactive placement of small sets of curves [116]. Research in the field of static sets of trajectories focuses on the efficient computation of their end points or quantities along them [72], and placement strategies for sets of whole trajectories. Research in placement of integral curves was initiated by the image-guided streamline placement due to Turk and Banks [179] and by evenly-spaced streamlines of arbitrary density due to Jobard and Lefer [81]. After several works on the placement of streamlines which, e.g., extended the concept to 3D [115] or took into account vector field topology for improved placement [206, 32], there have been works on the placement of streamsurfaces [45] in 3D and placement of streaklines and pathlines in 2D flow to avoid intersections and cusps by limiting their length [189], and on the placement of pathlines that avoids intersections of the curves by decoupling integration and visualization scales [71].

In dense flow visualization, where the curves fill the entire domain, texture-based methods give insight into the flow behavior by manipulating texture color according

[2] Parts of this section have been published in: [90] and [89]

to the flow characteristics. One of the early works in this field is LIC [30]. Here, this technique is employed in the 3D space-time field on surface representations to depict space-time flow behavior.

Vortex core line criteria provide a compact, topological visualization of complex fluid behavior. An overview of standard methods for vortex extraction can be found in the Fundamentals (Section 2.2.2). Weinkauf et al. [187] extended the instantaneous approach to 2D and 3D time-dependent flow by applying it to the respective 3D space-time and the 4D space-time representation. This way, they require vortex core lines in time-dependent flow to be tangent to pathlines instead of streamlines. Note that in the case of 2D flow, the vortex core lines in space-time represent the vortex centers of the 2D flow over time. Fuchs et al. [54] followed a similar approach by replacing acceleration with its time-dependent version in the formulation by Sujudi and Haimes [170]. Methods that use the parallel vectors operator, such as the core line definition by Sujudi and Haimes, provide accurate results when the core lines are straight, but they are rather inaccurate in regions where the core lines are curved [150]. A recent work [112] on the extraction of bifurcation lines according to Perry and Chong [136] employed a refinement of the resulting feature line to avoid this error, but a respective approach for refining vortex core lines is not yet available.

The space-time representation of 2D data was successfully employed in a wide range of time-dependent applications in computational visualization. Examples range from the tracking of critical points in vector fields [174, 176] to the analysis of eye tracking data from video streams [95]. The work most closely related is the vortex core line concept by Weinkauf et al. [187] that considered the time dependence of fields to define features. Weinkauf et al. compare their results to the standard vortex center extraction approach for 2D flow fields, i.e., they also extracted critical points of type focus and center [136]. Whereas they use the Sujudi-Haimes criterion for the space-time core line extraction, in this work Levy et al.'s criterion is employed which, although less accurate, provides a clearer structure of time-dependent vortical flow, because it results, at least in the presented experiments, in core lines that are less disrupted. Another example of the application of space-time representation is the work by Machado et al. [111], where hyperbolic trajectories in 2D unsteady vector field were extracted as space-time bifurcation lines, which allowed for efficient computation of Lagrangian coherent structures.

5.2.2 Space-Time Visualization

A useful property of the space-time representation is that 2D time-dependent vector fields can be transformed into steady 3D ones by treating time as additional dimension. In this representation, pathlines in the 2D time-dependent flow represent streamlines in the 3D space-time field. As discussed below, streamsurfaces in this 3D field represent streaklines or material lines in the original flow, providing an overall framework and facilitating their extraction.

Treating time as the third dimension is analogous to making an n-dimensional system of ordinary differential equations autonomous by lifting its dimension to $n+1$. The time-dependent 2D vector field is given by

$$\mathbf{u}(\mathbf{x},t) := \begin{pmatrix} u(x,y,t) \\ v(x,y,t) \end{pmatrix}, \tag{5.18}$$

with $\mathbf{x} := (x,y)^\top$ and time t. Its 3D space-time representation is obtained by treating the time domain as the third spatial axis:

$$\mathbf{u}'(\mathbf{x}') := \begin{pmatrix} u(x,y,t) \\ v(x,y,t) \\ 1 \end{pmatrix}, \tag{5.19}$$

with $\mathbf{x}' := (x,y,t)^\top$. Setting the third vector component to one ensures correct progress along the time axis when stepping according to the vector field $\mathbf{u}'$. This way, one obtains a stationary 3D vector field that encodes the dynamics of the original time-dependent 2D vector field and is invariant under Galilean transformations—adding constant velocity to $\mathbf{u}'$ changes the orientation of the vector field and hence skews the space-time streamlines, the effect is, however, compensated by the resulting skew of the space-time domain [111]. Please note that trilinear interpolation of $\mathbf{u}'$ is employed in the implementation, i.e., spatial bilinear interpolation is combined with linear interpolation in time.

Characteristic Curves

Streamlines of the original (time-dependent) field $\mathbf{u}$ represent instantaneous integral curves. In the space-time framework, they could be obtained by solving initial value problems in the modified representation $(u(x,y,t), v(x,y,t), 0)^\top$. Pathlines, in contrast, represent the true time-dependent trajectories in flow fields. Since $\mathbf{u}'$ represents a stationary 3D vector field of the original time-dependent 2D vector field $\mathbf{u}$, 3D streamlines in $\mathbf{u}'$ represent pathlines in $\mathbf{u}$, i.e., when projecting the 3D streamline along the time axis, one obtains the respective pathline in $\mathbf{u}$.

A streamsurface is obtained by densely seeding streamlines along a seeding curve. Since each particle of a material line moves along a pathline and because all particles move for the same time duration, one can obtain material lines by means of streamsurface integration in $\mathbf{u}'$. All constant-time sections (i.e., t = const.) of these streamsurfaces represent the material line at the respective time t. Since material lines are seeded only at one instant of time along a curve, the seeding curve of the streamsurface needs to be located in the domain of $\mathbf{u}'$ within an xy-plane, i.e., it has no extent in time. Figures 5.14(a)–5.14(d) demonstrate material lines at selected instants of time, which represent sections of the corresponding space-time streamsurface (obtained using the algorithm by Hultquist [76]) shown in Figure 5.14(e).

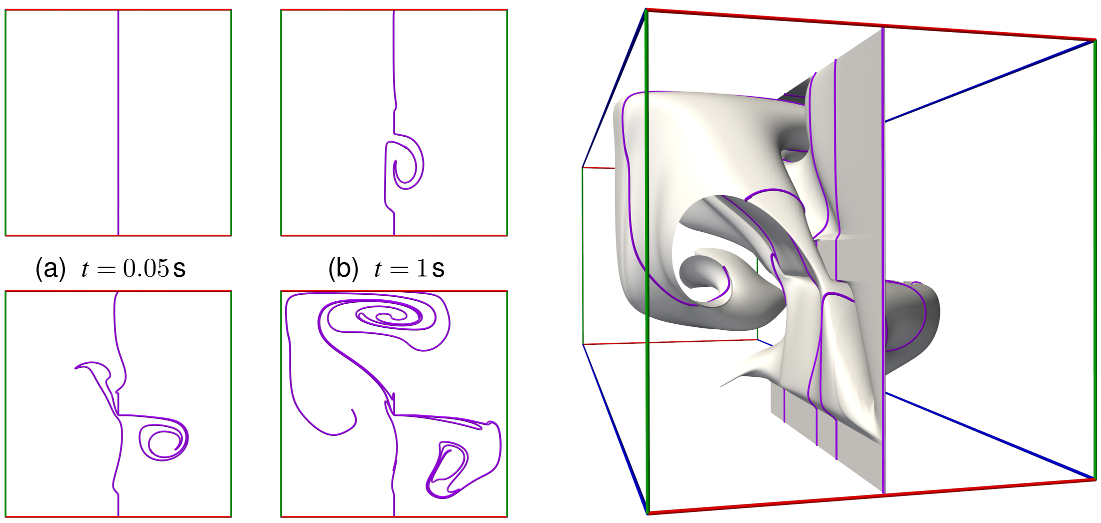


Figure 5.14: Material lines and space-time representation. (a) A material line (purple) is seeded as a vertical line at the center of the spatial domain. (b)–(d) The material line folds and stretches continuously in the unsteady flow. (e) Time sections of the streamsurface of $\mathbf{u}'$ represent the material lines from (a)–(d).

Streamribbons [180] can be seen as a special case between streamlines and streamsurfaces. They represent narrow constant-width strips which show twist near streamlines. Streamribbons are usually constructed by integrating a streamline and during integration, an initially randomly chosen orientation vector is propagated along the streamline. During this propagation, the direction vector is kept orthogonal to the streamline tangent and rotated based on the velocity gradient, i.e., rotated the way a particle in the vicinity of the streamline would locally rotate around the streamline. From this information, a narrow band is constructed by generating a triangulated surface strip along the streamline that approximates the streamribbon. While in true 3D vector fields streamribbons typically represent an appropriate approximation of swirling flow behavior, rotation in $\mathbf{u}'$ takes place in the xy-plane only. Hence, streamribbons whose streamline is not aligned with the t-axis would be subject to inappropriate representation, i.e., the edges of the ribbons could point reverse in time, which would be misleading. Therefore, a dedicated construction algorithm for the space-time vortex core ribbons is proposed which is a modified concept for visualizing vortical flow in 3D space-time.

Space-Time Line Integral Convolution

In the presented approach, 3D LIC [30] is employed to the space-time vector field $\mathbf{u}'$. Thus, the output texture is a 3D scalar field containing the smeared LIC noise, denoted here as space-time LIC. This space-time LIC is mapped to surfaces to visualize the space-time dynamics of the vector field at these surfaces, similar to Bachthaler et al. [11]. In cases where a surface region is aligned with $\mathbf{u}'$, i.e., locally a streamsurface of $\mathbf{u}'$, the resulting pattern on the surface exhibits the shape of the streamlines used for LIC computation. In cases where the surface region is perpendicular to $\mathbf{u}'$, the resulting pattern exhibits rather a dotted structure, since the surface is a cross section of LIC.

Space-Time Vortex Core Lines

As described in Section 2.2.2, there are several vortex extraction criteria that operate in 3D steady flow. It has been noted there that, although a vortex core line has to be tangent to streamlines, it is a common circumstance that vortex core lines do not represent single streamlines but rather consecutive short streamline segments [154]. Since $\mathbf{u}'$ is a 3D steady representation of 2D time-dependent flow, the streamlines in $\mathbf{u}'$ can be utilized for the analysis and definition of vortices in 2D unsteady flow, whereby the vortex core lines in $\mathbf{u}'$ are a space-time representation of the vortex centers of the 2D flow over time.

In this work, vortex core line criteria according to Levy et al. [38] and Sahner et al. [155] are applied to $\mathbf{u}'$ to obtain time-dependent definitions for vortex centers in unsteady 2D flow fields. The results are compared to those from the Sujudi-Haimes criterion in space-time [187] in Section 5.2.3. There, it turns out that while the Sujudi-Haimes approach tends to be more accurate, it strongly suffers from disrupted core lines in our applications, even for very large angle criteria. In contrast, Levy et al.'s and Sahner et al.'s criteria in $\mathbf{u}'$ provide core lines that are more coherent, however, at the cost that they are less accurate than those from the Sujudi-Haimes criterion, but still substantially more accurate than vortex center extraction by means of critical points (which is not Galilean-invariant).

The parallel vectors operator [134] is commonly used for the definition and extraction of line-type features: it identifies points where two vector fields are parallel or antiparallel. In this framework, the angle between the feature tangent, here the tangent to the vortex core line, and the two parallel vectors serves as a quality measure, which is denoted here as angle criterion. The smaller this angle, the more distinguished the vortex core line is. Nevertheless, in typical cases, one needs to allow quite large, e.g., 45 degrees, angles to avoid disrupted core lines, with the result that there will be parts with flux through the vortex core line. The presented vortex core line extraction employs the parallel vectors algorithm described by Peikert and Roth [134], where each quad face of a grid cell is split in two triangles to detect the intersection points of the core line with the cell faces. To suppress spurious solutions, those parts of the core lines violating the angle criterion or leading to too short core lines are rejected.

Vortex Core Streamsurfaces

One of the advantages of vortex core lines is that they provide a concise picture of flows with respect to vortical motion, i.e., they clearly indicate the location and longitudinal extent of vortices, and avoid visual clutter and occlusion. However, this advantage at the same time involves the drawback that they do neither convey the transversal extent of vortices nor the strength or direction of rotation. Using color coding to map scalar vortex criteria on the core lines can give a notion of the strength of vortices, but it cannot provide insight into their dynamics.

A common approach to address these issues is to augment the vortex core lines by selected streamlines or pathlines, either seeded interactively in the vicinity of the core lines, or by automatic seeding, e.g., on circles centered and oriented perpendicular to the core lines [154]. However, this approach likely leads to visual clutter due to overlapping lines. Hence, alternative approaches based on streamsurfaces of $\mathbf{u}'$ are investigated.

Since, depending on the quality of their extraction, vortex core lines in practice do not represent streamlines, seeding a single streamsurface at the upstream end of each vortex core line and integrating it until it reaches the length of the core line would in general not be useful—the streamsurface would likely deviate from the core line. Hence, one approach is to use regular time intervals and seed a streamsurface at the beginning of each interval and stop it at its end. This approach was investigated for straight or circular seeding curves, both aligned in the xy-plane of $\mathbf{u}'$, as shown in Figure 5.21. To support the perception of rotational dynamics and stretching of the surfaces, "candy stripe" texture was added to the streamsurfaces. It can be seen that this approach visualizes both the flow in the vicinity of the core lines as well as the transport away from the core lines. While the straight seeding strategy emphasizes the overall transport, the circular seeding gives better insight with respect to rotation. Note that these approaches are, with minor modification (e.g., replacing streamsurfaces with pathsurfaces), also applicable to true 3D flow fields.

There remains, however, a major issue with this approach, i.e., the difficulty to find an appropriate restart interval. Too frequent restarts lead to visual clutter, whereas too few restarts produce streamsurfaces that deviate too far from the core line and hence do not provide visualization of the vortical flow. These difficulties motivated the development of vortex core ribbons (Section 5.2.2), which are inspired by setting the restart interval to an infinitesimal value.

Vortex Core Ribbons

An approach inspired by streamribbons is derived in this section. Streamribbons are originally constructed along streamlines and show both the shape of the streamline together with twist. Since vortex core lines are close to streamlines in 3D flow fields (assuming small angle criteria, as discussed above), or to streamlines in $\mathbf{u}'$, ribbons along vortex core lines can be constructed to visualize swirling flow. This approach can be applied to any 3D flow field by constructing streamribbons along core lines. Since rotation takes place in $\mathbf{u}'$ only in the xy-plane, a modified ribbon construction scheme is introduced, as illustrated in Figure 5.15.

