Tobias Günther

Opacity Optimization and Inertial Particles in Flow Visualization

INF FACULTY OF COMPUTER SCIENCE

Opacity Optimization and Inertial Particles in Flow Visualization

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.-Ing.)

angenommen durch die Fakultät für Informatik der Otto-von-Guericke Universität Magdeburg.

vorgelegt von:	Tobias Günther
geboren am:	26. Juni 1988
in:	Gardelegen
Gutachter:	Prof. Dr. Holger Theisel
	Prof. Dr. Thomas Ertl
	Prof. Dr. Rüdiger Westermann

Magdeburg, den 30. Juni 2016

LOCATION:

Otto-von-Guericke University
Faculty of Computer Science
Department of Simulation and Graphics
Magdeburg

Tobias Günther: Opacity Optimization and Inertial Particles in Flow Visualization

Acknowledgements

Holger, I feel the deepest gratitude for the many opportunities you have offered me, and for everything you taught me. You have supported me long before I even started with my PhD. I guess, we had too many ideas back then to not take it on the road, and with your constant encouragement, ideas and feedback, we produced something that I can gladly be proud of. Let's continue on this course! I believe, many more things wait for us to be unraveled with Maple.

Alex, it appears that quite a number of coincidences led us to the study of inertial particles. With them, you paved the way for the entire third part of this thesis. Thanks, my friend, I wouldn't have had much to write, if it weren't for you and these lucky coincidences!

Kai, thanks for all the years of friendship, the time we spent studying, and the great work on all these projects that I had the pleasure to work on with you!

Maik, please don't take it the wrong way, but you have always been the 'older guy' among us PhD students to look up to (I didn't wrote elderly!). Already in my first week in the group, I learned extremely useful things from you and Janick, and it was instantly clear to me that I have arrived in the right place. Thanks for the great years!

Christian, you support the group in so many ways. Having you on a paper always assures that it will end up in world-class structure and writing. In your courses and in our work, you taught me so many useful things that I can take with me. Thank you!

Kai, my former (almost regularly Monday) Subway fellow and sports buddy! I got so out of shape since you moved. Though, I probably wouldn't have been in one if it weren't for you (and Ben, obviously!). Thanks for the great time, and the discussions on our work. For me it was both quite interesting and motivating to see such a productive paper machine at work.

Nora, thank you for the many conversations and that you always have my back!

And of course, without the everlasting love and support of my parents and sister, I wouldn't even have had the slightest chance to be here. Thank you so much!

This list is not complete and it can hardly ever be. There are many more companions and friends who have joined and supported me on my path. In some way or the other, you have contributed to this work, as you had a great influence on me. Rest assured that even if I haven't mentioned you personally, you are definitely not forgotten.

Abstract

Vector field visualization is a major discipline of scientific visualization that helps to push the frontiers of research in fluid mechanics, medicine, biology, astrophysics and many more. In particular, vector field visualization is concerned with the discovery of relationships in possibly large and complex vector fields, which serve as general descriptors of air and fluid flows, magnetic fields and dynamical systems. The visualization community found a number of different ways to assist in the analysis and exploration of these fields. Two major classes of approaches are the so-called *geometry-based* and *feature-based* / *topology-based* techniques. The first and second part of this thesis introduce techniques that reside in these two classes, respectively. The third part of the thesis addresses the analysis of *inertial particles*, i.e., finite-sized objects carried by fluid flows.

When it comes to 3D flow visualization, we often encounter occlusion problems when displaying dense sets of lines or multiple surfaces. A vital aspect is the careful selection of the primitives that best communicate the relevant features in a data set. In the first part of the thesis, we present optimization-based approaches that adjust the opacity of lines and surfaces to strive for a balance between the presentation of relevant information and occlusion avoidance.

The second part of the thesis is dedicated to novel rendering techniques for the visualization of unsteady flows. For this, we will apply techniques from light transport in heterogeneous participating media to the unbiased rendering of Lagrangian scalar fields, namely finite-time Lyapunov exponents. Further, we propose a new class of vortex definitions for flows that are induced by rotating mechanical parts, such as stirring devices, hydrocyclones, centrifugal pumps or ventilators.

In the third part of this thesis, we introduce inertial particles as a new application domain to the flow visualization community. Recent research in flow visualization focused on the analysis of massless particles. However, in many application scenarios, the mass of particles and their resulting inertia are essential, such as when sand particles interact with aircraft. The governing ODE of even simple inertial flow models is up to seven dimensional, which makes feature extraction a challenging task. We abstract the description of mass-dependent particle trajectories and apply existing flow visualization methods to the mass-dependent case. In particular, we extract

and visualize integral geometry, study the vortical motion and separation behavior of inertial particles, extend traditional vector field topology to the inertial case and present a new approach to the source inversion problem, i.e., the recovery of the source of dispersed pollutants.

We demonstrate the usefulness of our methods by applying them to a variety of synthetic and real-world data sets.

Zusammenfassung

Die Visualisierung von Vektorfeldern ist eine zentrale Disziplin der wissenschaftlichen Visualisierung. In der Strömungsmechanik, Medizin, Biologie, Astrophysik und in vielen weiteren Bereichen hilft sie den gegenwärtigen Forschungsstand voranzutreiben. Insbesondere spielt die Visualisierung in der Entdeckung von Zusammenhängen in großen und komplexen Vektorfeldern eine Rolle, die generell der Beschreibung von Luft- und Wasserströmungen, Magnetfeldern und dynamischen Systemen dienen. Die Visualisierungs-Community erschloss eine Vielzahl an Wegen um bei der Analyse und Exploration dieser Felder zu unterstützen. Zwei grundlegende Klassen von Ansätzen sind die sogenannten *geometrie-basierten* und *feature-/topologie-basierten* Techniken. Der erste und zweite Teil dieser Dissertation stellt Techniken vor, die in diesen zwei Klassen angesiedelt sind. Der dritte Teil der Arbeit adressiert die Analyse von *masse-behafteten Partikeln*, die in Strömungen transportiert werden.

In der 3D Strömungsvisualisierung begegnet man häufig Verdeckungsproblemen, sobald dichte Linienfelder oder mehrere Flächen dargestellt werden. Ein wichtiger Aspekt ist daher die sorgfältige Auswahl von Darstellungsprimitiven, sodass die relevanten Strukturen im Datensatz bestmöglich kommuniziert werden können. Im ersten Teil der Dissertation stellen wir optimierungsbasierte Ansätze vor, die die Sichtbarkeit von Linien und Flächen einstellen um eine Balance zwischen der Darstellbarkeit von Informationen und der Vermeidung von Verdeckung zu finden.

Der zweite Teil der Dissertation ist neuen Visualisierungsmethoden für zeit-abhängige Strömungen gewidmet. Hierfür wenden wir Techniken aus dem Lichttransport inhomogener partizipierender Medien auf die erwartungstreue Darstellung von integrationsbasierten Skalarfeldern an, nämlich dem sogenannten finite-time Lyapunov Exponent. Des Weiteren stellen wir eine neue Klasse von Wirbeldefinitionen für Strömungen vor, die von rotierenden mechanischen Teilen induziert werden, wie etwa von Rührwerken, Hydrokzyklonen, Zentrifugalpumpen oder Ventilatoren.

Im dritten Teil der Dissertation, führen wir masse-behaftete Partikel als ein neues Anwendungsfeld in der Strömungsvisualisierung ein. Bisherige Forschung im Bereich der Strömungsvisualisierung beschränkte sich auf die Analyse zeitabhängiger, jedoch masseloser Partikel. In vielen Anwendungsgebieten spielt die Masse von Partikeln, und die daraus resultierende Trägheit, allerdings eine essentielle Rolle in der Strö-

mungsmechanik, beispielsweise wenn Sandpartikel mit Luftfahrzeugen interagieren. Die zugrundeliegende gewöhnliche Differentialgleichung von selbst einfachen Partikelbewegungsmodellen ist bis zu sieben-dimensional, was die Merkmalsextraktion zu einer herausfordernden Aufgabe macht. Wir abstrahieren die Beschreibung masse-abhängiger Partikelflugbahnen und wenden existierenden Visualisierungsmethoden für Strömungen auf den masse-abhängigen Fall an. Im Einzelnen extrahieren und visualisieren wir Integralgeometrie, studieren das Wirbelverhalten und die Separation von masse-abhängigen Partikeln, erweitern traditionelle Vektorfeldtopologie auf den masse-abhängigen Fall und präsentieren einen neuen Ansatz zur Entdeckung der Herkunft von Schmutzpartikeln in Strömungen.

Wir demonstrieren die Anwendbarkeit unserer Methoden in einer Vielzahl von synthetischen und realen Datensätzen.

Parts of this thesis have previously been published in peer-reviewed journals.

1. T. Günther, A. Kuhn, B. Kutz and H. Theisel
   Mass-Dependent Integral Curves in Unsteady Vector Fields
   Computer Graphics Forum (Proc. EuroVis) 32, 3 (2013), 211–220.

2. T. Günther, C. Rössl and H. Theisel
   Opacity Optimization for 3D Line Fields
   ACM Transactions on Graphics (Proc. SIGGRAPH) 32, 4 (2013), 120:1–120:8.

3. T. Günther, C. Rössl and H. Theisel
   Hierarchical Opacity Optimization for Sets of 3D Line Fields
   Computer Graphics Forum (Proc. Eurographics) 33, 2 (2014), 507–516.

4. T. Günther, M. Schulze, J. Martinez Esturo, C. Rössl and H. Theisel
   Opacity Optimization for Surfaces
   Computer Graphics Forum (Proc. EuroVis) 33, 3 (2014), 11–20.

5. T. Günther and H. Theisel
   Vortex Cores of Inertial Particles
   IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization) 20, 12 (2014), 2535–2544.

6. T. Günther and H. Theisel
   Finite-Time Mass Separation for Comparative Visualizations of Inertial Particles
   Computer Graphics Forum (Proc. EuroVis) 34, 3 (2015), 471–480.

7. T. Günther, M. Schulze and H. Theisel
   Rotation Invariant Vortices for Flow Visualization
   IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization 2015) 22, 1 (2016), 817–826.

8. T. Günther and H. Theisel
   Inertial Steady 2D Vector Field Topology
   Computer Graphics Forum (Proc. Eurographics) 35, 2 (2016), 455–466.

9. T. Günther and H. Theisel
   Source Inversion by Forward Integration in Inertial Flow
   Computer Graphics Forum (Proc. EuroVis) 35, 3 (2016).

10. T. Günther, A. Kuhn and H. Theisel
    MCFTLE: Progressive Rendering of Finite-Time Lyapunov Exponent Fields
    Computer Graphics Forum (Proc. EuroVis) 35, 3 (2016).

Contents

Chapter 1. Introduction

The movement of air and fluids is astounding in many ways. It surrounds us at all times, yet often we cannot see, but only feel it. When made visible – even by chance – for instance by leaves circling in the air or by smoke rising from a blown out candle, we see complex patterns emerge that inevitably catch the eye. While such small phenomena might only be noticed by watchful eyes, nature repeatedly unleashes its inconceivable power, reminding us that we are bound to its laws. Major fluid dynamics such as hurricanes, tornadoes and tsunamis affect billions of lives. Aside from such uncontrollable forces, air and fluid flows are tamed on lower scales to our advantage. For instance, they can be harnessed for conversion into electrical energy in an ecologically compatible way. Not only industrial processes such as combustion of pulverized coal or spray drying are successfully utilizing optimized circular air flows, but also smaller problems are solved, such as air jets at revolving doors or ventilators. In engineering, aerodynamics play a role in the fuel consumption of airplanes and cars. In certain applications, not only the flow itself is of interest, but the behavior of small objects transported therein. These include desert sediment transport, sand blasting, and brownout in helicopter landing maneuvers. In fact, they influence our everyday life, even though we are not constantly fully aware of it. On the one hand, nature engenders phenomena that are in part dangerous and need to be further researched in order to forecast them better. These reach from oceanic currents that drive the global climate to the blood flow in our veins. On the other hand, aerodynamics affect many engineering problems that are optimized for efficiency and safety.

In all these fields, researchers seek to understand the laws at play and strive to model these typically highly complex phenomena. For this, *flow visualization* is an important tool that makes these processes visible. Thereby, it targets a vast range of applications, in which it allows to communicate information, explore relationships, reason about causes and even helps to predict future events.

Understanding flow data is challenging for a number of reasons. Data sets are not only getting larger in size, especially for time-dependent sequences, which is challenging from a computational point of view. Also, the presentation of such densely packed information raises perceptual issues that need to be solved, for instance by avoidance of occlusion or the extraction of descriptive features. Thus, a main challenge is to find interactive exploration techniques that deal with the complexity of such flow data.

This thesis is concerned with topics resided in *geometric and feature-based / topology-based flow visualization*. Traditionally, the approaches in these fields have in common that they are based on the observation of massless tracer particles, carried by the fluid. Their trajectories follow the flow tangentially and thus provide a compact picture that proved useful to identify areas of interest and to comprehend the overall behavior of the flow. The first part of the thesis addresses a very fundamental and long-standing issue, i.e., the *selection of meaningful geometry* for the exploration of both time-independent and time-dependent flow data. More specifically, we study the selection of relevant lines and surfaces by using global optimization-based adjustments of opacity that take domain knowledge into account. In the second part of the thesis, we present two novel techniques for feature-based / topology-based visualization of unsteady flows. We present a technique that allows to render Lagrangian scalar fields, such as finite-time Lyapunov exponent fields, without grid discretization and step size errors during ray marching, which enables the construction of a ground truth. Afterwards, we introduce a new class of vortex concepts that is invariant under uniform equal-speed rotation, i.e., rotation invariant vortices.

Since traditional flow visualization is based on the observation of massless particles, it does not allow to infer the behavior of finite-sized, so-called inertial particles, moving in the flow. Currently, this is a large gap in the visualization literature. Thus, the third part of the thesis is dedicated to the study of *inertial particles*. We will extend known concepts, like integral curves to the mass-dependent case, study finite-time mass separation and establish vortex core line extraction methods that are applicable to inertial particles. Further, we study inertial critical points and asymptotic forward integration behavior, and present a novel approach to the source inversion problem. The study of inertial particles opens a vast field of visualization applications, such as sediment transport, pollution problems or view restriction during landing maneuvers of aircraft. For many problems, it is to expect that we can extend and draw ideas from the large body of flow visualization literature, originally designed for traditional massless flows. We are hopeful that our work encourages more visualization research on inertial particles, which have eluded the visualization community long enough.

1.1. Thesis Structure

This thesis is divided into three parts. The first part resides in geometry-based flow visualization and is dedicated to the selection of lines and surfaces for the display of general flow data.

- Chapter 2 introduces vector fields, reviews the most recent literature on line and surface selection and places our work in the context of geometry-based flow visualization.
- Chapter 3 focuses on the interactive and frame-coherent selection of meaningful lines in 3D, such that the view is cleared on lines that exhibit a high user-defined importance. In this area, our approach is the first that formulates the selection problem as a global optimization-based adjustment of line opacities.
- Chapter 4 targets the yet completely unsolved problem of line selection for animated 3D lines. For this, we present a hierarchical and view-dependent partition of lines to speed up the opacity computation and additionally address time coherence for animated lines.
- Chapter 5 extends the concept of opacity optimization to surfaces, for which an optimization-based opacity adjustment was absent, too. We present an approach to the partition of the surfaces and the final interpolation of opacities across the surface.

The second part of this thesis presents novel feature-based / topology-based techniques for the visualization of unsteady flows.

- Chapter 6 introduces into the computation of finite-time Lyapunov exponents and revisits vortex invariance under certain types of motion.
- Chapter 7 uses unbiased free path sampling to construct ground renderings of finite-time Lyapunov exponent fields that are free of grid discretization and step size errors during ray marching.
- Chapter 8 presents a new class of vortex concepts for the use in rotating devices such as mixers or centrifugal pumps. For those, we introduce a general way to turn Galilean invariant vortex concepts into rotation invariant ones.

In the third part of the thesis, we introduce methods for the analysis of inertial particles in flow fields.

- Chapter 9 presents an overview of the modeling of inertial particle motion and its relation to massless particles.
- Chapter 10 introduces an abstract concept to describe inertial particle trajectories.

Based on this, we define mass-dependent extensions of traditional integral curves, as well as a new class of integral curves that depicts the separation of masses.

- Chapter 11 builds up on the mass-dependent integral curves and presents three coordinated views that assist in the exploration of mass-induced separation behavior.
- Chapter 12 takes a first step toward inertial steady 2D vector field topology by conducting a full classification of inertial critical points and a visualization of the asymptotic flow behavior in forward direction.
- Chapter 13 addresses the source inversion problem, which is useful to recover the origin of dispersed pollutants. We will show that for a given initial velocity all possible origins form a curve, which can be described as tangent curve of a derived vector field.
- Chapter 14 is dedicated to the extraction of the mass-dependent vortex cores of inertial particles. We present a parallel vectors-based technique that extends Sujudi-Haimes and its time-dependent counterpart, i.e., the cores of swirling particle motion. Further, for the extraction of attracting vortices, we apply the well-known particle density estimation.

We conclude the thesis in Chapter 15, where we present an outlook on future work.

1.2. Notation

General Notation

p, E	Scalar value or scalar-valued function
$\mathbf{x}$	Vector (column-wise) in n-D space
$\overline{\mathbf{x}}$	Vector (column-wise) in $(n+1)$-D space-time
$\tilde{\mathbf{x}}$	Vector (column-wise) in $(2n)$-D space-velocity
$\hat{\mathbf{x}}$	Vector (column-wise) in $(2n+1)$-D space-velocity-time
$\mathbf{A}$	Matrix
$\mathcal{A}$	Set
$\mathbf{x}^{\mathrm{T}}$	Transpose of a vector or matrix
$p \cdot \mathbf{x}$	Multiplication by a scalar
$\mathbf{x}^{\mathrm{T}}\mathbf{x}$	Dot product (always in matrix notation)
$\mathrm{d}f/\mathrm{d}t$	Total derivative of a (multivariate) function
$\partial f/\partial t$	Partial derivative of a multivariate function
$\mathbf{v}_1 \parallel \mathbf{v}_2$	Parallel vectors operator

Common Symbols

$\mathbf{u}(\mathbf{x})$	Steady vector field (time-independent)
$\mathbf{u}(\mathbf{x}, t)$	Unsteady vector field (time-dependent)
τ	Integration duration
$\Phi_t^\tau(\mathbf{x})$	Flow map
∇	Nabla operator
$\mathbf{J} = \nabla\mathbf{u}$	Jacobian matrix (first-order partial derivatives)
$\mathbf{0}$	Zero vector (column-wise)
$\mathbf{I}$	Identity matrix
$\mathbf{v}$	Particle velocity
r	Particle response time $[s]$
d_p	Particle diameter $[\mu m]$
ρ_p	Particle density $[kg/m^3]$
ν	Kinematic viscosity $[m^2/s]$
μ	Dynamic viscosity $[kg/(m\,s)]$
$\mathbf{g}$	Gravity vector $[m/s^2]$
λ	Eigenvalue (of Jacobian)
$\mathbf{e}$	Eigenvector (of Jacobian)
$\mathbf{f}$	Feature flow field

Part I.

Line and Surface Selection by Opacity Optimization

2 Chapter 2. Overview on Geometry-based Flow Visualization

Flow visualization is the discipline of visualizing vector field data for means of data presentation, visual exploration and visual analysis. Generally, flow visualization is classified into four classes, which are shortly described below.

Direct Methods. Direct visualization methods are characterized by a depiction of a set of primitives that directly encode properties of the data, e.g., arrow plots pointing in the flow direction. They are suitable for multi-variate data visualization, as multiple attributes can be encoded by glyphs [BKC*13]. While they are widely and successfully applied in 2D, their utility is limited when they are spatially embedded in 3D due to the arising occlusion.

Dense / Texture-based Methods. Texture-based methods, which are sometimes also referred to as dense methods, densely encode data on the flow domain by using textures. They are predominantly used in 2D and on 2D manifolds, again due to the occlusion problems. A famous example is the line integral convolution [CL93]. More details on texture-based methods are found in [LHD*04] and [LEG*08].

Feature-based / Topology-based Methods. In flow visualization, the term *feature* is often reserved for structures that are comprised in vector field topology [HH91]. The essence of these methods is to partition and compactly describe the flow as an ensemble of areas with coherent behavior. More on the extraction and tracking of those feature is compiled in [PVH*03]. More recent reports summarize the efforts in topology-based [LHZP07] and specifically unsteady topology-based [PPF*11] flow

visualization. Features will play a larger role in the second part of this thesis, when we discuss finite-time Lyapunov exponents and vortex concepts.

Geometry-based Methods. Geometry-based methods represent the flow by geometric primitives such as flow aligning lines and surfaces. A profound overview of integral geometry approaches is given by McLoughlin et al. [MLP*10]. Recent reports focused on surface-based [ELC*12a] and illustrative flow visualization [BCP*12].

The first part of this thesis is mainly concerned with geometry-based techniques. For this reason, the next section elaborates on various concepts required in geometry-based flow visualization. Parts of the following text are compiled from our publications [GRT13, GRT14, GSM*14] that the subsequent chapters are based upon.

2.1. Introduction to Geometry-based Flow Visualization

In the following, we formally introduce into vector fields and their properties, consider the equations of particle motion, describe the concept of flow map, and numerical integration schemes. Finally, we use the flow map to define the most common characteristic integral curves and surfaces.

2.1.1. Vector Fields, Particle Motion and the Flow Map

First, we distinguish between *steady* and *unsteady* flows. A steady flow is not changing over time and is formally given as a time-independent vector field $\mathbf{u}(\mathbf{x}) = \mathbf{u}(x, y\,[, z]) : D \to D$, defined in a spatial domain D, with $\mathbf{x} \in D$. Thereby, D is a subset of the n-dimensional ($n = 2, 3$) Euclidean space $\mathbb{E}^n$, i.e., $D \subset \mathbb{E}^n$. Steady flows are used to describe instantaneous fields, such as magnetic fields or generally fields that are not changing over time.

An unsteady flow, on the other hand, varies over time, and thus, it is given as a time-dependent vector field $\mathbf{u}(\mathbf{x}, t) = \mathbf{u}(x, y, [z,]\, t) : D \times T \to D$. Most physical processes are subject to temporal variation, including the motion of air and fluids. By interpreting a time-dependent field as an autonomous system in space-time, we can write it as an $(n+1)$-dimensional steady field $\overline{\mathbf{p}} = \begin{pmatrix} \mathbf{u} \\ 1 \end{pmatrix}$.

Note that we denote vectors in $(n+1)$-D space-time formally by a bar.

Derivatives. We denote the partial derivatives of a time-dependent vector field as:

$$\mathbf{u}_x = \frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial x}\,,\quad \mathbf{u}_y = \frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial y}\,,\quad \left[\mathbf{u}_z = \frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial z}\,,\right]\quad \mathbf{u}_t = \frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial t}\,.$$

Note that for steady flows $\mathbf{u}_t = \mathbf{0}$.

Nabla Operator The Nabla operator ∇ is a symbol that simplifies the notation of several derived differential quantities. It is a vector that contains the partial derivative symbols with respect to the spatial dimensions:

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)^{\mathrm{T}}$$

Divergence. The divergence $\operatorname{div}\mathbf{u}$ of a steady flow $\mathbf{u}(x,\, y\,[,\, z]) = (u,\, v\,[,\, w])^{\mathrm{T}}$ is a scalar field that characterizes the change in volume of a virtual finite-sized sphere that is advected with the flow. The divergence is given as:

$$\operatorname{div}\mathbf{u} = \nabla\cdot\mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\left[+\frac{\partial w}{\partial z}\right] = u_x + v_y\,[+w_z]$$

If the volume increases the divergence is positive; if it decreases the divergence is negative. If the volume remains constant, even though the shape of the sphere might deform, the divergence is zero. If a flow has zero divergence everywhere in the domain, the flow is known to be *divergence-free* or incompressible. The compressibility of a fluid depends on its molecular composition. Tightly packed fluids such as water are often modeled as incompressible. Since gases can easily be packed tighter under pressure, they are highly compressible and not divergence-free.

Curl. The curl $\operatorname{curl}\mathbf{u}$ of a steady 3D vector field $\mathbf{u}(x,\, y,\, z) = (u,\, v,\, w)^{\mathrm{T}}$ is a vector field that indicates how the flow swirls at a certain point. It is defined as:

$$\operatorname{curl}\mathbf{u} = \nabla\times\mathbf{u} = \begin{pmatrix} w_y - v_z \\ u_z - w_x \\ v_x - u_y \end{pmatrix}$$

When a virtual finite-sized particle is advected with the flow, it might spin. The axis around which it spins is parallel to the curl vector and the curl's magnitude corresponds to the angular speed of the rotation. A vector field in which the curl is the zero vector everywhere in the domain is called *curl-free* or irrotational.

Jacobian. The spatial Jacobian $\mathbf{J} = \nabla\mathbf{u} = (\mathbf{u}_x, \mathbf{u}_y[, \mathbf{u}_z])$ is an $n \times n$ matrix that contains a first-order description of how the flow behaves locally around a given location. The space-time Jacobian is an $(n+1)\times(n+1)$ matrix that additionally contains the temporal derivative. It is defined as $\overline{\mathbf{J}} = \nabla\overline{\mathbf{p}} = \begin{pmatrix} \mathbf{J} & \mathbf{u}_t \\ \mathbf{0}^{\mathrm{T}} & 0 \end{pmatrix}$.

The eigenvalues and eigenvectors of the Jacobian are of great interest to characterize isolated first-order critical points [HH89, HH91, Wei08], i.e., locations at which $\mathbf{u}(\mathbf{x}) = \mathbf{0}$ and $\mathbf{u}(\mathbf{x}+\epsilon) \neq \mathbf{0}$. We cover critical points in detail later in Section 9.3.1. We denote the eigenvectors of $\mathbf{J}$ as $\mathbf{e}_1, \mathbf{e}_2[, \mathbf{e}_3]$, respectively. For $n = 3$ and the case that two eigenvalues are complex, the eigenvector to the only real eigenvalue is $\mathbf{e}$. The eigenvectors will play a greater role later for the definition of vortex concepts.

Acceleration. The acceleration of a particle in space is $\mathbf{a} = \mathbf{J}\mathbf{u} + \mathbf{u}_t$, and the acceleration in space-time is $\overline{\mathbf{a}} = \overline{\mathbf{J}}\ \overline{\mathbf{p}}$. Note that $\overline{\mathbf{a}} = \begin{pmatrix} \mathbf{a} \\ 0 \end{pmatrix}$.

Feature Flow Field. The feature flow field is a derived vector field that was originally introduced to track critical points in space-time [TS03]. Later, we will use it for general vortex tracking. We define it in space-time as $\overline{\mathbf{f}}$ and in space as $\mathbf{f}$ by division by its last component. For $n = 2$ we have [TS03]

$$\overline{\mathbf{f}} = \begin{pmatrix} \det(\mathbf{u}_y, \mathbf{u}_t) \\ \det(\mathbf{u}_t, \mathbf{u}_x) \\ \det(\mathbf{u}_x, \mathbf{u}_y) \end{pmatrix}, \quad \mathbf{f} = \frac{1}{\det(\mathbf{u}_x, \mathbf{u}_y)} \begin{pmatrix} \det(\mathbf{u}_y, \mathbf{u}_t) \\ \det(\mathbf{u}_t, \mathbf{u}_x) \end{pmatrix}. \tag{2.1}$$

For $n = 3$ we have [WSTH07]

$$\overline{\mathbf{f}} = \begin{pmatrix} -\det(\mathbf{u}_y, \mathbf{u}_z, \mathbf{u}_t) \\ \det(\mathbf{u}_z, \mathbf{u}_t, \mathbf{u}_x) \\ -\det(\mathbf{u}_t, \mathbf{u}_x, \mathbf{u}_y) \\ \det(\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z) \end{pmatrix}, \quad \mathbf{f} = \frac{1}{\det(\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z)} \begin{pmatrix} -\det(\mathbf{u}_y, \mathbf{u}_z, \mathbf{u}_t) \\ \det(\mathbf{u}_z, \mathbf{u}_t, \mathbf{u}_x) \\ -\det(\mathbf{u}_t, \mathbf{u}_x, \mathbf{u}_y) \end{pmatrix}. \tag{2.2}$$

Note that $\mathbf{f}$ can only be computed if $\mathbf{J}$ is non-singular. Also note that $\overline{\mathbf{J}}$ has the eigenvectors $\begin{pmatrix} \mathbf{e}_1 \\ 0 \end{pmatrix}, \begin{pmatrix} \mathbf{e}_2 \\ 0 \end{pmatrix}, \left[\begin{pmatrix} \mathbf{e}_3 \\ 0 \end{pmatrix}, \right] \overline{\mathbf{f}}$, where $\overline{\mathbf{f}}$ has the corresponding eigenvalue 0. There is a simple relation

$$\mathbf{u} - \mathbf{f} = \mathbf{J}^{-1}\,\mathbf{a} \tag{2.3}$$

and a simpler definition of the feature flow field as

$$\mathbf{f} = -\mathbf{J}^{-1}\,\mathbf{u}_t \tag{2.4}$$

that follow both directly from the definitions of $\mathbf{J}$, $\mathbf{a}$ and $\mathbf{f}$, respectively[1].

Particle Motion. The heart of geometry-based flow visualization techniques is the observation of massless particles, which are often referred to as *tracer particles*. The location of a particle $\mathbf{x}(\tau)$, seeded in a *steady* flow at $\mathbf{x}_0$ and observed after integration time τ, is the solution of the autonomous ordinary differential equation (ODE):

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbf{x}(\tau) = \mathbf{u}(\mathbf{x}(\tau)) \quad \text{with} \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{2.6}$$

The location of a massless particle in an *unsteady* flow $\mathbf{u}(\mathbf{x}(t), t)$ is governed by:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \mathbf{u}(\mathbf{x}(t), t) \quad \text{with} \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{2.7}$$

The ODE in (2.7) can be rendered autonomous by modeling time as a state variable:

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\begin{pmatrix}\mathbf{x}\\ t\end{pmatrix}(\tau) = \begin{pmatrix}\mathbf{u}(\mathbf{x},t)\\ 1\end{pmatrix} \quad \text{with} \quad \begin{pmatrix}\mathbf{x}\\ t\end{pmatrix}(0) = \begin{pmatrix}\mathbf{x}_0\\ t_0\end{pmatrix}. \tag{2.8}$$

Flow Map. An abbreviated notation is the so-called *flow map* $\phi : D \to D$, whereby $\phi_t^\tau(\mathbf{x})$ describes the location where a particle seeded at $(\mathbf{x}, t)$ moves to during the integration in a vector field $\mathbf{u}$ over a time interval τ.

2.1.2. Numerical Integration

In the following, we explain how autonomous ODEs are solved by the example of particle tracing in unsteady flow. In some cases, autonomous ODEs can be solved analytically, e.g., if we are interested in the particle trajectories in linear vector fields [BDH92]. In most cases, an analytic solution is not available and we must rely on numerical integration techniques. Given an initial condition to start with, here, the seed point $(\mathbf{x}_0, t_0)$ of a particle, a numerical integrator repeatedly estimates the next location of the particle and advances it to sweep out the particle trajectory.

The accuracy of numerical integrators is a function of the step size h. During a numerical integration the error accumulates, which may quickly lead to quite inaccurate

[1] The relation (2.3) can be shown by rearranging:

$$\mathbf{J}(\mathbf{u} - \mathbf{f}) = \mathbf{J}\mathbf{u} - \mathbf{J}\mathbf{f} = \mathbf{J}\mathbf{u} + \mathbf{u}_t = \mathbf{a}\,. \tag{2.5}$$

With the fact that $\bar{\mathbf{f}}$ has eigenvalue 0, we have $\overline{\mathbf{J}\mathbf{f}} = \mathbf{0} \Leftrightarrow \bar{\mathbf{J}}\begin{pmatrix}\mathbf{f}\\ 1\end{pmatrix} = \mathbf{0} \Leftrightarrow \mathbf{J}\mathbf{f} + \mathbf{u}_t = \mathbf{0}$, which also leads to (2.4). The relationship in Eq. (2.3) was shown in [Sah09] as part of the proof of Galilean invariance of the cores of swirling particle motion [WSTH07], and goes back to Ronald Peikert.