Instead of initializing a random direction vector and propagating it along the streamline (or core line), as would be done for traditional streamribbons, two streamlines in $\mathbf{u}'$ are simultaneously seeded at the 'earlier' end of the space-time core line. Both streamlines are integrated for a small step using the fourth-order Runge-Kutta scheme in $\mathbf{u}'$ to obtain the new position of the two particles. Since after each integration step the parti-

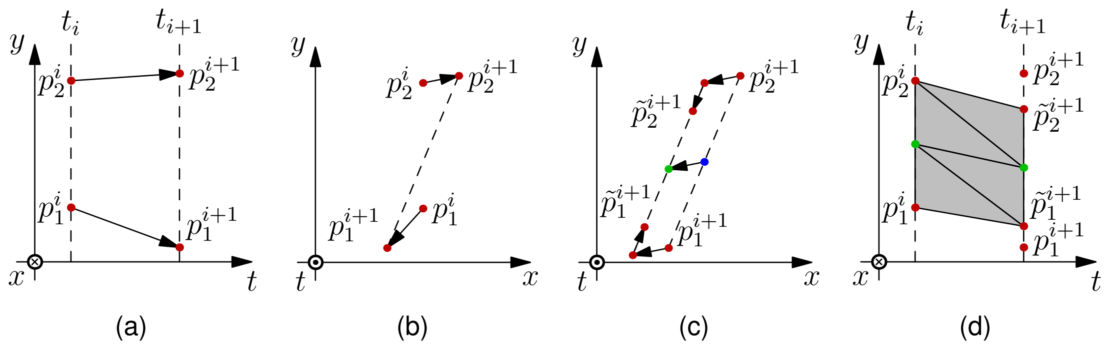


Figure 5.15: Construction of space-time vortex core ribbons. (a) Two particles p_1^i and p_2^i (red dots) are seeded at time t_i and integrated forward to t_{i+1}. In (b) the same as in (a) is shown, with the time axis pointing out of the page. (c) At $t = t_{i+1}$, the particles are translated so that their center of gravity (blue) lies on the vortex core line (green) and the distance between them is equal to the ribbon width (points $\tilde{p}_\cdot^{i+1}$). (d) Between the points $p_\cdot^i$ and $\tilde{p}_\cdot^{i+1}$, a triangle mesh is generated.

cles can deviate from the core line, an xy-translation is needed such that their midpoint is located on the vortex core line. Subsequently, the distance between each of the two points and the closest point on the core line is adjusted to half of the prescribed ribbon width. Finally, using the resulting two points, a triangulated surface patch is constructed that connects to the previous front of the ribbon.

There are two main differences between the space-time approach and the original streamribbon construction [180], irrespective if carried out along streamlines, as in the original definition, or along vortex core lines. First, in the traditional approach, the front of the ribbon is kept perpendicular to the streamline (or core line), while in the space-time approach, the front is kept aligned in the xy-plane, because all streamlines advance at the same pace in time (note the 1-component in Equation 5.19). Hence, the width of the space-time vortex core ribbon is not constant in space-time. Instead, its intersection with planes at $t = \text{const.}$ has constant length. This has the benefit that the tangents of the ribbon cannot point reverse in time. Second, while the twist of traditional ribbons would express differential rotation at the central streamline (or vortex core line), the twist of the space-time vortex ribbons expresses the combined twist of the two streamlines, locally positioned at both sides of the ribbon. This is less misleading, in particular for wide ribbons, because ribbon-based visualization implies that the shape of the ribbon not only shows flow at its medial axis but along its overall extent.

5.2.3 Results

The utility of the space-time visualization concepts is demonstrated using a 2D time-dependent CFD simulation of a buoyant flow.

2D Buoyant Flow The dataset represents a 2D CFD simulation of air flow in a square container with the bottom wall heated at 75 °C and the upper cooled to 5 °C. This drives

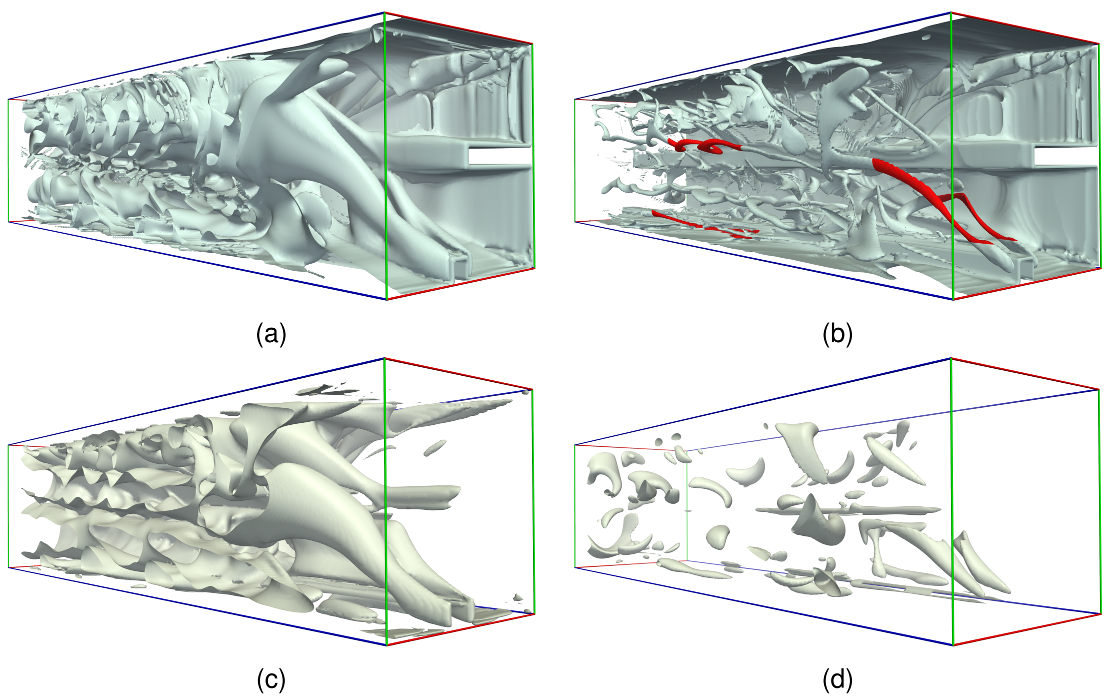


Figure 5.16: *2D Buoyant Flow* dataset in space-time representation (x: red, y: green, t: blue), with time increasing to the left. (a) The isosurface of normalized helicity of $\mathbf{u}'$ at isolevel 0.6 provides rather cluttered visualization. (b) Same as (a), but isolevel 0.95 nicely reveals individual vortices over time. Regions marked in red are visualized in Figures 5.19(a) and 5.19(b). (c) Isosurface of λ_2 of $\mathbf{u}'$ at isolevel -0.0001 results in clutter, similar to (a). (d) Same as (c), but, compared to (b), isolevel -70.0 fails to reveal dynamics of the vortices.

a buoyant convection flow. The side walls exhibit adiabatic boundary conditions and all walls are no-slip boundaries. There are two obstacles to induce the development of unsteady flow. The data is given on a structured grid with an overall resolution of 101×101 nodes and 401 time steps.

In Figure 5.16, visualizations by means of isosurfaces of vortex criteria are demonstrated. Figures 5.16(a) and (b) show isosurfaces of normalized helicity of $\mathbf{u}'$, at isolevels 0.6 and 0.95, respectively. The isosurface was clipped at the front left (x-max) face of the domain to provide view to the inside. In Figure 5.16(b), one can see how two vortices originate at the bottom square obstacle and start to rise (two red parts on the right-hand side of the figure). Normalized helicity provides results superior to λ_2 (Figures 5.16(c) and (d)) in this dataset.

Next, different vortex core line definitions are investigated in Figure 5.17. Critical points, Sujudi-Haimes core lines, and Levy core lines provide comparable results, whereas valley lines of λ_2 somewhat deviate. It is apparent that the Sujudi-Haimes core lines are severely disrupted and it is difficult to discern the vortical structure. In contrast, the space-time Levy criterion clearly reveals the vortex structure in the unsteady flow.

Projections along the time axis of vortex core lines according to Levy and Sujudi-Haimes' criterion are shown in Figure 5.18(a) and 5.18(b). The arrows show the velocity

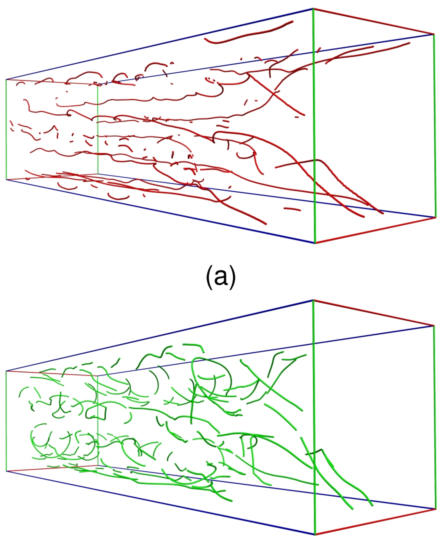
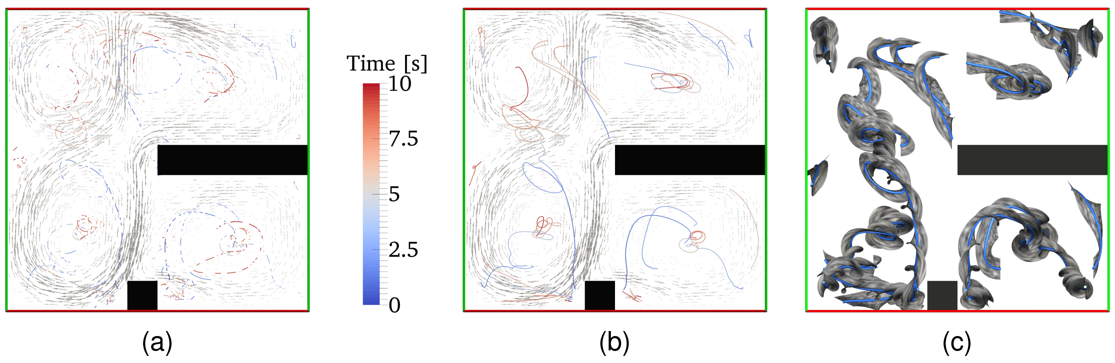

(a) (b) (c) (d)

Figure 5.17: Space-time vortex core lines in the *2D Buoyant Flow* dataset (time increases from right to left): (a) curves of critical points of type focus and center in $\mathbf{u}$, (b) Sujudi-Haimes vortex core lines in $\mathbf{u}'$, (c) λ_2 valley lines in $\mathbf{u}'$, and (d) Levy vortex core lines in $\mathbf{u}'$. Sujudi-Haimes vortex core lines are severely disrupted.



(a) (b) (c)

Figure 5.18: Projection of (a) Sujudi-Haimes and (b) Levy space-time vortex core lines along the time axis (color legend for time). Black obstacles and gray velocity arrows at time $t = 10.0\,\mathrm{s}$ are displayed for context. Core lines according to Levy's criterion are longer and thus easier to analyze. (c) Time projection of Levy vortex core ribbons.

field at the last simulation time step and reveal highly vortical flow. The vortex core lines in both figures give insight into the evolution of vortex centers over time. Some of the core lines remain in one region (e.g., in the center of the lower right quarter), others move across the domain (e.g., the core line starting to the left of the lower obstacle). Although Sujudi-Haimes vortex core lines (Figure 5.18(a)) are more accurate in this dataset, Levy core lines (Figure 5.18(b)) are easier to follow in the 2D projection, since they are less fragmented.

Figure 5.19 provides a more detailed comparison of the vortex core lines obtained

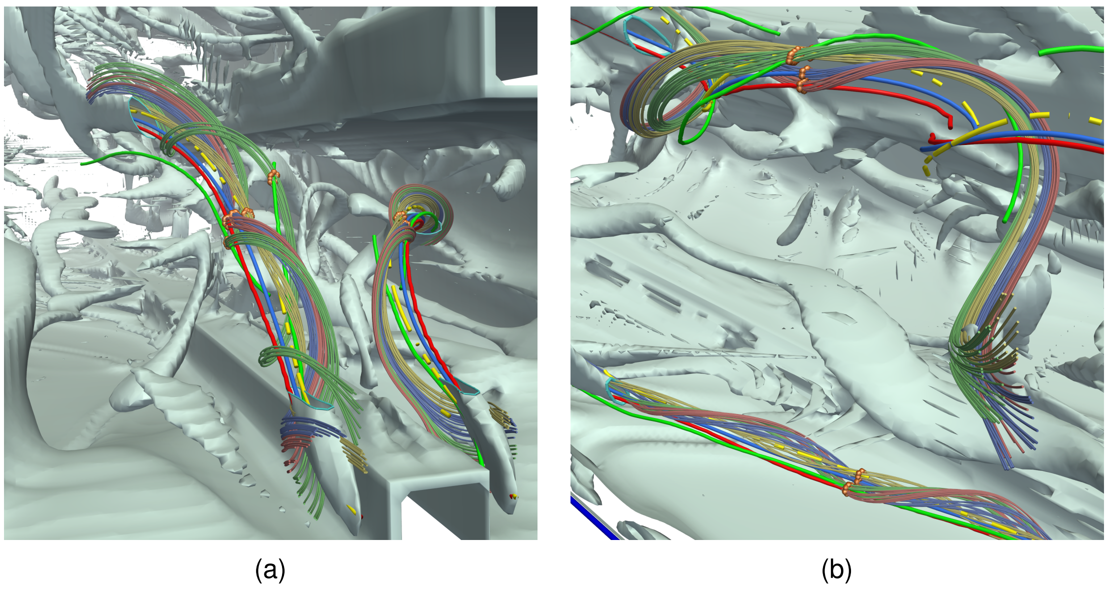


Figure 5.19: Selected vortex regions in the *2D Buoyant Flow* dataset. Vortex core lines shown with high-saturated thick tubes (colors from Figure 5.17), and pathlines with low-saturated thin tubes, seeded at the orange spheres at the respective core line. The isosurface of normalized helicity is displayed for context. Visualization of the regions depicted in red in Figure 5.16(b): (a) the front right region, and (b) the region back left.

by the different definitions. In Figure 5.19(a), pathlines seeded at Sujudi-Haimes core lines (yellow) exhibit least swirl, i.e., they follow the yellow core line tightly, whereas pathlines started at Levy core lines (blue) exhibit more swirl. Pathlines started at critical points (red) exhibit even more swirl, whereas pathlines seeded at λ_2 valley cores (green) exhibit the largest swirl radius and are therefore most inaccurate. Hence, Sujudi-Haimes core lines are most accurate in this dataset, however, with the drawback that they are strongly disrupted. This affects perception and can hinder derived visualization, e.g., the construction of our vortex core ribbons (Figure 5.23(b)). In Figure 5.19(b), one can see that pathlines seeded at the Sujudi-Haimes core line at the upper curved vortex are again closest to the vortex center. However, all definitions provide similar results at the vortex at the bottom of Figure 5.19(b), because this vortex is straight in $\mathbf{u}'$ and oriented along the t-axis, i.e., the vortex does not move within the examined time interval. All in all, best results were obtained with the space-time Levy criterion, because it provides a good tradeoff between accuracy and readability—although it deviates from the true vortex core more than the Sujudi-Haimes criterion in the experiments, it provides more continuous structures and it is more accurate than vortex visualization by means of critical points or valley lines of λ_2 in $\mathbf{u}'$.