Input: Initial particle position $\mathbf{x}_0$, start time t_0, integration duration τ, step size h
Output: Reached particle location $\mathbf{x}$ at time $t = t_0 + \tau$

$\mathbf{x} \leftarrow \mathbf{x}_0$, $t \leftarrow t_0$; // initialize current state
while $t + h < t_0 + \tau$ **do**
  $(\mathbf{x}, t) \leftarrow$ integrateStep($\mathbf{x}$, t, $\mathbf{u}$, h) ; // repeatedly take step, Eqs. (2.15) and (2.16)
end
$(\mathbf{x}, t) \leftarrow$ integrateStep($\mathbf{x}$, t, $\mathbf{u}$, $t_0 + \tau - t$) ; // final step to reach time $t_0 + \tau$ exactly

Algorithm 1: Implementation of the flow map $\phi_t^\tau(\mathbf{x})$ in a vector field $\mathbf{u}(\mathbf{x}, t)$.

results. The choice of the step size is therefore an important trade-off between higher numerical accuracy and faster computations. The easiest, fastest and unfortunately most inaccurate method is the *explicit Euler* method:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h \cdot \mathbf{u}(\mathbf{x}_i, t_i) \tag{2.9}$$

$$t_{i+1} = t_i + h\,. \tag{2.10}$$

It is sometimes used for explanatory brevity, but is not recommended for practical use. The majority of the results presented in this thesis is obtained with the *fourth-order Runge-Kutta* method [BDH92], which averages four slopes in the step size interval h to obtain a more accurate prediction of the next particle location. Computing the four slopes requires four vector field samples:

$$\mathbf{k}_1 = \mathbf{u}(\mathbf{x}_i, t_i) \tag{2.11}$$

$$\mathbf{k}_2 = \mathbf{u}(\mathbf{x}_i + \frac{h}{2}\mathbf{k}_1, t_i + \frac{h}{2}) \tag{2.12}$$

$$\mathbf{k}_3 = \mathbf{u}(\mathbf{x}_i + \frac{h}{2}\mathbf{k}_2, t_i + \frac{h}{2}) \tag{2.13}$$

$$\mathbf{k}_4 = \mathbf{u}(\mathbf{x}_i + h\,\mathbf{k}_3, t_i + h)\,. \tag{2.14}$$

Finally, the slopes are averaged with greater weight given at the midpoint:

$$\mathbf{x}_{i+1} = \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \tag{2.15}$$

$$t_{i+1} = t_i + h\,. \tag{2.16}$$

The implementation of the flow map is listed in Alg. 1 as a repeated application of the integrator.

Explicit methods require a fixed step size to be set in advance. On the one hand, this requires to set a conservatively small step size, which leads to increased computational cost. On the other hand, a fixed step size can also be an advantage if massive sets of particles are integrated simultaneously, since then all particles share the current time state and require the same vector field time slices to be in memory, which reduces I/O

traffic, increases caching performance and eases the implementation. Alternatively, adaptive methods are available that adjust the step size to stay below a certain error bound in each step (the error still accumulates). In its essence they compare two integrators with different order and reduce the step size if their results differ too strongly. Adaptive methods naturally allow better error-control and are faster compared to conservatively set fixed step sizes. Famous examples are the Heun-Euler method (order 1 and 2), and the Runge-Kutta-Fehlberg method (order 4 and 5) [HNW93].

Integrators require to sample a vector field at a given location. Depending on the discretization of the vector field, the cell that contains the queried sample is easy to find or requires an acceleration data structure for efficiency. In regular grids, the cell is easily determined analytically. Rectilinear grids require a binary search, and unstructured grids typical involve a search in a balanced kd-tree. Cell locations in spatial trees can typically be accelerated by starting the search at the cell of the previous query. Ueng et al. [USM96] and Garth and Joy [GJ10] have shown how cell locations are efficiently carried out in unstructured grids.

2.1.3. Particle-based Visualization

Most flow visualization techniques rely on virtual massless particles moving with the underlying flow field. A system for the interactive visualization of large particle sets, exploiting graphics hardware capabilities, is presented by Krüger et al. [KKKW05] and another system to explore multi-variate flow properties is presented by Fraedrich and Westermann [FW12]. Further GPU realizations for massive particle advection in steady and time-dependent vector fields have been presented by Peterka et al. [PRN*11], Kipfer et al. [KSW04] and Hlawatsch et al. [HSW11]. A recent particle advection benchmark was performed by Childs et al. [CBP*14] for several architectures. Staib et al. [SGG15] demonstrated the utility of transparency and ambient occlusion for the visualization of particle data.

The advection of additional flow field properties along particle trajectories, further allows to derive insight into the dynamics of those properties. Li et al. [LTH08] propose a physically oriented dye advection scheme using mass conservation, which was later improved by Karch et al. [KSW*12]. Recently, Sadlo [Sad15] introduced the Lyapunov time, which is the time over which a particle trajectory is predictable (i.e., not dominated by error). This consideration allows to incorporate uncertainty into integration-based visualizations. Similarly, McLoughlin et al. [MET*15] visualize the impact of a small modification of a seed point.

2.1.4. Integral Curves

In general, direct particle visualization does only allow for limited insight into the dynamic motion behavior. Hence, a common approach is the usage of integral geometry, based on the trajectories of moving particles, to visualize their long-term behavior and structural patterns therein.

The integration of particles in vector fields leads to different kinds of curves and trajectories, which are generally summarized as *integral curves*. It was shown that all of the classic kinds of integral curves are *tangent curves* of special vector fields [WT10a], [WHT12].

A curve $\mathbf{x}(\tau)$ is called *tangent curve* of a vector field $\mathbf{u}$, if the tangent of the curve coincides everywhere with $\mathbf{u}(\mathbf{x})$:

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbf{x}(\tau) = \mathbf{u}(\mathbf{x}(\tau)). \tag{2.17}$$

Such curves follow the path of a massless particle, which makes them meaningful tools for the visualization of flow data. For massless observations in time-dependent (unsteady) flows $\mathbf{u}(\mathbf{x}, t)$, four types of integral curves are established, namely streamlines, pathlines, streaklines and timelines, which we describe in the following.

Streamlines. The path of a massless particle in steady flow can be computed from Eq. (2.6). For time-independent (steady) vector fields $\mathbf{u}(\mathbf{x})$ such tangent curve is called *streamline*. For every point, there is just one curve passing through, making the map between particle trajectory and vector field bijective.

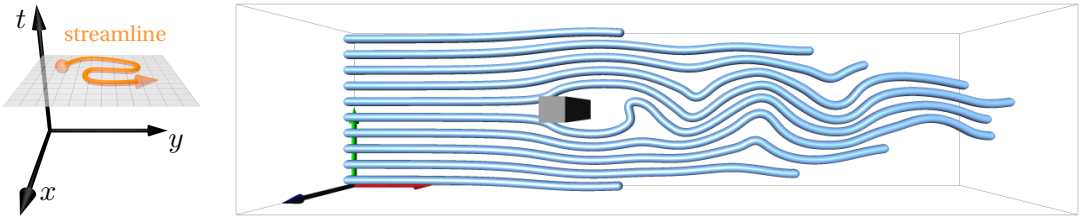
Similar to the steady case, *streamlines* are acquired in unsteady flow by integrating a particle trajectory at a constant time t_0. Thus, streamlines of an unsteady vector field $\mathbf{u}(\mathbf{x}, t)$ are tangent curves of the $(n+1)$-dimensional vector field $\bar{\mathbf{s}}(\mathbf{x}, t)$:

$$\bar{\mathbf{s}}(\mathbf{x}, t) = \begin{pmatrix} \mathbf{u}(\mathbf{x}, t) \\ 0 \end{pmatrix}. \tag{2.18}$$

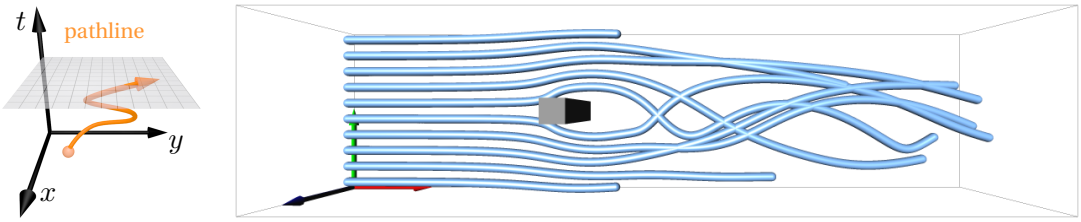
An example for streamlines in unsteady data are the instantaneous field lines of unsteady magnetic fields. Fig. 2.1a shows streamlines in the flow around an obstacle.

Pathlines. In unsteady flow, the trajectory of a massless particle is called *pathline*. It is the solution of Eq. (2.8) and therefore arises as tangent curve of the $(n+1)$-dimensional vector field

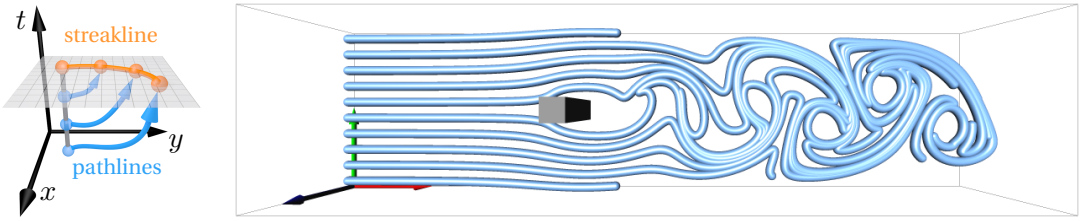
$$\bar{\mathbf{p}}(\mathbf{x}, t) = \begin{pmatrix} \mathbf{u}(\mathbf{x}, t) \\ 1 \end{pmatrix}. \tag{2.19}$$



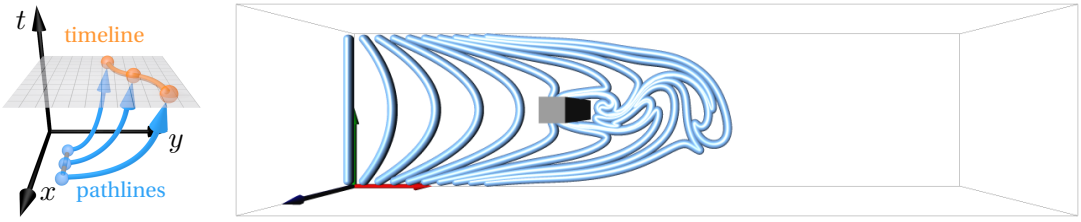
(a) **Streamlines**: Trajectories in a single time slice, e.g., magnetic field lines.



(b) **Pathlines**: Trajectories of tracer particles, advancing in time.



(c) **Streaklines**: Trail of smoke from a blown out (point-sized) candle.



(d) **Timelines**: Particle front that advances over time.

Figure 2.1.: Space-time illustrations and examples of integral curves in unsteady flow.

In terms of the flow map, it is simply expressed as

$$\mathbf{p}(\tau) = \phi_t^\tau(\mathbf{x}). \tag{2.20}$$

Here, an infinite number of pathlines passes through each point in the spatial domain, depending on the seed time of the curve, which makes the visualization of pathlines challenging. Fig. 2.1b gives an example.

Streaklines. A *streakline* arises by the continuous injection of particles at a location $\mathbf{x}$. The advected trail of particles forms a continuous curve that visualizes temporal coherence, i.e., the change of the flow at a certain location over time. A real-world

example is smoke, injected at a single point. Streaklines are expressed by:

$$\mathbf{s}(\tau) = \phi_{\tau}^{t-\tau}(\mathbf{x}). \tag{2.21}$$

In Fig. 2.1c, streaklines are released in the flow around an obstacle, revealing the characteristic von Kármán vortex street. Weinkauf and Theisel [WT10a] have shown how these curves can be computed as tangent curves of a higher-dimensional derived vector field. Varying the injection location over time results in the *generalized streaklines* in Wiebel et al. [WTS*07], which are useful to visualize the flow separation at boundaries of an object immersed in the flow.

Timelines. Injecting particles on a seed curve $\mathbf{c}(u)$ simultaneously and tracing them over time forms a *timeline*, as shown in Fig. 2.1d. These lines show the spatial coherence between particles that were seeded at the same time along a curve. Using the flow map they are defined as

$$\mathbf{t}(u) = \phi_{t}^{\tau}(\mathbf{c}(u)). \tag{2.22}$$

Weinkauf et al. [WHT12] derived these curves as tangent curves of a special higher-dimensional vector field.

Extraction of Integral Curves

Once defined as a tangent curve, integral curves can be computed by integrating the particle trajectory in the defining vector field numerically step-by-step. In practice, however, streaklines and timelines are most often computed by refinement of an advancing front line. Thereby, the line is represented by a discrete set of particles that is advected with the flow. Whenever adjacent particles get very close they can be merged, and when they move too far apart, a new particle is inserted between them. Ideally, the new particle is inserted on the seed curve at the seed time and is integrated up to the current time. This refinement strategy is computationally expensive, especially in out-of-core settings when not all time steps reside in memory, and would have to be streamed through once more just to insert a couple of particles (possibly after every integration step). Further, the selection of the seed point on the seed curve is only done heuristically by assuming that the mid-point on the seed curve partitions the front line well. This assumption might not hold well near saddle-like structures. For this reason, a practical heuristic is to insert the new particle at the center of the front line edge that needs refinement. While this introduces a small error, this is the most commonly used approach, especially when GPUs are involved. An improvement of the refinement criterion was proposed by Wiebel et al. [WWSS10] in order to reduce the number of particles, required to adequately represent the line. For this, they estimate

the accuracy of the polyline representation not only by the particle distance, but also by the angle and triangle area spanned between adjacent polyline segments.

Rendering of Integral Curves

Lines are basic geometric primitives that are directly supported by rasterization hardware in form of line lists or strips. An anti-aliased and fast display of large line sets is therefore technically not a problem. When it comes to their visualization, however, several issues arise, as they do not have a unique normal and no spatial extent that conveys depth information. Zöckler et al. [ZSH96] applied the Phong illumination model to streamlines by selecting a normal in the plane spanned by the tangent and light direction. Everts et al. [EBRI09] proposed a halo visualization for view-oriented line strips, in which the halo is folded slightly toward the background. Consequentially, the halos stay hidden if lines are close to each other, which creates an impression of line bundling. In later work, Everts et al. [EBRI11, EBRI15] further investigated different line styles. Aside from line strips, textured stream tubes have been proposed by Stoll et al. [SGS05]. Eichelbaum et al. [EHS13] added an ambient occlusion effect to convey a depth impression that improves the perception of nearby spatial relations. Further approaches to improve the depth perception were shown by Lawonn et al. [LLPH15], who redundantly encoded depth by (quantized) color and a hatching on the lines. We refer to Brambilla et al. [BCP*12] for a broader overview of the topic.

2.1.5. Integral Surfaces

The definitions of integral curves naturally extend to surfaces by parameterizing the seeding structure of the integral curves. For stream surfaces, path surfaces and streak surfaces, integral curves are not released from a single point but from a *seed curve* $\mathbf{c}(s)$ with $s \in [s_0, s_1]$. A time surface is released from a seed surface $\mathbf{c}(s,u)$ with an additional parameter $u \in [u_0, u_1]$.

Formally, a stream surface $\mathbf{x}(s,\tau)$ is the union of all streamlines released from the seed curve:

$$\frac{\partial \mathbf{x}(s,\tau)}{\partial \tau} = \mathbf{u}(\mathbf{x}(s,\tau)) \quad \text{with} \quad \mathbf{x}(s,0) = \mathbf{c}(s). \tag{2.23}$$

Isoparametric lines $s = \text{const}$ are *streamlines*. Examples are shown in Fig. 2.2.

Similarly, a path surface is obtained by releasing pathlines in unsteady flow from a seed curve:

$$\frac{\partial \mathbf{p}(s,t)}{\partial t} = \mathbf{u}(\mathbf{p}(s,t),t) \quad \text{with} \quad \mathbf{p}(s,0) = \mathbf{c}(s). \tag{2.24}$$

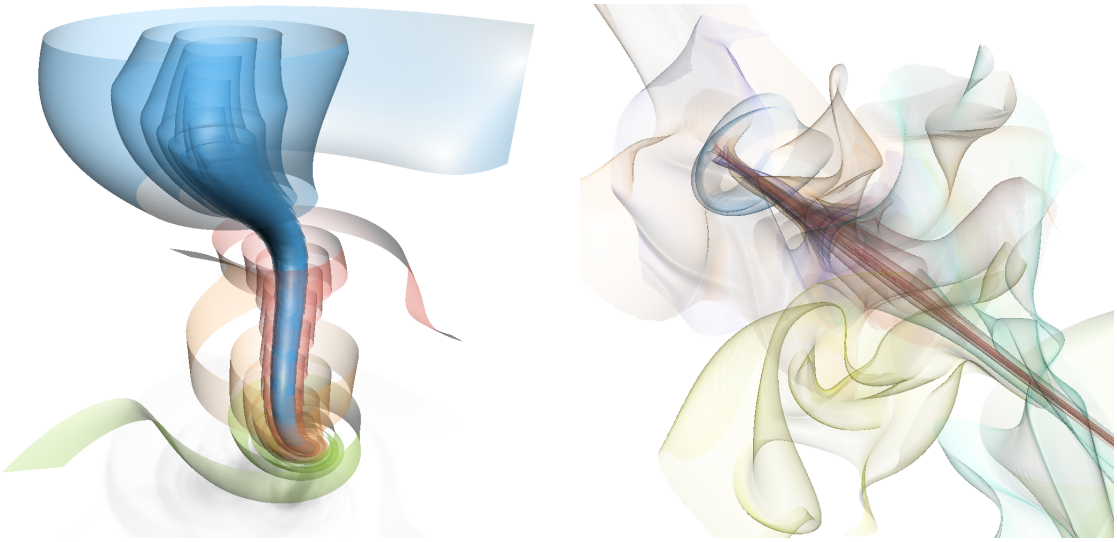


Figure 2.2.: Examples of integral surfaces in 3D vector fields.

Here, isoparametric lines $s = \text{const}$ are *pathlines* and isoparametric lines $t = \text{const}$ are timelines.

Path surfaces, streak surfaces and time surfaces can be expressed by the flow map as:

$$\mathbf{p}(s,\tau) = \phi_t^\tau(\mathbf{c}(s)) \tag{2.25}$$

$$\mathbf{s}(s,\tau) = \phi_\tau^{t-\tau}(\mathbf{c}(s)) \tag{2.26}$$

$$\mathbf{t}(s,u) = \phi_t^\tau(\mathbf{c}(s,u)). \tag{2.27}$$

Extraction of Integral Surfaces

Stream and path surfaces can be extracted by advancing front algorithms. Thereby, an active particle front line is advected with the flow, starting at the seed curve. Because of potential growing and shrinking of the front line, the particle front is adaptively refined. The refinement is similar to the refinement of a streakline or timeline. All locations that the front line sweeps over form the integral surface. The surface can be tessellated by creating triangles between the current front line and its previous position.

The tessellation of the surface can become quite difficult as the front line might undergo shearing, which leads to thin and elongated triangles. Schulze et al. [SGRT12] called this the *small angle problem*, as the angle between the isoparametric lines becomes small. Further, critical points can cause problems, as for instance at a saddle a front line could infinitely grow. Therefore, some surface extraction algorithms employ a special handling of these cases. The seminal work of Hultquist [Hul92] describes a quite robust approach that advances the front particles adaptively in a recursive

manner. The approach is greedy and implicitly tends to front lines that are orthogonal to the flow. The recursive nature of the algorithm makes it hard to parallelize the approach on GPUs.

The approach was later extended by Stalling [Sta98] to better handle saddles, and eventually by Schneider et al. [SRWS10] with a full topology-aware surface extractor. Garth et al. [GTS*04] modified Hultquist's method by making the streamline integration based on arc length in order to better handle flow of inhomogeneous magnitude. In addition, they added another refinement criterion that considers the curvature of the front in order to better handle flows in the vicinity of vortices. In a later work, Garth et al. [GKT*08] focused on the avoidance of the recursion. For this, they split the surface computation into two steps: a parallel timeline advancing front integration to obtain a rough skeleton and a subsequent adaptive meshing. The resulting method is easier parallelized, but it cannot handle the small angle problem, since the front line tessellation is local. For this, Schulze et al. [SGRT12] recognized the need for a global front line tessellation. They proposed an adaptive optimization of the step size of the individual front line particles so that after each integration step flow orthogonal front lines are obtained.

A higher order tessellation method for smooth surfaces was presented in Schneider et al. [SWS09]. Schafhitzel et al. [STWE07] presented a particle-based approach that is GPU-accelerated and completely avoids the necessity of a triangulation. They proposed a method to maintain a uniform sampling of the surface and then splatted individual particles. Scheuermann et al. [SBH*01] proposed solutions for analytic surfaces in tetrahedral cells, given linear seed curve segments. Recently, Lu et al. [LSP14] presented a workload balancing approach for a scalable and parallelized integration of stream surfaces across many processors.

The work of Weinkauf et al. [WT10a, WHT12] made it possible to integrate streak surfaces and time surfaces by the aforementioned advancing front algorithms, as they described streaklines and timelines as tangent curves of higher dimensional derived vector fields. When extracted on the original vector field, streak surfaces and time surfaces must be remeshed after every integration step, since not only the front line but the whole surface changes. For this, Bürger et al. [BFTW09] proposed two realtime approaches on the GPU. The first is patch-based for which individual patches are advected and refined. This method has the advantage that required memory for the added patches can be pulled from a global memory pool. However, T-junctions arise between the patches, which have to be handled by blending. The second approach is mesh-based and handles converging and diverging flows with potentially inhomogeneous magnitude. Here, the refinement along a timeline is limited up to a pre-defined upper bound. Additional timelines can, however, be added by pulling data from a

global memory pool. These methods cannot handle shear flows, i.e., the small angle problem is not considered. For this, Krishnan et al. [KGJ09] introduced a triangle-based approach that performs edge flips, edge collapse and edge split in order to improve the mesh quality. Their approach is implemented on the CPU, same as the quad-based approach of McLoughlin et al. [MLZ10]. More recently, Machado et al. [MSE14] proposed an image-based stream surface rendering approach that adaptively generates dense streamlines and thereby avoids the necessity of an intermediate triangulation for rendering. Biswas and Shen [BS13] proposed a number of metrics for the evaluation and verification of stream surfaces, in terms of tessellation and integration accuracy.

Rendering of Integral Surfaces

Surface rendering has a long history in computer graphics. From a visualization point of view, we will look in the following on methods to convey shape information, avoid occlusions and finally review some recent work on surface-based flow visualization.

Conveying Shape. Comprehensible rendering of 3D shapes has a long tradition and was pioneered by Saito and Takahashi [ST90], who extracted discontinuities, edges and contour lines in screen space. Interrante et al. [IFP96] placed curvature-oriented strokes on transparent iso-surfaces. Nienhaus and Döllner [ND04] rendered technical blueprints by peeling the layers of complex compound objects and extracted edges per layer. Luft et al. [LCD06] amplified high frequencies in the depth buffer by unsharp masking to emphasize depth discontinuities. Beside silhouettes, various classes of surface curves were suggested to enhance shape understanding like suggestive contours by DeCarlo et al. [DFRS03], apparent ridges by Judd et al. [JDA07] and Laplacian lines by Zhang et al. [ZHXC09]. To improve the perception of *transparent* surfaces, Wang et al. [WGM*08] proposed an interactive tool for color design in illustrative visualizations that respects intermixing of colors of transparent objects. Vergne et al. [VPB*09] warped environment light around surface features to enhance shape perception.

Occlusion Avoidance. Occlusion is typically avoided by transparency. Viola and Gröller [VG05] compiled a survey on smart visibility in visualization, including cutaways and ghosting, which are mainly utilized in direct volume rendering. There, Viola et al. [VKG04] adapted opacity dependent on importance. Bruckner and Gröller [BG07] improved on the perception of spatial relations by using volumetric halos. Chan et al. [CWM*09] adjusted opacity based on psychological principles, whereas Correa and Ma [CM11] used visibility histograms to assist in transfer function design. For surfaces, Diepstraten et al. [DWE02] addressed transparency in technical illustrations by cap-

turing styles and choices of technical artists. There exist other conceptually different ways to avoid occlusion like exploded views [APH*03] or interactive spatial separation by deformation [CSC07].

Flow Visualization. In flow visualization, surfaces are used to visualize complex data [BCP*12], which has a long tradition as can be observed for the manually crafted illustrations in text books [AS84]. The rendering of integral surfaces [ELC*12a], as a field of geometric flow visualization [MLP*10], was actively studied in recent years. To avoid occlusions Born et al. [BWF*10] interactively selected cuts and surface slabs, which were rendered by halftoning with additional line contours. Spencer et al. [SLCZ09] suggested the texturing of evenly-spaced streamlines on surfaces. Carnecky et al. [CSFP12] computed line integral convolution on transparent surfaces by an anisotropic diffusion of spot noise. Hummel et al. [HGH*10] enriched the surfaces by adaptive view-dependent line textures that represent tangent curves and timelines. Moreover, they proposed two transparency mappings that accentuate grazing angles. We briefly recall their mapping approaches as we compare them later on with our method.

Angle-based transparency maps the dot product between normal and eye vector to transparency:

$$\alpha_{\text{angle}} = \frac{2}{\pi} \arccos(\mathbf{n}^{\mathrm{T}}\mathbf{u}) \,. \tag{2.28}$$

Normal variation considers screen space partial derivatives $\partial/\partial u$ and $\partial/\partial v$ of the view space normal's z-component:

$$\alpha_{\text{normal}} = \left((\frac{\partial n_z}{\partial u})^2 \, + \, (\frac{\partial n_z}{\partial v})^2 \right)^{\gamma/2} \,. \tag{2.29}$$

The angle-based approach was used in Fig. 2.2.

Carnecky et al. [CFM*13] addressed the perception of the layer order of transparent surfaces by a diffusion of silhouettes and halos that makes surface crossings distinguishable. We make use of this technique later on.

2.2. Challenges in Line and Surface Selection

The following sections lay the foundation for the subsequent chapters by motivating the problems to solve and by summarizing the related work.

Typically, the path to a meaningful geometry-based flow visualization is laid out in three steps: integral geometry has to be *extracted* (integrated), *selected*, and *rendered*. In the previous section, we discussed extraction and rendering. The first part of this

thesis is dedicated to the *selection problem*. To provide a notion for the complexity of the selection problem, we first introduce the concept of line fields.

2.2.1. Steady Line Fields

Line fields consist of families of 3D curves that cover (part of) a 3D domain *densely*. This means that for every point in a 3D domain there is exactly one line passing through it. Examples are streamlines (of velocity vector fields), tensor lines or fiber bundles of DT-MRI data (medical imaging), magnetic field lines (physics) and speedlines depicting motion (computer graphics). The main challenge in rendering line fields is *line selection*: from the potentially infinite set of possible lines, a set of representatives has to be selected for rendering, and this selection should visually convey the main features of the data. On the one hand, displaying too many lines results in cluttered renderings where important features may be hidden. On the other hand, displaying too few or the wrong lines may also lead to missing features due to undersampling of the interesting regions.

Line selection for line fields was intensively studied, mainly in the field of flow visualization. So far, all existing methods use a local or greedy approach: a suitable line is found either by locally searching for a good seeding point for a line integration, using a greedy algorithm that repeatedly inserts new lines, or by computing local importance measures for a finite set of pre-selected lines. Furthermore, none of the existing approaches is readily applicable to a free navigation in a scene: existing methods depict lines to generate illustrations for a distant viewpoint, and they do not handle the massive occlusion that can be introduced by even a single line very close to the camera. (Lines are usually expanded to ribbons or tubes to provide depth cues, thus they typically cover more screen space when moving close to them). Chapter 3 presents the first *global* optimization-based approach to the line selection problem. Given a user-defined importance measure that provides a notion for relevant structures, our method trades the representation of information for occlusion avoidance.

2.2.2. Unsteady Line Fields

In many applications, not one single line field but sets of line fields occur, meaning that more than one line passes through a point in 3D. These sets of line fields depend on one or multiple parameters. Examples are pathlines and streaklines of time-dependent flow, as well as sequences of instantaneous magnetic field lines.

Chapter 4 presents the first global approach to line selection in sets of line fields. In particular, this is the first work in flow visualization presenting a global line selection

algorithm for 3D pathlines and streaklines. For this, we extend the work from Chapter 3 for which two problems have to be solved:

- Sets of line fields consist of significantly more lines than a single field. Since view-dependent line selection is an interactive process, our approach for single line fields has to be extended and accelerated to cope with larger amounts of line segments.
- Parametric sets of line fields can be naturally visualized by an animation, i.e., a time sequence of line fields. In this case, time coherence has to be ensured.

This work presents solutions to both problems. We solve the first problem using a hierarchical representation of the lines to make the problem size of the numerical optimization view-dependent. The second problem is addressed by an additional temporal smoothness term in the underlying energy minimization together with a propagation of opacity values to lines in adjacent time steps.

2.2.3. Steady Surfaces

Stream surfaces are another standard tool to visualize steady vector fields. Unfortunately, their use is often limited even for simple data sets: the user is interested in understanding a volume domain that is covered by surfaces that are typically complex and experience a high degree of occlusion. The perception of surfaces, however, is limited in the presence of occlusion to which the use of transparency is a common approach. For the rendering of transparent surfaces, visual cues such as opacity at grazing angles or silhouettes are key to convey shape. Still, multiple surfaces or even one single complex surface can result in heavily occluding cues. This may make it difficult to faithfully capture shape information from an image. In fact, this may even be impossible if occlusion hinders the view on the relevant structures.

As a consequence, it becomes difficult for the user to discover and to explore the relevant structures, e.g., vortex cores, and hence to understand the data set. Our goal is to facilitate the interactive rendering of complex surfaces with a high degree of (self-) occlusion such that important surface features and cues are visually communicated. Thus in Chapter 5, we pick up the general idea of Chapter 3 and extend our global opacity optimization to the rendering of transparent surfaces. For this, our method balances occlusion avoidance and an importance-driven representation of relevant structures to provide a clear view with visual cues supporting an intuitive comprehension of the visualized shapes.

2.3. Related Work

The rendering of integral geometry, such as streamlines, pathlines, streaklines or their corresponding surface extensions, is a fundamental and well-established tool in geometric flow visualization [MLP*10] and an active area of research [BCP*12]. A particular problem is the placement and selection of the geometry to display. This section reviews previous algorithmic solutions to this problem for both steady and unsteady vector fields.

2.3.1. Line Selection in Steady Vector Fields

Much work went into the placement of streamlines in 2D domains or 2D manifolds [TB96, JL97, VKP00, JL01, MAD05, LMG06, LHS08, SLCZ09, RPP*09]. The focus of these approaches was mainly on finding seeds to cover the domain densely with streamlines (evenly-spacing or according to a density field). Extensions to 3D have been developed by guiding the seeding by density-based [MTHG03, SHH*07], feature-based [YKP05, YWSC12], or similarity-based measures [CCK07, MJL*13].

In 3D, however, the maximization of perceivable information is not only a matter of evenly sampling the domain, but also of avoiding occlusions: it becomes a view-dependent problem. Li and Shen [LS07] applied the iterative 2D seeding strategy of Jobard and Lefer [JL97] to produce evenly-spaced streamlines by an image-space approach. Thereby, the depth of the lines is acquired by reprojection. Annen et al. [ATR*08] proposed a seeding algorithm, inspired by non-photorealistic rendering techniques. Hereby, the termination of streamline integration depends on local measures, such that selected lines have the most similar behavior to contours on surfaces. Xu et al. [XLS10] used an information-theoretic approach to measure the information conveyed by a given set of streamlines. The disparity to the original field's entropy guides the seeding of streamlines to add details where needed. Lines are faded out to reduce occlusion by mapping the scalar entropy field to transparency.

Selection-based approaches do not search for seed points for line integration but instead select lines from a precomputed set. This does not only speed up the search, it also makes frame coherence (i.e., avoidance of popping artifacts) easier to achieve and is therefore a good choice when aiming at interactive navigation. Marchesin et al. [MCHM10] joined lines from a precomputed set with additionally generated lines, depending on the on-screen footprint of the already chosen lines (accumulated in a so-called occupancy buffer), as well as local properties per line. Such local properties again originate in information theory and are the linear entropy [FI08] (i.e., variation of velocity along a line) and the angular entropy [MCHM10] (i.e., variation of a line's

direction). So far, these approaches are neither interactive nor frame coherent. We introduced another local, but view-dependent measure in Günther et al. [GBWT11], where we mapped the number of visible pixels to transparency to fade out lines with only minor contribution. Since this approach tends to favor lines closer to the camera it cannot remove lines covering up the viewport. Ma et al. [MWS13] combined lines from a view-independent and view-dependent set by considering coherence between local views and the last frame. For filling the viewport they compute occupancy [MCHM10], thus do not account for the order of occlusions.

A problem related to line selection – and sometimes solved simultaneously with it – is viewpoint selection. Lee et al. [LMSC11] steered the selection of lines from a precomputed pool in a greedy algorithm based on the maximum intensity projection (MIP) of a scalar entropy field, called maximum entropy projection (MEP). They also used the MEP to select the best viewpoint among 780 candidates placed on a sphere surrounding the data set. Tao et al. [TMWS13] coupled the selection of streamlines and viewpoints by modeling them as interrelated information channels [WS11], i.e., they selected a streamline set and a viewpoint simultaneously. The viewpoint is constrained to be located on a sphere surrounding the data set, and the streamline selection is invariant under camera movement but is based on the evaluation of multiple views. Additionally, they generate a camera path on the sphere.