Figure 5.20 demonstrates the use of space-time LIC on streamsurfaces of $\mathbf{u}'$, i.e., on material lines in space-time representation. Due to LIC, one can easily interpret the dynamics within the material line. The ridges in a slice of the FTLE field [64] exhibit high correlation with the material line because the material line is attracted by the ridges,

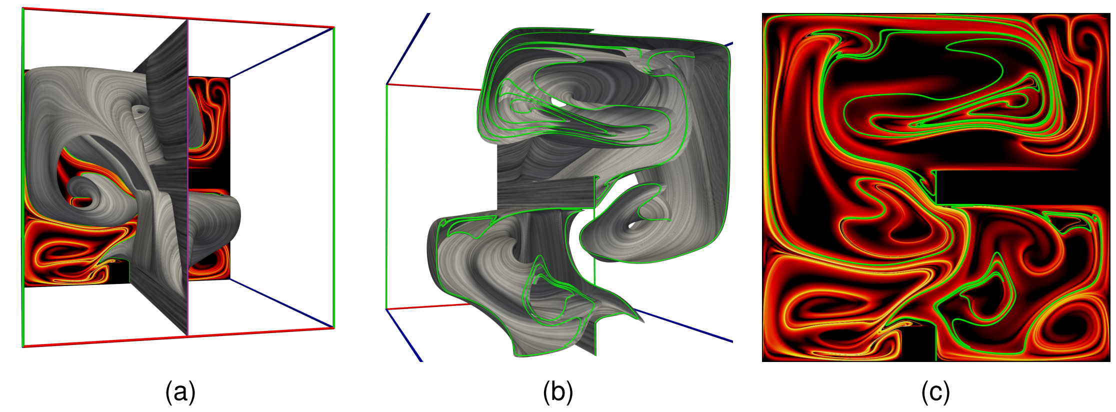


Figure 5.20: Space-time material line in the *2D Buoyant Flow* dataset, seeded at purple curve, with space-time LIC visualizing stretching and folding of material line (a). Time increasing along the blue axis to the back. Material line stretching is apparent, e.g., from the LIC at the top left region. (b) Green intersection curve represents a material line. (c) The FTLE field located at the respective instant in time exhibits high correlation with this material line.

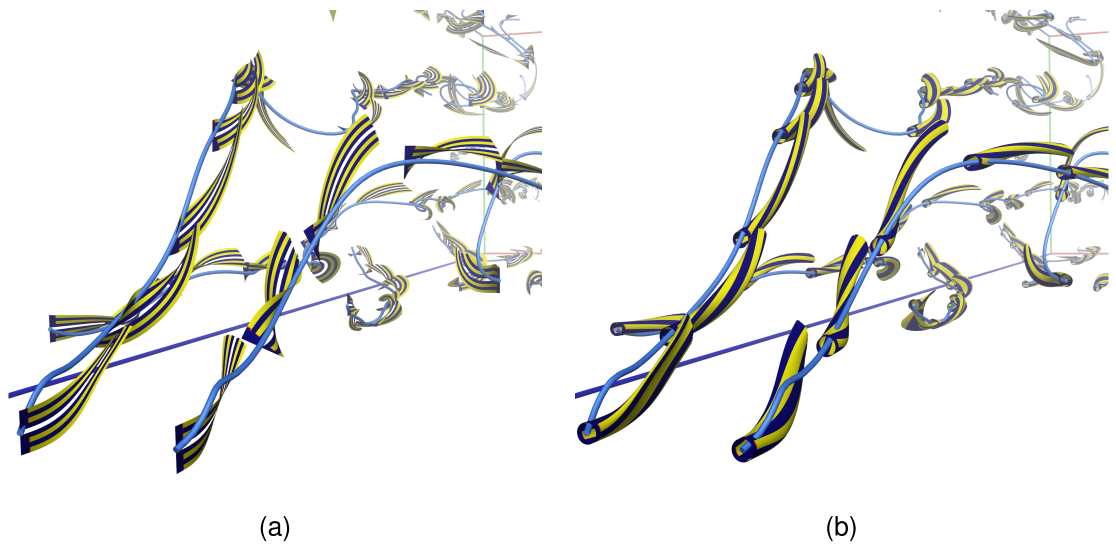


Figure 5.21: Vortex core surfaces in $\mathbf{u}'$ with (a) line seeding and (b) circular seeding along each Levy core line and reset every 0.375 s (the time domain of the simulation is 10 s). Stripes on the surfaces additionally reveal rotation and stretching.

which represent attracting Lagrangian coherent structures (LCS). The FTLE field is computed from trajectories started at the slice in space-time and integrated in reverse time until the start of the dataset. Since in the presented experiment, a material line was seeded at the arbitrarily-chosen purple curve instead, there are some deviations between the FTLE ridges and the material line. The space-time material lines give insight into topology-related flow behavior, without costly computations as would be required to obtain the FTLE field.

In Figure 5.21, vortex core streamsurfaces are visualized. For each core line, a

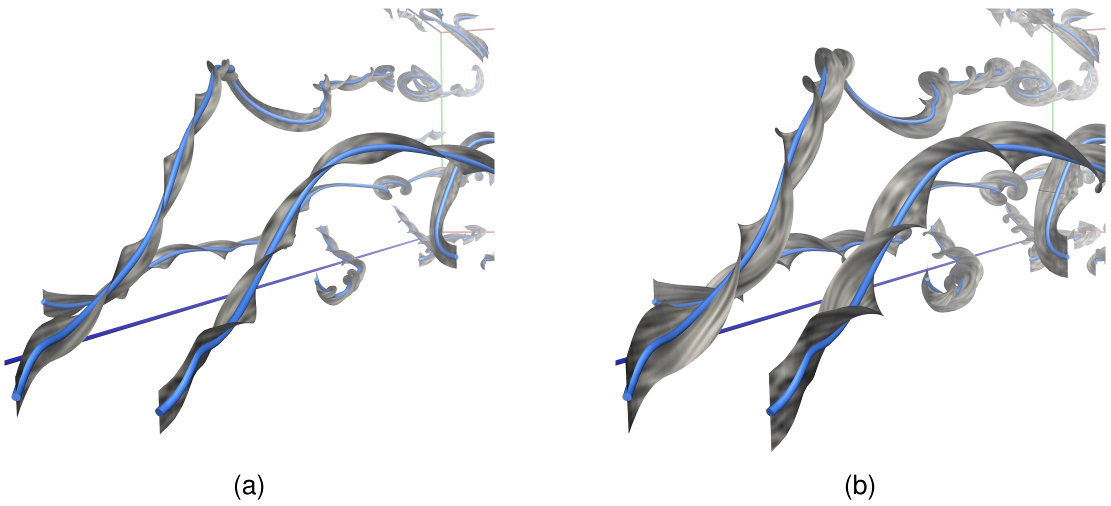


Figure 5.22: Space-time vortex core ribbons textured with space-time LIC in the *2D Buoyant Flow* dataset, with focus on the region from Figure 5.19(a). Time increases to the right along the blue axis. (a) Narrow vortex core ribbons along Levy core lines in $\mathbf{u}'$ nicely visualize twist along core line, i.e., the rotational dynamics of the vortices. However, the bands are too narrow to visualize the LIC texture well. (b) Same as (a), but with wider ribbons to better show the space-time LIC texture.

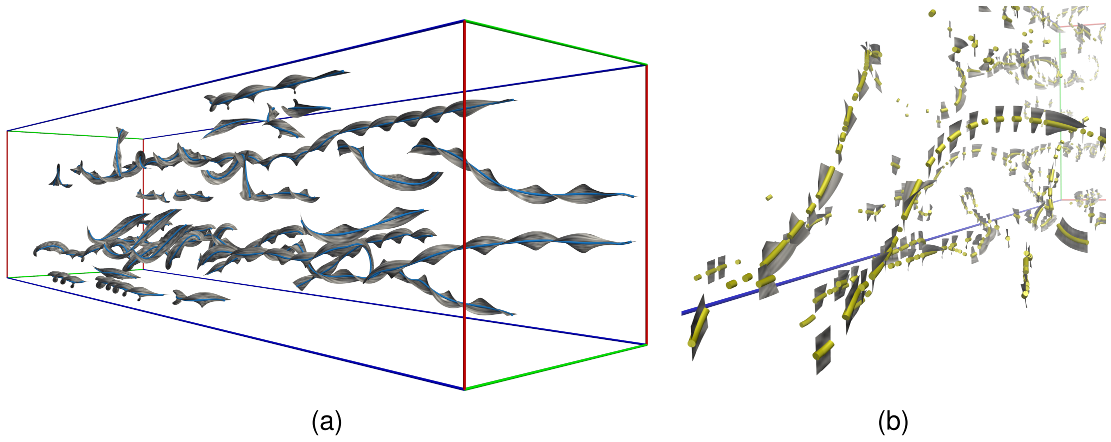


Figure 5.23: (a) Top view of vortex core ribbons from Figure 5.22(b). Time increases to the left along the blue axis. (b) Same as 5.22(a), but applied to Sujudi-Haimes core lines, suffers from their disrupted geometry, resulting in visual clutter.

vortex core streamsurface is generated at its starting point and reseeded every $0.375\,\mathrm{s}$. The advection is stopped at the time at which the core line ends. Line seeding and circular seeding are used in Figures 5.21(a) and 5.21(b), respectively. While the former better reveals the overall transport along and from the core line, the latter emphasizes rotation.

Finally, Figures 5.22 and 5.23 demonstrate the utility of the vortex core ribbons. While narrow ribbons reduce occlusion and better visualize twist along the core lines, they are often too narrow to appropriately visualize the LIC texture. Increased ribbon width shows the LIC texture well (Figure 5.22(b)). One can identify regions where the LIC patterns cross the core line or where the patterns are more point-like than line-like. These are locations where the flow is not tangent to the core line, i.e., where its quality is rather poor with respect to the angle criterion. Even the purely spatial (2D) time projection of vortex core ribbons (Figure 5.18(c)) reveals the vortical flow around the core lines. The swirling motion is clearly visible for the core lines starting near the lower obstacle, as well as for the core lines in the lower right and upper left corners. Finally, although the Sujudi-Haimes criterion provides more accurate results, the frequent disruption of core lines leads to disintegrated and inferior vortex core ribbons that are very hard to interpret (Figure 5.23(b)). This motivates the use of our proposed space-time Levy core line criterion also for vortex ribbons in this dataset.

5.3 Implicit Vector Field Topology

Vector field topology is concerned with the identification and extraction of regions of qualitatively different streamline behavior within the domain of a steady vector field. The behavior is determined with respect to spatial convergence as time reaches $\pm\infty$. The structures that the streamlines converge to include critical points (isolated zeros of the vector field) and periodic orbits (isolated closed streamlines) which can be classified in terms of their attraction/repulsion behavior. Critical points exhibiting both attracting and repelling behavior are classified as *hyperbolic* or *saddle-type*. Please see Chapter 2 for a detailed description of different types of topological features in 2D vector fields.

In traditional vector field topology, the topological flow structure, i.e., *topological skeleton*, is visualized explicitly by means of graphical representation [68] of the critical points, periodic orbits, and their manifolds, i.e., so-called *separatrices*, consisting of streamlines that converge to saddle-type critical points in forward or reverse time direction. Although this approach is very successful, it involves several difficulties. First, it depicts the boundaries of the different regions by means of separatrices, but does not depict the regions themselves, hence complicating interpretation. Second, it may require the extraction of further topological structures together with the respective separatrices that converge to those structures in forward or reverse time direction. Extraction of each of these structures must be handled individually and the respective implementation is usually non-trivial. Beyond that, the approach may require explicit treatment of more intricate structures such as strange attractors.

These challenges motivated the development of implicit visualization of vector topology where, instead of extracting the boundaries of the regions, the focus is on the visualization of the regions directly. This is accomplished by extracting all critical points of a vector field, densely seeding streamlines, and assigning to each seed the ID of the critical point that the streamline converges to. Since it would take infinite time for a streamline to reach a critical point, a sphere with radius ε is defined for each critical point and it is tested if the streamline enters such a sphere. To reduce the costly streamline integration, a refinement scheme is presented that increases the sampling at the region boundaries, i.e., at the implicitly visualized separatrices. The approach is demonstrated on 2D vector fields, it can be, however, readily extended to 3D. [3]

5.3.1 Related Work

The concept of vector field topology was introduced in the context of visualization by Helman and Hesselink [68] who provided a space-time representation of vector field topology. For visualization of 3D vector field topology, Weinkauf et al. [188] extended the concept of saddle connectors to boundary switch connectors that represent separation surfaces emanating from boundary switch curves. Closed streamlines were investigated by Wischgoll and Scheuermann [202] who analyzed grid cells reentered by

[3] Parts of this section have been published in: [91]

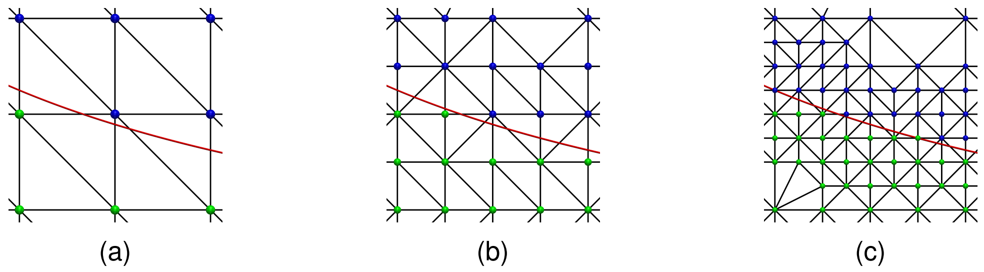


Figure 5.24: Grid refinement. The red curve represents the separatrix to be captured, with streamlines seeded on both of its sides reaching different critical points. (a) After initial streamline integration from the grid nodes, the ID of the respective critical point is stored at the node (blue/green). (b) If two nodes with different labels share an edge, all edges sharing those nodes are subdivided by inserting a new node at the edge midpoint. These new nodes are incorporated into the grid using Delaunay triangulation. (c) The process is repeated until the minimum edge length threshold is reached.

streamlines to detect periodic orbits. Theisel et al. [175] extracted the closed streamlines in 2D vector fields by detecting intersection curves of streamsurfaces advected forward and backward in extruded 3D space-time vector field representation. Visualization of another topological feature was proposed by Machado et al. [112] who locally extracted bifurcation lines and their manifolds using a modified method for vortex core line extraction. Friederici et al. [52] presented an approach for segmentation of 2D vector fields using finite time separation characteristics along separatrices. Their method avoided costly computation of flow maps.

5.3.2 Visualization

The input to the visualization technique is the vector field sampled on a 2D uniform grid with cell size δ, a refinement threshold σ, the positions of its critical points, and the radius ε of the spheres around the critical points. The radius $\varepsilon > \delta/2$ is chosen to make sure that there is at least one node of the sampling grid within each sphere. The algorithm consists of two phases.

In the initial phase, each critical point is given a unique non-negative ID, and a streamline is seeded at each node of the sampling grid and integrated over the predetermined time with the fourth-order Runge-Kutta scheme. After integration, each streamline p consists of N_p vertices such that $\mathbf{x}_{p,0}$ is the seed and $\mathbf{x}_{p,N_p-1}$ is the last vertex of the streamline. Subsequently, the first vertex along the streamline is found that is contained in one of the spheres centered at the critical points, and the respective ID of that critical point is stored at the seed node of the sampling grid. The first encountered critical point is determined by subsequently computing the distance $s = \|\mathbf{x}_{p,i} - \mathbf{x}_{c_j}\|$ for points $\mathbf{x}_i$, $i = 0, \ldots, N_p - 1$ on the streamline p to the critical points c_j until $s < \varepsilon$ for any c_j. If no vertex could be found in any of the spheres, the ID at the respective node is set to -1. After this phase, IDs are assigned to all grid points.