The aforementioned techniques all tried to come up with information-theoretic or geometry-based properties that characterize the relevance of a line. These properties, however, do not work well in all scenarios. In fact, dependent on the exploration task or the application domain, different aspects of a vector field might be of interest, such as presence of vortices, flow behavior close to boundaries etc. Therefore, it makes sense to let the user provide application-dependent importance measures. However, a direct mapping of such importance measure to opacity is only local and does not incorporate the occlusions introduced by the selected lines. Such straightforward method would neither provide context information nor does it allow navigation in the scene without suffering from potentially vast occlusions. We will provide a technique that incorporates the importance information in a more sophisticated way.

All existing approaches for 3D line selection are – to the best of our knowledge – either local or greedy. “Local” means that a line is selected based on certain importance measures without considering its relation to other selected lines. “Greedy” means that a new line is selected based on its relation to already selected lines, making the process of line selection dependent on the order of line insertion. Furthermore, the restriction of some techniques for having the viewpoint outside of the domain is a limitation that makes viewpoint selection approaches not an option for free scene navigation.

Table 2.1 compares features of the most relevant techniques for 3D line selection and

our new approach. We consider (in columns left to right) the following features: view dependence (selected lines change when changing the viewpoint), frame coherence (small changes of the viewpoint lead to small changes in the shown lines without popping artifacts of new lines; view-independent techniques are trivially frame coherent), occlusion (occlusion information is used for line selection), interactivity (line selection is either precomputed or fast enough for interactive navigation, trivially yes for view-independent), GPU-accelerated (availability of GPU-accelerated implementation), transparency (semi-transparency used for line rendering), and strategy (either local, greedy, or global). The yellow boxes could not be uniquely classified: the image-based approach of Li and Shen [LS07] produces for a given view a line set that does not intersect in image-space, thus there is no occlusion. For more complex and larger data sets they propose to combine the lines acquired from multiple views. In that case, they do not address the arising occlusion. Albeit Li and Shen [LS07] and Ma et al. [MWS13] first validate lines from the previous frame, frame coherence is still an issue, as newly added lines introduce popping artifacts. McLoughlin et al. [MJL*13] and Yu et al. [YWSC12] present hierarchical streamline clustering algorithms in which the user steers the degree of occlusion by interactively selecting a level-of-detail.

In a later work, we tailored a local transparency mapping for blood flow data [LGP14]. For this, first vortices are extracted by the λ_2 criterion [JH95]. Then, the flow around them is separated into convex and concave flow (as seen from the camera), which distinguishes between the flow in front and behind the vortex. Those are separately mapped to transparency in order to clear the view on the vortices.

Method	**View-dep.**	**Frame coh.**	**Occlusion**	**Interact.**	**GPU-acc.**	**Transp.**	**Strategy**
[CCK07]	✗	✓	✗	✓	✗	✗	greedy
[LS07]	✓	✓	✓	✗	n/a	✗	greedy
[ATR*08]	✓	✓	✗	✓	✓	✗	local
[MCHM10]	✓	✗	✓	✗	✓	✗	greedy
[XLS10]	✗	✓	✗	✓	n/a	✓	greedy
[GBWT11]	✓	✓	✗	✓	✓	✓	local
[LMSC11]	✓	✗	✓	✓	✓	✗	greedy
[MJL*13]	✗	✓	✓	✓	n/a	✗	greedy
[YWSC12]	✗	✓	✓	✓	✓	✗	greedy
[TMWS13]	✗	✓	✗	✓	✓	✗	greedy
[MWS13]	✓	✓	✓	✓	✓	✗	greedy
Our approach	✓	✓	✓	✓	✓	✓	**global**

Table 2.1.: Comparison of features of related methods and our approach.

2.3.2. Line Selection in Unsteady Vector Fields

There is considerably less work on line (placement and) selection for unsteady flow data in 2D and 3D domains likewise. For 2D, Jobard and Lefer[JL00] suggested to depict a coherent animation of streamlines instead of displaying pathlines. For each streamline in one time step a corresponding line in the next time step is found by integrating candidate lines from sample points in the next time step and choosing the closest line. Weinkauf et al. [WTS12] proposed a selection strategy for creating static illustrations of pathlines and streaklines. They reduced visual clutter by avoiding cusps in the projections of the lines. To cope with the visual complexity of possibly heavily intersecting pathlines, Hlawatsch et al. [HSJW14] proposed a glyph visualization, in which each glyph contains a single pathline. When zooming out, patterns emerge that provide an impression of areas with coherent flow behavior. Until now, the selection problem was neither addressed for animated pathlines nor for streaklines.

For 3D line fields, Günther et al. [GBWT11] proposed a view-dependent and frame-coherent approach, which sets the opacity of a line based on its contribution to the viewport, i.e., the number of pixels visible in the rendered image. Since this is a local approach, the method is potentially fast enough for line selection from sets of line fields. However, we will show later that even for steady data the screen contribution blending method is of limited use, because occlusion is not considered in the selection process. So far, there exists no satisfying solution that allows explorative navigation in sets of line fields, such as pathlines or streaklines.

2.3.3. Surface Selection in Steady Vector Fields

Even more complicated than the placement of lines is the placement of surfaces. Most often the seed curves of integral surfaces are placed manually by the user. The placement of a 2D line in a 3D domain is thereby a rather tedious and cumbersome task. The manual placement can be facilitated by dragging flow-aligning surfaces [MSRT13b] for a more intuitive interaction.

There is also a thread of research on the automatic placement of stream surfaces. Early work of van Wijk [vW93] and Cai and Heng [CH97] proposed methods to built evenly-spaced stream surfaces in flows for which a stream function exists, i.e., these methods are limited to helicity-free flows. Martinez Esturo et al. [MSRT13a] extracted *one single* representative surface that is globally optimal w.r.t. a quality measure. They have shown that their approach extracts the surface that was manually chosen by experienced visualization researchers in related visualization papers, which gives evidence for the usefulness of their method. Instead of displaying just a single surface, Edmunds

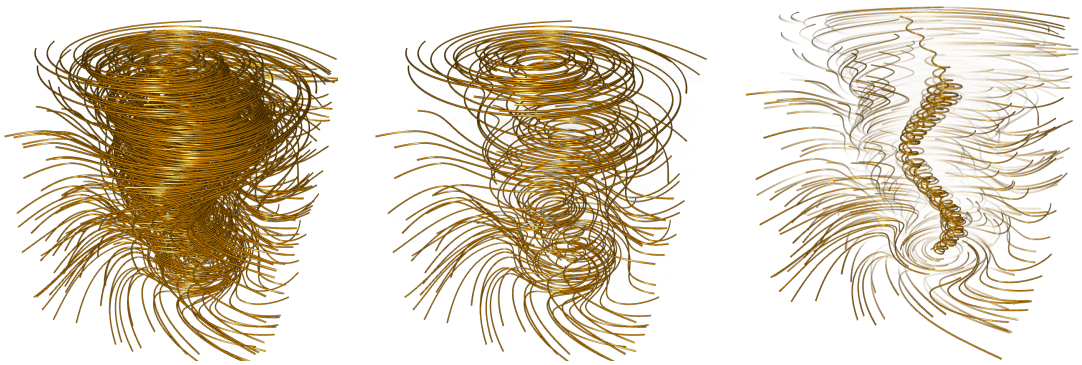
et al. [ELC*12b] densely filled the domain with *multiple* stream surfaces automatically. For this, they used isolines on the domain boundaries as seed curves. While the full set of their extracted surfaces is not necessarily useful for direct visualization due to rather strong occlusions, it might serve as input to a subsequent selection procedure. Further, Edmunds et al. [ELM*12] clustered seed curves to obtain multiple stream surfaces. They do, however, not consider in the selection process how well a resulting stream surface represents the flow. In Schulze et al. [SMG*14], we extended the work of Martinez Esturo et al. [MSRT13a] to produce multiple characteristic stream surfaces that are suitable for an exploration task. Here, quality criteria are evaluated for the extracted surfaces. Recently, Brambilla and Hauser [BH15] constructed multiple stream surfaces, based on the similarity of the streamlines therein. They aim for surfaces that consist of most similar streamlines, whereas streamlines of different stream surfaces maximize dissimilarity. Bartoň et al. [BKC15] extracted stretch-minimizing stream surfaces in divergence-free vector field.

3 Chapter 3. Opacity Optimization for 3D Line Fields

In this chapter, we describe an energy-based optimization for the opacity of 3D lines such that the view is cleared on important line parts. The following text is in large parts based on our publication:

T. Günther, C. Rössl and H. Theisel
Opacity Optimization for 3D Line Fields
ACM Transactions on Graphics (Proc. SIGGRAPH) 32, 4 (2013), 120:1–120:8.

The selection of lines that represent a vector field best is a long-standing and very fundamental problem. If we show too many lines, as in Fig. 3.1a, we get occlusions



(a) Too many lines cause occlusion.

(b) The wrong lines miss important features.

(c) The right lines highlight features and show context.

Figure 3.1.: Line selection is a difficult problem that requires to find a balance between the presentation of relevant lines and the avoidance of occlusion. In this chapter, we tackle this problem, leading to visualizations as in Fig. 3.1c.

that hinder the view at relevant structures. If we remove a few lines without regarding their role in the image, we might miss relevant parts as in Fig. 3.1b. Still, even for the lines that we did not miss, we might still get occlusions. In the end, it is always a *trade-off* between the presentation of information (i.e., relevant lines) and the avoidance of occlusion. But, there is more. When looking at a visualization, there is always something that we are searching for, something that is of interest to us, like the eye of the tornado shown in Fig. 3.1c. Thus, what we rather want is a visualization technique that clears the view on those relevant parts. So, given a set of lines, and a user-defined importance measure, this chapter describes a technique that adjusts line opacities so that we can see the important structures more clearly. Fig. 3.2 demonstrates several results of our method.

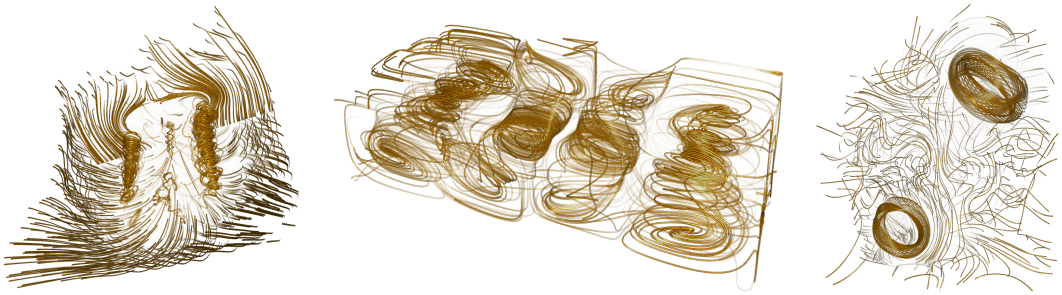


Figure 3.2.: Applications of our interactive, global line selection algorithm. Our bounded linear optimization for the opacities reveals user-defined important features, e.g., vortices in rotorcraft flow data, convection cells in heating processes (Rayleigh-Bénard cells) and field lines of decaying magnetic knots (from left to right).

3.1. Problem Setting and Error Function

This work is based on the insight that line selection should be formulated as a *global optimization problem*: if a line is detected to be important, but at the same time occludes more important structures, it should not be rendered. On the other hand, if a line is of only moderate importance and does not occlude more important structures, it can (and should) safely be rendered. This means that the decision of selecting a particular line is a compromise between having a maximal amount of conveyed information and having a minimal amount of occlusion of other features. There are a number of challenges associated with this problem:

View Dependence. Occlusion is an inherently view-dependent problem, i.e., it must be solved for every possible viewpoint anew to enable free scene navigation.

Interactivity. Results should be provided at interactive rates. Solving the problem per pixel might therefore be too expensive. Instead, a coarser discretization should be devised.

Frame Coherence. A critical challenge is to obtain frame coherence under camera movement. This means lines should not suddenly pop up, be removed or flicker. (Greedy approaches are particularly prone to this problem, e.g., [MCHM10].)

Continuity. The opacity should vary slowly along the lines. This continuity is important as artificial discontinuities might grab attention and distract. If lines appear chopped it is only because they left the domain, which is a useful information to present, as it supports the spatial perception.

Error Function

Similar to existing methods, our approach starts with a finite set of polylines that covers a 3D domain densely. Instead of selecting a subset of these lines, we render all lines but assign varying opacities to their segments. For this, every polyline is split into a number of segments, producing in total n polyline segments. Figs. 3.3a and 3.3b illustrate this for two polylines with $n = 6$. We compute the optimal opacity $\alpha_i \in [0,1]$ for each segment $1 \leq i \leq n$ as solution to a bounded-variable least-squares problem (Fig. 3.3c). Thereby, $\alpha_i = 0$ means completely invisible, and $\alpha_i = 1$ is completely opaque. For rendering, the opacities are interpolated between adjacent segments to yield vertex opacities, see Fig. 3.3d. This way, we resolve occlusions by locally fading out line segments.

Each segment is equipped with a local importance $g_i \in [0,1]$, which can be chosen depending on the application. (A discussion of possible choices for g_i follows in Section 3.3.1.) The higher g_i the more the segment should be emphasized. If no particular importance is given, g_i is set to 0.5 for all segments. In addition, the following properties are computed for every pair (i,j) of segments

(a) Given is a set of polylines.

(b) Discretize polylines into n segments (here: $n = 6$).

(c) Compute per-segment opacity α_i by energy minimization.

(d) Interpolate opacities between adjacent segments for rendering.

Figure 3.3.: Illustration of the general idea.

- a_{ij} encodes adjacency of segments: $a_{ij} = 1$ if the segments i and j are adjacent on the same polyline; otherwise $a_{ij} = 0$. The example in Fig. 3.3c has $a_{12} = a_{21} = a_{23} = a_{32} = a_{45} = a_{54} = a_{56} = a_{65} = 1$ and all remaining $a_{ij} = 0$,
- $h_{ij} \geq 0$ describes how much segment j is occluded by segment i as seen from the particular viewpoint. Its continuity during camera movement induces frame coherence. In Fig. 3.3c we have $h_{25} > 0$ and all remaining $h_{ij} = 0$.

We find opacities α_i as minimizers of the quadratic error function

$$E = \frac{p}{2} \sum_{i=1}^{n} (\alpha_i - 1)^2 \tag{3.1}$$

$$+ \frac{q}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \Big(\alpha_i \, (1 - g_i)^\lambda \, h_{ij} \, g_j \Big)^2 \tag{3.2}$$

$$+ \frac{r}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \Big(\alpha_i \, (1 - g_i)^\lambda \, h_{ji} \, g_j \Big)^2 \tag{3.3}$$

$$+ \frac{s}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{a_{ij}}{2} (\alpha_i - \alpha_j)^2 \tag{3.4}$$

with *bounded* variables $0 \leq \alpha_i \leq 1$. The four terms of E can be interpreted as follows:

Strive for Visibility. Term (3.1) is a regularization to prevent almost empty renderings, i.e., prevent all α_i from being close to 0 by penalizing a variation of α_i from 1. That is, all segments try to be visible.

Penalize Occlusion. Term (3.2) introduces a penalty if an unimportant segment i (i.e., $1 - g_i$ is large) occludes to some degree an important segment j (i.e., g_j is large). Then, $h_{ij} > 0$ and in this case the product $(1 - g_i)^\lambda \, h_{ij} \, g_j$ gets a high value. To minimize the entire term, the optimization will assign a low opacity to the segment in front, i.e., α_i is reduced. The parameter λ steers the fall-off of g_i from 1 and thereby allows to put further emphasis on the important structures.

Remove Background Clutter. Term (3.3) is used to remove background clutter behind important segments. It therefore covers the opposite case, i.e., it acts if an unimportant segment i is occluded by an important segment j. Similarly, the opacity of segment i is forced to be small in order to have a "more empty" background behind the important segment j.

Smoothness. Term (3.4) is a smoothness term that enforces a slow change of the opacity along a line. Intuitively speaking, the term expresses: if two segments are adjacent, their opacity should be similar, i.e., we penalize the opacity deviation of

adjacent segments.

Finally, the weights p, q, r, s balance the contribution of the terms. They can be restricted by a normalization or global scale, e.g., we choose $p = 1$. Section 3.3.2 discusses the parameters in more detail.

3.2. Details and Implementation

The essence of our approach is to use transparency to locally fade out line segments that cause occlusion. This requires a method for correctly blending many transparent layers, see Maule et al. [MCTB11] for a survey of raster-based transparency techniques. We decided to create on the GPU for each pixel a linked list of the rasterized fragments [YHGT10], as this data structure serves two different purposes: the blending of transparent layers and the generation of occlusion degrees h_{ij} that measure occlusion caused by segment i on j. For the latter, fragment linked list elements store – in addition to color and depth – a value that is used to identify the polyline segments. Based on the segment index and the fragment depth, the order of occlusions is taken into account.

3.2.1. Initial Line Set

Firstly, we obtain the initial polyline set from the given line field in a preprocess. For this, the only requirement is a dense covering of the domain, i.e., for every point in the domain there is a line passing by closely. For the examples on flow visualization and medical imaging, we used a random distribution of seeding points for line integration. If random seeding does not yield a fair uniform distribution of line geometry, sophisticated view-independent seeding strategies may serve as input [CCK07, XLS10, MJL*13].

Secondly, we partition each polyline into a fixed number of k polyline segments. The total number of segments is n, which is given for all images later in Section 3.4.1. The value k trades the potential variation in opacity along each polyline for the size of the system to solve. We observed that k can be rather small because the computed opacities are blended smoothly along polylines for rendering. We used $k = 8$ for all examples.

In the current chapter, the n segments are uniformly distributed along the lines. Note that an importance-driven subdivision is not helpful, i.e., having smaller segments in more important regions. This is because we want unimportant segments to be able to locally fade out. In an importance-driven subdivision, unimportant lines might consist of only one segment, making the entire line disappear even if only a small part

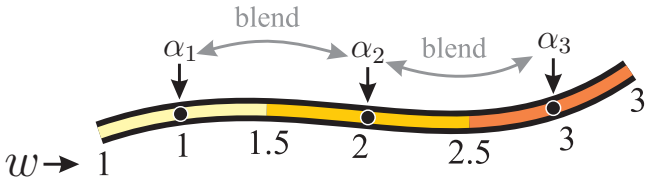


Figure 3.4.: The precomputed parameterization w provides lookup up of nearest line segment centers and blending weights.

of the line causes occlusion. Which part of a line needs to reduce its opacity is view-dependent, and can thus not be known a priori. For that reason, a uniform distribution of segments is advisable to give all segments a similar chance to purposefully fade out. On a further note, an adaptive, view-dependent partition of the lines, so that all segments have a similar size in screen space, helps to reduce the total number of segments n. This improvement is subject of Chapter 4.

3.2.2. Fragment Linked List Construction and Rendering

All element data of the fragment linked lists resides in a global memory pool that is shared by all pixels, and is managed by atomic operations in the fragment shader. The element data to append is obtained by rendering all lines, and expanding them in a geometry shader to viewport-aligned triangle strips. Since our technique only focuses on the computation of opacities, it can be combined with all kinds of rendering styles and techniques. Depth-dependent halos [EBRI09] are one option to provide depth cues (e.g., when two lines cross). We used this method, i.e., in the pixel shader we alter the depth and color of the boundary region to add halos. Other depth enhancing methods are imaginable here as well, e.g., line ambient occlusion [EHS13] or unsharp masking of the depth buffer [LCD06]. Techniques related to ambient occlusion excel especially in very dense fields. Further, we use an illuminated streamline shading (ISL) by Zöckler et al. [ZSH96], combined with multi-sampling (16× coverage-sampling anti-aliasing/CSAA, i.e., 4 samples per pixel). We run the pixel shader only once per pixel (i.e., on pixel-frequency) to create only one fragment in the linked list, and evaluate multiple samples explicitly to smooth the sharp edge between center and halo region.

Every opacity value is associated with the center of the respective polyline segment. The arc length distance of an arbitrary point on the line to its closest left and right centers is used as weight for blending opacities, i.e., to define the transparency of a line's fragment. Blending weights and the indices of the closest opacity centers are easy to obtain from the blending weight parameterization w, shown in Fig. 3.4, which is precomputed and added as a vertex attribute. We evaluate index i and opacity α for

a weight w as

$$\begin{aligned} i &\leftarrow \text{floor}(w) \\ \alpha &\leftarrow \text{lerp}\big(\alpha_i, \alpha_{i+1}, \text{frac}(w)\big) \,. \end{aligned}$$

The transparent colored fragment is appended to the pixel's fragment linked list. We store the depth (quantized to 24 bit) and the coverage bit vector (8 bit) together. Color and blending weight w are stored as RGBA vector with 8 bit per component and as full 32-bit float, respectively. A second pass sorts the lists, which is required for computing h_{ij} (see below). We remark that also rendering benefits from pre-sorting per pixel instead of sorting per sample.

3.2.3. Computation of Occlusion Degrees

To gain higher throughput, we use an otherwise unoccupied CPU core for the assembly of the system matrix. Thus, in order to compute the values h_{ij}, we stream the fragment linked lists asynchronously to the CPU after sorting. We iterate all lists and sum up the weights of the occluded fragments for all segments, see Alg. 2.

Input: fragment linked list per pixel
Output: h(i,j)
h(i,j) ← 0 ;
foreach *pixel p* **do**
  foreach *fragment A in linked list of pixel p* **do**
    i ← floor(A.w + 0.5) ;
    foreach *fragment B behind A* **do**
      j ← floor(B.w) ;
      v ← frac(B.w) ;
      h(i,j) ← h(i,j) + 1-v ;
      h(i,j+1) ← h(i,j+1) + v ;
    end
  end
end

Algorithm 2: Penalty h_{ij} for segment i occluding j.

3.2.4. Minimization of the Error Function

The minimization of the quadratic error function E refers to solving a bounded-variable least-squares problem which can be formulated in the normal equations as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x} + \text{const} \\ \text{subject to} \quad & 0 \leq \mathbf{x}_i \leq 1. \end{aligned}$$

with $\mathbf{x} = (\alpha_1, ..., \alpha_n)^{\mathrm{T}}$. The sparse, positive definite matrix $\mathbf{Q}$ captures the error terms (3.1)–(3.4) together with the constant vector $\mathbf{c} = (-p, \ldots, -p)^{\mathrm{T}}$, which stems from the constant -1 in (3.1). The operator $\mathbf{Q}$ can be assembled as

$$\mathbf{Q} \;=\; p \cdot \mathbf{I} \;+\; q \cdot \mathbf{W} \;+\; r \cdot \mathbf{V} \;+\; s \cdot \mathbf{D}^{\mathrm{T}}\mathbf{D}\,,$$

where $\mathbf{I}$ is the identity matrix. The matrices $\mathbf{W}$ and $\mathbf{V}$ are both diagonal matrices with entries $\mathbf{W}_{ii} = (1-g_i)^{2\lambda} \sum_{j=1}^{n} h_{ij}^2\, g_j^2$ and $\mathbf{V}_{ii} = (1-g_i)^{2\lambda} \sum_{j=1}^{n} h_{ji}^2\, g_j^2$. such that $\mathbf{W}$ and $\mathbf{V}$ refer to (3.2) and (3.3), respectively. The matrix $\mathbf{D}$ is the backward difference operator that is applied to adjacent polyline segments in (3.4). Due to the smoothness term, the final system matrix $\mathbf{Q}$ becomes a tridiagonal matrix. Appendix A contains a derivation of the reformulation of the energy into matrix notation.

The original publication [GRT13] contained an error in the reformulation of the energy into matrix notation, which has now been fixed[1]. We wish to thank Florian Ferstl for reporting this. Unfortunately, the error carried into the follow-up papers [GRT14, GSM*14], as well. For this thesis, the experiments have been repeated. With the corrected version, we obtained visually similar results for a slightly different parameter setting. The parameter listings and solver timings have been updated accordingly. On the plus side, the correction increased the performance of the system, as the system matrix $\mathbf{Q}$ became tridiagonal. We elaborate on the performance in Section 3.4.1.

We use the reflective Newton method [CL96] that is implemented in MATLAB's `quadprog`[2] function to solve the system. Due to frame coherence of the solutions, we obtain a significant speed-up by using the previous solution as an initial guess for the next solution. We perform the setup of the system matrix $\mathbf{Q}$ for the *next* time step in one

[1] The original implementation minimized a different energy for the occlusion term (and similarly the background clutter term):

$$\frac{q}{2} \sum_{j=1}^{n} \left(\sum_{i=1}^{n} \alpha_i \, (1-g_i)^{\lambda} \, h_{ij} \, g_j \right)^2$$

instead of

$$\frac{q}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\alpha_i \, (1-g_i)^{\lambda} \, h_{ij} \, g_j \right)^2$$

[2] We found that an equivalent formulation for MATLAB's `lsqlin` was slower, and same for quadratic/conic solvers in MOSEK (see `mosek.com`). Thanks to Christian Rössl for the experiments.

CPU-thread, while another thread is dedicated to solving the system for the *current* time step, i.e., we pipeline the computation. Once a solution is available, it is streamed to the GPU to smoothly blend the current opacities to the latest solution. A technical reason for this fading is that we do not get every frame a new solution, i.e., we want to hide this typically short latency. An additional benefit is that a smooth and soft transition is more comfortable than sudden changes, as it avoids the perceptual issues of popping artifacts. Thus, even if a solution was available every frame, an artificial fading is still helpful.

3.3. Parameter Studies

In the following, we discuss the importance function g_i and parameters of our system.

3.3.1. Choosing the Importance

As the meaning of *importance* depends on the application and goals of data exploration, we do not prescribe a particular importance function. Instead, we provide a generic way to incorporate *any* suitable importance measure, which is defined as g_i per polyline segment i. This enables the user to either explore the presence of, or to communicate a specific feature in the data. For all examples, we use either the polyline length (i.e., all segments of a line have identical importance) or the segment curvature (i.e., integrated curvature). These measures are generic and available for any application. In addition, various methods exist that extract application-independent line features, for instance linear entropy [FI08], angular entropy [MCHM10], scalar entropy fields [XLS10], or the number of pixels that a line covers on the screen [GBWT11]. Alternatively, importance may be application-specific, e.g., distance to a tumor in medical imaging, cluster sizes in fiber tracking, uncertainty in data, or it can simply be sampled from a scalar field. With our approach, we can incorporate any of those sophisticated importance functions. Different choices for importance are shown for a synthetic TORNADO data set in Fig. 3.8 (line length) and Fig. 3.2 (curvature). The latter brings out the vortex core. (For both images we set $q = 1.2$, $\lambda = 5$.)

Note that setting the importance of a segment to $g_i = 1$ guarantees that the segment will always be visible, since the energy terms (3.2) (penalize occlusion) and (3.3) (remove background clutter) both evaluate to zero. Then, only the smoothness term (3.4) may reduce the opacity. The ability to specify lines that should definitely be seen is a valuable tool for the user, as it allows to represent critical parts in the data set, e.g., an aneurysm in blood flow.

3.3.2. Optimization Parameters

The parameters p, q, r, s act as weights for the error terms, and λ emphasizes important lines. We set $p = 1$ and select q, r, s relatively to this value. The parameter q weights occlusion and is probably most important: it scales the amount of opacity in the final image. Setting $q = 0$ keeps all lines opaque (if r is chosen relative to q, as described later). An increasing q gradually fades out occluding lines. Effectively, this is the main tool to resolve occlusions introduced by lines in the foreground, as shown in Fig. 3.5: Here, we placed a camera behind a helicopter in slow forward flight close to the ground. The data set we used is described in [KKKK12] and is a validated simulation in a wind tunnel (test case No. 12 in the paper), computed in about $4\frac{1}{2}$ weeks wall clock time on 256 processors on the cray cluster at the High Performance Computing Center in Stuttgart, Germany. We uniformly resampled 400 degree rotor revolution at 10 degree per time step at a resolution of $700 \times 100 \times 130$ voxels, yielding in total 4.2 GB of vector field data. In the remainder of the thesis, we refer to this data set as the HELICOPTER HOVER flow. The line field represents air flow, and the direct visualization of the original data, shown in the left image, suffers from vast occlusion. In the right image, we use curvature as importance to reveal the main vortices: visible are not only the two vortices, released from the rotor blades and transported back, but also the vortex on the ground in front of the helicopter. The latter vortex is critical as it is the main source of small dust particles that cause hazardous brownout conditions, which endanger ground personnel as well as passengers.

The parameter r has a more subtle effect: it accounts for fading out unimportant lines in the background that are occluded by important lines in the foreground. This effect is demonstrated in the flow around a short WALL-MOUNTED CYLINDER, simulated by [FWT08] and shown in Fig. 3.6. Behind the interesting region there is an unimportant laminar layer that clutters the view for $r = 0$ (left). For $r > 0$ we obtain a much clearer visualization, as disturbing unimportant lines in the background are faded out (right).

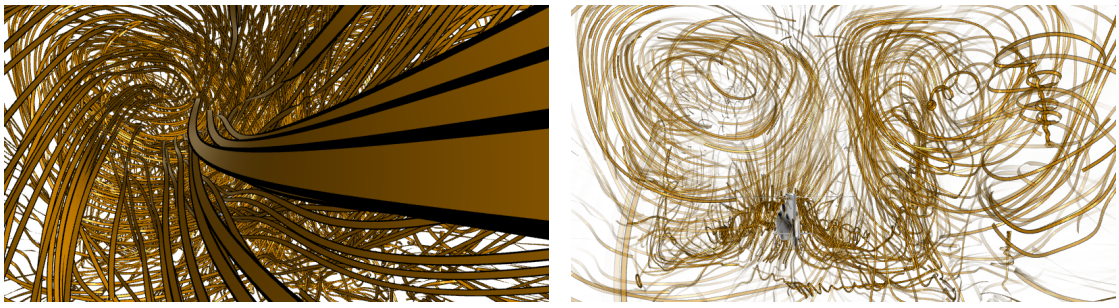


Figure 3.5.: Streamlines visualize the air flow around a helicopter in forward flight close to the ground. The camera is placed in the wake of the helicopter. Left: Input line set is cluttered ($q = r = 0$). Right: We set $q = 1$, $r = 0.4$, $\lambda = 3$ to fade out unimportant parts.

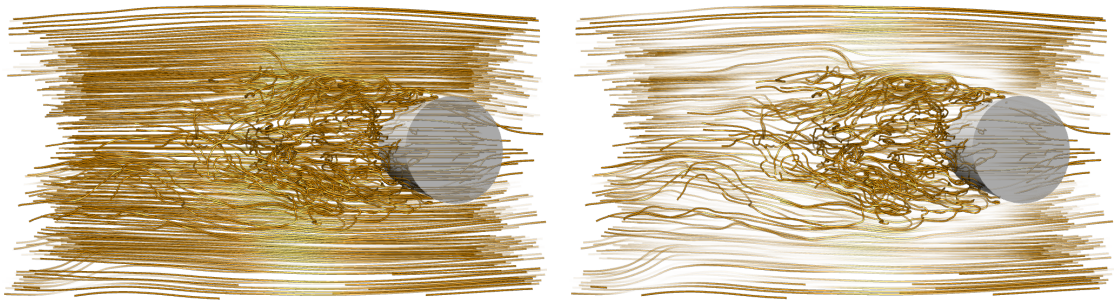


Figure 3.6.: Visualization of a turbulent area behind a wall-mounted cylinder. In both images, we set $q = 0.8$, $\lambda = 1.5$. Left: Visual clutter in the background for $r = 0$. Right: Clutter can be removed by setting $r = q/10$.

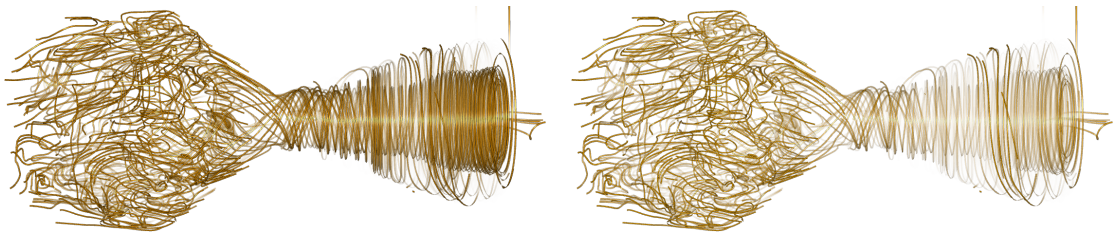


Figure 3.7.: Streamlines in a hydrocyclone. An increase of λ emphasizes important lines, i.e., the conical outflow part. We set $q = 1$, $r = 0.01$, left: $\lambda = 28$ and right: $\lambda = 25$.