In the second phase, the sampling grid is iteratively refined to capture the details of

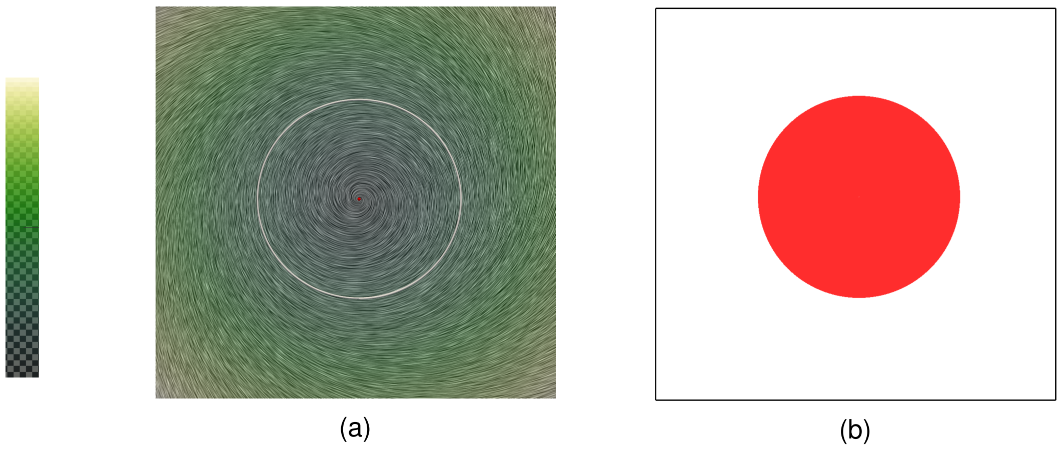


Figure 5.25: *Periodic Orbit* dataset. (a) LIC [30] representation of the vector field with attracting-focus critical point at its center (red dot), and color legend for the vector field magnitude on the left. (b) Implicit visualization reveals topological region (red) where streamlines are attracted toward the critical point. The boundary of this region implicitly depicts the periodic orbit.

the topology of the vector field. First, Delaunay triangulation of the sampling nodes is employed to obtain a triangular mesh. From the mesh, all edges are extracted, and for each edge with edge length $\geq \sigma$, IDs at both ends are compared. If the IDs are different, sample nodes are added at midpoints of all edges sharing either of these grid points (see Figure 5.24). Afterwards, streamlines are seeded at the new sample positions and their IDs are determined as described above. This process is repeated, i.e., the Delaunay triangulation step is performed for the whole grid, including the sample nodes generated in the last iteration. Since the edges with different IDs are halved in each step, this procedure converges to a sampling of the vector field topology at resolution σ.

The whole algorithm is performed forward and backward in time in order to capture the regions of different behavior, e.g., with respect to convergence to sinks and sources. The visualization is implemented as a ParaView plugin [10], where the vector field and critical points are given as input.

5.3.3 Results

Periodic Orbit The first data set is a synthetic vector field with a repelling periodic orbit around a critical point of type attracting focus (see Figure 5.25(a)). The presented technique extracts in forward time the topological region connected to the critical point (Figure 5.25(b), i.e., the region within the periodic orbit. The disk boundary implicitly reveals the periodic orbit, since the streamlines started outside the periodic orbit reach the domain boundary, resulting in an assignment of invalid ID (-1) at the seed points.

2D Buoyant Flow and Rotated Flow The *2D Buoyant Flow* dataset is a single time step from a computational fluid dynamics simulation of buoyant flow in a closed container with resolution of 102×102 nodes (see Figure 5.26(a)). This dataset exhibits criti-

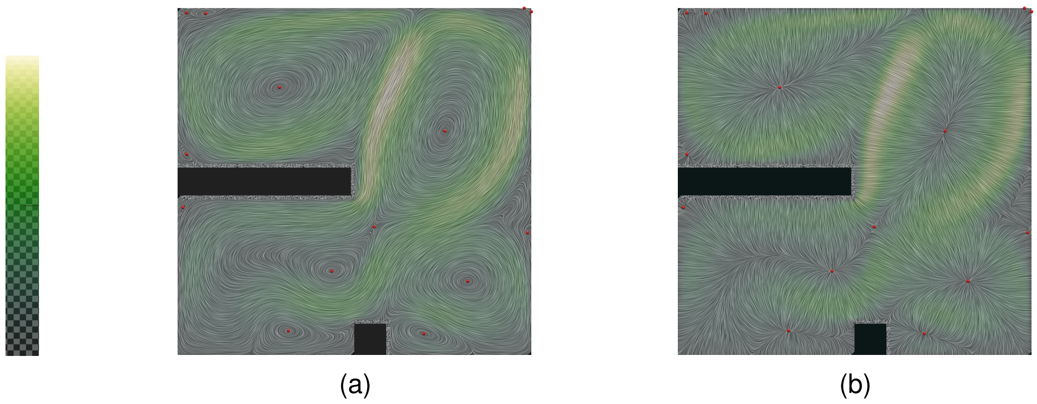


Figure 5.26: (a) LIC representation of the 2D *2D Buoyant Flow* dataset, with color indicating velocity magnitude (color legend on the left). Critical points by red dots. (b) *Rotated Flow* dataset.

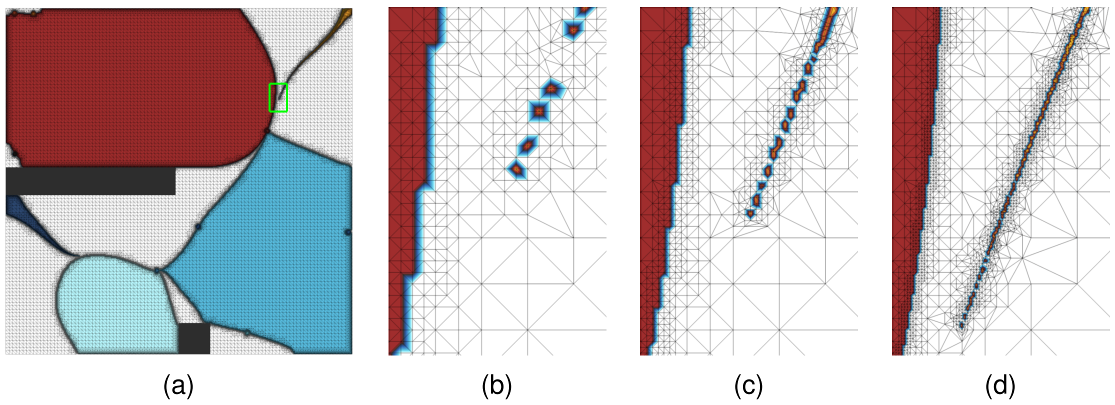


Figure 5.27: *2D Buoyant Flow* dataset (forward integration, as in Figure 5.28(a)), with region marked with green box in (a) enlarged for (b) $\sigma = 2.4 \times 10^{-4}$, (c) $\sigma = 1.2 \times 10^{-4}$, and (d) $\sigma = 0.6 \times 10^{-4}$. Smaller values of σ provide improved accuracy but cannot reveal additional regions.

cal points of type focus (very close to the center type, i.e., very small outflow/inflow) and saddle. The *Rotated Flow* dataset was obtained by rotating the vectors of the *2D Buoyant Flow* dataset by 90 degrees counterclockwise (Figure 5.26(b)). This way, sources and sinks were obtained from centers with clockwise and counterclockwise rotation, respectively, while saddles have retained their type.

For both datasets, the threshold σ for the minimum edge length was set to 1.2×10^{-4}, which corresponds to $0.12d$, where d is the diagonal of a cell from the dataset. For the *Rotated Flow* dataset, the refinement resulting from different values of σ is shown in Figure 5.27, where already with the chosen σ (Figure 5.27(c)) the details are well captured.

The critical point radius ε was set to $\varepsilon = 7.071 \times 10^{-4}$ for the *Rotated Flow* dataset. As will be shown below, this value was insufficient to capture the whole topological structure in the *2D Buoyant Flow* dataset, in which case $\varepsilon = 14.142 \times 10^{-4}$ was used.

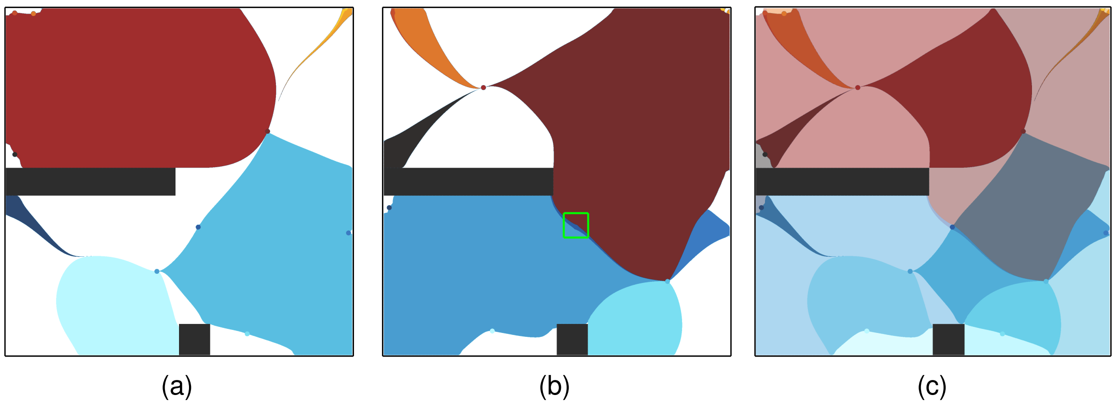


Figure 5.28: *Rotated Flow* dataset. Implicit visualization with (a) forward and (b) backward integration. (c) Overlay of (a) and (b) reveals the topological structure of the vector field.

In the *Rotated Flow* dataset, Figure 5.28(a) and (b) shows topological regions for forward and backward streamline integration. The critical points, representing sources and sinks, are the colored dots which have been captured by the refinement. The color of the regions indicates the respective critical point ID. The overlay of the results from both integration directions (see Figure 5.28(c)) reveals the topological structure of the vector field, with region boundaries representing separatrices. Interestingly, a partially developed region corresponding to the saddle point in the green box in Figure 5.28(b) can bee seen. The backward integration captures the diagonal region (top left to bottom right) corresponding to a separatrix, since streamlines do not diverge strongly from the separatrix, and hence more remote streamlines reach this critical point. In case of forward integration, however, the streamlines diverge faster from the separatrix (bottom left to top right) and hence the separatrix is not captured at the given maximum resolution σ. Interestingly, in the *2D Buoyant Flow* dataset, the regions corresponding to the separatrices emerging from that critical point are captured as fully developed closed curves (see green box in Figure 5.29(c)). It is worth noting that, as can be seen in Figure 5.29(a) and (b), the smaller critical point radius $\varepsilon = 7.071 \times 10^{-4}$ leads to disconnected separatrices corresponding to the saddle point (green box), since, due to relatively high streamline divergence around the separatrices, few of them reach the critical point. Only after doubling the radius to $\varepsilon = 14.142 \times 10^{-4}$, do the streamlines reach the saddle point, therefore closing the separatrix.

Performance Analysis

To analyze the characteristics of the refinement procedure, the number of seed points generated per iteration has been recorded for the *2D Buoyant Flow* and *Rotated Flow* datasets. The results are shown in Figure 5.30, where the 0th iteration corresponds to initial grid node positions. For both datasets and in both integration directions, the first iterations are very similar—the first refinement produces less points than in the original grid, but in the next few iterations, the number of inserted seed points increases consid-

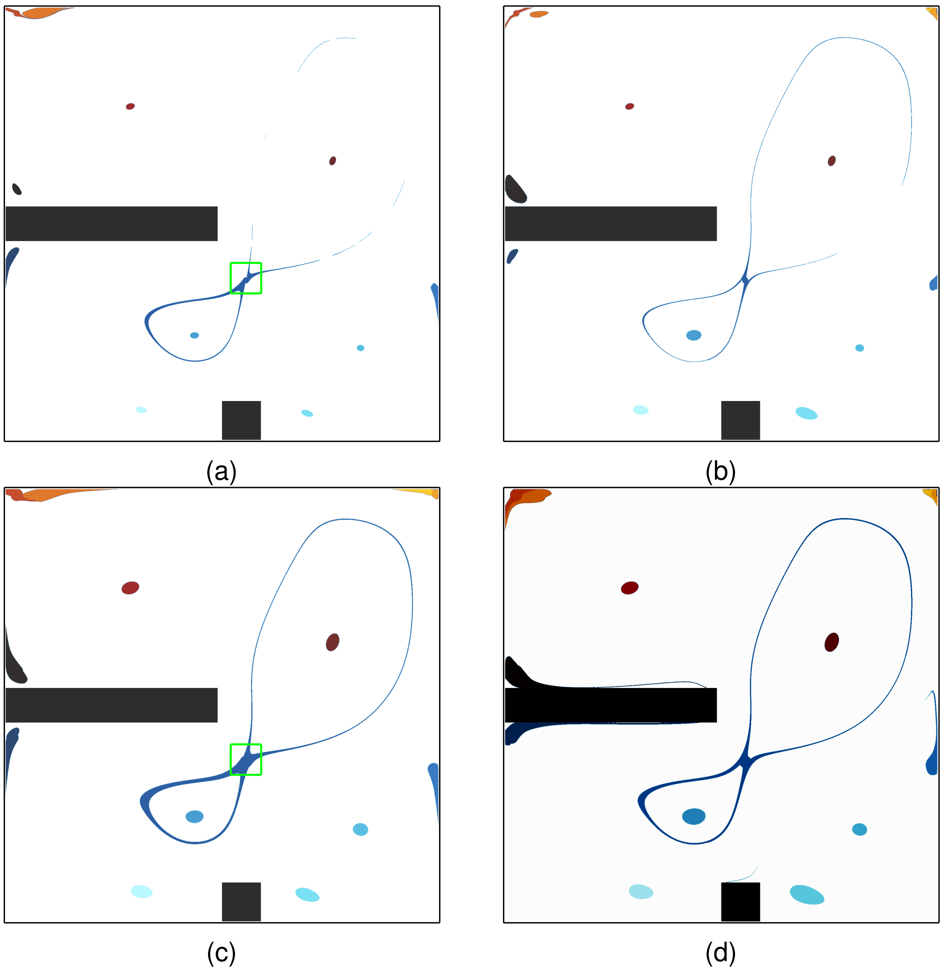
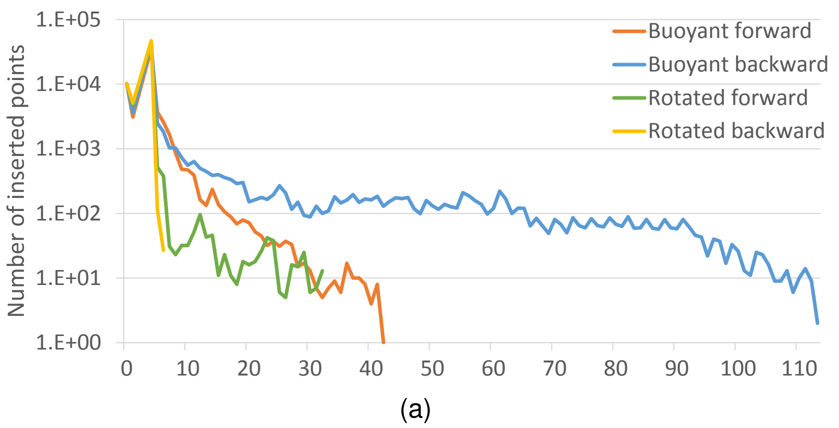


Figure 5.29: *2D Buoyant Flow* dataset. With tolerance $\varepsilon = 7.071 \times 10^{-4}$, the blue thin lines representing separatrices are disconnected for (a) forward and (b) backward integration due to limited sampling. With tolerance $\varepsilon = 14.142 \times 10^{-4}$, the lines are closed for both integration directions ((c) and (d)), and therefore clearly show the topological region constrained by the separatices.

erably. This is followed by steady overall decrease in the number of new seed points for all configurations. The interesting refinement characteristics in the first iterations can be explained by the fact that the initial node positions have no correspondence to the topological region boundaries, which leads to the initial low-quality refinement. After the first refinement, however, the method starts to capture the fine details as the new seed points are inserted at positions that now correspond to the potential region boundaries.

Regarding the performance characteristics, shown in Figure 5.30(b), streamline integration time is proportional to the number of new seed points inserted and hence it uses most computation in the first iterations. For the Delaunay triangulation (dashed line), the computation time is dependent on the total number of points, and therefore it increases considerably in the first iterations, due to the large number of new seed points,



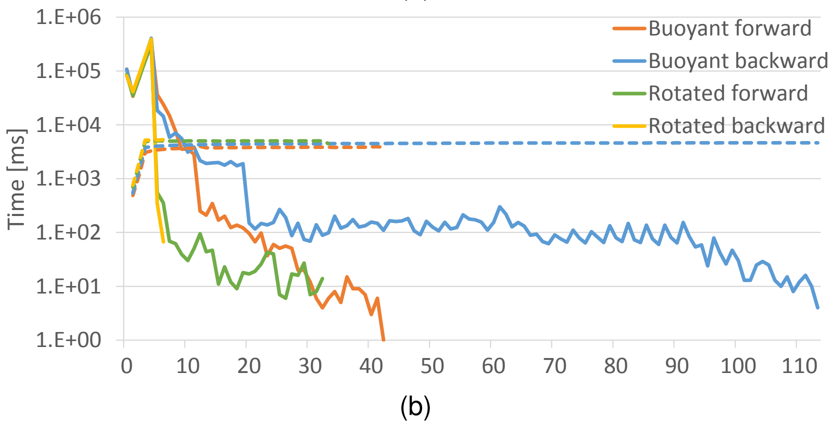


Figure 5.30: (a) Number of inserted seed points after each iteration and (b) computation times per iteration for streamline integration (continuous line) and Delaunay triangulation (dashed line). The low number of inserted seed points in the first iteration is due to uniform node positions that do not correspond to the topological region boundaries.

but remains roughly constant for the following refinement steps, where the number of new seed points decreases with each iteration.

CONCLUSION | 6

In this thesis, visualization techniques have been proposed for the analysis of two-phase flow dynamics with the focus on droplet-related processes. The great complexity and large variety of the simulated physical phenomena necessitate a comprehensive visualization approach that ensures effective analysis of the involved droplet interactions, dynamics of interfaces, and material transport in the related single-phase flow. These aspects reflect three scales that have been identified as crucial for better understanding of two-phase flow dynamics. Accordingly, the proposed visualization methods provide a multi-scale approach designed to effectively support visual investigation of process at these three different scales. Moreover, this thesis addresses many challenges inherent to flow visualization, such as dealing with high dynamics of flow, especially in two-phase flow with interfaces, providing innovative solutions for specific requirements of investigated phenomena, as well as efficiently processing large data produced by CFD simulations. In the following, the presented techniques are summarized. That is, details are provided on how they accomplish the research goals outlined in the Introduction in Chapter 1 and how they handle the related challenges.

6.1 Summary

This thesis provides visualization techniques designed for the analysis of different, yet interdependent problems in two-phase flow dynamics. To handle the complexity of the two-phase flow phenomena, the proposed visualizations were designed to support the analysis of three important aspects: droplet interactions, interface dynamics, and material transport in single-phase flow configuration.

Interactions in Droplet Groups The vast amount of interacting inclusions and frequent topology changes resulting from splits and merges require visualization methods that can manage large amounts of data to extract the important flow characteristics. As has been demonstrated in Chapter 3, static representation of inclusion dynamics has proven to be a powerful instrument for this purpose. In the visualization of inclusion dynamics presented in Section 3.1, visualization of highly complex two-phase flow, in-

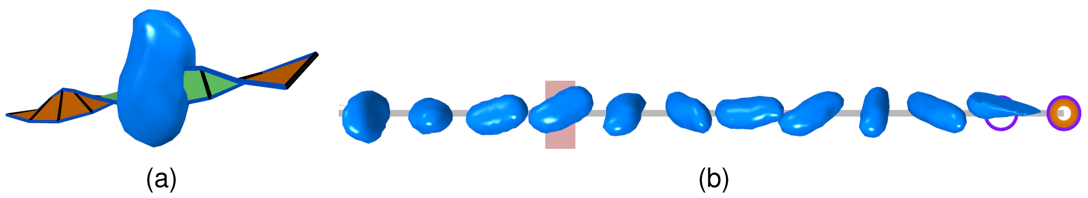


Figure 6.1: Visualization of droplet dynamics in the *Peripheral Collision* dataset. (a) This droplet detached from a larger ring-shaped component. Its initial oscillation transforms into rotational motion. (b) The graph representation reveals the overall droplet motion until it leaves the domain (purple rings).

cluding droplet collisions and scattering of jets, has been accomplished with a novel tightly-coupled 3D spatial and 2D space-time representations of simulation data (Figure 6.1). In the latter, an abstract graph visualization of droplet breakups and merges is employed that provides an overall static view of the whole simulation. Geometrical representation of segmentation in the visualization of inclusion separation (Section 3.2) provides a detailed view on the volumetric contributions of droplets. In the presented method, the extracted segmentation boundaries determine the volumes of inclusions that will eventually break up, while the separation surfaces reveal the information on the time point at which a given separation has occurred.

The two visualization techniques presented in Chapter 3 allow for a detailed and effective analysis of droplet interactions, a task particularly difficult due to the overwhelming complexity of the involved processes. This has been achieved by focusing on the different droplet deformation mechanisms, as well as on relevant physical characteristics whose visualization could give new insights into the investigated phenomena. With the visualization of inclusion dynamics, it is possible to better understand the transitions between oscillation and rotation, and how these two types of motion influence the droplet breakup, as has been shown in the demonstrated datasets. It has been achieved by introducing novel visualization techniques based on principal component analysis, frequency analysis, and derived physical measures, and integrating them into both 2D and 3D views. This approach proved to be a powerful solution, since in the former it allows to detect interesting droplets in the whole dataset, while in the latter it enables close examination of the dynamics of the investigated droplet. Furthermore, the second method reveals how the inclusions and the segments inside them evolve until the breakup. For instance, it revealed that the ligaments detaching from the jet core originate from elongated core sections that, due to folding enforced by gas resistance, later form more compact shapes. For direct representation of the inclusion separation, the particles stored during advection can be additionally used to show the temporal evolution of inclusion segments. To ensure that the particles do not stray from the original phase, a corrector scheme has been introduced which results in a clearer visual representation of the segmentation. The visualization method has been successfully applied in the investigation of jets [44] where some of the flow characteristics would be very difficult to discern without a proper visualization approach. The usability of this method has been further extended by tracking inclusions backward in time. In this case, the segmentation within an inclusion shows the contributions of merged inclusions. Such an approach could be particularly useful in the analysis of multi-component flow simulations where droplets with different chemical components collide. As has been demonstrated in this thesis, the two methods complement each other in the goal of providing a better understanding of droplet interactions: the first one analyses the reasons of particular inclusion breakups and merges, while the second one supports close investigation of these events by showing the contributions of inclusion volumes.

The analyzed phenomena at this scale often necessitate high spatial resolution to accurately resolve small droplets and their interactions. The resulting large datasets incur additional burden during visualization. To deal with large data, in the static graph visualization, edge clustering according to a predefined measure has been employed to reduce visual clutter. Additionally, the level-of-detail approach enables folding and unfolding of clustered edges, and therefore allows for exhaustive analysis of large datasets. In the visualization of inclusion separation, domain parallelization is used to permit the processing of large datasets. To reduce the storage requirements of simulation data, the datasets have typically substantially reduced temporal resolution. This poses a particular problem in this visualization method, since it requires reliable particle tracking in order to expose fine details of the separation. Hence, to account for low temporal resolution, a novel corrector scheme has been introduced that exploits the VOF-field to ensure phase-consistent particle trajectories.

Liquid Interface Dynamics In Chapter 4, the dynamics of fluid interfaces were scrutinized. Fluid interfaces have a great impact on the behavior of two-phase flow on larger scales due to the interplay of surface tension and fluid dynamic forces. They typically lead to intricate interface formations and potential breakups. On the other hand, in the presence of an electric field, static discharges at the droplet-insulator contact line can inflict damage on the insulators. Therefore, it is important to visually analyze the interfaces and the related processes to gain more insight into the overall two-phase flow. To help with the investigation of interface dynamics, several visualization approaches have been developed that address diverse, yet equally important aspects related to phase interfaces. This chapter also showed that in order to develop relevant and effective visualization techniques, close collaboration with simulation experts is vital.

In two-phase flow simulation, the PLIC interface reconstruction method strikes a good balance between the reconstruction accuracy and computational efficiency. However, because of this trade-off, it is reasonable to expose the intricacies of phase tracking in the simulation to visually assess the simulation quality. The presented visualization in Section 4.1 is based on a novel interpretation of PLIC reconstruction as the first-order Taylor approximation of the VOF field. With the Taylor approximation, it is possible to generalize the PLIC reconstruction to higher-order approximations. This allows for a derivation of error bounds on the implicit approximation of this field and, additionally, it provides several geometry-based error measures with respect to the shape of the re-

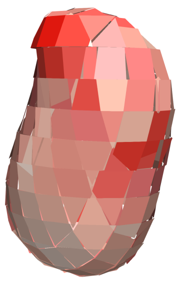


Figure 6.2: Same droplet as in Figure 6.1, analyzed in terms of interface reconstruction in the simulation. Here, discontinuities between interface patches are color-coded. Large discontinuities in the top of the droplet, revealed by intense red color, indicate possible loss of accuracy in the simulation advection step.

construction and the discontinuities at cell boundaries (Figure 6.2). To handle highly dynamic processes and to gain insight into advection processes, the visualization provides geometric representations of the volume fluxes used in the solver to track phases. As has been demonstrated on the example datasets, the volume visualization can help understand how the reconstruction can sometimes lead to the deterioration of the interface, or how it artificially enforces division of liquid inclusions.

A problem inherent to two-phase flow is that the interface instability and the potential breakups are difficult to follow visually due to the clutter introduced by the presence of interfaces. A suitable approach is therefore required to convey the interface deformation and at the same time visualize the interface itself in order to indicate unstable regions. The visualization of interface deformation in Section 4.2 focused on the deformation characteristics and how these deformations transform into inclusion breakups. The method employs a metric tensor based on the Jacobian of the local interface displacement that shows the directions of stretching and the corresponding magnitudes. It therefore provides useful insights into interface instabilities. Since the metric tensor does not capture the actual shape change, a novel technique for the analysis of the curvature change has been introduced. It utilizes the difference of shape tensors of the geometric interfaces to reveal the bending strength and direction and hence provides a direct and descriptive measure for interface deformation. The dynamics of interfaces is represented by glyphs whose orientation and color indicate deformation direction and strength, respectively. At the same time, the glyphs outline the phase interface, and therefore provided the context for the analysis of interface dynamics. This in turn facilitates the analysis of highly dynamic processes.

Simulations of water droplets in the presence of electric fields are carried out to help understand static discharges on wetted insulators. Consequently, the visualization presented in Section 4.3 has been developed to help in the analysis of such coupled fluid dynamic and electric field problems. The method transforms the circular contact line of each simulation time step into a stripe, and stacks them together to form a space-time representation of the time-dependent electric field. This novel approach avoids the occlusion of the time-dependent process and provides a static representation of the time-dependent process.

All techniques presented in this chapter utilize the solver proximity to various extents in order to provide useful information on the simulation behavior and to support the analysis of droplet-specific phenomena. The PLIC visualization method exposes the solver-based interface reconstruction to help in the assessment of simulation quality. In the visualization of interface deformation, on the other hand, the reimplementation of the algorithm for curvature computation allowed to gain better understanding of the role of the surface tension on inclusion instabilities. Finally, in the visualization of electric field on interfaces, special handling of edge-based data representation conveyed field discontinuities at material boundaries. These techniques show that in order to effectively visualize various phenomena at phase interfaces, specially tailored solver-aware approaches are advantageous, although the inherent disadvantage of such techniques is their relatively limited application.

Material Transport A number of physical phenomena related to droplet dynamics can be investigated using single-phase flow simulations. This is possible either when the droplet internal flow is considered, or when the presence of interfaces is not relevant for the analysis. Although visualization of single-phase flow is an established research field, the flow dynamics still poses problems during the visual analysis. Therefore, in Chapter 5, several visualization methods have been presented that substantially improve the investigation of single-phase flows, either by effectively conveying the analyzed physical phenomena, or by reducing the flow complexity and extracting relevant flow features.

The transport of diffusive quantities simulated with CFD cannot be captured with traditional flow visualization techniques that only take into account the advective transport. To address this issue, a dye-based visualization for advection-diffusion processes was presented in Section 5.1. In this method, the concept of passive diffusion to visualize diffusion fluxes was introduced that reveals the transport of diffusive quantities due to both advection and diffusion (Figure 6.3). The employed finite volume scheme with a WENO reconstruction technique ensures high-quality interactive dye advection with substantially reduced numerical diffusion. For smooth and interactive visualization of datasets with high temporal resolution, the method exploits parallelization and memory hierarchies on a GPU to efficiently process and render the streamed simulation time steps.

This chapter showed the advantage of extracting relevant features in flow fields. It has been employed to overcome high dynamics of fluid flow by reducing the analyzed data to relevant information. To handle complex flow dynamics, the method presented in Section 5.2 uses space-time representation of 2D time-dependent data, where time is represented by the z-axis of a three-dimensional domain in a Cartesian frame. With the third vector component corresponding to time, the 3D streamlines represent pathlines from the original 2D time-dependent flow field. Since these pathlines are Galilean-invariant, so are the concepts derived in the space-time representation. In this space-time domain, several 3D streamline-based techniques were investigated for the analysis of 2D time-dependent flow. The employed vortex core line extraction criteria provide a good balance between accuracy and readability. For the visualization of vortical flow in the 3D space-time field, vortex cores are augmented by the introduced vortex core

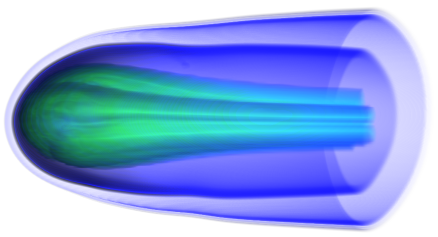


Figure 6.3: Dye-based visualization of advection-diffusion in the simulation of an evaporating droplet. Blue transparent surfaces represent vapor isocontours. Dye released at the droplet interface is advected downstream. Advection transport dominates the transport mechanism.

ribbons which clearly visualize the twist around vortices. By mapping the space-time LIC texture on the ribbons, they also provide a notion of vortex core line quality. Additionally, space-time LIC was applied on streamsurfaces to visualize both the extrinsic and intrinsic dynamics of material lines. This visualization also supports the argument that the space-time can be useful also in the analysis of single-phase flow.