It is reasonable to select $r \leq q$, and our experiments show that setting r relative to q, as $r = \frac{1}{10}\, q$ is generally a good choice. We use this setting in the remainder of the chapter unless stated otherwise.

The remaining parameter s weights a smoothing term that penalizes high variation of opacities along polylines. We always set $s = 0.3$.

Finally, the coefficient λ controls the emphasis of important lines. In Fig. 3.7, we demonstrate its utility in a HYDROCYCLONE, a device to separate particles in a liquid suspension. The steady data set was provided by Markus Rütten and was uniformly resampled to $256 \times 369 \times 536$ voxels. If not stated otherwise, we set $\lambda = 1$ neutral, i.e., no increased emphasis on important lines. In all the examples above, we used curvature as importance measure.

3.4. Results and Discussion

We compare [MCHM10] and [GBWT11] with our algorithm in Fig. 3.8. The columns show the initial line set (left) and the different methods. The different data sets are arranged in the rows.

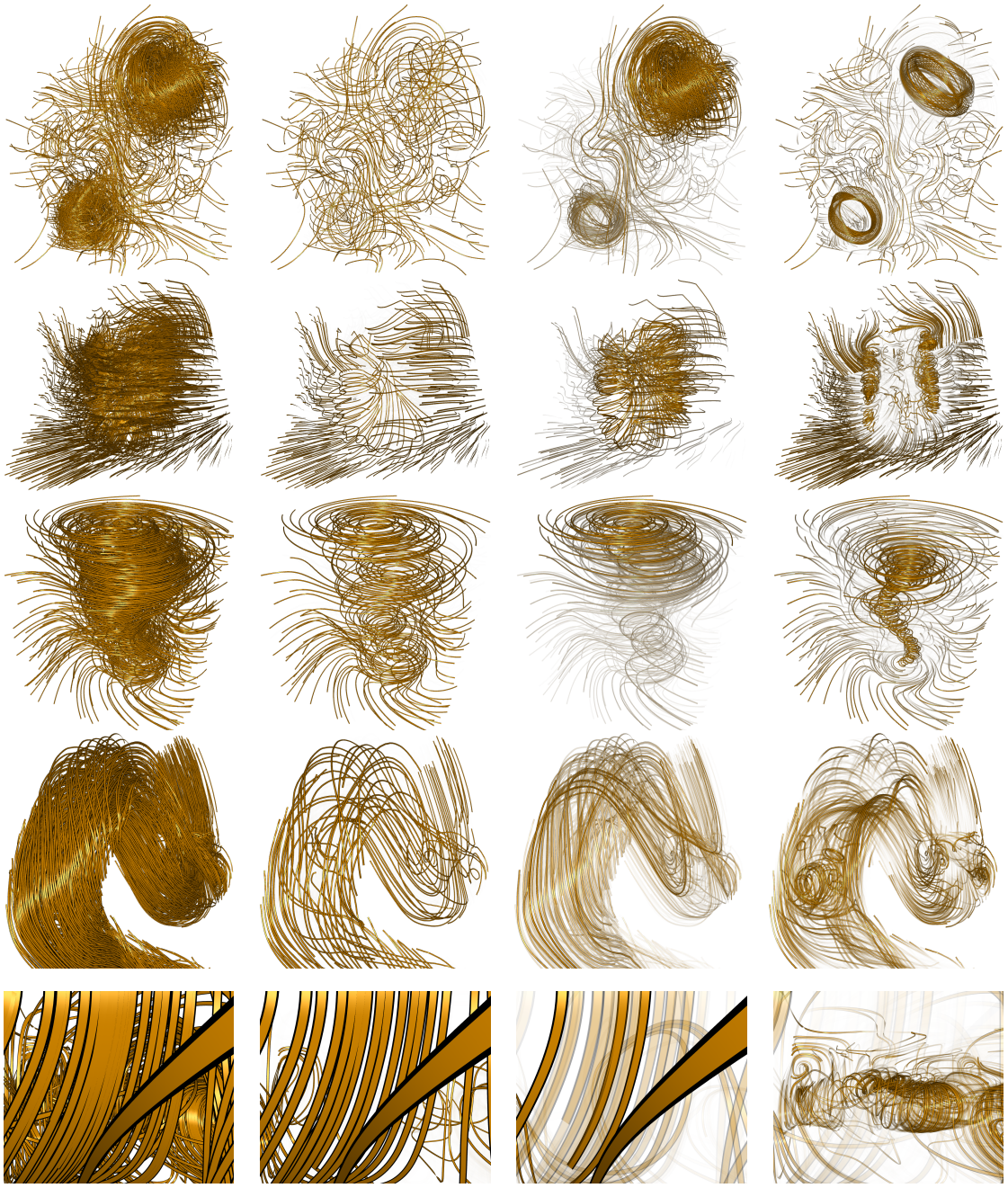


Figure 3.8.: Comparison with other line selection algorithms. Columns from left to right: input line set (i.e., all lines), [MCHM10], [GBWT11] and our approach. Data sets are in rows, top to bottom: Borromean rings, flow around a descending helicopter, tornado, aneurysm and Rayleigh-Bénard convection. Our method brings out specific features that are otherwise occluded, i.e., rings, vortex cores, the aneurysm (cf. Fig. 3.9), and the convection cells (cf. Fig. 3.2 for an overview image).

The first row shows magnetic field lines in the decay of magnetic knots, as present in astrophysical objects. Here, the topological reconnection of rings was studied by Candelaresi and Brandenburg [CB11]. The rings temporarily form the shape of BORROMEAN RINGS. The objective of the visualization is to bring out the highly occluded, symmetric rings, a task at which our technique performed best. We use line lengths as importance with $q = 1.8$, $r = 0.18$, and $\lambda = 3$.

The second row analyzes experimental wind tunnel data by [YTvdW*02] of a descending helicopter (HELICOPTER DESCENT), viewed from below without the helicopter body. Of interest are the vortices that detach from the tips of the rotor blades: their impact on the rotor and on the noise generation (by blades cutting the vortices) should be analyzed. Our method produces clear views on the vortices. Here, curvature was used as importance, with $q = 1.3$, $r = 0.65$, and $\lambda = 2$.

The TORNADO data set shown in the third row is described in Section 3.3.1. Since our approach is able to locally fade out segments, it can cover the entire viewport with lines (by using all lines) and at the same time reveal features like the tornado's vortex core (in contrast to [MCHM10]). Furthermore, this test case shows the advantage of being able to fade out line segments locally over using a global opacity value per line as in [GBWT11].

The fourth row shows medical data, i.e., blood flow through an ANEURYSM, simulated at the Institute of Fluid Dynamics and Thermodynamics, based on vessel geometry acquired at the University Hospital, both in Magdeburg. In the input data, the aneurysm is entirely occluded by flow through the blood vessel. Not only the aneurysm but also convection due to wall reflection is made visible by our approach. Here, we used curvature as importance and set $q = 3$ and $\lambda = 3.5$. An overview, taken from a distant viewpoint, is shown in Fig. 3.9, using curvature as important and setting $q = 2$ and $\lambda = 2.5$.

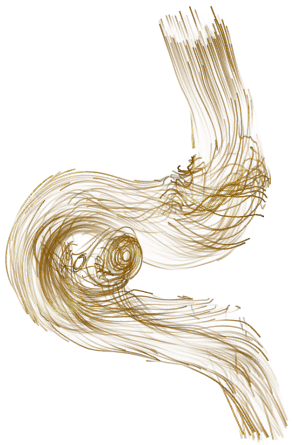


Figure 3.9.: Aneurysm.

The fifth row shows a RAYLEIGH-BÉNARD convection, i.e., the forming of separate convection cells by fluids heated from below (simulated using the free software NaSt3DGP [GDN98]). The center left image in Fig. 3.2 depicts all four cells. To demonstrate the disadvantage of greedy algorithms, we placed the camera inside a convection cell. Both methods, [MCHM10] and [GBWT11], select highly occluding lines, while our approach reveals the occluded cells (curvature as importance and $q = 0.5$, $r = 0.1$, $\lambda = 3$).

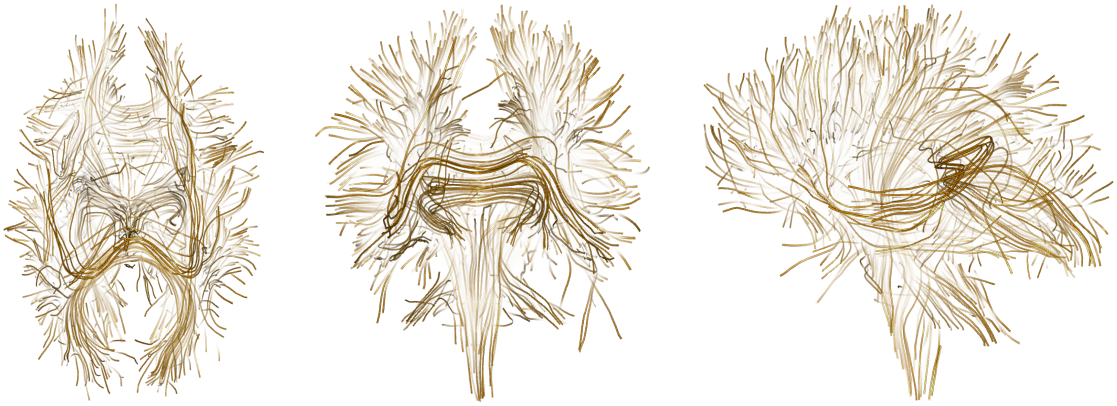


Figure 3.10.: Tensor data in medical imaging: Diffusion tensor lines, using line length as importance with $q = 4$, $\lambda = 5$.

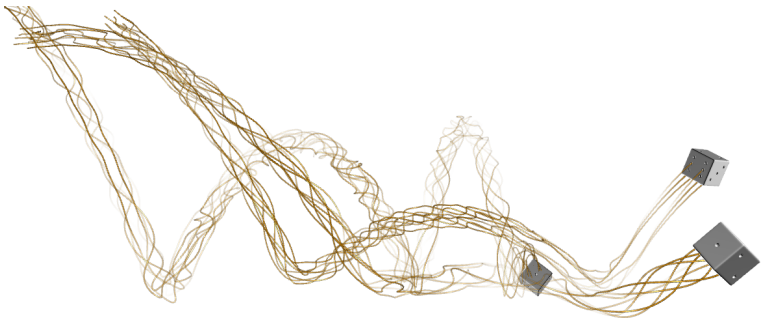


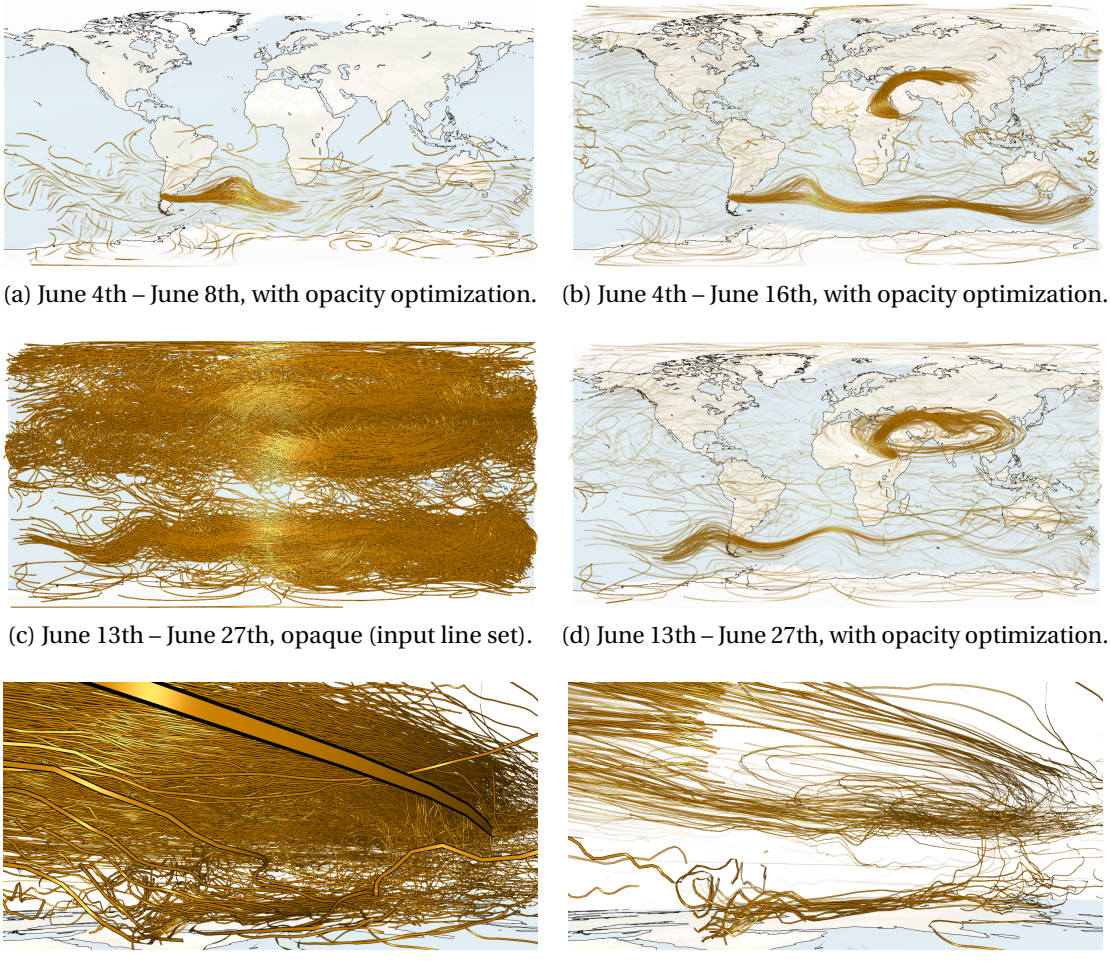
Figure 3.11.: Trajectories of three falling dice. Each casts eight speed lines, which are partially faded out to reveal the helical trajectory. (curvature as importance, $q = 0.15$, $r = 0.375$, $\lambda = 1$)

Fig. 3.10 shows visualizations of diffusion tensor data, i.e., tractography in medical imaging, which allows to infer the white-matter connectivity of the brain to diagnose vascular strokes or cancer. The DT-MRI data was acquired at the Central Hospital of Bremen.

In Fig. 3.11, we visualize trajectories of three falling dice to depict motion. We show speed lines for every corner to demonstrate that our technique is also suited for automatic fading of lines in illustrative renderings. This scene is courtesy of Maik Schulze.

We applied the opacity optimization technique in our IEEE Scientific Visualization Contest 2014 submission [GSFT16]. The contest was hosted by RWTH Aachen University and the data and tasks were provided by the Simulation Laboratory Climate Science at the Jülich Supercomputing Centre and the Institute of Energy and Climate Research, both resided at the Research Center Jülich, Germany. The contest centered around the fusion of multiple satellite data modalities to reconstruct and assess the movement of

volcanic ash and sulfate aerosol emissions. Among others, the provided data set contained simulated trajectories of individual sulfate aerosol particle, which are visualized in Fig. 3.12. The simulation of these trajectories is based on the ERA-Interim [DUS*11] meteorological reanalysis and is calculated by the Chemical Lagrangian Model of the Stratosphere (CLaMS) [MKG*02, KGM*07, KGG*10] that has recently been extended to the troposphere [PKM*12]. While the input line set is highly cluttered, the sulfate aerosol clouds of the volcanoes Puyehue-Cordón Caulle in Chile (eruption on June 4th, 2011) and Nabro in Eritrea (eruption on June 13th, 2011) can clearly be seen when opacity optimization is applied. In the images, an index measure related to sulfate aerosol concentration was used to specify importance.



(a) June 4th – June 8th, with opacity optimization.

(b) June 4th – June 16th, with opacity optimization.

(c) June 13th – June 27th, opaque (input line set).

(d) June 13th – June 27th, with opacity optimization.

(e) June 21st – July 5th, opaque (input line set).

(f) June 21st – July 5th, with opacity optimization.

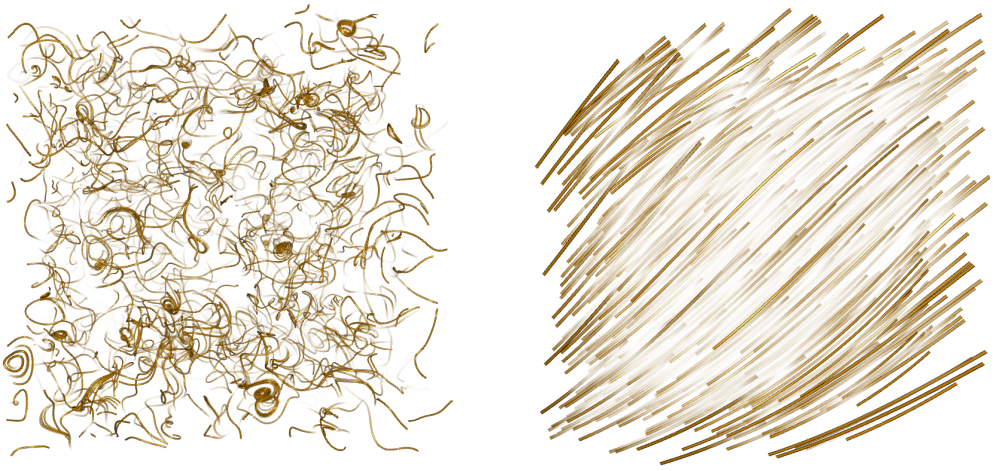
Figure 3.12.: Trajectories of sulfate aerosol particles after volcanic eruptions.

3.4.1. Performance

We measured the performance for an Intel Core i7-2600K CPU with 3.4 GHz, 24 GB RAM and a Nvidia GeForce GTX 560 Ti GPU with 2 GB VRAM. The number of frames, h_{ij} assemblies, and solves per second are listed for all referenced figures in Table 3.1. Note that the assembling, solving, and rendering run in parallel. Thus, the slowest component poses the bottleneck, which was in all but one case the assembly of the system matrix (including the traversal of fragment linked list for computing the occlusion degrees). The only exception was the dice in Fig. 3.11, for which the optimization problem was small so that the rendering was slower. The streaming of the fragment linked lists to the CPU took always 22 ms (due to the constant fragment pool size). In the examples throughout this chapter, the fragment pool size was set to an average fillrate of 8 fragments per pixel. If the entire memory pool is consumed, the buffer could dynamically be enlarged (reallocated on the GPU). All results were computed for a resolution of 1200×1000 with $16\times$ coverage-sampling anti-aliasing (CSAA).

Table 3.1.: Performance: Frames per second (Fps), assemblies of h_{ij} per second (Alg. 2) and solves per second (including computation of Q), with n being the number of segments. The bottleneck is printed **bold**.

Data set	**Figs.**	**Fps**	**Asm./s**	**Sol./s**	n
Aneurysm	3.8	16.3	**4.3**	12.8	3,816
Aneurysm	3.9	20.8	**6.3**	14.7	3,816
Bénard	3.2	7.6	**1.1**	16.9	2,224
Bénard	3.8	10.5	**2.8**	12.5	2,224
Borromean	3.2, 3.8	37.0	**9.9**	14.7	2,936
Brain (axial)	3.10	66.6	**13.3**	21.3	5,808
Cylinder	3.6	27.0	**9.5**	34.5	3,776
Dice	3.11	**46.3**	166.6	156.2	576
Heli Brownout	3.5	19.2	**5.1**	14.1	8,136
Heli Descent	3.2, 3.8	14.5	**5.9**	14.5	6,896
Hydrocyclone	3.7	16.9	**3.9**	28.6	2,016
Tornado	3.2, 3.8	28.4	**13.7**	27.7	2,648



(a) Turbulent flow *with only* important lines. ($q = 40, r = 80$)

(b) Laminar flow *without any* important lines. ($q = 1, r = 1$)

Figure 3.13.: Limitations: for extreme cases with no prominent feature to bring out, the results of our method resemble depth cueing.

3.4.2. Limitations

Our approach strives to keep a balance between information presentation and occlusion avoidance. There are two cases, shown in Fig. 3.13, when it does not get meaningful results:

- There are *too many important structures*: for the example of streamlines in a turbulent flow, the whole domain may be densely covered with important lines. In this case, it is difficult to decide on the parts to emphasize, leading to a simple depth cueing as optimum.
- There are *too few important structures*: if all lines are of low interest (e.g., streamlines in a laminar flow), the optimum also gives a simple depth cueing. This is because there is no feature to direct the focus on.

Although both cases can occur, we do not consider them as critical. For turbulent flows, as in the first case, streamlines are generally not an adequate approach for visual representation, since the shape of particular lines is dominated by randomness. In the second case, the rendering of a dense line field is not really necessary because of its simple structure.

An inherent disadvantage of line selection methods compared to seed placement methods (such as Jobard and Lefer [JL01]) is the inability to provide unlimited level of detail. In other words, if the viewer zooms into the data set, no further lines are added in sparse regions. Some line selection approaches, such as Marchesin et al. [MCHM10],

fill the viewport in a post-process by integrating further lines. Similarly extending our method requires special care to preserve the frame coherence, since newly added lines would influence the opacity of existing lines.

The rendering of transparent objects often makes it harder to infer the depth order of object, cf. Carnecky et al. [CFM*13]. Especially the crossing of lines provides "context" that helps to resolve this ambiguity. Since in our setting all lines have equal width, their projected size tells whether an unimportant line is in front or behind an important line. A constant width, however, limits the ability to zoom in, since eventually every line will cover large parts of the viewport, causing tremendous occlusion. Again, this is an issue that falls into the unlimited level-of-detail category described above.

Another limitation is the fact that our optimization does not explicitly incorporate the density of the lines. If many lines are spatially close to each other, the algorithm will assign similar opacities to them. A simple (and limited) workaround is to add small noise to the importance of the lines so that individual lines are emphasized more. A better approach would be to assign one representative a high opacity and to make adjacent lines less visible, which requires a notion of line density. While the minimization of our quadratic energy leads to considering a linear gradient, higher-order energies such as density estimators lead to non-linear gradients. Note that due to the presence of bounds on variables the problem becomes significantly more expensive than standard linear least-squares settings. We leave the challenge of effectively incorporating density into the energy to future research. The aforementioned proximity problem can be avoided by a suitable filtering of the input line set, which if executed in a view-dependent way is far from trivial, since frame coherence must be ensured. Having a general method to deal with this problem therefore seems desirable.

Also, fading lines might be confused with depth cueing, which might interfere with depth perception. There are different aspects that could be studied here. Lawonn et al. [LLPH15] studied a number of different visual cues that help with depth perception in the context of vascular trees, which to some extend might directly transfer to general line depictions. An application of this in our context likely improves the perception, though more work could be spend on understanding *when* fading lines relate to depth cueing. For instance, restricting the opacity profiles along the lines to a series of smoothstep functions of a specified width that connect plateaus of either entirely invisible ($\alpha_i = 0$) or entirely visible ($\alpha_i = 1$) line segments could remove the impression that lines slowly bend away from the camera, since they no longer vary opacity slowly. Instead, it might only appear that lines bend away from the camera very sharply. Whether this association is triggered can be questioned, as it is not represented by the line shape/width or the shading. Of course, answering these questions requires further study.

3.5. Summary

In this chapter, we introduced the first method for line selection in 3D line fields that is based on a global optimization: We computed a globally optimal opacity for polyline segments by minimizing a quadratic error function, which formalizes desired properties (favor opaque lines, reduce occlusions, smooth opacity gradient along the lines). Our approach can be steered by a-priori domain knowledge in terms of an importance function, and it attains interactive, frame coherent and view-dependent visualizations. Moreover, it is the first approach that resolves occlusions effectively for every viewpoint position.

So far, we narrowed the user parameters down to a small and intuitive set. All parameters are stable, i.e., no sudden changes occur when interactively varying them. A deeper study of the behavior of the parameters, and ultimately a reduction of the number of parameters are necessary steps that have to be taken to deploy the technique into standard visualization software. That is, the focus should shift from technical research to user-centered development. An automatic setup that adjusts the parameters to a given scene is an interesting idea that could be further investigated, especially if the processing of a large number of data sets is anticipated. Tao et al. [TMWS13] already worked on viewpoint selection, which is quite helpful for that task.

Another aspect of future work is the mapping of the optimization result to other visual parameters. So far, color was intentionally left open for the visualization task. It is undoubtedly useful to direct attention and communicate information. Varying the width of lines could compromise depth understanding, as we have argued before, yet for the purpose of continuous level-of-detail a compromise has to be found. Further, a number of different rendering styles (cf. [EBRI11],[EBRI15]) could be combined with our methods, such as using shaded tubes instead of line strips [SGS05], ambient occlusion [EHS13], depth encoding by (quantized) color and redundant depth encoding by using hatching on the lines [LLPH15], and so on. See Brambilla et al. [BCP*12] for a broader overview.

The performance of the system is another factor to improve. Our practice showed that CPU-based solvers are fast for sparse systems. In fact, we attained sufficient performance with a standard solver on the otherwise unoccupied CPU cores for the scenes that we have worked with. Nevertheless, complexity and scene sizes are never bound and are constantly growing. We think that the setup of the system (Alg. 2) may benefit from using the GPU.

The next step is to apply our line selection to unsteady line fields. The following chapter is dedicated to this problem.

Chapter 4.

4 Hierarchical Opacity Optimization for Sets of 3D Line Fields

In this chapter, we address the yet completely unsolved problem of line selection for animated 3D lines, as described in:

T. Günther, C. Rössl and H. Theisel
Hierarchical Opacity Optimization for Sets of 3D Line Fields
Computer Graphics Forum (Proc. Eurographics) 33, 2 (2014), 507–516.

For this, we extend the method from Chapter 3, for which two problems have to be solved. First, the performance has to be increased to be able to rapidly adapt the opacity to the deforming and growing lines. Second, time coherence has to be assured. For this, we present a hierarchical and view-dependent partition of lines to speed up the opacity computation. Additionally, we address time coherence by an additional energy term and a correspondence heuristic for opacity transport between time steps. Fig. 4.1 shows an example: the display of pathlines and streaklines in a time-dependent cylinder flow.

Optimizing opacities requires the solution of a bounded-variable least-squares prob-

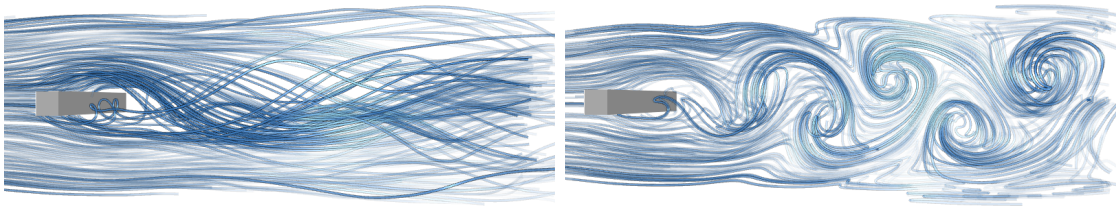


Figure 4.1.: View-dependent, frame-coherent and interactive selection of meaningful lines for sets of line fields: pathlines (left, 48 ms/opacity update) and streaklines (right, 53 ms/opacity update) in a SQUARE CYLINDER flow.

lem. The assembly of the system matrix and the numerical solving dominate the computation time. The rate at which solutions are computed depends on the number of unknowns, i.e., the total number of polyline segments n used for opacity optimization. For the interactive exploration of line sets, we reported in the previous chapter in Section 3.4.1 a maximum problem size of $n < 10,000$ polyline segments; this bound may vary depending on the data (and the desired rate of solves), $n \approx 3,000$ is a typical value. This is not sufficient for the interactive exploration of parametric sets of line fields such as pathlines or streaklines, because these fields consist of many more, possibly time-dependent, lines. Generally, the line representation used for rendering is decoupled from the representation used for solving the numerical problem. In the previous chapter, this was achieved by splitting each polyline into only few polyline segments of equal length. This coarse representation was used in the numerical optimization, and the computed opacity values were then interpolated on the original polyline. The splitting solely depended on arc length. Even though the lines were typically split into only few – at most 8 – polyline segments the total number of segments n could have become high, as all lines were treated equally regardless of their contribution to the image.

The idea of this chapter is to advance from a static arc length-based splitting to a *view-dependent* distribution of polyline segments. This way, not only the problem size is decreased, but effectively a better distribution of opacity samples is achieved in a sense that lines closer to the camera obtain a locally higher line sampling rate, see Fig. 4.2, and lines outside the viewport are not taken into account. We achieve this by a hierarchical representation of polylines that enables an efficient extraction of n coarse, adaptively sampled polyline segments used for the opacity optimization from a potentially large total number of candidate segments c. In all our examples, we use $c = 20,000$.

To accelerate the assembly of the system we propose a faster approximation to calculate the occlusion degree. Previously, we used a multi-sampled full resolution pass to construct fragment linked lists, which were used for both rendering and determining the occlusion degrees. Approximating the occlusion degree in a single-sampled quarter resolution pass produces less fragments to process on the CPU side and allows to discard invisible lines and segments from the rendering pass completely. Both improvements accelerate opacity optimization greatly and apply similarly for the steady case.

Further, we show how to achieve temporal coherence for sets of line fields in the presence of time-dependent data. This requires an extension of the energy term as well as an interpolation of opacities across hierarchy levels (see Section 4.2).

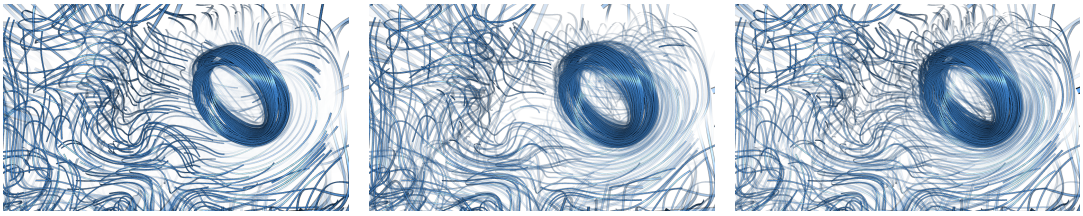


Figure 4.2.: BORROMEAN RINGS. A better distribution and higher number of polyline segments yield details around important structures, here, the field lines close to the ring. Left: original opacity optimization from Chapter 3 for $n = 6\,k$ with 431 ms/solve. Center: novel hierarchical approach for $n = 1.4\,k$ with 65 ms/solve, and right for $n = 4\,k$ with 147 ms/solve.

4.1. Hierarchical Opacity Optimization

4.1.1. Hierarchical Polyline Representation

In an off-line preprocess, we build a hierarchical representation for each polyline in the input set of lines as follows. The initially given polyline is assumed to be sampled uniformly with respect to the arc length, possibly at a high sampling rate. These uniform segments represent the finest level of the hierarchy. The coarser levels are built iteratively in a bottom-up approach: for the next coarser level, every pair of adjacent segments in the current level is collapsed into one single segment. For an uneven number, the remaining segment is left as is. Doing this recursively until there remains only one root segment yields a binary tree with leaf nodes referring to the original segments and internal nodes referring to recursively collapsed segments. Finally, doing this for all lines, the union of all line representations defines a *forest* of binary trees. If the input is a set of line fields (e.g., an animated line field), we process every field in the set individually and obtain a set of forests, which are discussed further in Section 4.2. Fig. 4.3 illustrates the setting of a single line field containing two lines.

4.1.2. View-dependent Adaptation of the Problem Size

From the hierarchical representation of lines as a forest of binary trees, we determine the discretization of polylines for the numerical optimization dynamically based on the current view. The selection of an appropriate set of resampled lines refers to finding a cut through the forest. An ideal cut should make the projected lengths of the joined polyline segments, i.e., the nodes' footprints, as equal as possible. The rationale of the heuristic is to obtain an as uniform as possible discretization of opacity samples. Note that while a low cut yields fine granularity of opacity samples, a higher cut would still

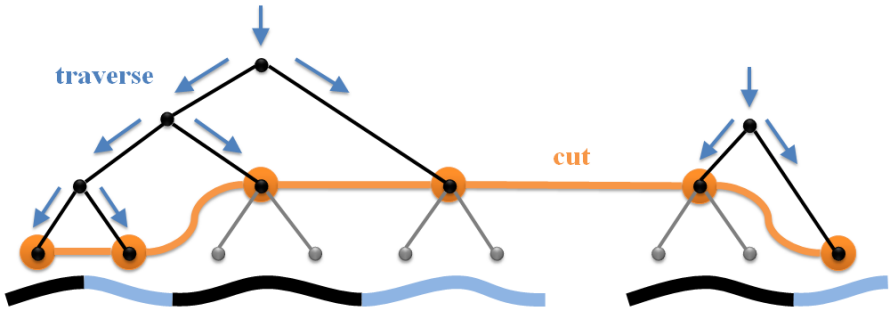


Figure 4.3.: Illustration of the hierarchical representation (Section 4.1.1) for two lines. For finding a cut (orange nodes) the tree is traversed (blue arrows) top-down (Section 4.1.2).

maintain a fair distribution of samples, which improves visual quality (see Fig. 4.2). We use a breadth-first search to efficiently find a good approximation to such a cut. Thereby, each node is associated with the bounding box of the represented polyline segment. Starting from the root nodes of the trees, we traverse toward the leaf nodes with two stopping criteria: First, the traversal stops at a node when the bounding box of this node is completely outside the view frustum. In this case, the node is "trivially" rejected and neither considered for the opacity optimization nor the rendering. Second, the traversal stops when a node reached a desired viewport footprint.