In the last presented method (Section 5.3), feature extraction has been applied to gain a concise representation of complex flows. In this method, a novel, implicit topology extraction method has been proposed that, instead of providing the topology skeleton, directly reveals regions of different flow behavior. This is achieved by revealing the asymptotic behavior of streamlines seeded in the whole domain. Hence, separatrices and periodic orbits are displayed implicitly as the boundaries of these regions. Additionally, the presented grid refinement method improves the quality of the extracted regions.

The visualization techniques presented in Chapter 5 can provide better understanding of physical processes and additionally support effective analysis of highly dynamic data. A clear strength of the visualization techniques presented in this chapter is that their application range extends beyond the analysis of droplet dynamics to all types of phenomena investigated in single-phase configuration. The advantage of proposed advection-diffusion analysis is that, depending on the application problems, various visualization techniques can reveal the advection-diffusion transport. In fact, a recent publication by Sadlo et al. [153] shows that advection-diffusion can be employed for topology extraction in simulations with both advective and diffusive transport. Additionally, as proposed in the work by Hochstetter et al. [74], the advection-diffusion visualization can be extended to simulations of solvents by decomposing the concentration transport into the advective and diffusive components using the mean and maximum diffusion velocity.

A Multi-Scale Approach for Visualization of Two-Phase Flow Dynamics

The range of phenomena related to two-phase flow dynamics and their interplay is a challenging topic. As the visualization of two-phase flow has so far gained only moderate attention, it was the goal of this thesis to propose visualization techniques that provide knowledge on the dynamics of droplets as well as general two-phase flow dynamics. Consequently, the presented visualization techniques provide a comprehensive analysis for the investigation of multitude aspects of two-phase flow with the focus on droplet dynamic processes. In this thesis, droplet interactions, interface dynamics, as well as material transport in single-phase configuration have been identified as important problems. Accordingly, the developed visualization techniques complement each other to form a comprehensive approach for visual analysis of two-phase flow dynamics. The introduced visualization of droplet interactions enables a detailed investigation of the dynamics of breakups and merges and reveals the origins and characteristics of these processes. The interactions are greatly influenced by the interface dynamics. Thus, various aspects of interface characteristics have been scrutinized in this thesis to provide better understanding of both the simulation behavior and physical phenomena at the interface. Additionally, in the investigation of droplet dynamics, there are simulation scenarios where phase interface does not have to be explicitly considered. For these cases, the introduced single-phase visualization methods allow for investigation of the highly complex flows. With such a wide spectrum of visualization techniques, this thesis addresses many aspects related to the investigation of two-phase flow dynamics and can thus help the scientists and engineers in the study of natural phenomena and in engineering applications.

6.2 Outlook

The presented methods were designed to comprise an exhaustive visualization approach for the analysis of multiple aspects in two-phase flow. Regarding implementation, these methods could be integrated into a single framework, possibly by exploiting existing visualization frameworks, such as ParaView. In fact some of the presented methods have already been implemented as ParaView plugins. Although this thesis concentrated on liquid-gas flows, the presented techniques that take the interface explicitely into account are readily applicable to multiphase flows, specifically, flows with two immiscible liquids of different chemical components. This has been demonstrated in the visualization of interface deformation, where oil-water configuration has been investigated. For visualization of droplet interactions, identification of inclusions across multiple time steps leads to a correspondence problem, which involves many challenges. The particle tracking is particularly difficult in two-phase flow which is highly nonlinear. The involved collisions, as well as surface tension force are difficult to capture and require high temporal resolution for accurate tracking. It is anticipated that for detailed analysis of such data, the in-solver particle advection will be necessary, at least to provide a ground-truth solution to the particle tracking problem. In case of interface visualization, the existing techniques are somewhat limited to the VOF-based simulations. As has been discussed, interface analysis techniques benefit from the proximity to the fluid solver and it would be beneficial to extend these visualization techniques to other solver types and interface reconstruction methods. In the analysis of droplet internal flow, a very interesting yet challenging direction would be to consider the deformations of the droplets when analyzing the internal flow. This is in fact outlined in an ongoing research project at the University of Stuttgart. In this thesis, droplets in the presence of an electric field have been investigated. There are, however, many other processes—such as icing, evaporation, or mixing—that demand specialized visualization techniques that could handle these complex multi-physics problems.

Since many aspects of two-phase flow have been so far rather unexplored in the scientific visualization, it it the author's hope that this thesis will shed some light on the phenomena related to the two-phase flow dynamics with the focus on droplets. As two-phase flow is a complex and intriguing research topic, it is also the author's hope that it will get more attention and therefore spur more fascinating research in flow visualization.

Appendix

Parallelization

As stated in the Introduction (Chapter 1), the constantly growing computational power of both personal desktops and computing clusters allows for flow simulations with increasing spatial and temporal resolutions. This poses a particular challenge in the visualization of the resulting data, and in this thesis two approaches have been adopted to ensure effective flow visualization: utilization of the GPUs and distributed systems. The former enables interactive visualization and rendering of dye advection (Section 5.1), while the latter allows for processing of large datasets that do not fit into memory of standard desktops (Section 3.2). What follows is a brief description of the characteristics of both solutions that were taken into account in the visualization techniques presented in this thesis.

GPU Architecture

GPUs allow for considerable acceleration of data processing for algorithms that can benefit from parallelization. Since the GPU architecture differs greatly from the CPUs, care must be taken to properly design the algorithm implementation so that the full potential of a graphics card can be exploited. Below, several aspects are briefly described that must be considered when developing GPU code. Please note that the description—by no means exhaustive—is based on the terminology used in NVIDIA CUDA [123].

Thread Organization In the GPU programming model, the code that runs on the graphics card is called *kernel*, and one kernel is executed by one core processor. Modern graphics cards contain more than thousand computing cores that on a hardware level are organized in a hierarchy that is reflected in the programming model. From a programming point of view, at the lowest level, threads, i.e., execution units, are grouped into *warps* of typically 32 threads. At the warp level, instruction branching should be avoided on the current GPU architecture, since otherwise the execution of the branches is serialized. The thread warps are further grouped into *thread blocks* with shared memory accessible to all threads within a given block. Additionally, thread execution can be synchronized to allow for coordinated data exchange within a block. At the highest level, thread blocks are organized in a grid where a number of thread blocks can execute simultaneously, depending on the block size, resources required per block, and resources available on the device. It is important to note that currently, synchronization is not possible across thread blocks.

Memory Design In the graphics cards there are several memory types visible to a programmer that differ in access speed and latency, as well as size. The device memory is a global memory type accessible to all GPU threads. This memory is also the only

GPU memory type visible to the host (i.e., code running on the CPU), and is used to transfer data between CPU and GPU. The size of the device memory is of the order of few gigabytes but it offers the slowest access. Hence, data transfers and accesses should be reduced as much as possible. The shared memory is available for each thread block separately, such that threads within one block can exchange data through this memory. It is much faster than the device memory, however, its size is orders of magnitude smaller (typically, $48\mathrm{KB}$). The fastest and at the same time most sparse memory resource are the registers, i.e., small memory blocks assigned for storage of temporary variables during kernel execution.

Memory Access Optimization There are some nuances that must be attended to when developing GPU code. The thread warps mentioned earlier require some number of cycles to fetch data before they can execute the next instruction. This latency may range from a few tens of cycles for registers to hundreds of cycles for global memory access. It is therefore essential to hide this warp inactivity by other warps that are ready to execute their code. To increase the number of active warps (i.e., warps that reside on the GPU multiprocessors), one should decrease the number of registers used per thread. To further reduce the impact of global memory transactions, some memory access patterns should be considered. *Memory coalescence* ensures that consecutive threads read from or write to successive memory addresses. *Memory alignment* of data types means that their addresses are multiples of their size. Both solutions reduce the number of transactions needed and therefore speed up memory transfer.

Distributed Architecture

While parallelization on a GPU allows for considerable acceleration of visualization techniques, some simulation data are too large to fit into graphics cards memory and require parallelization on distributed systems. Moreover, the simulation data can sometimes be too large for transfers from clusters/supercomputers to commodity desktops, and hence require processing on the computing nodes, which usually are not equipped with GPUs.

Message Passing Interface For parallel computation on clusters and supercomputers, the Message Passing Interface (MPI) protocol is typically used [50]. MPI is an architecture-independent API that provides a set of functions for data transfers across computing nodes. For visualization, either the tasks or the simulation data are distributed on the nodes. A task is a program module responsible for processing data required for visualization. For example, in the particle advection, the task is the integration of the particle position in a given time interval. In the case of data parallelism, which has been employed in the presented visualization of inclusion separation (Section 3.2), the simulation domain is split into blocks that reside on separate computing nodes. In the employed particle tracking, particles that leave the assigned block are transferred to the process responsible for the block they enter. This approach is relatively simple to implement, and allows for processing large datasets. A disadvantage of data parallelism is that for particles that follow similar paths, computational load is poorly distributed, since the computed particle positions are concentrated in relatively few subdomains.

BIBLIOGRAPHY

[1] R. Acar and P. Boulanger. Digital marbling: a multiscale fluid model. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):600–614, 2006. 101

[2] A. Agranovsky, D. Camp, C. Garth, E. W. Bethel, K. I. Joy, and H. Childs. Improved post hoc flow analysis via Lagrangian representations. In *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*, pages 67–75, 2014. 44

[3] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Springer London, 1st edition, 2011. 23

[4] U. Alim, T. Möller, and L. Condat. Gradient estimation revitalized. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1495–1504, 2010. 64

[5] J. C. Anderson, C. Garth, M. A. Duchaineau, and K. I. Joy. Discrete multi-material interface reconstruction for volume fraction data. *Computer Graphics Forum*, 27(3):1015–1022, 2008. 19

[6] J. C. Anderson, C. Garth, M. A. Duchaineau, and K. I. Joy. Smooth, volume-accurate material interface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 16(5):802–814, 2010. 19

[7] J. D. Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill, 5th edition, 2010. 9

[8] R. Aris. *Vectors, Tensors and the Basic Equations of Fluid Mechanics*. Dover Books on Mathematics. Dover Publications, 1990. 14

[9] N. Ashgriz and J. Y. Poo. FLAIR: Flux line-segment model for advection and interface reconstruction. *Journal of Computational Physics*, 93(2):449–468, 1991. 13

[10] U. Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., 2015. 51, 83, 97, 132

[11] S. Bachthaler, F. Sadlo, C. Dachsbacher, and D. Weiskopf. Space-time visualization of dynamics in Lagrangian coherent structures of time-dependent 2D vector fields. In *Proceedings of International Conference on Information Visualization Theory and Applications (IVAPP)*, pages 573–583, 2012. 120

[12] S. Bachthaler, F. Sadlo, R. Weeber, S. Kantorovich, C. Holm, and D. Weiskopf. Magnetic flux topology of 2D point dipoles. *Computer Graphics Forum*, 31(3pt1):955–964, 2012. 91

[13] M. Bartoň, J. Kosinka, and V. M. Calo. Stretch-minimising stream surfaces. *Graphical Models*, 79:12–22, 2015. 75

[14] B. G. Becker, N. L. Max, and D. A. Lane. Unsteady flow volumes. In *IEEE Visualization 1995*, pages 329–335, 1995. 44

[15] A. S. Berres. *Discrete Geometric Methods for Surface Deformation and Visualisation*. PhD thesis, Technische Universität Kaiserslautern, 2015. 76

[16] A. Bock, E. Sunden, B. Liu, B. Wunsche, and T. Ropinski. Coherency-based curve compression for high-order finite element model visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2315–2324, 2012. 91

[17] M. Boger, J. Schlottke, C.-D. Munz, and B. Weigand. Reduction of parasitic currents in the DNS VOF code FS3D. In *Proceedings of 12th Workshop on Two-Phase Flow Predictions*, 2010. 78

[18] M. Bojsen-Hansen, H. Li, and C. Wojtan. Tracking surfaces with evolving topology. *ACM Transactions on Graphics*, 31(4):53:1–53:10, 2012. 20

[19] N. Bonneel, M. van de Panne, S. Paris, and W. Heidrich. Displacement interpolation using lagrangian mass transport. *ACM Transactions on Graphics*, 30(6):158:1–158:12, 2011. 20

[20] K. S. Bonnell, M. A. Duchaineau, D. R. Schikore, B. Hamann, and K. I. Joy. Material interface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):500–511, 2003. 18

[21] A. Bossavit. A rationale for 'edge-elements' in 3-D fields computations. *IEEE Transactions on Magnetics*, 24(1):74–79, 1988. 90

[22] J. U. Brackbill, D. B. Kothe, and C. Zemach. A continuum method for modeling surface tension. *Journal of Computational Physics*, 100(2):335–354, 1992. 76

[23] A. Brambilla, P. Angelelli, Ø. Andreassen, and H. Hauser. Comparative visualization of multiple time surfaces by planar surface reformation. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, pages 88–95, 2016. 75

[24] P.-T. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):248–260, 2010. 23

[25] P.-T. Bremer, G. Weber, J. Tierny, V. Pascucci, M. Day, and J. Bell. Interactive exploration and analysis of large-scale simulations using topology-based data segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 17(9):1307–1324, 2011. 23

[26] R. Bridson. *Fluid Simulation for Computer Graphics*. CRC Press, 2008. 11

[27] K. Brodlie. *Scientific Visualization - Techniques and Applications*. Springer Berlin Heidelberg, 1992. 1

[28] P. Buning and J. Steger. Graphics and flow visualization in computational fluid dynamics. In *Proceedings of 7th Computational Physics Conference*, Meeting Paper Archive. American Institute of Aeronautics and Astronautics, 1985. 1

[29] D. Bürkle, T. Preußer, and M. Rumpf. Transport and anisotropic diffusion in time-dependent flow visualization. In *IEEE Visualization 2001*, pages 61–68, 2001. 102

[30] 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, pages 263–270. ACM, 1993. 101, 118, 120, 132

[31] W. Cao, W. Huang, and R. Russell. A moving mesh method based on the geometric conservation law. *SIAM Journal on Scientific Computing*, 24(1):118–142, 2002. 12

[32] G. Chen, K. Mischaikow, R. S. Laramee, P. Pilarczyk, and E. Zhang. Vector field editing and periodic orbit extraction using morse decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):769–785, 2007. 117

[33] C.-S. Chou and C.-W. Shu. High order residual distribution conservative finite difference WENO schemes for convection–diffusion steady state problems on non-smooth meshes. *Journal of Computational Physics*, 224(2):992–1020, 2007. 102

[34] P. Colella and P. R. Woodward. The Piecewise Parabolic Method (PPM) for gas-dynamical simulations. *Journal of Computational Physics*, 54(1):174–201, 1984. 102

[35] C. D. Correa, R. Hero, and K. L. Ma. A comparison of gradient estimation methods for volume rendering on unstructured meshes. *IEEE Transactions on Visualization and Computer Graphics*, 17(3):305–319, 2011. 81

[36] S. J. Cummins, M. M. Francois, and D. B. Kothe. Estimating curvature from volume fractions. *Computers & Structures*, 83(6-7):425–434, 2005. 76

[37] N. Cuntz, A. Kolb, R. Strzodka, and D. Weiskopf. Particle level set advection for the interactive visualization of unsteady 3D flow. *Computer Graphics Forum*, 27(3):719–726, 2008. 101

[38] D. Degani, A. Seginer, and Y. Levy. Graphical visualization of vortical flows by means of helicity. *AIAA Journal*, 28(8):1347–1352, 1990. 17, 121

[39] U. Diewald, T. Preusser, and M. Rumpf. Anisotropic diffusion in vector field visualization on Euclidean domains and surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):139–149, 2000. 102

[40] H. Doleisch and H. Hauser. Smooth brushing for focus+context visualization of simulation data in 3D. In *Journal of WSCG*, pages 147–154, 2001. 23

[41] M. Dumbser and M. Käser. Arbitrary high order non-oscillatory finite volume schemes on unstructured meshes for linear hyperbolic systems. *Journal of Computational Physics*, 221(2):693–723, 2007. 102, 107, 109, 111, 112

[42] K. Eisenschmidt, M. Ertl, H. Gomaa, C. Kieffer-Roth, C. Meister, P. Rauschenberger, M. Reitzle, K. Schlottke, and B. Weigand. Direct numerical simulations for multiphase flows: An overview of the multiphase code FS3D. *Applied Mathematics and Computation*, 272, Part 2:508–517, 2016. 11, 13

[43] M. Ertl and B. Weigand. Direct numerical simulations of surface waves on shear thinning praestol jets in the near nozzle region. In *Proceedings of ICLASS 2015, 13th Triennial International Conference on Liquid Atomization and Spray Systems*, 2015. 12

[44] M. Ertl, B. Weigand, G. K. Karch, F. Sadlo, and T. Ertl. Investigation and visual analysis of direct simulations of quasi-steady primary break-up of shear thinning liquids. In *Proceedings of 9th International Conference on Multiphase Flow: ICMF 2016*, 2016. 138

[45] J. M. Esturo, M. Schulze, C. Rössl, and H. Theisel. Global selection of stream surfaces. *Computer Graphics Forum*, 32(2pt1):113–122, 2013. 117

[46] C. A. Felippa. Advanced finite element methods (ASEN 6367) course material. University of Colorado at Boulder. 92

[47] O. Fernandes, D. S. Blom, S. Frey, S. H. Van Zuijlen, H. Bijl, and T. Ertl. On in-situ visualization for strongly coupled partitioned fluid-structure interaction. In *Proceedings of the VI International Conference on Computational Methods for Coupled Problems in Science and Engineering*, 2015. 64

[48] M. S. Floater and K. Hormann. Surface parameterization: A tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. Springer Berlin Heidelberg, 2005. 76, 80

[49] C. Focke and D. Bothe. Direct numerical simulation of binary off-center collisions of shear thinning droplets at high Weber numbers. *Physics of Fluids (1994-present)*, 24(7):073105, 2012. 12

[50] M. P. Forum. MPI: A Message-Passing Interface Standard. Technical report, University of Tennessee, Knoxville, TN, USA, 1994. 13, 148

[51] M. M. Francois, S. J. Cummins, E. D. Dendy, D. B. Kothe, J. M. Sicilian, and M. W. Williams. A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework. *Journal of Computational Physics*, 213(1):141 – 173, 2006. 77

[52] A. Friederici, C. Rössl, and H. Theisel. Finite time steady 2D vector field topology. In *Proceedings of TopoInVis'15*, 2015. 131

[53] M. Frigo and S. G. Johnson. The Design and Implementation of FFTW3. In *Proceedings of the IEEE*, volume 93, pages 216–231, 2005. 28

[54] R. Fuchs, R. Peikert, H. Hauser, F. Sadlo, and P. Muigg. Parallel vectors criteria for unsteady flow vortices. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):615–626, 2008. 118

[55] D. Fuster, G. Agbaglah, C. Josserand, S. Popinet, and S. Zaleski. Numerical simulation of droplets, bubbles and waves: state of the art. *Fluid Dynamics Research*, 41(6):065001, 2009. 12

[56] D. Fuster, A. Bagué, T. Boeck, L. Le Moyne, A. Leboissetier, S. Popinet, P. Ray, R. Scardovelli, and S. Zaleski. Simulation of primary atomization with an octree adaptive mesh refinement and VOF method. *International Journal of Multiphase Flow*, 35(6):550–565, 2009. 12

[57] C. Garth, R. S. Laramee, X. Tricoche, J. Schneider, and H. Hagen. Extraction and visualization of swirl and tumble motion from engine simulation data. In *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 121–135. Springer Berlin Heidelberg, 2007. 43

[58] C. Garth and X. Tricoche. Topology- and feature-based flow visualization: Methods and applications. In *Proceedings of SIAM Conference on Geometric Design and Computing*, 2005. 15, 16

[59] J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transactions on Graphics*, 23(1):45–63, 2004. 76

[60] H. Gomaa, N. Roth, J. Schlottke, and B. Weigand. DNS calculations for the modeling of real industrial applications. *Atomization and Sprays*, 20(4):281–296, 2010. 13

[61] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 62–67, 2003. 76

[62] S. Grottel, G. Reina, J. Vrabec, and T. Ertl. Visual verification and analysis of cluster detection for molecular dynamics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1624–1631, 2007. 23

[63] Y. Gu and C. Wang. TransGraph: Hierarchical exploration of transition relationships in time-varying volumetric data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2015–2024, 2011. 23

[64] G. Haller. Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D: Nonlinear Phenomena*, 149(4):248–277, 2001. 15, 44, 126

[65] F. H. Harlow. The particle-in-cell method for numerical solution of problems in fluid dynamics. Technical report, Los Alamos Scientific Lab., New Mexico, 1962. 1

[66] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids*, 8(12):2182–2189, 1965. 1, 11

[67] C. Harrison, H. Childs, and K. P. Gaither. Data-parallel mesh connected components labeling and analysis. In *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization*, EGPGV '11, pages 131–140. Eurographics Association, 2011. 51

[68] J. L. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *Computer*, 22(8):27–36, 1989. 1, 23, 130

[69] C. Hirsch. Chapter 5 - finite volume method and conservative discretization with an introduction to finite element method. In *Numerical Computation of Internal and External Flows (Second Edition)*, pages 203–248. Butterworth-Heinemann, 2007. 104

[70] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225, 1981. 12, 13

[71] M. Hlawatsch, F. Sadlo, H. Jang, and D. Weiskopf. Pathline glyphs. *Computer Graphics Forum*, 33(2):497–506, 2014. 117

[72] M. Hlawatsch, F. Sadlo, and D. Weiskopf. Hierarchical line integration. *IEEE Transactions on Visualization and Computer Graphics*, 17(8):1148–1163, 2011. 117

[73] M. Hlawatsch, F. Sadlo, and D. Weiskopf. Predictability-based adaptive mouse interaction and zooming for visual flow exploration. *International Journal for Uncertainty Quantification*, 3(3):225–240, 2013. 117

[74] H. Hochstetter, M. Wurm, and A. Kolb. Vector field visualization of advective-diffusive flows. *Computer Graphics Forum*, 34(3):481–490, 2015. 142

[75] Z. Hossain, U. R. Alim, and T. Moller. Toward high-quality gradient estimation on regular lattices. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):426–439, 2011. 64

[76] J. P. M. Hultquist. Constructing stream surfaces in steady 3D vector fields. In *IEEE Visualization 1992*, pages 171–178, 1992. 119

[77] B. M. Irons and O. C. Zienkiewicz. *The Isoparametric Finite Element System: A New Concept in Finite Element Analysis*. Royal Aeronautical Society, 1969. 90

[78] J. Jeong and F. Hussain. On the identification of a vortex. *Journal of Fluid Mechanics*, 285:69–94, 1995. 17

[79] G.-S. Jiang and C.-W. Shu. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126(1):202–228, 1996. 105

[80] B. Jobard, G. Erlebacher, and M. Y. Hussaini. Hardware-accelerated texture advection for unsteady flow visualization. In *IEEE Visualization 2000*, pages 155–162, 2000. 101

[81] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In A. P. D. W. Lefer and D. M. Grave, editors, *Visualization in Scientific Computing '97*, Eurographics, pages 43–55. Springer Vienna, 1997. 117

[82] H. Jänicke and G. Scheuermann. Visual analysis of flow features using information theory. *IEEE Computer Graphics and Applications*, 30(1):40–49, 2010. 23

[83] G. K. Karch, F. Beck, M. Ertl, C. Meister, K. Schulte, B. Weigand, T. Ertl, and F. Sadlo. Visual analysis of inclusion dynamics in two-phase flow. *IEEE Transactions on Visualization and Computer Graphics*, 2017. 4, 21, 22

[84] G. K. Karch, F. Sadlo, S. Boblest, M. Ertl, B. Weigand, K. Gaither, and T. Ertl. Visualization of inclusion separation in two-phase flow. *arXiv:1705.05138 [cs]*, 2017. 43

[85] G. K. Karch, F. Sadlo, C. Meister, P. Rauschenberger, K. Eisenschmidt, B. Weigand, and T. Ertl. Visualization of piecewise linear interface calculation. In *Proceedings of 2013 IEEE Pacific Visualization Symposium (PacificVis)*, pages 121–128, 2013. 4, 51, 61, 63, 66

[86] G. K. Karch, F. Sadlo, H. Songoro, E. Gjonaj, T. Weiland, and T. Ertl. Visualizing edge-conforming discrete field quantities in electromagnetic field problems with interfaces. In *Proceedings of ILASS Europe, 25th European Conference on Liquid Atomization and Spray Systems, 2013*, 2013. 5, 62, 90

[87] G. K. Karch, F. Sadlo, D. Weiskopf, and T. Ertl. Streamline-based concepts for space-time analysis of 2D time-dependent flow. In *Proceedings of International Symposium on Flow Visualization (ISFV16)*, 2014. 5

[88] G. K. Karch, F. Sadlo, D. Weiskopf, and T. Ertl. Visualization of 2D unsteady flow using streamline-based concepts in space-time. *Journal of Visualization*, 19(1):115–128, 2016. 5, 99

[89] G. K. Karch, F. Sadlo, D. Weiskopf, C. D. Hansen, G. S. Li, and T. Ertl. Dye-based flow visualization. *Computing in Science Engineering*, 14(6):80–86, 2012. 5, 99, 101, 117

[90] G. K. Karch, F. Sadlo, D. Weiskopf, C.-D. Munz, and T. Ertl. Visualization of advection-diffusion in unsteady fluid flow. *Computer Graphics Forum*, 31(3pt2):1105–1114, 2012. 5, 99, 101, 117

[91] G. K. Karch, A. Straub, F. Sadlo, and T. Ertl. Implicit visualization of vector field topology. In *Proceedings of TopoInVis'17*, 2017. 5, 100, 130

[92] T. L. Kay and J. T. Kajiya. Ray tracing complex scenes. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 269–278. ACM, 1986. 51

[93] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *IEEE Visualization 2003*, VIS '03, pages 513–520, 2003. 76

[94] P. Kundu and I. Cohen. *Fluid Mechanics*. Elsevier Science, 2010. 9

[95] K. Kurzhals and D. Weiskopf. Space-time visual analytics of eye-tracking data for dynamic stimuli. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2129–2138, 2013. 118

[96] D. Laney, P. T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1053–1060, 2006. 22

[97] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum*, 23(2):203–221, 2004. 14, 101

[98] R. S. Laramee, B. Jobard, and H. Hauser. Image space based visualization of unsteady flow on surfaces. In *IEEE Visualization 2003*, pages 131–138, 2003. 101

[99] T. Lasinski, P. Buning, D. Choi, S. Rogers, and G. Bancroft. Flow visualization of CFD using graphics workstations. In *Proceedings of 8th Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, 1987. 1

[100] A. Lefebvre. *Atomization and Sprays*. Combustion (Hemisphere Publishing Corporation). Taylor & Francis, 1988. 12

[101] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988. 114

[102] G.-S. Li, X. Tricoche, and C. Hansen. Physically-based dye advection for flow visualization. *Computer Graphics Forum*, 27(3):727–734, 2008. 102

[103] G. S. Li, X. Tricoche, D. Weiskopf, and C. D. Hansen. Flow charts: Visualization of vector fields on arbitrary surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1067–1080, 2008. 101

[104] X. Li, M. Arienti, M. Soteriou, and M. Sussman. Towards an efficient, high-fidelity methodology for liquid jet atomization computations. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, 2010. 12

[105] X. Li, X. He, X. Liu, J. J. Zhang, B. Liu, and E. Wu. Multiphase interface tracking with fast semi-Lagrangian contouring. *IEEE Transactions on Visualization and Computer Graphics*, 22(8):1973–1986, 2016. 63

[106] D. R. Lipşa, R. S. Laramee, S. J. Cox, J. C. Roberts, R. Walker, M. A. Borkin, and H. Pfister. Visualization for the physical sciences. *Computer Graphics Forum*, 31(8):2317–2347, 2012. 14

[107] B. Liu, A. Bock, T. Ropinski, M. Nash, P. Nielsen, and B. C. Wünsche. GPU-accelerated direct volume rendering of finite element data sets. In *Proceedings of the 27th Conference on Image and Vision Computing*, IVCNZ '12, pages 109–114. ACM, 2012. 91

[108] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200–212, 1994. 102

[109] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169. ACM, 1987. 17

[110] J. Ma, C. Wang, C.-K. Shene, and J. Jiang. A graph-based interface for visual analytics of 3D streamlines and pathlines. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1127–1140, 2014. 22

[111] G. Machado, S. Boblest, T. Ertl, and F. Sadlo. Space-time bifurcation lines for extraction of 2D Lagrangian coherent structures. *Computer Graphics Forum*, 35(3):91–100, 2016. 15, 118, 119

[112] G. M. Machado, F. Sadlo, and T. Ertl. Local extraction of bifurcation lines. In *Vision, Modeling, and Visualization*. The Eurographics Association, 2013. 118, 131

[113] G. M. Machado, F. Sadlo, T. Müller, D. Müller, and T. Ertl. Visualizing solar dynamics data. In *Proceedings of Vision, Modeling and Visualization (VMV)*, pages 95–102, 2012. 91

[114] S. Mann. Extending the A-patch single sheet conditions to enable the tessellation of algebraics. In *Proceedings of 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, pages 337–342. ACM, 2009. 19

[115] 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, pages 213–222. ACM, 2003. 117

[116] T. McLoughlin, M. W. Jones, R. S. Laramee, R. Malki, I. Masters, and C. D. Hansen. Similarity measures for enhancing interactive streamline seeding. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1342–1353, 2013. 117

[117] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum*, 29(6):1807–1829, 2010. 14

[118] J. S. Meredith and H. Childs. Visualization and analysis-oriented reconstruction of material interfaces. *Computer Graphics Forum*, 29:1241–1250, 2010. 18

[119] M. M. Mesmoudi, L. D. Floriani, and P. Magillo. Discrete curvature estimation methods for triangulated surfaces. In *Applications of Discrete Geometry and Mathematical Morphology*, pages 28–42. Springer Berlin Heidelberg, 2012. 76

[120] M. Meyer, B. Nelson, R. Kirby, and R. Whitaker. Particle systems for efficient and accurate high-order finite element visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1015–1026, 2007. 91

[121] G. Mur. Edge elements, their advantages and their disadvantages. *IEEE Transactions on Magnetics*, 30(5):3552–3557, 1994. 90

[122] J. C. Nedelec. Mixed finite elements in R3. *Numerische Mathematik*, 35(3):315–341, 1980. 90

[123] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with CUDA. *Queue*, 6(2):40–53, 2008. 114, 147

[124] W. F. Noh and P. Woodward. SLIC (simple line interface calculation). In *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics*, number 59 in Lecture Notes in Physics, pages 330–340. Springer Berlin Heidelberg, 1976. 13