The footprint threshold f is a user parameter that balances the solve rate (and hence interactive response) versus the number of possible opacity changes along the lines. An exact calculation of the viewport footprint of a given polyline is view-dependent and quite expensive as it would (likely) require a rasterization of the polyline. Instead, we approximate the footprint. The footprint of a leaf node is approximated by assuming that the polyline segment that is represented by the leaf node is a straight line. For a sufficiently fine discretization of the polyline into segments this assumption is valid. We project the bounding box corners of the polyline segment (which are view-independent and precomputed) onto the viewport. For small (straight) segments, the diagonal of the screen space bounding box that contains the eight projected corner vertices, is a practical approximation to the footprint. That is, the footprint can be easily evaluated without rasterization. For internal nodes the footprint of the children is summed up. Fig. 4.3 illustrates the finding of a cut conceptually and Fig. 4.4 visualizes a cut for $f = 30\%$ of the viewport diagonal.

The view-dependent selection is particularly helpful in large scenes, for instance the blood vessel in Fig. 4.5. There, only about 33% of the polyline segments are visible. As shown in the figure, the view-dependent selection of polyline segments from a large set of candidates produces a better set for the optimization compared to the fixed, precomputed subdivision. For using computed opacity values as an initial guess in the

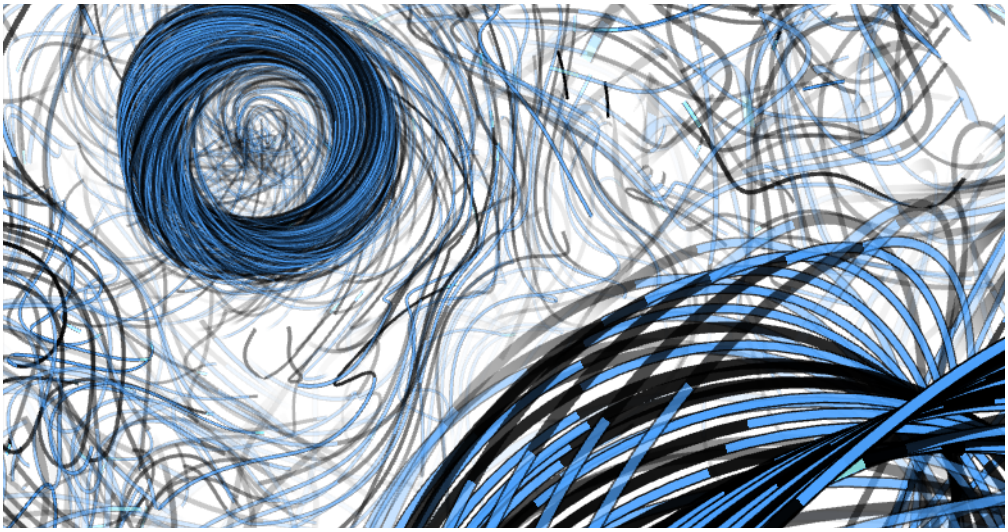


Figure 4.4.: Visualization of a cut with color-coded segments (alternating blue/black). Note that more polyline segments are spent in the foreground. ($n = 2.5k$ segments).

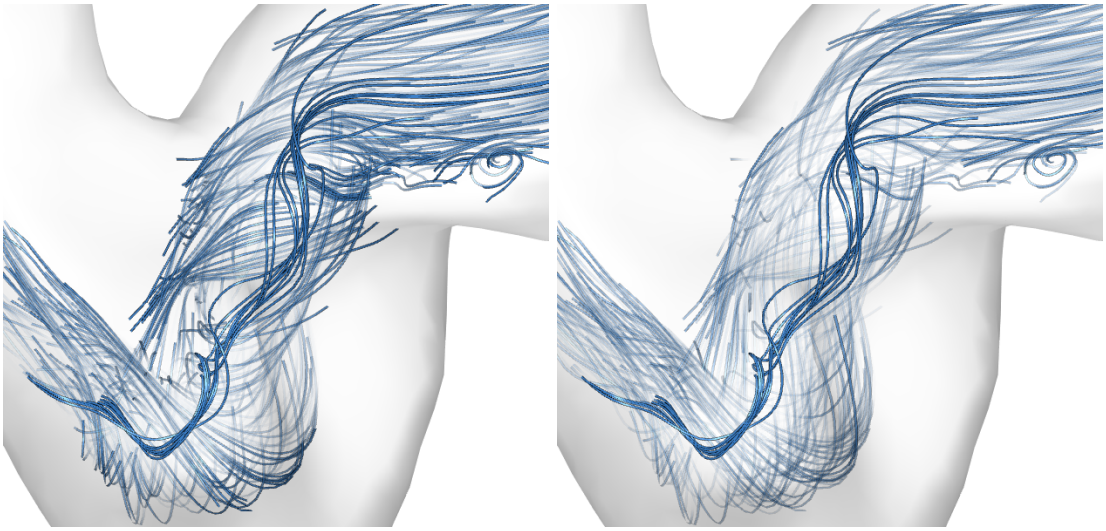


Figure 4.5.: BYPASS. Vortex core in a blood vessel. Left: standard opacity optimization ($n = 8k$ at 365 ms/solve). Right: novel approach provides clearer view at the vortex core (cut contains $n = 1.8k$ polyline segments at 60 ms/solve).

next solving iteration, the opacities are propagated in the hierarchy from the old cut to the new cut. Fig. 4.6 illustrates this and the aforementioned culling of segments.

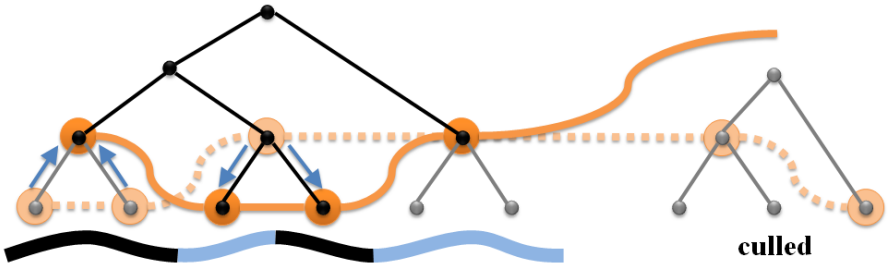


Figure 4.6.: Propagation of opacity values between cuts in the hierarchy. Thereby, invisible polyline segments are culled. The dashed line represents the old cut, the solid line is the new cut.

4.2. Processing Parametric Sets of Line Fields

Our goal is to visualize sets of line fields that may depend on additional parameters such as time. Examples are animated pathlines and streaklines, as shown in Fig. 4.1. Another class of parameter-dependent integral curve is the so-called *massline*. Masslines arise when considering finite-sized particles in fluids, which will be subject of the third part of this thesis. We formally introduce these lines in Chapter 10. For now, we can consider them as another type of time-dependent line field.

For an overview of the *space of pathlines*, there is an alternative to animation: Rendering *all* lines in a static scene and letting the opacity optimization put focus on the most important events of the entire sequence. In this case, time is an additional dimension which leads to a larger amount of data. This case can be handled effectively by the hierarchical optimization as is.

4.2.1. Enforcing Temporal Coherence in Opacity Values

Typically, the visualization of parametric sets of line fields is an animation of lines over time. As an example, Fig. 4.7 shows a sequence of time steps of a streakline animation. Animations require to solve an additional problem: the sequence of frames should be coherent over time. Thereby, we have to carefully distinguish between coherent opacity values on the lines – this is the problem we solve – and the possibly incoherent behavior of the lines themselves. The latter means that lines may suddenly disappear, when line parts leave the domain or when integral lines suddenly change their direction due to an insufficient temporal resolution of the animation. This behavior is inherent to the data and should therefore not be faded out (even though it may be due to artifacts, e.g., from sampling).

In order to ensure temporal coherence of opacity values, we introduce an additional

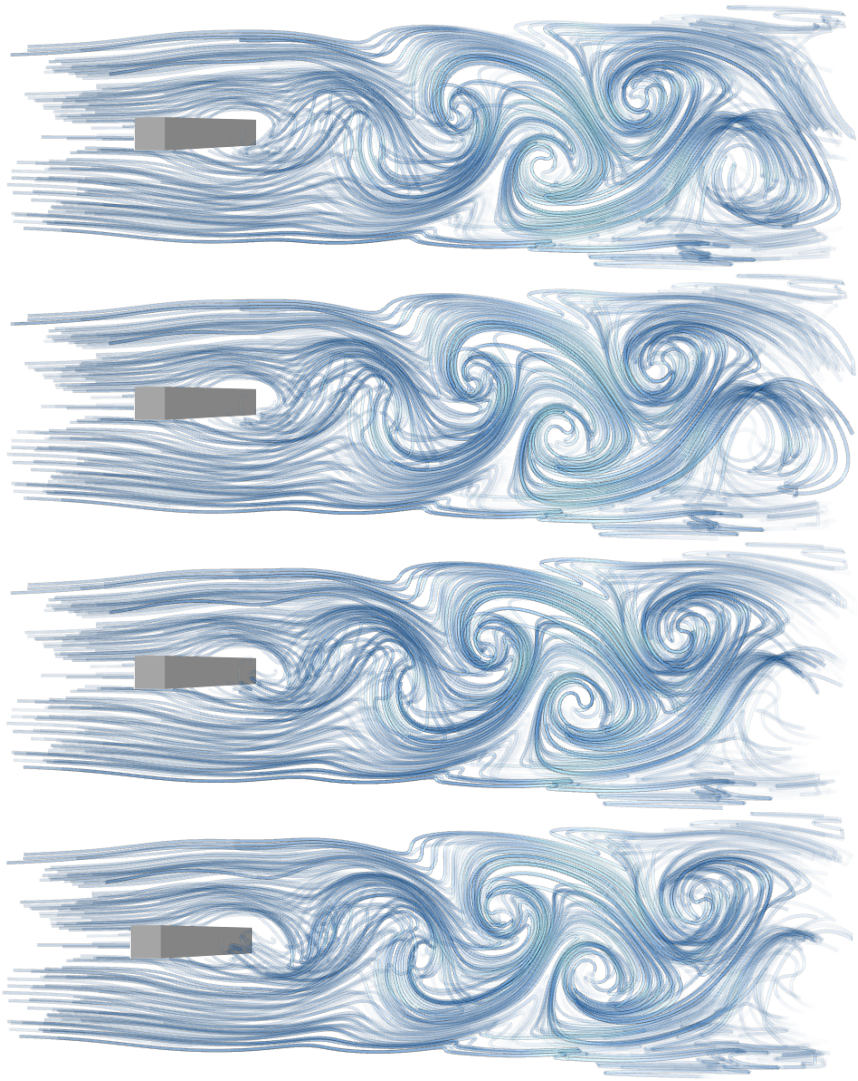


Figure 4.7.: Time steps of the evolution of streaklines forming a vortex street behind a SQUARE CYLINDER. Time advances from top to bottom.

term in the global energy:

$$
\begin{aligned}
E = & \frac{p}{2} \sum_{i=1}^{n} (\alpha_i - 1)^2 \\
& + \frac{q}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\alpha_i \, (1 - g_i)^{\lambda} \, h_{ij} \, g_j \right)^2 \\
& + \frac{r}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\alpha_i \, (1 - g_i)^{\lambda} \, h_{ji} \, g_j \right)^2 \\
& + \frac{s}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{a_{ij}}{2} (\alpha_i - \alpha_j)^2 \\
& \boxed{+ \frac{u}{2} \sum_{i=1}^{n} \left(\alpha_i - \alpha_i' \right)^2}
\end{aligned}
$$

again, with *bounded* variables $0 \leq \alpha_i \leq 1$. The new (boxed) term penalizes deviation from the solution α' of the previous iteration and hence introduces "inertia" to suppress sudden changes. Note that thereby inertia depends on the solving rate. The new parameter u balances smoothness of the animation versus accuracy in a sense of deviation from the minimizer of the original energy with $u = 0$. The choice $u = 1$ (w.r.t. the normalization $p = 1$ suggested in Section 3.1) showed good results in all our experiments.

4.2.2. Inter-frame Interpolation of Opacity Values

To improve the performance of the numerical solver, we provide the previously computed opacity values α' as initial guess to the next solving iteration. As mentioned in Section 4.1, we represent time-parametric sets of line fields as sequence of hierarchical representations, i.e., one forest per time step. Consequently, we have to interpolate opacity values not only within cuts in one forest but between two forests: Then, a correspondence problem arises for the transport of opacities α_i from nodes of the current cut to a possibly topologically different cut in the next time step. Our approach is visualized in Fig. 4.8. First, we propagate opacity values in the current time step down to the leaf nodes. From there, opacities are copied to leaf nodes in the next time step, i.e., the next forest. Since leaf nodes represent polyline segments of roughly equal length, leaf nodes of the same index can be assumed to be spatially close to each other. If a line has grown and no corresponding segment exists, its opacity value is set to 0, i.e., "invisible". Then, opacity values are propagated upward from the leaf nodes to the new cut.

There are several heuristics imaginable that establish segment correspondence at the base level. An advection of opacities (copy to next index) induces a flow of opacities along lines, while the index-to-index correspondence (which we used) holds opacities in position. Which is better depends on the lines behind and in front, since if the location of the crossing does not move it is better to hold the opacities in position. If the location of the crossing is advected with the flow instead, advection gives a better estimate for the next time step. In short, we can find examples where one yields the better result or the other. In practice, we found the index-to-index correspondence to give overall better visuals.

4.2.3. Hiding the Latency of the Solver

In the original opacity optimization in Chapter 3, opacity values of the polyline segments fade slowly toward the newly computed solution to hide the latency of the solver.

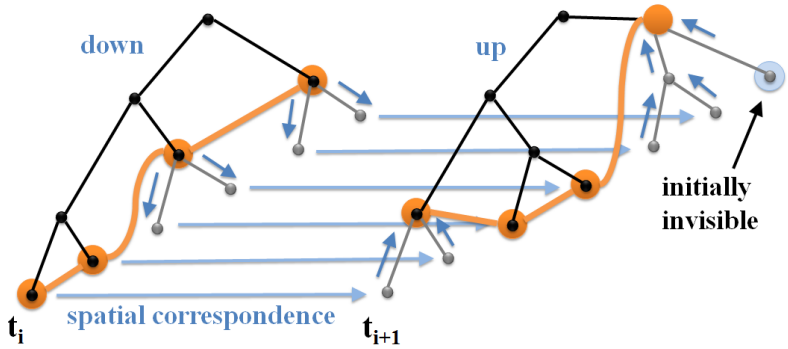


Figure 4.8.: For propagating opacity to the next time step a correspondence at the leaf nodes is required. Polyline segments without a corresponding segment in the previous time step are initialized as invisible, i.e., $\alpha_i = 0$.

Fading opacity in animated sequences requires their projection onto the line set in the next time step, i.e., a correspondence has to be established. Depending on the type of integral curve, a meaningful correspondence is naturally given, i.e., opacity can be copied to the subsequently released particle (e.g., pathlines or streaklines) or it can be advected with the vertices (e.g., timelines or masslines). Thus, we store the current opacity values used for display not per node (polyline segment) as in Chapter 3, but as a vertex attribute. We acquire the opacity values to fade to by interpolation of solutions from nearby polyline segments. For this, we can use the blending weight parameterization from Fig. 3.4, which was used in the original algorithm to interpolate opacities on the lines.

4.3. Implementation and Technical Details

4.3.1. Initial Line Set

For the display of all pathlines, we place seeds on the space-time boundaries to ensure a dense sampling. Animated pathlines and streaklines were seeded in inlet regions. Masslines in the HELICOPTER HOVER data set are released from the sediment bed, as they depict sand particle behavior. Clustering lines and choosing representatives [CYY*11, YWSC12, MJL*13] is a possible improvement to reduce the size of the input set required to represent the field. Given the line set, we divide all lines into a total of c polyline segments such that all segments have roughly equal lengths. We use $c = 20k$ in all our examples. Afterwards, the forests are formed bottom-up for each line field individually.

4.3.2. Matrix Setup Using the Hierarchy

Compared to the original method in Chapter 3, we made a few detailed modifications due to the hierarchical data structure when setting up the system matrix. Counting the occlusion degrees $h_{i,j}$ requires to identify the corresponding polyline segment (i or j) for each fragment. In the previous chapter, we proposed a parameterization that yields in the hierarchical case the index of the polyline segment at the leaf level. In addition, we maintain for each leaf node a pointer to the representing inner node that is currently selected by the cut. For identifying whether cut nodes are adjacent ($a_{ij} = 1$), the left-most and right-most leaf nodes of the cut nodes are tested for neighborhood. To speed up the test, the indices of these children are stored for each inner node after cut construction. For bottom-up propagation of opacities (and initially the importance g) in the hierarchy, the entries of child nodes are averaged. For top-down propagation opacities are copied to the child nodes.

Reformulating the bounded-variable least-squares problem from Section 4.2 to:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x} + \text{const} \\ &\text{subject to} && 0 \leq \mathbf{x}_i \leq 1. \end{aligned}$$

with $\mathbf{x} = (\alpha_1, ..., \alpha_n)^{\mathrm{T}}$, yields

$$\begin{aligned} \mathbf{Q} &= p \cdot \mathbf{I} + q \cdot \mathbf{W} + r \cdot \mathbf{V} + s \cdot \mathbf{D}^{\mathrm{T}}\mathbf{D} + u \cdot \mathbf{I} \\ \mathbf{c} &= (-p - u \cdot \alpha'_1, \ldots, -p - u \cdot \alpha'_n)^{\mathrm{T}} \end{aligned}$$

As before, $\mathbf{I}$ is the identity matrix. The matrices $\mathbf{W}$ and $\mathbf{V}$ are both diagonal matrices with entries $\mathbf{W}_{ii} = (1-g_i)^{2\lambda} \sum_{j=1}^{n} h_{ij}^2 \, g_j^2$ and $\mathbf{V}_{ii} = (1-g_i)^{2\lambda} \sum_{j=1}^{n} h_{ji}^2 \, g_j^2$, and the matrix $\mathbf{D}$ is the backward difference operator. A derivation is contained in Appendix A.

4.3.3. Decoupling Rendering and System Setup

In Chapter 3, we used a single rendering pass of *all lines* to acquire fragment linked lists that were used for both rendering and occlusion degree extraction on the CPU. We found that splitting them into two passes led to a significant frame rate improvement, as measured later in Section 4.4.2. In the first pass, we construct the linked lists at quarter viewport resolution and with single sampling. Using these shorter, separate lists to compute approximations of the occlusion degrees reduces the amount of data to copy to the CPU and shortens the time spent on iterating them when counting occlusion pairs. A reduction of the resolution is justified here, since the lines typically cover multiple pixels due to their expansion to triangle strips in view space. Moreover, the subsequent high quality rendering pass is no longer required to render all lines, but only polyline segments in the view frustum with an opacity $\alpha_i > 0$.

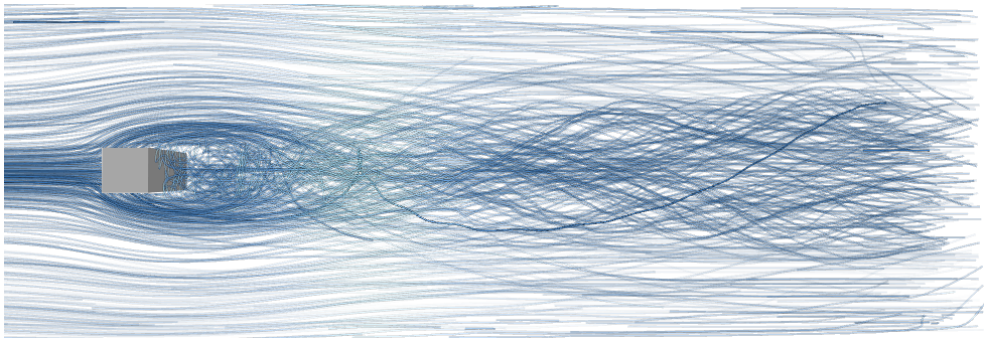


Figure 4.9.: Set of all pathlines in the SQUARE CYLINDER sequence. The opacity optimization shows an overview of events for any point in time (74 ms/solve, $n = 2k$).

4.4. Results

In this section, we show examples for visualizations generated by our hierarchical approach. Afterward, we discuss its performance. As we are processing time-dependent data, we would also like to refer to the video of the corresponding paper [GRT14].

4.4.1. Visualization of Steady and Time-dependent Data

Square Cylinder. Fig. 4.1 shows the unsteady wake of a square cylinder by animated pathlines and streaklines—the latter revealing a vortex street. The uniformly resampled version of this vector field sequence was provided by Tino Weinkauf (see [vFWTS08]) and is based on the Navier-Stokes simulation by Camarri et al. [CSBI05]. The set of all pathlines is shown in Fig. 4.9, depicting events from any point in time.

Helicopter Hover. The second unsteady sequence captures the simulation of the flow regime around a helicopter in slow forward flight close to the ground (cf. Section 3.3.2). The data was simulated by Kutz et al. [KKKK12] and is studied to understand how sand and dust clouds develop around rotorcrafts. This phenomenon is called brownout and is subject of Chapter 10. Fig. 4.10 depicts the data from two perspectives using streaklines. In this data set, masslines were used to analyze relationships of the finite-sized sand particles. A massline connects all the locations that finite-sized particles of different mass reach after a certain integration time, when released from the same seed point. In Fig. 4.11 a vortex ring is visible that stirs up particles from the loose sediment bed, hindering the view of the pilot.

Wall-mounted Cylinder. Frederich et al. [FWT08] simulated the air flow around a short wall-mounted cylinder (see Section 3.3.2). The view on animated pathlines in

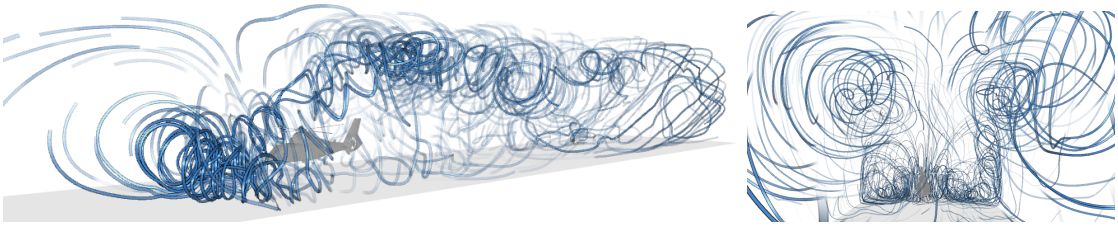


Figure 4.10.: Animated streaklines in the HELICOPTER HOVER sequence from outside and inside the domain. The location of the vortex on the ground is best seen in the left image (65 ms/solve, $n = 1k$). In the image on the right, the camera looks at the two vortices in the air. There, the vortex on the ground is visible in the distance (73 ms/solve, $n = 1k$).

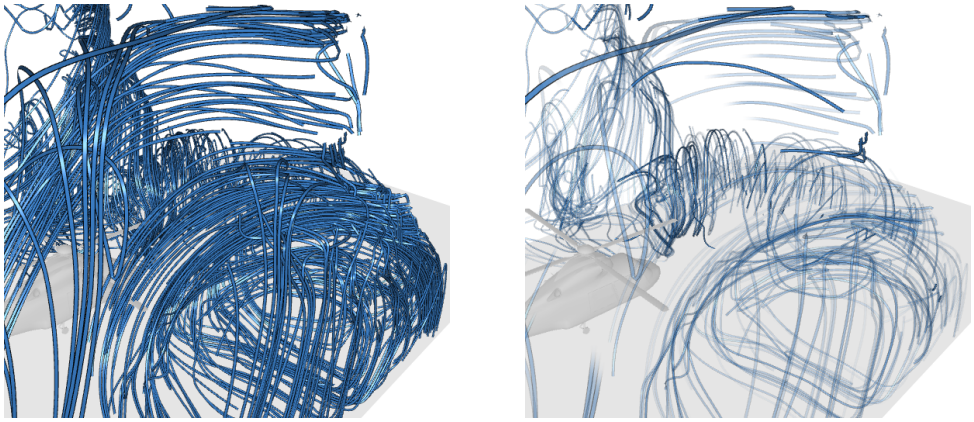


Figure 4.11.: Masslines forming a vortex ring in the HELICOPTER HOVER sequence. Left: cluttered input line field, right: our method (77 ms/solve, $n = 612$).

Fig. 4.12 in the wake of the cylinder is hindered by a laminar layer that is removed by the opacity optimization.

Borromean Rings. If the spatial relation of events permits it, individual time steps of an unsteady sequence can be shown simultaneously in one static view, which consequently contains an increased number of lines. Candelaresi and Brandenburg [CB11] simulated the decay of a magnetic field, with field lines initially interlocked in the shape of Borromean rings (cf. Section 3.4). During the simulation, the rings decay to two small magnetic knots that drift apart, as visualized in Fig. 4.13.

Helicopter Descent. The following steady data sets were used to demonstrate the improved performance (Section 4.4.2) and visual quality of the novel hierarchical optimization. In Fig. 4.14 (left) streamlines depict the air flow around a descending helicopter, viewed from below without the helicopter geometry as in Section 3.4.

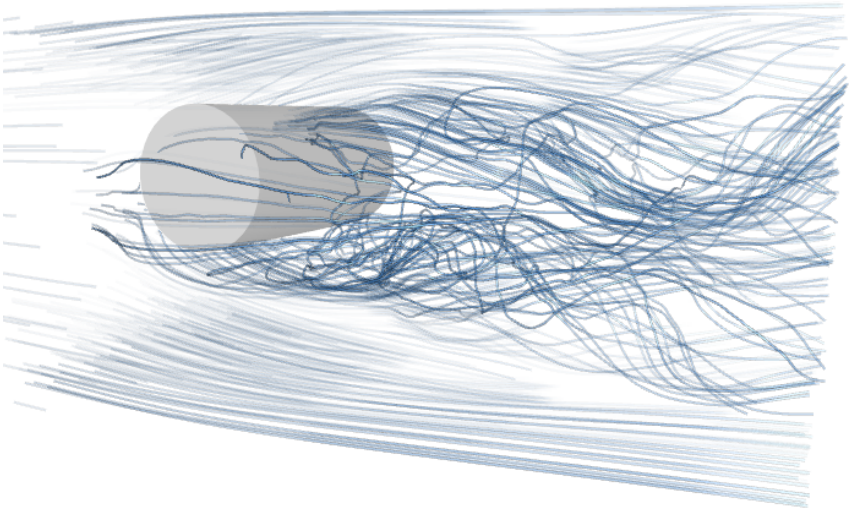


Figure 4.12.: Wake of a WALL-MOUNTED CYLINDER shown using animated pathlines (83 ms/solve, $n = 1.4k$).

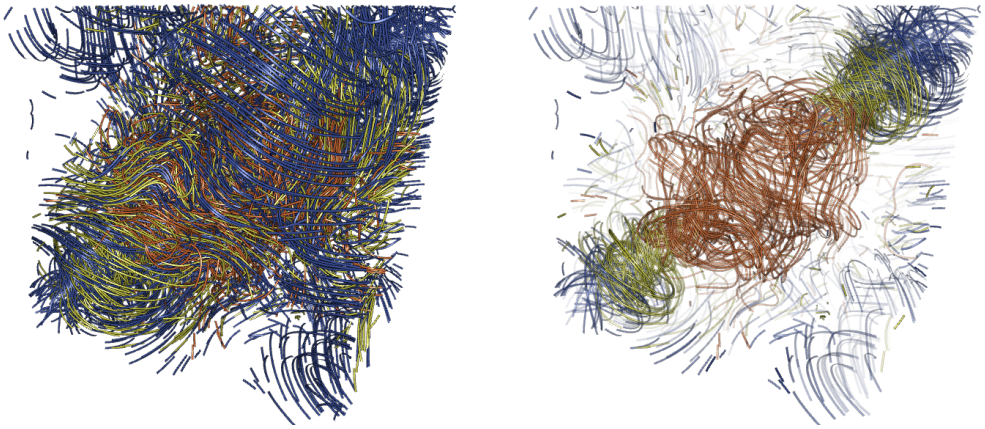


Figure 4.13.: BORROMEAN RINGS. Multiple magnetic fields from a time series of a magnetic flux decay simulation. The time steps are color-coded from orange to yellow to blue. Left: input line fields, right: visualization of the two knots therein moving apart (160 ms/solve, $n = 4k$).

Blood Flow in Bypass and Aneurysm. As an example for medical applications, we use hemodynamics data acquired at the Leipzig Heart Center and the University Hospital of Magdeburg (processed at Institute of Fluid Dynamics and Thermodynamics in Magdeburg). We visualize blood flow in an aortic bypass (Fig. 4.5), measured by PCMRI and in an aneurysm at a pathological vessel (Fig. 4.14, center) as in Section 3.4.

Rayleigh-Bénard Convection. In Fig. 4.14 (right) the camera is placed inside a Rayleigh-Bénard convection that arises if a thin layer of liquid is heated from below (cf. Section 3.4). The opacity optimization clears the view on the cells.

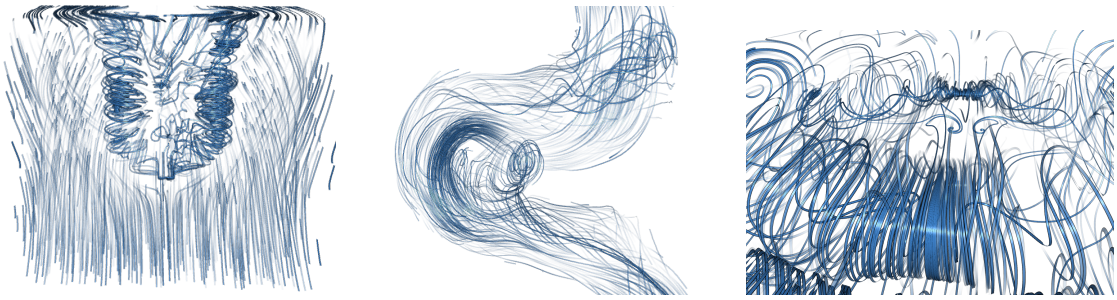


Figure 4.14.: Results in steady data sets. From left to right: Vortices behind a HELICOPTER in descent (95 ms/solve, $n = 2k$), blood flow through an ANEURYSM of a pathological vessel (55 ms/solve, $n = 2k$), and field lines in a RAYLEIGH-BÉNARD CONVECTION (110 ms/solve, $n = 1.4k$).

4.4.2. Performance

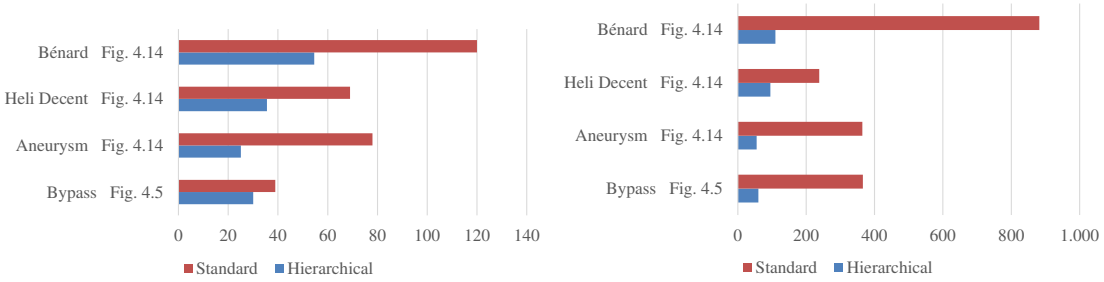
For the following performance measurements, we used an Intel Core i7-2600K CPU with 3.4 GHz and an Nvidia GeForce GTX 560 Ti GPU. All results were computed for a resolution of 1200×1000 with $16\times$ coverage-sampling anti-aliasing (CSAA).

The hierarchy construction took $4-8$ ms per line set, which corresponds to $50-60\,\%$ of the total preprocessing time. Timings for our sequences and an average time per set (which corresponds to the steady case) are listed in Table 4.1. The hierarchy data structure contains parent and child pointers, bounding boxes, importance values and several pointers for faster access, see Section 4.3.2. For 500 lines, each consisting of 40 segments, this yields less than 2.5 MB in total per line set.