[125] H. Obermaier, F. Chen, H. Hagen, and K. I. Joy. Visualization of material interface stability. In *2014 IEEE Pacific Visualization Symposium (PacificVis)*, pages 225–232, 2012. 19, 64

[126] H. Obermaier, M. Hering-Bertram, J. Kuhnert, and H. Hagen. Volume deformations in grid-less flow simulations. *Computer Graphics Forum*, 28(3):879–886, 2009. 75

[127] H. Obermaier and K. I. Joy. Derived metric tensors for flow surface visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2149–2158, 2012. 75, 76, 80

[128] Y. Ohtake and H. Suzuki. Edge detection based multi-material interface extraction on industrial CT volumes. *Science China Information Sciences*, 56(9):1–9, 2013. 63

[129] OpenMP Architecture Review Board. *OpenMP Application Programming Interface*, 2015. 13, 51

[130] S. Ozer, D. Silver, K. Bemis, and P. Martin. Activity detection in scientific visualization. *IEEE Transactions on Visualization & Computer Graphics*, (1), 2013. 22

[131] C. Pagot, D. Osmari, F. Sadlo, D. Weiskopf, T. Ertl, and J. Comba. Efficient parallel vectors feature extraction from higher-order data. *Computer Graphics Forum*, 30(3):751–760, 2011. 91

[132] C. Pagot, J. Vollrath, F. Sadlo, D. Weiskopf, T. Ertl, and J. ao Luiz Dihl Comba. Interactive isocontouring of high-order surfaces. In *Scientific Visualization: Interactions, Features, Metaphors*, pages 276–291. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011. 91

[133] N. M. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer Berlin Heidelberg, 1st edition, 2009. 83

[134] R. Peikert and M. Roth. The "parallel vectors" operator: A vector field visualization primitive. In *IEEE Visualization 1999*, pages 263–270, 1999. 121

[135] R. Peikert and F. Sadlo. Flow topology beyond skeletons: Visualization of features in recirculating flow. In *Topology-Based Methods in Visualization II*, Mathematics and Visualization, pages 145–160. Springer Berlin Heidelberg, 2009. 15

[136] A. E. Perry and a. M. S. Chong. A description of eddying motions and flow patterns using critical-point concepts. *Annual Review of Fluid Mechanics*, 19(1):125–155, 1987. 118

[137] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow map layout. In *Proceedings of the 2005 IEEE Symposium on Information Visualization*, pages 219–224, 2005. 23

[138] J. E. Pilliod Jr. and E. G. Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *Journal of Computational Physics*, 199(2):465–502, 2004. 13

[139] A. Pobitzer, R. Peikert, R. Fuchs, B. Schindler, A. Kuhn, H. Theisel, K. Matković, and H. Hauser. The state of the art in topology-based visualization of unsteady flow. *Computer Graphics Forum*, 30(6):1789–1811, 2011. 23

[140] S. Popinet. An accurate adaptive solver for surface-tension-driven interfacial flows. *Journal of Computational Physics*, 228(16):5838–5866, 2009. 76, 77, 78

[141] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, 2003. 22, 43

[142] A. Preston, R. Ghods, J. Xie, F. Sauer, N. Leaf, K. L. Ma, E. Rangel, E. Kovacs, K. Heitmann, and S. Habib. An integrated visualization system for interactive analysis of large, heterogeneous cosmology data. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, pages 48–55, 2016. 23

[143] I. Prilepov, H. Obermaier, E. Deines, C. Garth, and K. I. Joy. Cubic gradient-based material interfaces. *IEEE Transactions on Visualization and Computer Graphics*, 19(10):1687–1699, 2013. 63

[144] F. Reinders, F. H. Post, and H. J. W. Spoelder. Visualization of time-dependent data with feature tracking and event detection. *The Visual Computer*, 17(1):55–71, 2001. 19, 22, 23

[145] J.-F. Remacle, N. Chevaugeon, E. Marchandise, and C. Geuzaine. Efficient visualization of high-order finite elements. *International Journal for Numerical Methods in Engineering*, 69(4):750–771, 2007. 90

[146] W. J. Rider and D. B. Kothe. Reconstructing volume tracking. *Journal of Computational Physics*, 141(2):112–152, 1998. 4, 13

[147] P. Riehmann, M. Hanfler, and B. Froehlich. Interactive sankey diagrams. In *Proceedings of the 2005 IEEE Symposium on Information Visualization*, pages 233–240, 2005. 23, 32

[148] K. Robbins, C. Jeffery, and S. Robbins. Visualization of splitting and merging processes. *J. Vis. Lang. Comput.*, 11(6):593–614, 2000. 23

[149] L. Rokach and O. Maimon. Clustering methods. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 321–352. Springer US, 2005. 32

[150] M. Roth. *Automatic extraction of vortex core lines and other line-type features for scientific visualization*. theses, Diss. Technische Wissenschaften ETH Zürich, Nr.13673, 2000, 2000. 17, 118

[151] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, 3DPVT '04, pages 486–493, 2004. 76

[152] F. Sadlo. Lyapunov time for 2D Lagrangian visualization. In J. Bennett, F. Vivodtzev, and V. Pascucci, editors, *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, pages 167–181. Springer Berlin Heidelberg, 2015. 44

[153] F. Sadlo, G. K. Karch, and T. Ertl. Topological features in time-dependent advection-diffusion flow. In *Topological Methods in Data Analysis and Visualization III*, pages 217–231. Springer Cham, 2014. 142

[154] F. Sadlo, R. Peikert, and E. Parkinson. Vorticity based flow analysis and visualization for Pelton turbine design optimization. In *IEEE Visualization 2004*, pages 179–186, 2004. 16, 121, 122

[155] J. Sahner, T. Weinkauf, and H.-C. Hege. Galilean invariant extraction and iconic representation of vortex core lines. In *EUROVIS 2005: Eurographics / IEEE VGTC Symposium on Visualization*. The Eurographics Association, 2005. 17, 121

[156] A. Sanderson, G. Chen, X. Tricoche, D. Pugmire, S. Kruger, and J. Breslau. Analysis of recurrent patterns in toroidal magnetic fields. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1431–1440, 2010. 91

[157] A. R. Sanderson. An alternative formulation of Lyapunov exponents for computing Lagrangian coherent structures. In *2014 IEEE Pacific Visualization Symposium (PacificVis)*, pages 277–280, 2014. 44

[158] A. R. Sanderson, C. R. Johnson, and R. M. Kirby. Display of vector fields using a reaction-diffusion model. In *IEEE Visualization 2004*, pages 115–122, 2004. 102

[159] F. Sauer, H. Yu, and K. L. Ma. Trajectory-based flow feature tracking in joint particle/volume datasets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2565–2574, 2014. 19, 22, 23, 43, 44

[160] T. Schafhitzel, J. E. Vollrath, J. P. Gois, D. Weiskopf, A. Castelo, and T. Ertl. Topology-preserving λ_2-based vortex core line detection for flow visualization. *Computer Graphics Forum*, 27(3):1023–1030, 2008. 17

[161] J. Schlottke and B. Weigand. Direct numerical simulation of evaporating droplets. *Journal of Computational Physics*, 227(10):5215–5237, 2008. 116

[162] W. J. Schroeder, F. Bertel, M. Malaterre, D. Thompson, P. P. Pebay, R. O'Bara, and S. Tendulkar. Methods and framework for visualizing higher-order finite elements. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):446–460, 2006. 91

[163] H.-W. Shen, C. R. Johnson, and K.-L. Ma. Visualizing vector fields using line integral convolution and dye advection. In *Proceedings of the 1996 Symposium on Volume Visualization*, pages 63–ff. IEEE Press, 1996. 101

[164] K. Shi, H. Theisel, T. Weinkauf, H. C. Hege, and H. P. Seidel. Visualizing transport structures of time-dependent flow fields. *IEEE Computer Graphics and Applications*, 28(5):24–36, 2008. 43

[165] J. Solomon, F. de Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas. Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics*, 34(4):66:1–66:11, 2015. 20

[166] Sonderforschungsbereich transregio 75. www.sfbtrr75.de. 2

[167] J. Stam and R. Schmidt. On the velocity of an implicit surface. *ACM Transactions on Graphics*, 30(3):21:1–21:7, 2011. 19

[168] A. Straub. Visualization of interface instabilities in two-phase flow, 2016. Master thesis, University of Stuttgart. 5

[169] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981. 23, 32

[170] D. Sujudi and R. Haimes. Identification of swirling flow in 3-D vector fields. In *Proceedings of 12th Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, 1995. 17, 118

[171] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114(1):146–159, 1994. 12

[172] F. Taponecco and M. Alexa. Vector field visualization using Markov random field texture synthesis. In *Proceedings of the Symposium on Data Visualisation*, VISSYM '03, pages 195–202. Eurographics Association, 2003. 102

[173] A. Telea and J. J. v. Wijk. 3d IBFV: hardware-accelerated 3D flow visualization. In *IEEE Visualization 2003*, pages 233–240, 2003. 101

[174] H. Theisel and H.-P. Seidel. Feature flow fields. In *Proceedings of the Symposium on Data Visualisation 2003*, VISSYM '03, pages 141–148. Eurographics Association, 2003. 118

[175] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Grid-independent detection of closed stream lines in 2D vector fields. 2004. 131

[176] H. Theisel, T. Weinkauf, H. C. Hege, and H. P. Seidel. Stream line and path line oriented topology for 2D time-dependent vector fields. In *IEEE Visualization 2004*, pages 321–328, 2004. 118

[177] G. Tryggvason, R. Scardovelli, and S. Zaleski. *Direct Numerical Simulations of Gas–Liquid Multiphase Flows*. Cambridge University Press, 2011. 12

[178] G. Turk. Generating Textures on Arbitrary Surfaces Using Reaction-diffusion. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pages 289–298. ACM, 1991. 102

[179] G. Turk and D. Banks. Image-guided streamline placement. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 453–460. ACM, 1996. 117

[180] S.-K. Ueng, C. Sikorski, and K.-L. Ma. Efficient streamline, streamribbon, and streamtube constructions on unstructured grids. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):100–110, 1996. 120, 123

[181] M. Üffinger, S. Frey, and T. Ertl. Interactive high-quality visualization of higher-order finite elements. *Computer Graphics Forum*, 29(2):337–346, 2010. 91

[182] A. Vilanova, S. Zhang, G. Kindlmann, and D. Laidlaw. An introduction to visualization of diffusion tensor imaging and its applications. In P. J. Weickert and P. H. Hagen, editors, *Visualization and Processing of Tensor Fields*, Mathematics and Visualization, pages 121–153. Springer Berlin Heidelberg, 2006. 102

[183] R. L. V. Wal, G. M. Berger, and S. D. Mozes. The splash/non-splash boundary upon a dry surface and thin fluid film. *Experiments in Fluids*, 40(1):53–59, 2005. 30, 31

[184] C. Wang. A survey of graph-based representations and techniques for scientific visualization. 2015. 23

[185] J. P. Webb. Edge elements and what they can do for you. *IEEE Transactions on Magnetics*, 29(2):1460–1465, 1993. 90

[186] J. P. Webb. Hierarchal vector basis functions of arbitrary order for triangular and tetrahedral finite elements. *IEEE Transactions on Antennas and Propagation*, 47(8):1244–1253, 1999. 90

[187] T. Weinkauf, J. Sahner, H. Theisel, and H. C. Hege. Cores of swirling particle motion in unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1759–1766, 2007. 118, 121

[188] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Boundary switch connectors for topological visualization of complex 3D vector fields. In *Proceedings of the Sixth Joint Eurographics - IEEE TCVG Conference on Visualization*, VISSYM'04, pages 183–192. Eurographics Association, 2004. 130

[189] T. Weinkauf, H. Theisel, and O. Sorkine. *Cusps of Characteristic Curves and Intersection-Aware Visualization of Path and Streak Lines*, pages 161–175. Springer Berlin Heidelberg, 2012. 117

[190] D. Weiskopf. Dye advection without the blur: A level-set approach for texture-based visualization of unsteady flow. *Computer Graphics Forum*, 23(3):479–488, 2004. 101

[191] D. Weiskopf. Iterative twofold line integral convolution for texture-based vector field visualization. In *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, Mathematics and Visualization. Springer Berlin Heidelberg, 2009. 101

[192] D. Weiskopf, M. Hopf, and T. Ertl. Hardware-accelerated visualization of time-varying 2D and 3D vector fields by texture advection via programmable per-pixel operations. In *Proceedings of Vision, Modeling and Visualization (VMV)*, pages 439–446, 2001. 101

[193] D. Weiskopf, T. Schafhitzel, and T. Ertl. Texture-based visualization of unsteady 3D flow by real-time advection and volumetric illumination. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):569–582, 2007. 101

[194] H. Whitney. *Geometric Integration Theory*. Courier Corporation, 2005. 90

[195] W. Widanagamaachchi, J. Chen, P. Klacansky, V. Pascucci, H. Kolla, A. Bhagatwala, and P. T. Bremer. Tracking features in embedded surfaces: Understanding extinction in turbulent combustion. In *2015 IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV)*, pages 9–16, 2015. 23

[196] A. Wiebel, X. Tricoche, D. Schneider, H. Janicke, and G. Scheuermann. Generalized streak lines: Analysis and visualization of boundary induced vortices. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1735–1742, 2007. 14

[197] J. J. v. Wijk. Implicit stream surfaces. In *IEEE Visualization 1993*, pages 245–252, 1993. 44

[198] J. J. v. Wijk. Image based flow visualization. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 745–754. ACM, 2002. 101

[199] J. J. v. Wijk. Image based flow visualization for curved surfaces. In *IEEE Visualization 2003*, pages 123–130, 2003. 101

[200] D. F. Wiley, H. R. Childs, B. Hamann, and K. I. Joy. Ray casting curved-quadratic elements. In *Proceedings of the Sixth Joint Eurographics - IEEE TCVG Conference on Visualization*, VISSYM'04, pages 201–210. Eurographics Association, 2004. 90

[201] J. O. Wilkes and S. W. Churchill. The finite-difference computation of natural convection in a rectangular enclosure. *AIChE Journal*, 12(1):161–166, 1966. 1

[202] T. Wischgoll and G. Scheuermann. Detection and visualization of closed streamlines in planar flows. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):165–172, 2001. 130

[203] A. Witkin and M. Kass. Reaction-diffusion textures. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pages 299–308. ACM, 1991. 102

[204] C. Wojtan, N. Thürey, M. Gross, and G. Turk. Physics-inspired topology changes for thin fluid features. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 50:1–50:8. ACM, 2010. 19

[205] D. Xue, C. Zhang, and R. Crawfis. Rendering implicit flow volumes. In *IEEE Visualization 2004*, pages 99–106, 2004. 44

[206] X. Ye, D. Kao, and A. Pang. Strategy for seeding 3D streamlines. In *IEEE Visualization 2005*, pages 471–478, 2005. 117

[207] K. Yokoi. A numerical method for free-surface flows and its application to droplet impact on a thin liquid layer. *Journal of Scientific Computing*, 35(2-3):372–396, 2008. 12

[208] D. L. Youngs. Time-dependent multi-material flow with large fluid distortion. *Numerical methods for fluid dynamics*, 24(2):273–285, 1982. 13, 63

[209] D. L. Youngs. An interface tracking method for a 3D Eulerian hydrodynamics code. *Atomic Weapons Research Establishment (AWRE) Technical Report*, (44/92):35, 1984. 63

[210] Y. Zhou and M. Garland. Interactive point-based rendering of higher-order tetrahedral data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1229–1236, 2006. 91