We compared the solve rate and frame rate of the original method from Chapter 3 with the novel hierarchical approach in Figs. 4.15a and 4.15b for the benchmark data shown in Figs. 4.5 and 4.14. We observed a $1.3-3.1\times$ speedup for rendering and a $2.5-8.0\times$ speedup for solving while maintaining a similar visual quality for the generated images. Using the standard method, the solver provided at least one solution per second for almost all data sets, which matches the results reported in the previous section. The

Table 4.1.: Preprocessing timings for the SQUARE CYLINDER (streaklines), WALL-MOUNTED CYLINDER (pathlines) and HELICOPTER HOVER (masslines) sequences.

Preprocessing	**Sq. Cyl.**	**Wall-mt.**	**Helicopt.**
Number of line sets	200	400	250
Average per set	14.04 ms	6.97 ms	10.60 ms
Total Time	2.81 s	2.79 s	2.65 s



(a) Rendering time in milliseconds.

(b) Solving time in milliseconds.

Figure 4.15.: Rendering and solving timings.

blood flow in the BYPASS data set in Fig. 4.5 contains off-screen vessel geometry that is filled with irrelevant streamlines. Generally, as soon as the camera moves into the domain (e.g., into the vessel's bounding box), the speedup factor for solving increases due to the culling in the view-dependent selection of polyline segments. (Note that *culling* in this context refers to decreasing the problem size for the solver.)

Next, we evaluate the use of the previous solution as initial guess for the next solve. For this, we quantify the speedup for solving and the cost for propagation of opacities between solves for a *still camera*. Then, in the steady case, the given initial solution is a "perfect" guess, which thus results in the best possible performance. However, this performance is generally unreachable for animated line fields, where the system of equation changes. But it states an upper bound that we strive to approach. Not using an initial guess resulted in a worse performance. The timings for solving in Table 4.2 indicate that the propagation of the solutions from the previous frame to the next solver iteration yields a constant speedup of $5-12$ ms in our tests.

Table 4.3 summarizes timings for the different steps of the algorithm for both rendering and solving (including setup of the system matrix), which are executed in different threads. The given timings are averages for one complete turn-over in the threads' main loops, including all synchronization barriers.

Table 4.2.: Benefits of an initial guess (on/off) for the next solve iteration. Shown are the solver turn-over times in milliseconds for the SQUARE CYLINDER, WALL-MOUNTED CYLINDER and HELICOPTER HOVER data, each for a fixed view.

Test Case	**Sq. Cyl.**	**Wall-mt.**	**Helicopt.**
Steady	51.7	75.6	71.6
Unsteady (on)	52.8	83.2	73.2
Unsteady (off)	64.2	88.4	80.3

In the following, the steps and their bottlenecks are discussed. The fading toward new opacity solutions is fast and primarily depends on the number of vertices in the scene. The speed of the low resolution pass depends on the image overdraw. Incidentally, the time for assembling the system matrix goes up, as more time is required to compute the occlusion degrees. The high quality rendering pass, however, does not need to render all the lines. Thus, it has less overdraw and is therefore faster. It depends on the number of segments chosen by the solver, which in turn depends on the user parameters of the energy. The computation time for the cut depends on the number of segments to begin with and the recursion depth. The propagation of opacity values from the previous frame depends on the number of segments. The same holds for the solving performance. If the input line set is larger, while targeting the same problem size, fewer line segments are available per line. In summary, the time-critical parts are the system setup and the solving. The first scales with the number of input lines and the latter depends on the number of polyline segments chosen in the cut.

The total solving performance is suitable for a visualization of animated line fields. For most scenes, we were below $60\,\mathrm{ms}$ and observed up to $83\,\mathrm{ms}$ only in scenes with high overdraw. For steady scenes, we increased the number of lines and polyline segments, thus, the solving time went up. Our worst case example was the BORROMEAN RINGS scene in Fig. 4.2 with up to $147\,\mathrm{ms}$. There, we have shown that a higher cut at $65\,\mathrm{ms}$ produced better results than the original method from Chapter 3 with $431\,\mathrm{ms}$.

Compared to the original method, the hierarchical approach performs significantly

Table 4.3.: Performance of the different algorithm steps for the SQUARE CYLINDER, WALL-MOUNTED CYLINDER and HELICOPTER HOVER data (all unsteady) with timings given in milliseconds. The upper part is executed in the rendering thread, the lower part is executed in the solver thread.

Pipeline Step	**Sq. Cyl.**	**Wall-mt.**	**Helicopt.**
Segments n (cut)	2.5k	1.4k	1k
Alpha Fading	0.01	0.01	0.01
Low Res Pass	1.06	4.17	7.82
HQ Pass	13.51	20.75	19.76
Turn-Over	**14.58**	**24.93**	**27.59**
Compute Cut	1.02	0.83	0.63
Assemble Matrix	42.17	61.45	62.95
Propagate α'	0.18	0.18	0.16
Solve	9.43	20.78	9.52
Turn-Over	**52.80**	**83.24**	**73.26**

better with a speedup factor of about 8. This performance improvement stems from our view-dependent (per frame) selection of polyline segments for which the opacity optimization is solved. The hierarchical data structure to accelerate this selection is preprocessed. Reusing previous solutions as initial guess requires a scheme to transport opacities between hierarchies at runtime.

4.4.3. Discussion and Limitations

We demonstrated that the proposed hierarchical opacity optimization enables interactive exploration of sets of line fields. For meeting the higher performance demands of animated line field exploration, the novel method does not need to sacrifice visual quality for speedup – quite the contrary. The improved polyline discretization for the opacity computation leads to an improved rendering quality at a smaller problem size. However, this makes it somewhat difficult to faithfully compare the standard and hierarchical methods: for the same problem size n they yield generally different results. In our experiments, the effective values of n are usually lower for the hierarchical approach in the comparisons. The magnitude of n is steered by the maximum footprint threshold (Section 4.1.2), which is the user parameter that defines the visual quality achieved by the optimization. We do not see much benefit in having a true "load balancing" in a sense of finding the best cut for a prescribed n: first, even a greedy search for such cut is more expensive. Second, a dynamic adaptation is preferable in practice, such that the footprint threshold is higher during interaction (low n) and lower at a user-defined value when the interaction pauses (high n).

Our hierarchy construction is based on binary trees. Other representations are imaginable as well and they might have additional benefits though most likely certain difficulties, too. Imagine the use of a graph instead, i.e., the connection of one base node to multiple parents. For example, consider the graph in Fig. 4.16. There, node 3 has two parents to blend values from. This is advantageous as downward propagation results in smoother transitions at the base level, since we can blend parent opacities at overlapping nodes. However, this configuration makes the definition of a seamless cut more complicated, as collapsed line parts overlap. Further, we thought of grouping child nodes during the bottom-up construction in a greedy manner, steered by the segment importance. This raises issues in the temporal coherence of the tree's topology. The same problem arises with more general tree topologies. The fix topology of a binary tree worked best for us, and already produced satisfactory results.

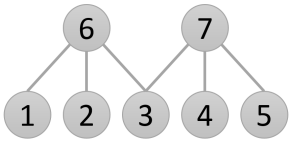


Figure 4.16.: Multiple parents.

In Section 4.1.2, we described an approximation of the viewport footprint of a curve

by summing up the bounding box extends of the leaf nodes in screen space. An exact calculation of the footprint is imaginable as well, but depends linearly on the number of vertices. Our approximation depends on the number of segments. Recomputing the exact footprint for every curve is currently too expensive for us. However, it could be reconsidered when a GPU acceleration is targeted for all pipeline stages. At present, this was not required and our approximation worked fine.

Our method shares the limitations of the standard opacity optimization: adequate input data is required such that all relevant features are represented. A local oversampling could be addressed in a preprocess, e.g., using a clustering approach on lines and visualizing cluster representatives. Furthermore, the animation of parametric line sets requires a correspondence between lines. This is not always given, for instance, in time-dependent magnetic fields. Streamline animations [JL00] might be a good start to search for a correspondence heuristic.

4.5. Summary

In this chapter, we extended our work from Chapter 3 so that sets of 3D line fields can be visualized at interactive rates. The main contributions consist in the therefore required design of a hierarchical representation of line sets and a modification of the energy together with an efficient interpolation of opacity values between time steps. The hierarchical representation enables a view-dependent and feasible problem size for opacity optimization even though the amount of input data may be magnitudes larger than for “steady” line sets. In addition, the opacity sampling is regularized which improves visual quality for similar problem size. The modification of the energy and the opacity interpolation provide temporally-coherent opacity changes during rendering and help with guessing initial solutions. With this we proposed the first global line selection approach for pathlines, streaklines and masslines in 3D time-dependent flow, and accelerated opacity optimization for general 3D line fields.

In the following chapter, we extend the steady opacity optimization (Chapter 3) in another direction: We move from lines to surfaces.

5 Chapter 5. Opacity Optimization for Surfaces

In this chapter, we apply the concept of opacity optimization to surfaces, which was described in the publication:

T. Günther, M. Schulze, J. Martinez Esturo, C. Rössl and H. Theisel
Opacity Optimization for Surfaces
Computer Graphics Forum (Proc. EuroVis) 33, 3 (2014), 11–20.

The idea of our method is similar as for lines, i.e., we adapt opacity to clear the view on important parts in the data. Here, the problem is even more challenging, because surface partition, opacity interpolation and the computation of the energy terms are not straightforward. In contrast to the line approaches, our surface method additionally respects visual cues of the rendering style, i.e., it controls the fading of silhouettes and halos. Examples are shown in Fig. 5.1.

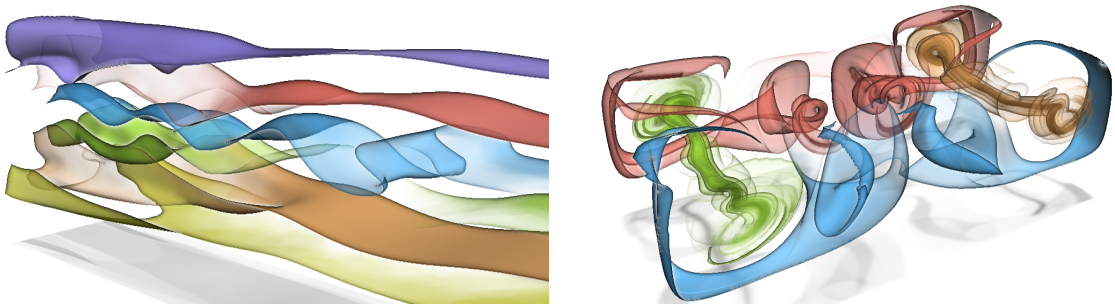


Figure 5.1.: Stream surfaces in a SQUARE CYLINDER flow and in a RAYLEIGH-BÉNARD convection (left to right). Our method fades out unimportant surfaces to clear the view on interesting structures, e.g., vortex cores. The visualizations are view-dependent, frame-coherent, at interactive rates (13–50 fps), and with 3–15 opacity updates per second.

5.1. Problem Statement and Contribution

Surface rendering is a classic tool in computer graphics. The perception of surfaces, however, is limited in the presence of occlusion to which the use of transparency is a common approach. Though, for multiple surfaces or even a single complex one, it might still be difficult to faithfully capture the data. See Fig. 5.2 for an example. In this chapter, we aim for a surface visualization that clears the view on the relevant parts, whereas relevance is again specified by the user.

The perception of surface order thereby resides in two key aspects: surface opacity and presentation of edges. The standard opacity mappings are *local* and accentuate grazing angles: angle-based transparency (2.28) and normal variation (2.29). When surfaces have low curvature or their boundary is not seen at a grazing angle, they are difficult to perceive. The more opaque a surface is, the better surfaces behind can be classified as such. Thus, our *first objective* is to only use transparency where strictly needed, which requires global occlusion information.

Carnecky et al. [CFM*13] applied insights from cognition to produce silhouettes that allow to deduce the order of surfaces. However, both the local transparency mapping approaches, as well as the straightforward and smart silhouette renderings tend to produce too many visual cues that occlude each other if surfaces are complex. Under certain views edges might appear that confuse the viewer, see Fig. 5.3. Therefore, our *second object* is to reduce the opacity of unimportant visual cues if they hide important structures.

While the established visual cues, such as opacity at grazing angles and edge enhancements, work well for many cases, they start to fail in complex scenarios. In this chapter, we propose a filtering of visual cues by transparency that is not just based on local

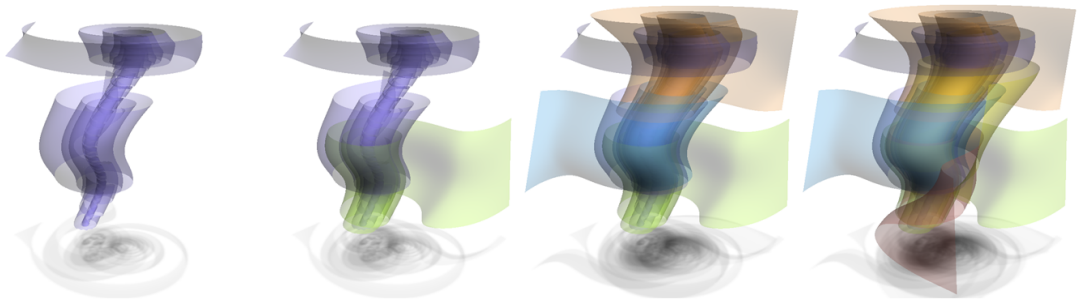


Figure 5.2.: A common approach to address occlusion of surfaces is to render them semi-transparently. When using multiple surfaces, or even a single complex one, perception might still be limited due to occlusion and difficulties in perceiving layer order.

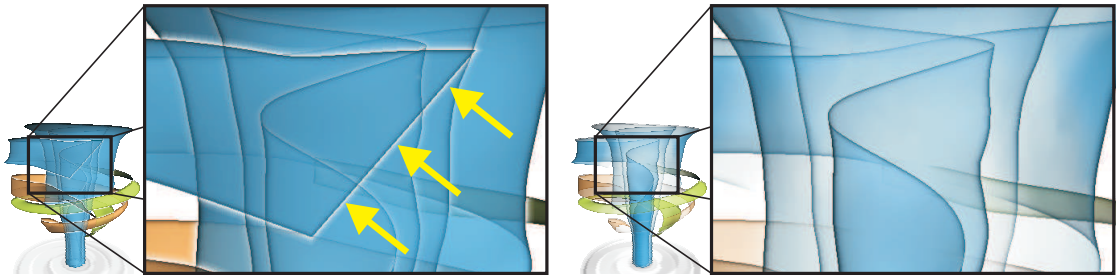


Figure 5.3.: Even for rather simple surfaces, silhouettes may be distracting if the object of interest is hidden. Opacity optimization: off (left) / on (right).

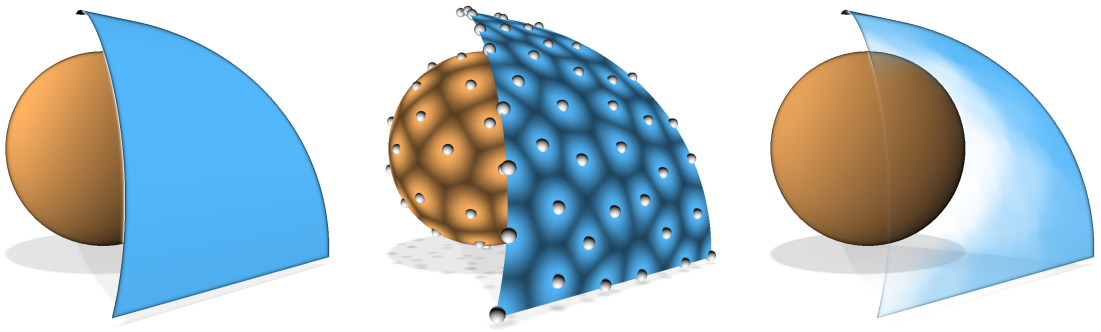
measures, but respects the occlusion of structures and their associated user-defined importance. Moreover, since we are only computing opacities, the method can be combined with any favorable rendering style or technique. For our purposes, we applied our view-dependent opacities to the silhouette rendering method of Carnecky et al. [CFM*13].

5.2. Opacity Optimization for Surfaces

In Chapters 3 and 4 we solved the occlusion problem for line rendering. In this section, we seize the formulation of opacity optimization as a global optimization problem, and we reformulate it such that it can be used for the rendering of steady surfaces. An overview of our method is shown in Fig. 5.4. Starting with a partition of the initial surfaces 5.4a into patches 5.4b, we compute for each patch an opacity by minimizing an energy that balances occlusion avoidance and presentation of visual cues. In our examples, cues originate from the smart transparency technique of Carnecky et al. [CFM*13]. For rendering, the resulting opacities are interpolated across the surface 5.4c.

5.2.1. Surface Partition

In a preprocessing step, we partition the given surfaces into patches. The preprocessing tool was implemented by Maik Schulze and is based on the geodesics in heat [CWW13] implementation of Janick Martinez Esturo. The partition of the surfaces corresponds to the subdivision of lines into polyline segments as in Section 3.1, however, this step is nontrivial for surfaces. In order to solve this problem, we apply a vertex clustering based on geodesic distance. This can be interpreted as computing discrete Voronoi cells on the manifold surfaces. Assume we are given a set of seed vertices on the surface that act as centers of cells or equally as cluster representatives. We compute



(a) Initial surfaces (b) Patches & centers (c) Result

Figure 5.4.: Overview: Starting with an initial set (a), we partition all surfaces into n patches (b) and compute an optimal opacity, which is interpolated on the surface (c). Here, we used $n = 100$ patches for illustration.

the approximate geodesic distance from each center to all other non-center vertices, and we assign each of these vertices to the cell whose center is nearest. Fortunately, the computation of highly accurate approximate geodesic distances can be done very efficiently by reformulation as a discrete heat-flow problem [CWW13]: This requires only the factorization of two sparse symmetric positive definite linear operators as the most expensive step. After that the computation of *all* distances can be done by back substitution. We use a farthest point seeding to place the centers. Starting from a randomly chosen vertex, we iterate the following step: place a new center at a vertex that has maximum geodesic distance to all existing centers. The iteration stops if the partition is fine enough, i.e., the cell size is below a threshold or a certain number of cells is reached.

The partition based on geodesic distance is reasonable as it tends to produce compact, "convex" cells with a good ratio of circumference to area. We observe that the partition by greedy farthest point seeding is, even though it is not perfectly regular, good enough for the purpose. This means that we can avoid global patch rearrangement, e.g., by a Lloyd relaxation [Llo82], which has a slow convergence.

5.2.2. Averaging and Interpolation

Given is a set of surface meshes with vertices $\mathcal{V}$. The partition of the surfaces is given as a map $C : \mathcal{V} \to \mathcal{C}$ where $\mathcal{C} \subset \mathcal{V}$ denotes the cell centers, which in turn represent patches. We use the preimage $C^{-1}(i) = \{\ell \in \mathcal{V} \,|\, C(\ell) = i\}$ to describe the set of all vertices in a cell. In the remainder of this section, we use indices i, j for centers and ℓ for general vertices. For the subsequent steps, we require operators for averaging and interpolation of values.

The first one is straightforward. Given some quantity x that is defined per vertex (denoted by x_ℓ), we define

$$\text{avg}_x(i) \;:=\; |C^{-1}(i)|^{-1} \sum_{\ell \in C^{-1}(i)} x_\ell \;,$$

where $|\cdot|$ denotes the cardinality of a set.

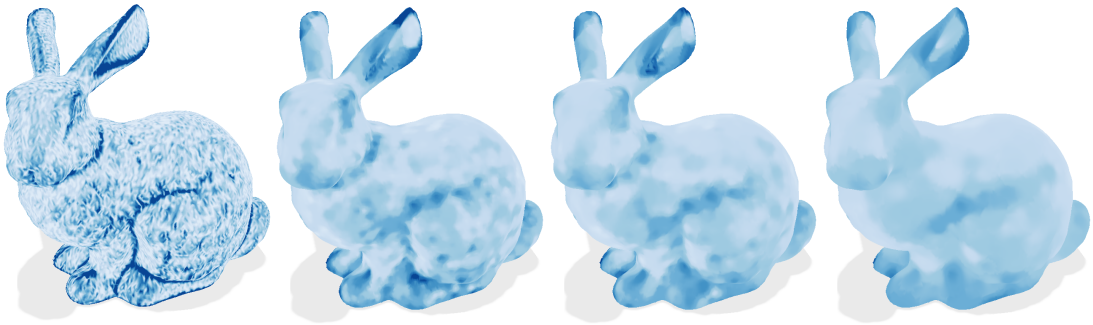
By *interpolation* we refer to computing new values at arbitrary vertices from values given at centers. This is a bivariate scattered data interpolation problem with samples distributed on a manifold. Finding a "high quality", smooth interpolating function is a nontrivial problem that is possibly computationally expensive, e.g., requires the solution of a linear system. For our application, we require first of all a highly efficient interpolation scheme. Thus, we locally apply Shepard's method (see, e.g., [HL93]), which is a kind of inverse distance weighting. Distances $d_g(\ell, i)$ from vertex ℓ to center i are the geodesic distances (see previous section). Given any vertex $\ell \in \mathcal{V}$ we define $\mathcal{N}_k(\ell) \subseteq \mathcal{C}$ as its k nearest centers w.r.t. d_g. To evaluate a quantity x we use the subset $\{x_i \,|\, i \in \mathcal{N}_k(\ell)\}$ of the values given at centers to define

$$\text{interp}_x(\ell, k) := \frac{\sum_{i \in \mathcal{N}_k(\ell)} w_i \, x_i}{\sum_{i \in \mathcal{N}_k(\ell)} w_i} \quad \text{with} \quad w_i = d_g(\ell, i)^{-2} \;.$$

with $\text{interp}_x(i, k) = x_i$ for centers $i \in \mathcal{C}$. The function $\text{interp}_x(\ell, k)$ provides interpolation of values at centers with constant precision but without any theoretical guarantees on smoothness and continuity when evaluated, e.g., for adjacent vertices. This is sufficient for our purpose, even though it is not a high quality interpolation. The main advantage is that this simple scheme can be evaluated efficiently. Its overall smoothness can, to some extent, be steered by the exponent for the inverse distance; we found that weights d_g^{-2} provided best results. Here we use $k = 3$, which is small for typical uses of Shepard's method. We do this because first, we found that in our setting the quality of interpolation is still good enough. Fig. 5.5 shows the quality of an interpolation of mean curvature that was averaged at centers for partitions of decreasing number of patches n and $k = 3$. Second, this allows to store the indices of nearest centers simply as a 3-vector, i.e., $\mathcal{N}_3(\cdot)$ is a coordinate look-up, which enables efficient interpolation on the GPU. We precompute nearest centers and interpolation weights once in a preprocess.

5.2.3. Optimization

Our surface opacity optimization is based on opacity optimization for 3D steady line fields, which is described in Section 3.1. The main differences consist in the use of surface patches instead of polyline segments and a modification of the energy



per vertex $n = 2000$ $n = 1000$ $n = 400$

Figure 5.5.: Mean curvature is estimated per vertex, then averaged per patch and interpolated for n patches.

functions. In particular, the coefficients g_i, h_{ij} and a_{ij} are computed differently. Note that all sums in the energy function now iterate over patches or their centers $i, j \in \mathcal{C}$, respectively. We abstain from a formal redefinition.

Change of Energy. We compute opacities for discrete elements. In the surface case, these are patches that result from partitioning. All quantities are measured on patches: patch *importance* g_i is an average of the importance values prescribed per vertex, for which choices are discussed in Section 5.2.4. For all examples, we use $g_i = avg_H(i)$, where H is the mean curvature estimated at every vertex. The *occlusion degree* h_{ij} is determined from rasterization on fragment level, which is detailed below in Section 5.2.5. We use a different notion of *adjacency* than for lines: The coefficients $a_{ij} > 0$ that contribute to the smoothness term are now no more either 0 or 1 but continuous values that reflect geodesic distance. Thereby, centers closer to each other receive higher penalty if their opacity differs. We define for centers $i, j \in \mathcal{C}$

$$a_{ij} := \frac{d_g(i,j)^{-2}}{\sum_{k' \in \mathcal{N}_{k+1}(i) \setminus \{i\}} d_g(i,k')^{-2}},$$

with the convention that $a_{ij} = 0$ whenever j is not one of the $k+1$ nearest neighbors of i, which can be interpreted as the denominator approaching infinity. Note that $i \in \mathcal{N}_{k+1}(i)$, since i is the closest center to itself. Thus, in total k neighboring centers are looked up.

Opacity values α_i, with $i \in \{1, \ldots, n\}$, are computed for each patch as minimizer of an

energy E that formalizes desired properties:

$$E = \boxed{\frac{p}{2}\sum_{i=1}^{n}(\alpha_i - t)^2} \tag{5.1}$$

$$+ \frac{q}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\Big(\alpha_i\,(1-g_i)^{\lambda}\,h_{ij}\,g_j\Big)^2 \tag{5.2}$$

$$+ \frac{r}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\Big(\alpha_i\,(1-g_i)^{\lambda}\,h_{ji}\,g_j\Big)^2 \tag{5.3}$$

$$+ \frac{s}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}(\alpha_i - \alpha_j)^2\,. \tag{5.4}$$

Compared to the original opacity optimization, we replaced the regularization by the boxed term, which includes a user-specified *target opacity* $0 < t \leq 1$. Thereby, we can prescribe a desired opacity, which allows to render generally transparent surfaces. The modified energy is still frame-coherent and leads to a least-squares problem with unknown opacity values bounded in $[0,1]$ (0 for transparent and 1 for opaque). We use the same algorithm for the numerical solution as in Chapter 3.

Evaluation of Opacity. Finally, the computed optimal opacity values a_i at centers $i \in \mathcal{C}$ are interpolated within patches such that every vertex $\ell \in \mathcal{V}$ is assigned an opacity

$$\alpha_\ell \;=\; \text{interp}_\alpha(\ell, k)\,.$$

At rasterization, opacity is linearly interpolated in triangles, resulting in a fragment opacity. As silhouettes are key to understand surface ordering, we apply a transfer function to the opacity of silhouette fragments to fade them out slower

$$\alpha \leftarrow 1 - (1-\alpha)^{\eta} \tag{5.5}$$

Increasing η keeps silhouettes longer, see later Section 5.3.2.

5.2.4. Importance Function

The importance g_i plays an important role, as it is the designated tool to characterize the regions and parts of the data worth to look at. In fact, the success of our method depends on its correlation to relevant structures. It might originate in local geometric information, might be sampled from scalar fields, or might be global information such as distance to a an object (such as a tumor in medical data). Consequentially, importance is strongly application-dependent and driven by domain knowledge. However,

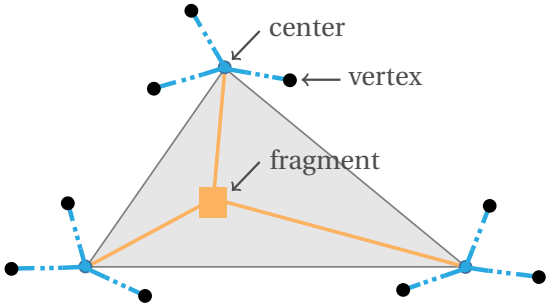


Figure 5.6.: Occlusion degrees h_{ij} are obtained by rasterization: The contribution of centers (•) to the rasterized fragment's (■) opacity is reconstructed. For each vertex (•) of the rasterized triangle the $k = 3$ closest centers are considered. (For illustration, their distances are shortened.) The center-to-fragment contribution is the product of interpolation weight (- -) and barycentric weight (—).

simple geometric measures exist that work well in many cases, e.g., in all examples, we used mean curvature. Note that for $g_i = 1$ surface parts do not vanish since Eqs. (5.2) and (5.3) evaluate to zero.

5.2.5. Computing Occlusion Degree on Fragment Level

The coefficients h_{ij} in the energy term measure the occlusion that patch i causes by occluding another patch j. Similarly to Chapter 3, we determine the occlusion after rasterization on fragment level, see Fig. 5.6. For each fragment we look up the centers and interpolation weights used for opacity interpolation. These are the $k = 3$ nearest centers for each vertex of the rasterized triangle and thus $3k$ centers in total per fragment (see Fig. 5.6). We maintain for every pixel a sorted linked list of the rasterized fragments [YHGT10], from which we obtain all "occluder-occludee" fragment pairs. Then we identify for every fragment pair the $3k$ centers $F \subset \mathcal{C}$ of the fragment in *front* as well as for every fragment *behind* the respective centers $B \subset \mathcal{C}$. Every fragment pair introduces a small amount of occlusion for the $3k \times 3k$ front-back center pairs i, j, with $i \in F$ and $j \in B$. The amount of occlusion is the product of the contribution of i to the occluder fragment and of j to the occluded fragment's opacity. For deducing the contribution of a center to a fragment, we store for each fragment the primitive ID (generated by the graphics runtime) and the barycentric coordinate of the fragment in the rasterized triangle. Based on the primitive ID, we look up contributing vertices, their k nearest centers and the interpolation weights. These weights are w.r.t. the triangle vertices, and they are multiplied by the barycentric weights to get per-fragment occlusion degrees. The total occlusion degree h_{ij} is determined as the sum of all contributions to fragments. We do not penalize self-occlusion of patches to prevent fading at grazing angles, i.e., we set $h_{ii} = 0$.

5.3. Usage and Parameters

This section elaborates on the choice of surfaces for visualizing flow data, the selection of parameters and their effect, as well as implementation details on rendering and solving.

5.3.1. Input Surface Set

Surface opacity optimization only reveals structures that are contained in the set, thus the set of input surfaces should be selected carefully. For the examples in this chapter, we used Hultquist's algorithm [Hul92] for extraction and placed the seed curve manually. All stream surfaces that were used in this chapter were provided by Maik Schulze. Dragging flow-aligning surfaces [MSRT13b] or using automatic placement methods [ELM*12, MSRT13a, SMG*14] as discussed earlier in Section 2.3.3 are possible alternatives.

5.3.2. Parameter Selection

We inherit the parameters of the original opacity optimization, which was introduced in Chapter 3. We discuss these parameters in the following. First, the total number of surface patches n is chosen. A higher number of patches leads to a more local adaptation of the surface opacity and thus preserves details, however, at the cost of higher solving timings. Fig. 5.7 depicts different choices for n. Most often, we used $n = 400$, cf. Table 5.2. The effect of parameters in the energy function E (see Section 5.2.3) are shown in Fig. 5.8: The weight q of the occlusion term (5.2) adjusts the overall opacity in the image. In Fig. 5.8 (left), a surface layer is removed to clear the view on the wake turbulences of a DELTA WING. The weight r of term (5.3) steers the removal of background clutter. In our example, the laminar stream surfaces in the background are removed to increase the visibility of the wake turbulence behind a WALL-MOUNTED CYLINDER. The weight s of the smoothness term (5.4) is used to enforce continuity of the surface opacity, as shown for the STATIC MIXER data set. If not mentioned otherwise, we set $s = 50$. Finally, the emphasis exponent λ in (5.2) and (5.3) stresses important structures: Fig. 5.8 (right) depicts the aerodynamics around a HELICOPTER in ground effect, studied for brownout conditions. Of particular interest is the green vortex in front of the helicopter, as it entrains sand particles that hinder the pilot's view.

In Section 5.2.3, we introduced two additional parameters both with a recommended default value. First, the target opacity t in (5.1) is used to let surfaces strive for being

almost, but not completely, opaque. A high value proved useful, as it supports the perception of layer order. We recommend $t = 0.9$, which is used in all examples. Second, silhouette opacity is steered by parameter η, introduced in (5.5), to control how fast silhouettes fade out. The effect is demonstrated in Fig. 5.9. As edges are helpful to hint how nearly invisible surfaces pass in front of important structures, a high value is recommended. In all our examples, $\eta = 10$ is used, as shown in Fig. 5.10.

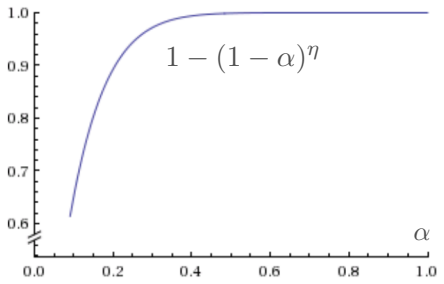
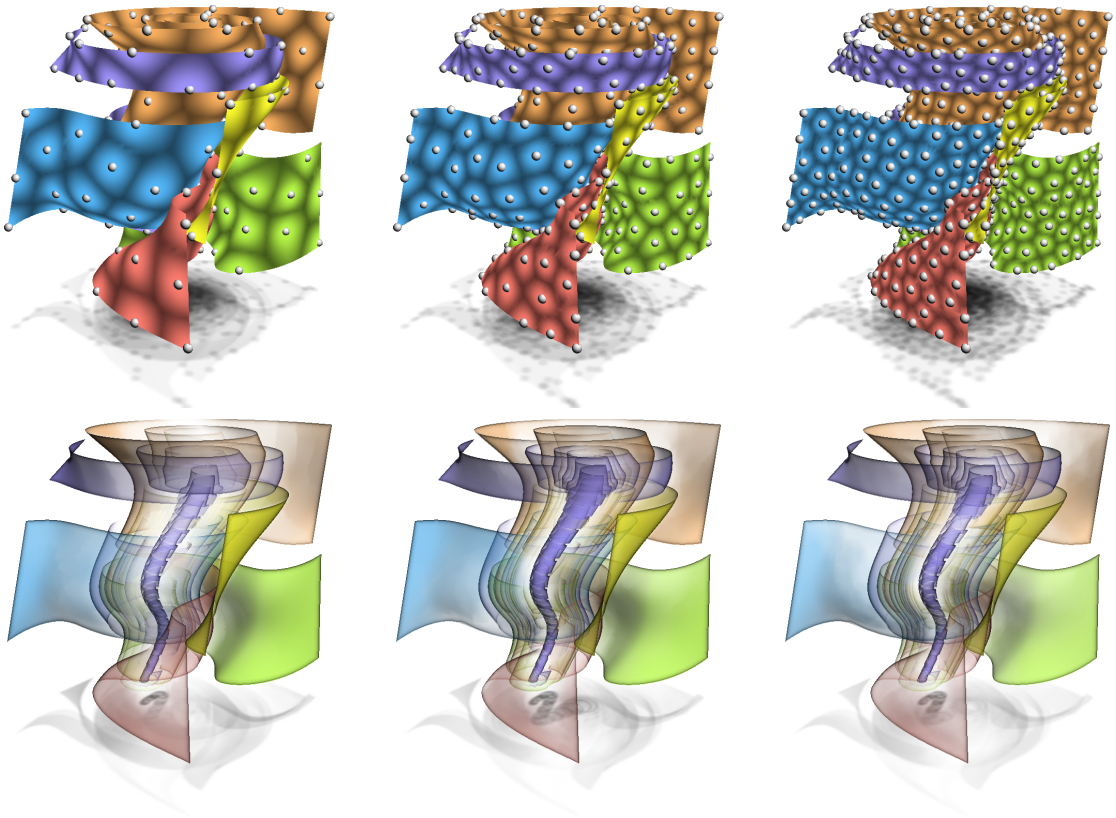


Figure 5.10.: Silhouette opacity transfer function plot for $\eta = 10$.

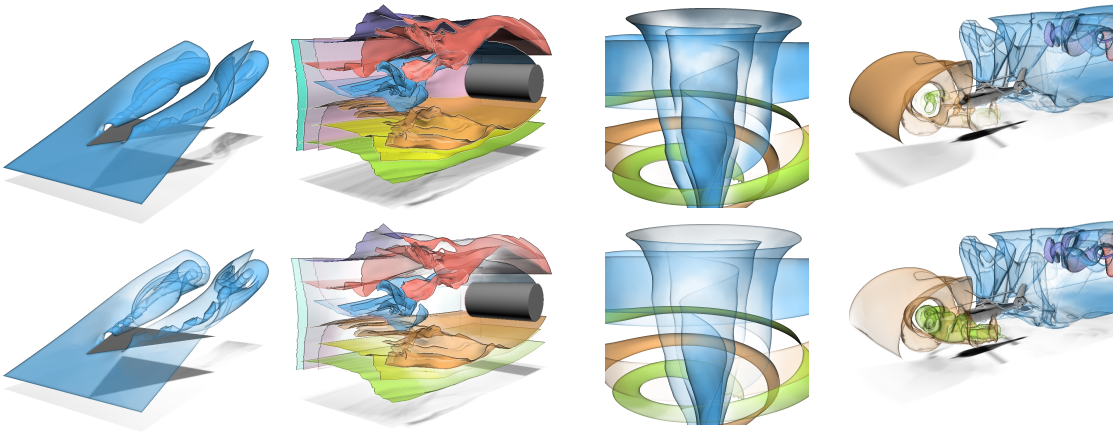


$n = 400$,
solve: $18\,ms$

$n = 1000$,
solve: $44\,ms$

$n = 2000$,
solve: $105\,ms$

Figure 5.7.: Solve rates for varying number of patches n. Using a higher number of patches preserves details, though $n = 400$ already proved useful.



Occlusion term (5.2), top: $q = 0$, bottom: $q = 7.8$

Background clutter term (5.3), top: $r = 0$, bottom: $r = 3$

Smoothness term (5.4), top: $s = 0$, bottom: $s = 50$

Emphasis exponent λ (5.2)&(5.3), top: $\lambda = 1$, bottom: $\lambda = 10$

Figure 5.8.: Study of the parameters inherited from opacity optimization in Chapter 3

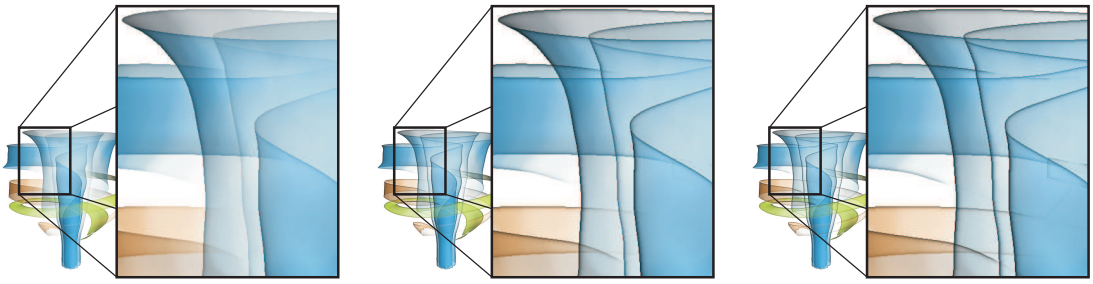


Figure 5.9.: Silhouette edges depend on the surface opacity, and can be longer conserved by an η-exponent. Left to right: $\eta = 1$ (neutral), $\eta = 10$ (recommended), $\eta = 50$.

5.4. Details on Rendering and Solving

Similar to the hierarchical version from Chapter 4, we execute rendering and solving in different threads to enable an interactive response. We use fragment linked lists[YHGT10] for both the rendering of transparent surfaces (cf. Maule et al. [MCTB11]) and the extraction of occlusion degrees. The latter is done in the solver thread on the CPU, thus fragment linked lists are copied to RAM. We construct two separate lists, one for rendering and one at quarter resolution for computing occlusion degrees. The solver thread assembles the system matrix and right-hand-side and solves the system as described in Section 5.2.3. The rendering thread repeatedly executes two steps:

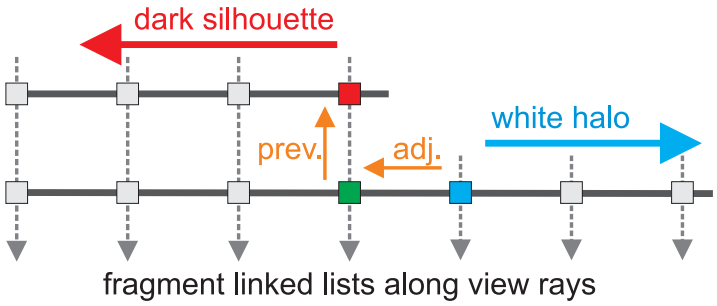


Figure 5.11.: Black silhouette gradients (←) and white halo gradients (→) are injected at the occluder fragment (■) and next (■) to the occluded fragment (■), see [CFM*13] for details. We modulate their intensity (which is then diffused) by the opacity of the occluder fragment (■).

1 Render surfaces at low resolution (for computing h_{ij})
   a) Create and sort fragment linked lists
   b) Stream lists to RAM and pass to solver thread
2 Render surfaces on full resolution
   a) Upload new opacities (if solver has new solution)
   b) Create and sort fragment linked lists
   c) Establish adjacency pointers (for diffusion)
   d) Inject black silhouette and white halo intensities
   e) Diffuse silhouettes and halos
   f) Blend fragment lists front-to-back
   g) Render shadow on ground plane using Fourier opacity mapping [JB10]

Opacity optimization and the silhouette extraction by Carnecky et al. [CFM*13] (steps 2c-2e) are easily combined as they are both based on fragment linked lists. The data stored per fragment is expanded by pointers to adjacent fragments and intensity scalars for the black borders and white halos, which are both diffused, here in five iterations of (2e).

As a visual hint for the layer order, Carnecky et al. inject and diffuse white and black color at silhouettes, see Fig. 5.11. We modulate the injected intensity by the opacity of the fragment that caused the silhouette. For the white halos, we require in addition to the adjacent fragments access to the predecessor of a fragment to lookup the occluder's opacity. Thus, we create doubly-linked lists in (2b).

5.5. Results

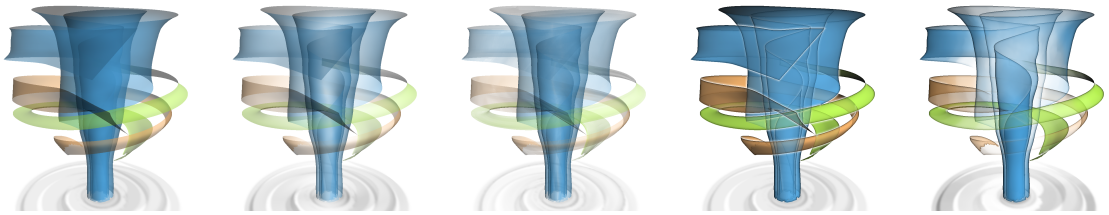
Fig. 5.12 compares our global method with established local approaches. These are (from left to right): rendering with constant opacity: angle-based transparency and normal variation [HGH*10] (see (2.28) and (2.29), resp.), and smart transparency [CFM*13], i.e., constant opacity adapted to the number of layers with silhouettes and halos to visualize surface layer order. The last column depicts results of our surface opacity optimization, for which we used mean curvature as importance. The first row shows stream surfaces in a STATIC MIXER. Here, the central vortex core is of interest (blue surface). Opacity optimization removes distracting surface parts, e.g., the blue part pointed out in Fig. 5.3, as well as the orange and green surface parts in front of and behind the core. This yields a clearer view of the interesting region.

The second row depicts the RAYLEIGH-BÉNARD CONVECTION from Section 3.4. Among the local methods, especially normal variation (2.29) excels in outlining contours of curved surfaces. Due to its low opacity the perception of the layer order is difficult. This is better preserved by angle-based transparency (2.28) and by [CFM*13], though at the cost of occlusions. Opacity optimization removes the occlusions and clears the view on the four convection cells. While simple importance measures such as curvature often do a good job, here, the part of the blue surface that connects the cells is not well preserved, as it occludes the more important red surface behind.

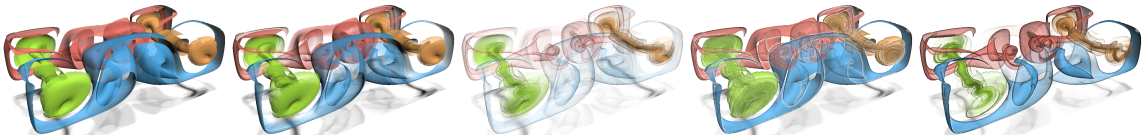
The third row shows the synthetic TORNADO data set, previously used in Section 3.4. Here, the vortex core is of interest, which is hidden behind a number of space-filling surfaces that represent the outer flow region. As seen in the center, normal variation (2.29) cannot reproduce planar surfaces faithfully. Surface opacity optimization performed best in accentuating the vortex core and maintaining context by removal of surface parts in front and behind the vortex core—the latter producing halos.

In the fourth row, the RAYLEIGH-BÉNARD CONVECTION is viewed from inside—a scenario that is useful for interactive exploration. The orange surface in the foreground hinders the view on the cells, depicted in blue and green. The removal of unimportant occluding surface parts is not addressed by local approaches. Therefore, our global surface opacity optimization performed best and removed the occlusions. We see our method best suited for interactive exploration scenarios with free camera navigation.

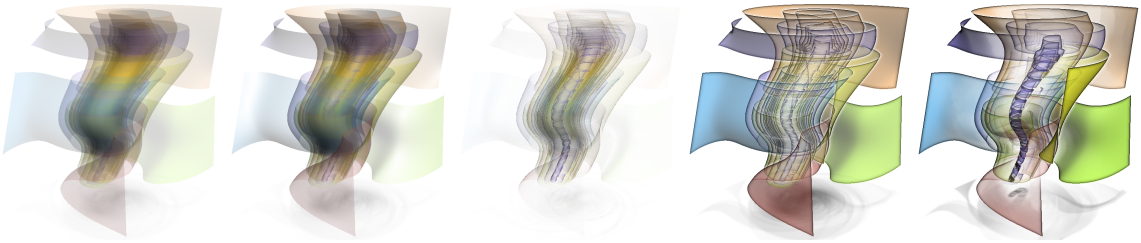
The fifth row illustrates blood flow in an ANEURYSM. The flow in the aneurysm is slow and results in highly folded and complex surfaces. For such data, all local approaches extract a high number of visual cues. Both, angle-based transparency (2.28) and normal variation (2.29), show important cues but cannot hint layer ordering. Carnecky et al.'s method [CFM*13] produces complex silhouettes that are hard to comprehend of due to their sheer number. Our method removes visual cues to reveal the twisting surfaces



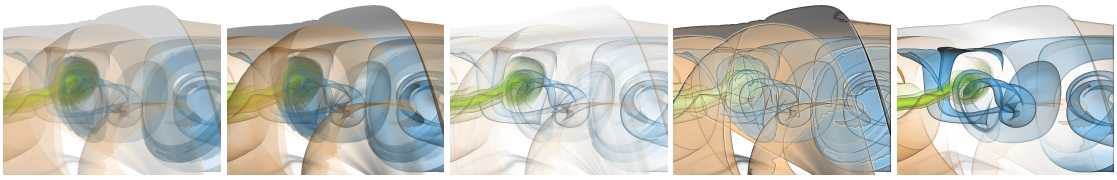
(a) Stream surfaces in a mixer ($q = 8.5$, $r = 14.3$, $\lambda = 25$).



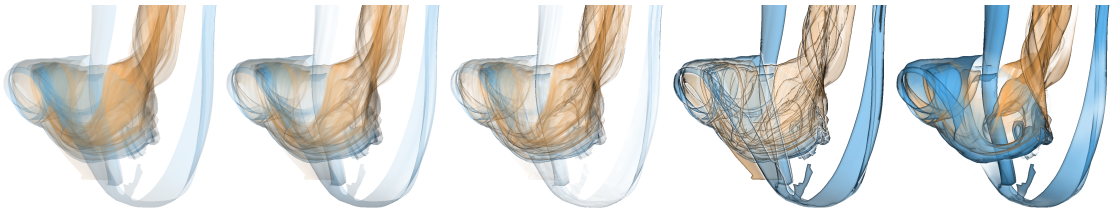
(b) Stream surfaces in a Rayleigh-Bénard convection ($q = 9.0$, $r = 5.0$, $\lambda = 15$).



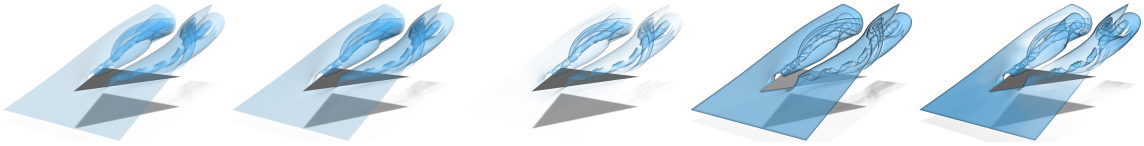
(c) Stream surfaces in a tornado ($q = 33.8$, $r = 35.1$, $s = 20$, $\lambda = 15$).



(d) Stream surfaces in a Rayleigh-Bénard convection, viewed from inside ($q = 1.0$, $r = 1.1$, $\lambda = 12$).



(e) Stream surfaces in a blood vessel with an aneurysm ($q = 4$, $r = 6$, $s = 5$, $\lambda = 11$).



(f) A single highly folded stream surface approaching a delta wing ($q = 7.85$, $r = 4.98$, $\lambda = 14$).

Figure 5.12.: Comparison with standard techniques, from left to right: constant transparency, angle-based [HGH*10], normal variation [HGH*10], smart transparency [CFM*13], and our novel surface opacity optimization.

in the inflow (orange) and the vortex in the aneurysm (blue).

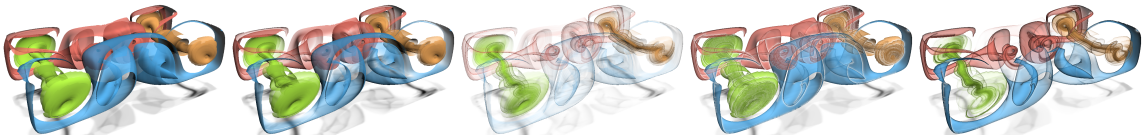
The last row shows a single stream surface approaching a DELTA WING, which is provided by Markus Rütten. It starts with a laminar flow, which is problematic for normal variation (2.29), and then begins to fold as wake turbulences detach. Even for simple surfaces, such as these, surface opacity optimization adds additional benefit by removing the topmost layer, clearing the view on the vortex cores. The DELTA WING is also shown in the first column of Fig. 5.8.

Fig. 5.1 (left) shows stream surfaces in the flow around a SQUARE CYLINDER. This flow was used and described earlier in Section 4.4.

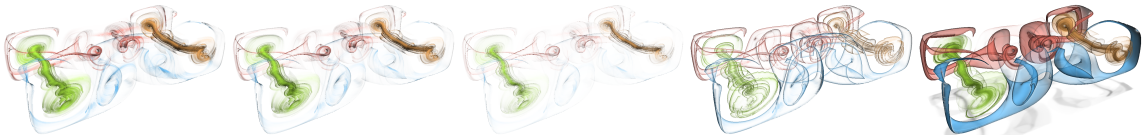
The second column of Fig. 5.8 displays stream surfaces in the flow around a WALL-MOUNTED CYLINDER, which was described in Section 3.3.2. We show that distracting, laminar layers in the background are effectively removed by surface opacity optimization, leading to more visual clarity.

The last column of Fig. 5.8 shows a HELICOPTER in slow forward flight close to the ground (cf. Section 3.3.2). This simulation by Kutz et al. [KKKK12] was used to study brownout, i.e., the entrainment of sand particles by aerodynamic uplift and impacting particles. Brownout clouds hinder the pilot's view and are mainly introduced by a vortex in front of the helicopter that is shown by the green surface.

It should be noted that in the comparison in Fig. 5.12 our technique has an edge over the others, as it includes an additional importance value that none of the standard techniques is making use of. Given that our technique uses more information the comparison could be considered not perfectly fair. Thus, it might occur to additionally weight the opacity of the standard techniques by the same importance value that we used in our technique. We present such experiments, once for the Rayleigh-Bénard convection in Fig. 5.13, in which important regions are spread throughout the domain, and again for the tornado data set in Fig. 5.14, in which the important region is small and centered in the domain. It can be seen that the modification of the standard techniques, i.e., the weighting of the opacity by the importance, results in loss of information in the image. This is because it makes unimportant surface parts always invisible. In contrast, our method preserves unimportant surfaces and only intervenes if they start to occlude important structures. With this, more context is preserved. In fact, it is this intelligent handling of occlusions and importance, which goes beyond mere weighting or blending, that is the key to success and the central contribution of our method.

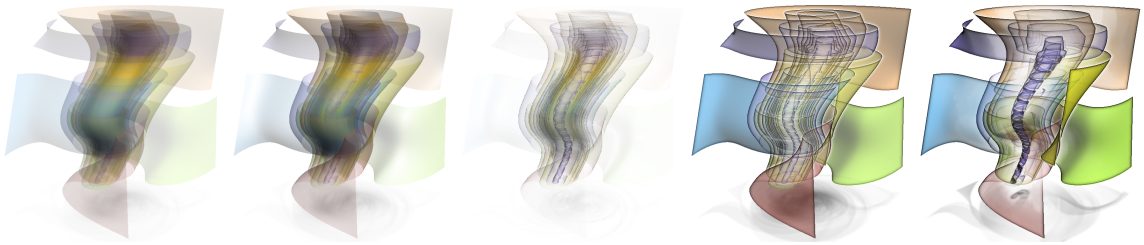


(a) Comparison with standard techniques as proposed in the literature (same as in Fig. 5.12b).

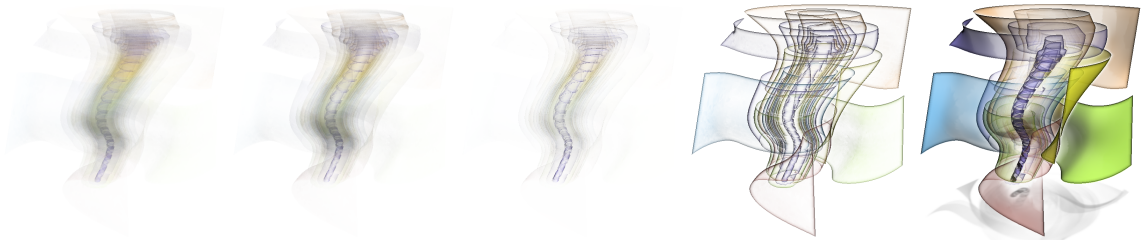


(b) Comparison with standard techniques with an additional weighting of the opacity by importance.

Figure 5.13.: Rayleigh-Bénard convection, displayed using different techniques. From left to right: constant transparency, angle-based transparency, normal variation, smart transparency and our opacity optimization.



(a) Comparison with standard techniques as proposed in the literature (same as in Fig. 5.12c).



(b) Comparison with standard techniques with an additional weighting of the opacity by importance.

Figure 5.14.: Tornado, displayed using different techniques. From left to right: constant transparency, angle-based transparency, normal variation, smart transparency and our opacity optimization.

5.5.1. Performance

We measured the performance of our system with an Intel Core i7-2600K CPU with 3.4 GHz, 16 GB RAM and Nvidia GeForce GTX 560 Ti GPU with 2 GB VRAM. The parameters (q, r, s, λ) have again been adapted to match the results obtained with the original implementation (see Section 3.2.4), and the solver and assembly timings have been updated. All images were rendered with Direct3D at a resolution of 1000×1000 pixels. Table 5.1 gives an overview of the test case configurations, and provides preprocessing timings (surface partition). Table 5.2 summarizes the timings for rendering and solving. The column labels refer to algorithm steps in Section 5.4. The shadow map used for Fourier opacity mapping [JB10] in step 2f has a resolution of 512×512 pixels and is filtered by a separated 7×7 Gaussian kernel.

The bottleneck of the system is the time required for streaming the fragment linked lists to RAM. This is because streaming is two frames deferred after enqueuing draw calls in the command buffer. Therefore, the streaming time depends on the rendering time per frame. The solving time was in our examples generally lower than the streaming time. Since both run asynchronously, we could have used more patches without loss of performance. The key to faster opacity updates therefore resides in optimizing the rendering. Fig. 5.15 compares the performance of enhanced silhouettes [CFM*13] with simple silhouettes, obtained by detecting depth discontinuities in image space. The latter is twice as fast in both rendering and solving, though at lower visual quality. One way to improve the scalability of the system is to use the simple silhouette detection or fewer diffusion iterations in [CFM*13] during interactive navigation and to switch to a high quality parameter set whenever the camera stops moving.

Table 5.1.: Configuration and partition time (in secs.) of our test scenes.

Data set	**Figs.**	**Partition time**	**Triangles**	**Patches** n
Aneurysm	5.12e	271 s	454 k	400
Bénard (in)	5.12d	97 s	246 k	400
Bénard (out)	5.12b	97 s	246 k	400
Delta Wing	5.12f	182 s	580 k	200
Helicopter	5.8	260 s	484 k	400
Square Cyl.	5.1	48 s	155 k	400
Static Mixer	5.12a	4.7 s	33 k	200
Tornado	5.12c	220 s	438 k	400
Wall. Cyl.	5.8	62 s	177 k	400

Table 5.2.: Performance analysis with time in milliseconds. Column labels refer to algorithm steps, see Section 5.4. The low resolution and high resolution pass are executed in the rendering thread. The solver thread comprises assembling of the system matrix (3a) and solving (3b). The asynchronous streaming of fragment linked lists is the bottleneck (**bold**).

	Low Res. Pass		Full Resolution Pass					Solver	
Data set	**1a**	**1b**	**2a**	**2b**	**2c–e**	**2f**	**2g**	**3a**	**3b**
Aneurysm	2.24	**206.6**	0.40	9.53	49.40	1.94	5.03	7.15	5.31
Bénard (in)	1.53	**198.7**	0.26	9.81	48.82	1.95	2.08	4.15	3.37
Bénard (out)	1.36	**168.2**	0.26	6.61	43.80	1.49	2.08	4.48	3.41
Delta Wing	2.53	**91.2**	0.48	7.88	16.19	0.87	6.37	3.34	2.74
Helicopter	2.27	**117.7**	0.42	7.29	22.89	1.02	5.33	4.97	5.02
Square Cyl.	0.69	**53.9**	0.20	2.85	10.94	0.60	2.33	3.33	3.23
Static Mixer	0.21	**98.8**	0.12	4.30	24.77	0.99	0.35	3.96	2.80
Tornado	2.37	**194.7**	0.40	9.25	45.30	2.31	4.82	5.83	3.49
Wall. Cyl.	0.92	**115.5**	0.22	4.83	25.46	1.15	3.23	6.12	5.37

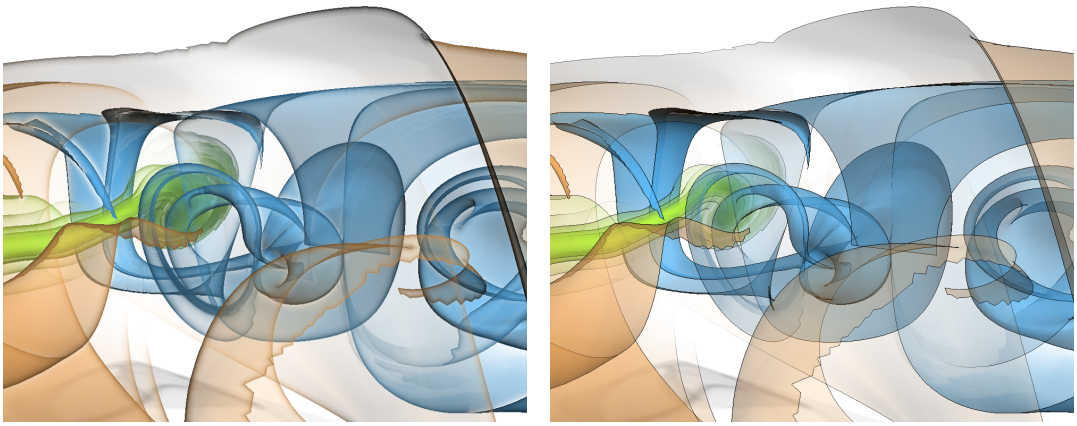


Figure 5.15.: The performance of the silhouette rendering has great impact on the frames and solves per second. Left: enhanced silhouettes [CFM*13] (12.8 fps, 3.8 sol/s), right: silhouettes by depth discontinuity (25.8 fps, 7.5 sol/s).

In summary, our rendering achieves interactive rates at about 13–50 frames per second, depending on the data set and the view. Due to the streaming bottleneck, opacity solutions are updated only 3–15 times per second. This latency is hidden similarly as in the original method by slowly fading toward the latest opacity solution.

5.5.2. Discussion and Limitations

Surface opacity optimization removes unimportant surface parts that occlude important scene elements. This requires the presence of visual clutter in the first place, which is only the case if both unimportant and important regions are present in the scene. If too many surfaces of likewise high importance occlude each other, it is impossible to decide, which surface should be shown. Similarly, surfaces with equally low importance, such as parallel planar surfaces, have no distinctive feature to accentuate. For such scenes, users are required to supply the importance terms with additional information to decide upon relevance. Besides, important structures might occlude each other, thus a notification of such an event is a possible step for future work. Furthermore, the results depend on the initial surface set, as opacity optimization only reveals information contained in the set.

In this chapter, we followed a patch-based approach that uses a fixed object-space subdivision. A fragment-based approach is to minimize our energy per pixel and to then smooth the resulting opacities in object space along the surfaces. The result would be qualitatively similar, but the runtime of the fragment-based algorithm is determined by the screen resolution, which is not desirable. Due to the high number of pixels (for each a quadratic programming problem is solved), the algorithm is very likely to be slower than solving once for $n = 400$ unknowns. Moreover, an object-space approach allows to update the solutions every few frames, whereas a fragment-based approach must either update every frame or find a correspondence to the new view, which is not easy and likely to be achieved by an intermediate mapping to the surface, i.e., the object space. Of course, the performance difference alleviates for an aggregation of pixels to tiles, which is not quite straightforward due to frame coherence issues. Also, an object-space subdivision is view-independent and can thus be computed once in a preprocessing step, which is a strong performance benefit. Though, there is an advantage to the view-dependent subdivision of a screen-space approach, i.e., only visible scene content is considered. However, the favorable independence from the screen resolution suggests an object-space subdivision approach into patches, such as ours, followed by a view-dependent grouping of patches, in the spirit of our hierarchical approach from Chapter 4. Then, frame coherence is easier to achieve due to the fix topology of the subdivision.

5.6. Summary

In this chapter, we addressed the occlusion problem for surfaces and developed the first global approach that determines a surface opacity that balances occlusion avoidance

versus importance-driven shape representation. For this, we fade out surface parts in a view-dependent and frame-coherent way so that important structures are no longer occluded. By decoupling rendering and solving, we obtain interactive rates. Our method targets specifically scenes with multiple surfaces or highly complex scenes. For such, improvements over existing local methods are significant.

Future work includes the extension to animated surfaces and accelerated solving by partitioning the surfaces in a view-dependent way in the spirit of our hierarchical method for line fields in Chapter 4. For the latter, the intermediate farthest point sets of the subdivision could be used to build a hierarchy of patches. Choosing the patches to compute opacities for then corresponds to finding a view-dependent cut based on the patch size on the viewport. As for animations, the surface subdivision has to be frame coherent so that opacities can be propagated to the surface patches of the next frame. Again, inverse distance-weighted k-nearest neighbor interpolation would be a first step. Another challenge is the performance that needs to be improved to deliver opacity updates fast enough for animated surfaces.

Aside from this, opacity optimization could also be extended to point sets, since geodesics can be computed for them, too, cf. Crane et al. [CWW13]. In the current state, surface opacity optimization can readily be applied to other prominent surface types, such as isosurfaces. The study of our method in other application contexts is an avenue for future work.

Part II.

Progressive FTLE and Rotation Invariant Vortices

Chapter 6.

6 Overview on FTLE and Vortex Concepts

The second part of this thesis focuses on the rendering of finite-time Lyapunov exponent fields without bias (Chapter 7) and on novel rotation invariant vortex concepts (Chapter 8). In this chapter, we start with an introduction of finite-time Lyapunov exponents (FTLE) for the use in unsteady flow visualization. Afterwards, using the concept of domain deformation, we formally introduce Galilean invariance and objectivity. Given this background, we motivate the problems to be solved in the subsequent chapters. Finally, we cover related work, i.e., we explain the relevant terminology of consistent volume rendering and we explain the most common vortex concepts. This chapter is in part compiled from our previous work [GST16, GT16a, GKT16].

6.1. Finite-Time Lyapunov Exponents

In flow visualization, finite-time Lyapunov exponent (FTLE) fields became one of the most popular tools to analyze coherent structures in unsteady vector fields [PPF*11, SLM05]. They describe how nearby-released particles separate over time, which allows to approximate regions of coherent behavior and barriers, i.e., Lagrangian coherent structures (LCS) or material structures, that particles will never cross [HY00, Hal01]. LCS have been shown to be valuable in a wide range of application scenarios such as predicting ocean pollutant transport due to oil mishaps, studying spread of algae in water currents, or to observe the feeding of jellyfish (see [Hal15] for an introduction).

Finite-time Lyapunov exponents characterize the separation behavior of particles in time-dependent fluid flows. In such flows, the *flow map* $\phi_t^\tau(\mathbf{x}) = \phi(\mathbf{x}, t, \tau)$ maps a particle seeded at $(\mathbf{x}, t)$ to its destination after pathline integration for duration τ (cf.

Section 2.1.1). The (spatial) gradient of the flow map $\nabla\phi(\mathbf{x},t,\tau) = \frac{\partial}{\partial\mathbf{x}}\phi(\mathbf{x},t,\tau)$ describes the behavior of particles released close to each other. We are interested in their separation behavior, which is characterized by the right Cauchy-Green deformation tensor $\nabla^{\mathrm{T}}\nabla$. Its largest real, positive eigenvalue $\lambda_{\max}$ denotes the (squared) largest magnitude of separation. Accounting for the exponential growth and normalizing by the integration duration τ yields the finite-time Lyapunov exponent (FTLE) [HY00, Hal01]:

$$\mathrm{FTLE}(\mathbf{x},t,\tau) = \frac{1}{|\tau|}\ln\sqrt{\lambda_{\max}(\nabla^{\mathrm{T}}\nabla)}, \tag{6.1}$$

Commonly, ridges of FTLE fields are often used as indicators for Lagrangian coherent structures (LCS) [SLM05]. In the literature, a variety of alternative LCS extraction techniques can be found (see [OHH15] for an overview) that typically exploit differential properties of the flow map and/or Lagrangian properties of the flow around tracer trajectories. For FTLE, the probably most wide-spread implementation (and the one we used) is to seed particles close to one another and approximate the flow map gradient by taking finite differences of their reached destinations, as in Haller and Yuan [HY00, Hal01]. The flow visualization community proposed alternative methods, such as localized FTLE [KPH*09], streak surface-based extraction [ÜSE13] and timeline tracking [KER*14]. A benchmark comparison of further computation methods was compiled by Kuhn et al. [KRWT12].

FTLE computations can be accelerated, e.g., by adaptive refinement of the flow map by Catmull-Rom interpolation [GGTH07] by the observation of filtered height ridges [SP07] or around automatically detected geometric structures [BT13]. Brunton and Rowley [BR10] reduced the number of redundant particle integrations by composition of intermediate flow maps. Nouanesengsy et al. [NLL*12] investigated a new scheduling strategy to distribute the particle tracing work across a large number of processors, minimizing idle times. Further, higher order flow map approximation [ÜSK*12] and timeline refinement [KER*14] schemes have been proposed. There exist several integral curve approximation schemes, such as hierarchical lines [HSW11], interpolation [COJ15, AOGJ15] and edge maps [BJB*12], which can be used to speed up any Lagrangian analysis technique. Generally, this requires a compromise between memory consumption, interactive response (speed) and accuracy (quality). An interactive 3D FTLE visualization was developed by Barakat et al. [BGT12], who interleave a computation and rendering phase to view-dependently build an adaptively sampled hierarchical representation of the FTLE field. Similar to our method, it produces more accurate FTLE approximations within multiple rendering passes. The accuracy of their method, however, is bound by memory and as it is based on ray marching, it cannot deliver a consistent solution. In this thesis, we strive for a consistent method that operates on a small and fixed memory bound, and avoids discretization onto (adaptive) grids and ray marching errors.

6.2. Domain Deformation, Galilean Invariance and Objectivity

Much of the argumentation in the second part of this thesis will be based on the concept of *domain deformation*. In fact, we use it for two purposes. First, it is used to formally define the concept of Galilean invariance and objectivity. Later in Chapter 8, we use it again to define rotation invariance and for finding practical computations of rotation invariant vortex measures.

6.2.1. Domain Deformation

Given the vector field $\mathbf{u}(\mathbf{x},t)$ in the spatial domain D and the temporal domain T, we consider a domain deformation as a differentiable map

$$\mathbf{g} : D \times T \to D \tag{6.2}$$

which is a diffeomorphism in its reduction to any $t \in T$. This means that there is a unique inverse map $\mathbf{h} : D \times T \to D$ with

$$\mathbf{h}(\mathbf{g}(\mathbf{x},t),t) = \mathbf{g}(\mathbf{h}(\mathbf{x},t),t) = \mathbf{x} \tag{6.3}$$

for any $(\mathbf{x},t) \in D \times T$. Based on $\mathbf{g}$, we define a new vector field in the domain $\mathbf{g}(D)$ that transforms pathlines: for every pathline $(\mathbf{q}(t),t)$ in $\mathbf{u}$, the transformed line $(\mathbf{g}(\mathbf{q}(t),t),t)$ is a pathline in $\mathbf{w}$. This property uniquely defines $\mathbf{w}$ as [KRWT12]

$$\mathbf{w}(\mathbf{x},t) = (\nabla\mathbf{h}(\mathbf{x},t))^{-1}(\mathbf{u}(\mathbf{h}(\mathbf{x},t),t) - \mathbf{h}_t(\mathbf{x},t)) \tag{6.4}$$

where $\nabla\mathbf{h}$ is the spatial gradient of $\mathbf{h}$ and $\mathbf{h}_t$ is its derivative with respect to time. We call $\mathbf{w}$ the *domain transformed vector field* inducted by the domain transformation $\mathbf{g}$. Fig. 6.1 illustrates this. Eq. (6.4) can be inverted to transform $\mathbf{w}$ back to $\mathbf{u}$:

$$\mathbf{u}(\mathbf{x},t) = (\nabla\mathbf{g}(\mathbf{x},t))^{-1}(\mathbf{w}(\mathbf{g}(\mathbf{x},t),t) - \mathbf{g}_t(\mathbf{x},t)) \tag{6.5}$$

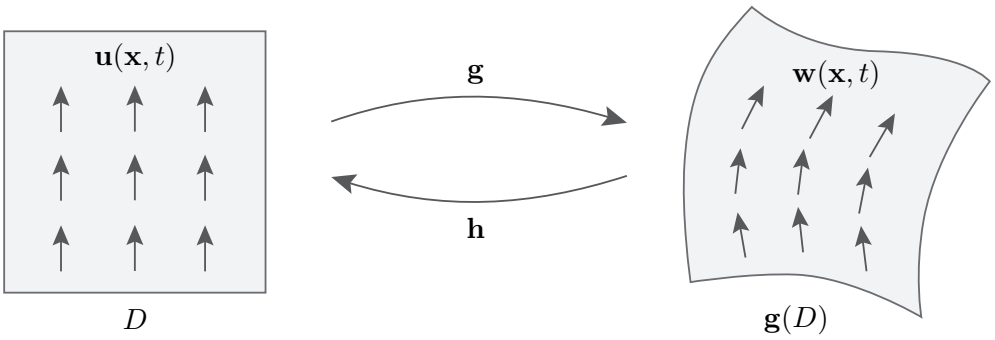


Figure 6.1.: Creating a new vector field $\mathbf{w}$ by a domain transformation $\mathbf{g}$.

6.2.2. Galilean Invariance

A vortex measure is *Galilean invariant* if it is invariant under an equal-speed translation of the underlying coordinate system. By using the concept of domain deformation we can formalize this by

Definition 1 *A vortex measure is Galilean invariant if for any transformation*

$$\mathbf{g}(\mathbf{x},t) = \mathbf{x} + \mathbf{c}_0 + t\,\mathbf{c} \tag{6.6}$$

where $\mathbf{c}_0$ *is a constant point and* $\mathbf{c}$ *is a constant vector, the following holds: the vortex measure classifies a point* $(\mathbf{x},t)$ *to be on the vortex in* $\mathbf{u}$ *iff the vortex measure classifies the point* $(\mathbf{g}(\mathbf{x},t),t)$ *to be on a vortex in the domain transformed field* $\mathbf{w}$.

Common local Galilean invariant measures are Jacobian $\mathbf{J}$ and acceleration $\mathbf{a}$. Any measure that contains only $\mathbf{J}, \mathbf{a}$, their derivatives or a Lagrangian aggregation of them over time is Galilean invariant as well. Examples are cores of swirling particle motion [WSTH07] (also known as *pathline cores*), λ_2 [JH95], Okubo-Weiss [Oku70, Wei91], FTLE [Hal01] and FSLE [BLRV01]. For $n = 2$ we also have the following Galilean invariant conditions that are all equivalent in areas of non-vanishing Jacobian:

$$\mathbf{a} = \mathbf{0} \quad\Leftrightarrow\quad \mathbf{u} - \mathbf{f} = \mathbf{0} \quad\Leftrightarrow\quad \overline{\mathbf{f}} \parallel \overline{\mathbf{p}} \quad\Leftrightarrow\quad \overline{\mathbf{J}}\,\overline{\mathbf{p}} \parallel \overline{\mathbf{p}}. \tag{6.7}$$

The equivalence of the four expressions[1] in (6.7) follows directly from Eq. (2.3), $\overline{\mathbf{J}}\,\overline{\mathbf{p}} = \begin{pmatrix}\mathbf{a}\\0\end{pmatrix}$ and $\overline{\mathbf{p}} = \begin{pmatrix}\mathbf{u}\\1\end{pmatrix}$.

As a side note, it means that for $n = 2$ the cores of swirling particle motion [WSTH07] and vortices by vanishing acceleration [KRHH11, KHNH11] are identical. For $n = 3$ there are the following equivalent conditions for a non-singular Jacobian [WSTH07]:

$$\overline{\mathbf{p}},\ \overline{\mathbf{f}},\ \begin{pmatrix}\mathbf{e}\\0\end{pmatrix} \text{ are coplanar} \quad\Leftrightarrow\quad \mathbf{e} \parallel \mathbf{u} - \mathbf{f} \quad\Leftrightarrow\quad \mathbf{J}(\mathbf{u} - \mathbf{f}) \parallel \mathbf{u} - \mathbf{f}, \tag{6.8}$$

all of them giving Galilean invariant vortices.

[1] Pathlines in 2D unsteady flow can be equivalently computed as streamlines of the space-time flow $\overline{\mathbf{p}}$. Thus, pathline cores are streamline cores of $\overline{\mathbf{p}}$, and thus Sujudi-Haimes $\overline{\mathbf{J}}\,\overline{\mathbf{p}} \parallel \overline{\mathbf{p}}$ can be applied for their extraction (cf. Section 6.4.2).

In 2D space-time, we have three eigenvectors. In case of swirling motion, two eigenvectors span the swirling plane (complex-conjugate eigenvalues) and the remaining eigenvector (the feature flow $\overline{\mathbf{f}}$) has a real-valued eigenvalue (zero). According to the reduced velocity criterion, the flow must be parallel to the eigenvector with real-valued eigenvalue: $\overline{\mathbf{f}} \parallel \overline{\mathbf{p}}$ (cf. Section 6.4.2).

Further, it follows $\overline{\mathbf{J}}\,\overline{\mathbf{p}} \parallel \overline{\mathbf{p}} \Leftrightarrow \begin{pmatrix}\mathbf{a}\\0\end{pmatrix} \parallel \begin{pmatrix}\mathbf{u}\\1\end{pmatrix} \Leftrightarrow \begin{pmatrix}\mathbf{a}\\0\end{pmatrix} = \gamma \begin{pmatrix}\mathbf{u}\\1\end{pmatrix}$. With $\gamma = 0$, $\begin{pmatrix}\mathbf{a}\\0\end{pmatrix} = \begin{pmatrix}\mathbf{0}\\0\end{pmatrix} \Leftrightarrow \mathbf{a} = \mathbf{0}$.

From $\mathbf{u} - \mathbf{f} = \mathbf{J}^{-1}\,\mathbf{a}$ in Eq. (2.3) follows $\mathbf{u} - \mathbf{f} = \mathbf{0} \Leftrightarrow \mathbf{a} = \mathbf{0}$.

Generally, standard Sujudi-Haimes [SH95] is known to *not* be Galilean invariant. However, based on the insights from Eq. (6.7) and Eq. (6.8), we can make Sujudi-Haimes applicable by a small change. For $n = 2$, we simply have to view the flow in space-time $\overline{\mathbf{p}}$, and apply Sujudi-Haimes there [KSWE15], because this gives: $\overline{\mathbf{J}}\ \overline{\mathbf{p}} \parallel \overline{\mathbf{p}}$.

For $n = 3$, we have to subtract the right ambient flow, i.e., in first-order approximation the feature flow field $\mathbf{f}$, which describes how features are moving forward and apply Sujudi-Haimes to $\mathbf{u} - \mathbf{f}$, which gives:

$$\nabla(\mathbf{u} - \mathbf{f}) \cdot (\mathbf{u} - \mathbf{f}) \parallel \mathbf{u} - \mathbf{f} \tag{6.9}$$

If vortices are subject to equal-speed translations, then $\mathbf{f} = \text{const}$ and thus $\nabla\mathbf{f} = \mathbf{0}$. The PV condition therefore reduces to $\mathbf{J}(\mathbf{u} - \mathbf{f}) \parallel \mathbf{u} - \mathbf{f}$. Eq. 6.9 will play a role in Chapter 8.

6.2.3. Objectivity

Definition 2 *A vortex measure is* objective *if it is invariant under any smooth translation and rotation of the reference system*

$$\mathbf{g}(\mathbf{x}, t) = \mathbf{Q}(t)\ \mathbf{x} + \mathbf{c}(t) \tag{6.10}$$

where $\mathbf{Q}(t)$ *is a time-dependent rotation matrix and* $\mathbf{c}(t)$ *is a time-dependent translation vector. Then, the following holds: the vortex measure classifies a point* $(\mathbf{x}, t)$ *to be on the vortex in* $\mathbf{u}$ *iff the vortex measure classifies the point* $(\mathbf{g}(\mathbf{x}, t), t)$ *to be on a vortex in the domain transformed field* $\mathbf{w}$.

The strain tensor $\mathbf{S} = \frac{1}{2}(\mathbf{J} + \mathbf{J}^{\mathrm{T}})$ is known to be objective. Any Lagrangian aggregation of it or its derivatives is objective as well, as done in the $\mathbf{M}_z$ criterion in [Hal05].

6.3. Challenges with FTLE Rendering and Vortex Tracking

6.3.1. Systematic Bias in Finite-Time Lyapunov Exponent Renderings

The FTLE field is a *Lagrangian measure*, i.e., a scalar field that arises from integration in a vector field and poses specific challenges with respect to the required computational resources and analysis complexity. All integration-based scalar fields and Lagrangian features have a key property in common: due to the integration, features are non-local and are defined at sub-grid resolution compared to the resolution of the underlying vector field. Since observable features (e.g., ridges) can become arbitrarily thin, the discretization and sampling of these fields is a fundamental challenge. In 3D, these

fields are typically precomputed on grids and are afterwards view-dependently ray casted, which introduces both grid discretization errors and sampling errors during ray marching. Further, even adaptive discretization [SP07, BGT12] can still be quite memory intensive.

In Chapter 7, we aim for high accuracy renderings of integration-based scalar fields, with focus on FTLE, by the use of Monte Carlo rendering techniques. Similar to [KPB12], we adopt progressive volumetric radiance transfer simulations to obtain high quality, aliasing-free renderings. In this work, however, we concentrate on renderings of *Lagrangian measures*. A scalable implementation therefore requires an acceleration data structure that allows to quickly determine an upper bound for the integration-based scalar value in a certain region. Since this cannot be decided a-priori due to sampling problems, the technical novelty of this chapter is that we adapt the underlying rendering method to progressively converge to the correct upper bounds. The remaining technical aspects (free path sampling, transmittance estimator and acceleration data structure) are applications of recent research from the offline rendering community.

Monte Carlo techniques start noisy and progressively converge, which typically requires time. Our method is therefore not meant to replace recent interactive systems [BGT12], but is rather aiming at HQ renderings for marketing, and the generation of exact ground truth data.

6.3.2. Vortex Tracking Beyond Equal-Speed Translations

In recent years, the visualization community contributed new definitions of vortex concepts and efficient algorithms to their numerical computation. Several vortex concepts are not invariant to any type of reference frame motion, which means that they cannot correctly extract vortices that move. A common useful property of many vortex definitions is Galilean invariance, i.e., the invariance of the vortex under an equal-speed translation of the underlying coordinate system. This property guarantees that a method successfully tracks vortices that perform equal-speed translations. However, if vortices perform a different type of motion, these techniques fail as well. In addition, there are a few approaches demanding the much stronger property of objectivity of a vortex, i.e., invariance under any smooth translation and rotation of the coordinate system. Such an invariance is rather restricting and there is only a rather small set of derived differential vector field quantities that can be used to formulate vortex criteria (e.g., the symmetric part of the Jacobian). There is evidence [Sah09] that objective measures are not descriptive enough to reveal known vortex patterns, e.g., the von Kármán vortex street in the wake of a cylinder.

Thus, we look for vortex concepts that are invariant to another relevant template of

vortex movement. In Chapter 8, we propose a new class of vortex definitions for flows that are induced by rotating mechanical parts, such as stirring devices, hydrocyclones, centrifugal pumps, or ventilators. Instead of a Galilean invariance, we enforce a rotation invariance, i.e., the invariance of a vortex under a uniform-speed rotation of the underlying coordinate system around a fixed axis.

6.4. Related Work

In the following, we explain the notion of *bias* and *consistency* in the context of Monte Carlo rendering and review recent work in consistent volume rendering. To pave the way for the subsequent chapter on rotation invariant vortices, we summarize related work on vortex extraction afterwards.

6.4.1. Consistent Volume Rendering

In rendering, Monte Carlo sampling methods are deeply established as a practical and general approach to calculate light transport [Vea98]. Generally, a Monte Carlo estimator computes the average of a sequence of n measurements M_i with $i \in \{1, ..., n\}$ that is supposed to match some unknown quantity Q. Thereby, each measurement has an error $M_i - Q$. If the expected value of this error is zero, the method is called *unbiased*. A famous example in light transport is bidirectional path tracing [LW93]. However, even *biased* methods can be made *consistent* if the bias converges to zero as n increases, such as in progressive photon mapping [HOJ08]. Note that being biased does not necessarily mean that convergence is reached slower. In Chapter 7, we seize an *unbiased* approach (Section 7.1). Our subsequent acceleration, however, is *biased* (Section 7.2). Nevertheless, the method remains *consistent*. Consistent light transport in participating media has also been extensively researched [CPP*05], including improved importance sampling [KF12], efficient beam estimates [JNT*11] and their union with general light transport [KGH*14], free path sampling [RSK08] and its acceleration [YIC*10, SKTM11], as well as efficient transmittance estimators [NSJ14].

In visualization, direct volume rendering has found many applications [MHB*00, HLSR08]. Photorealistic light transport on regular scalar fields was considered in the Exposure renderer by Kroes et al. [KPB12]. Note that integration-based scalar fields (such as FTLE) are much more expensive to evaluate. A scalable implementation requires additional changes to the necessary acceleration data structures [YIC*10, SKTM11] that could be neglected in [KPB12]. We describe how we tailored the acceleration to FTLE fields in Section 7.2. A general sampling and reconstruction framework for progressive rendering was described by Frey et al. [FSME14]. Recent surveys on GPU-based volume

rendering have been compiled by Beyer et al. [BHP14] and Balsa Rodríguez [BRGIG*14]. Typically, volume rendering approaches employ ray marching. Ray marching, however, results in unpredictable bias [NSJ14], and becomes much slower in high resolution volume data (in integration-based volumes the features can become arbitrarily thin) than Monte Carlo methods, as demonstrated by Yue et al. [YIC*10].

6.4.2. Vortex Extraction

Vortices are among the most important features in fluid flows and for this reason much research was devoted to their quantification, extraction and tracking. As there is no universal definition that captures all desired properties, a number of different vortex measures have been proposed in the literature [Hal05, Hun87, JH95, RP98]. Overall, they can be categorized into region-based methods and line-based methods.

In region-based methods, volumes of vortex-like behavior are extracted. Simple examples are thresholding pressure, vorticity or helicity, though they require a threshold to be set that is not necessarily constant along the vortex, thus they might be somewhat impractical. In the CFD community, region-based measures such as the Q criterion by Hunt [Hun87] and the λ_2-criterion by Jeong and Hussain [JH95] are well-established. Biswas et al. [BHD*15] combined four local region-based vortex detectors via majority voting, namely λ_2, Q, Δ [CPC90] and Γ_2 [GMG01]. Okubo [Oku70] and Weiss [Wei91] independently developed a criterion related to Q and more recently Haller [Hal05] derived the objective $\mathbf{M}_z$ criterion. A region-based method finding nested vortices was developed by Petz et al. [PKPH09]. Kasten et al. [KRHH11] extracted Galilean invariant vortex regions by use of acceleration. Further, Kasten et al. [KHNH12] tracked vortex merging events over time by use of combinatorial scalar field topology. Combinatorial topology was also of concern in the vortex core region detection of Jiang et al. [JMT02b].

Line-based methods, on the other hand, search for lines that particles rotate around. For this, Banks and Singer [BS95] suggested a curve following velocity-predictor, pressure-corrector method. Sahner et al. [SWH05a] extracted extremum lines of region-based methods, naming the λ_2 criterion and Q criterion by the use of feature flow fields [TS03]. Later, Sahner et al. [SWTH07] computed vortex and strain skeletons as extremal structures of derived scalar quantities. Schafhitzel et al. [SVG*08] set further focus on the topology of λ_2-based vortex corelines.

For a 3D steady flow $\mathbf{u}$, Sujudi and Haimes [SH95] defined the reduced velocity criterion, which considers the eigenvalues and eigenvectors of the Jacobian $\mathbf{J}$. A vortex coreline is present if two conditions are fulfilled. First, a pair of complex-conjugate eigenvalues exists, which is the necessary condition for swirling. Their two corresponding eigenvectors span the swirling plane, i.e., the plane in which rotating motion occurs.

Second, the remaining eigenvector $\mathbf{e}$ to the remaining real eigenvalue fulfills:

$$\mathbf{u} - (\mathbf{u}^{\mathrm{T}}\mathbf{e})\mathbf{e} = \mathbf{0}\,.$$

That is, the projection of the flow vector $\mathbf{u}$ onto the swirling plane gives zero. This means that precisely on the coreline, the particles only move forward and are not rotating. This method extracts the coreline of swirling streamlines in 3D flow and has found many applications [GLT*07]. Peikert and Roth [PR99] formally defined the parallel vectors (PV) operator, which returns the set of points at which two vector fields are parallel. Using the PV operator, Sujudi-Haimes is equivalently expressed as $\mathbf{u} \parallel \mathbf{J}\,\mathbf{u}$, i.e., $\mathbf{u}$ is parallel to an eigenvector of $\mathbf{J}$. A higher-order method was described later by Roth and Peikert [RP98] to extract bent vortex corelines. Van Gelder [VG12] made aware that inaccurate results with local methods are not only attributed to (possible) lack of Galilean invariance, but also arise when swirling occurs at different scales.

Aside from physically-based vortex definitions, geometric methods can be found in the literature [PHR99], which are useful for handling weak vortices. With the curvature centre method and the winding-angle method, Sadarjoen and Post [SP99] presented two geometric approaches that are based on streamline geometry. Köhler et al. [KGP*13] presented a semi-automatic method based on line predicates to assess vortices in cardiac blood flow. For unsteady data, Bauer and Peikert [BP02], and Theisel et al. [TSW*05] tracked the cores of swirling streamlines over time. This makes sense for vortex tracking in instantaneous vector fields, such as magnetic fields. Fuchs et al. [FPH*08] and Weinkauf et al. [WSTH07] found that pathlines swirl around a different coreline than streamlines and thus extended the method of Sujudi and Haimes in different ways to find cores of swirling pathlines, i.e., the cores of particles in unsteady flow. Weinkauf et al. [WSTH07] observed the four eigenvectors of the 4D space-time Jacobian $\bar{\mathbf{J}}$:

$$\underbrace{\begin{pmatrix}\mathbf{e}_1\\0\end{pmatrix}}_{\text{real}}, \underbrace{\begin{pmatrix}\mathbf{e}_2\\0\end{pmatrix}, \begin{pmatrix}\mathbf{e}_3\\0\end{pmatrix}}_{\text{complex}}, \underbrace{\begin{pmatrix}\mathbf{f}\\1\end{pmatrix}}_{\text{real}\,(0)}$$

Below the vectors, we denoted whether the corresponding eigenvalues are real-valued or complex. W.l.o.g., we assume the vector containing $\mathbf{e}_1$ to be the eigenvector with real eigenvalue. One of the eigenvectors contains the feature flow field from Eq. 2.2[2]. Following the idea of Sujudi and Haimes [SH95], Weinkauf et al. search for space-time locations at which the flow is in the plane spanned by the two eigenvectors to the

[2] Recalling, the feature flow field is defined as suitable combination of the partial derivatives of $\mathbf{u}$:

$$\mathbf{f} = \frac{1}{\det \mathbf{J}} \begin{pmatrix} -\det(\mathbf{u}_y, \mathbf{u}_z, \mathbf{u}_t) \\ +\det(\mathbf{u}_z, \mathbf{u}_t, \mathbf{u}_x) \\ -\det(\mathbf{u}_t, \mathbf{u}_x, \mathbf{u}_y) \end{pmatrix}. \tag{6.11}$$

The eigenvector that contains the feature flow field has always the real-valued eigenvalue 0.

two real remaining eigenvalues. Closer inspection of the vectors reveals that the 4D coplanarity condition can be reduced to a 3D parallel vectors operation, as the fourth dimension becomes zero after setting $\gamma = 1$ and subtracting the 4D feature flow field on both sides:

$$\begin{aligned}
\begin{pmatrix}\mathbf{u}\\1\end{pmatrix}, \begin{pmatrix}\mathbf{e}_1\\0\end{pmatrix}, \begin{pmatrix}\mathbf{f}\\1\end{pmatrix} \text{ are coplanar} &\Leftrightarrow \begin{pmatrix}\mathbf{u}\\1\end{pmatrix} = \beta \begin{pmatrix}\mathbf{e}_1\\0\end{pmatrix} + \gamma \begin{pmatrix}\mathbf{f}\\1\end{pmatrix} \qquad \text{set: } \gamma = 1 \\
&\Leftrightarrow \mathbf{u} - \mathbf{f} = \beta\, \mathbf{e}_1 \\
&\Leftrightarrow \mathbf{u} - \mathbf{f} \parallel \mathbf{e}_1 \\
&\Leftrightarrow (\mathbf{u} - \mathbf{f}) \parallel \mathbf{J}(\mathbf{u} - \mathbf{f})
\end{aligned}$$

The remaining two eigenvalues must be complex, which is the necessary condition for swirling.

Fuchs et al. [FPH*08] based their extension of Sujudi-Haimes to the unsteady case on the acceleration. The acceleration of an unsteady flow is $\mathbf{a} = \mathbf{J}\,\mathbf{u} + \mathbf{u}_t$. In the steady case, the temporal derivative is zero $\mathbf{u}_t = \mathbf{0}$, thus Sujudi-Haimes can likewise be interpreted as $\mathbf{u} \parallel \mathbf{a}$. In order to include the temporal derivative in unsteady flows, Fuchs et al. [FPH*08] use the unsteady acceleration in the parallel vectors condition: $\mathbf{u} \parallel \mathbf{J}\,\mathbf{u} + \mathbf{u}_t$. Since $\mathbf{J}(\mathbf{u} - \mathbf{f}) = \mathbf{a}$ from Eq. (2.5), their method is equivalently expressed by:

$$\mathbf{u} \parallel \mathbf{a} \quad \Leftrightarrow \quad \mathbf{u} \parallel \mathbf{J}\,\mathbf{u} + \mathbf{u}_t \quad \Leftrightarrow \quad \mathbf{u} \parallel \mathbf{J}(\mathbf{u} - \mathbf{f})\,.$$

Note that these conditions are not Galilean invariant. The last condition differs only by the subtraction of $\mathbf{f}$ on the left side from Weinkauf et al. [WSTH07].

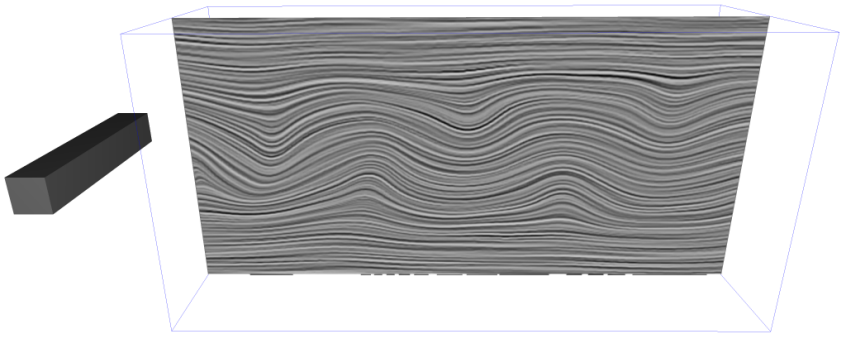
Aside from the geometric methods, all aforementioned automatic extraction approaches have in common that they are local, and thus easily parallelized. However, it was shown that there are classes of vortices that cannot be extracted by local methods, for instance attracting vortices that move on non-linear paths. Thus, instead, integration-based methods were developed, such as particle density estimation by Wiebel et al. [WCW*09]. They proposed to inject a number of particles and observe their attraction behavior over time. Weinkauf and Theisel [WT10a] found attractors by analyzing the Jacobian of a derived vector field in which streaklines are tangent curves.

Another Lagrangian detector was developed by Cucitore et al. [CQB99]. They extracted vortices by observing the neighboring particles around a particle to test, i.e., they let the reference frame move with the tested particle. Further, Lagrangian smoothing as proposed by Fuchs et al. [FPS*08] and Shi et al. [STH*09] can be applied to any local vortex detector that was originally designed for steady flow by smoothing the extraction results along pathlines over time.

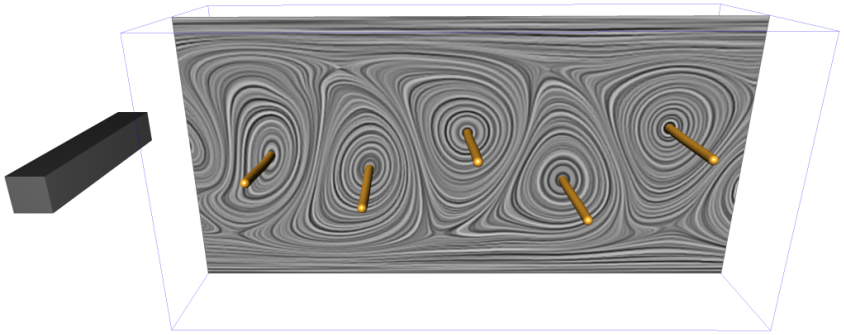
Aside from the definition of extraction methods that are invariant under certain types of reference frame motion (Galilean invariance, objectivity, rotation invariance), there

is a thread of research on finding a suitable reference frame in which the flow appears (nearly) steady. A very common example is the subtraction of a mean flow (or a certain percentage of it that is based on domain expert experience), which is obsolete for Galilean invariant extraction methods [WSTH07]. More sophisticated reference frame choices use decompositions of the flow to find a harmonic vector field that can be subtracted to eliminate the general motion. Since harmonic fields are divergence-free and curl-free, the resulting field keeps its local divergence and rotation properties. Wiebel et al. [WTS*07] decomposed the flow into a localized and a harmonic component, and studied the localized component, with the restriction that the "localized" flow is confined into a domain. The Helmholtz-Hodge decomposition (HHD) [BNPB13] decomposes a vector field into a scalar potential (curl-free), a vector potential (divergence-free) and a harmonic vector field. If the latter is present, the resulting components and the uniqueness of the decomposition strongly depend on the boundary conditions. Bathia et al. [BPKB14] used their *natural* HHD to extract vortices in the resulting (near-)steady flow. Aside from using the HHD to perform a change of the reference frame, vortices have also been characterized as extremal structures of the magnitude of the vector potential, e.g., by Tong et al. [TLHD03] and Wiebel et al. [Wie04]. Fig. 6.2 illustrates the two conceptually different approaches to vortex tracking in unsteady flows: subtraction of the ambient flow and invariance to certain types of motion.

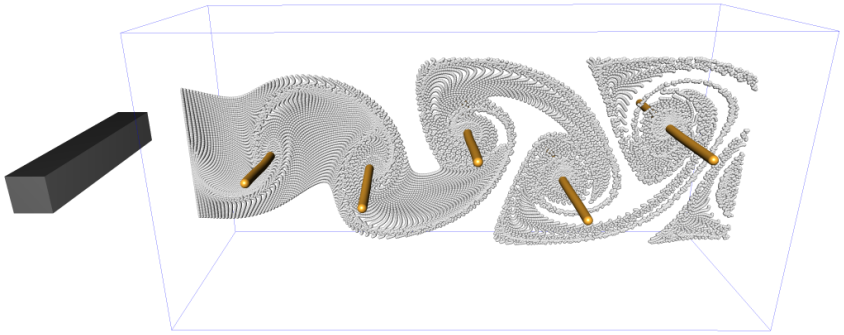
Orthogonal to vortex definitions are the verification of numerical extraction results and their further processing for visualization. Jiang et al. [JMT02a] presented a method to verify corelines based on the geometry of streamlines. To improve the vortex core visualization, Garth et al. [GTS*04] computed hull surfaces around vortex cores based on the Rankine vortex model. Sahner et al. [SWH05a] proposed an iconic representation to indicate scale and extent. Shafii et al. [SOK*13] encoded rotation direction, rotational strength and the spatial extent of vortex cores and show the behavior of the flow in the vicinity of the vortex. For a more detailed overview on vortex extraction and visualization methods we refer to [LHZP07, PR99, PVH*03].



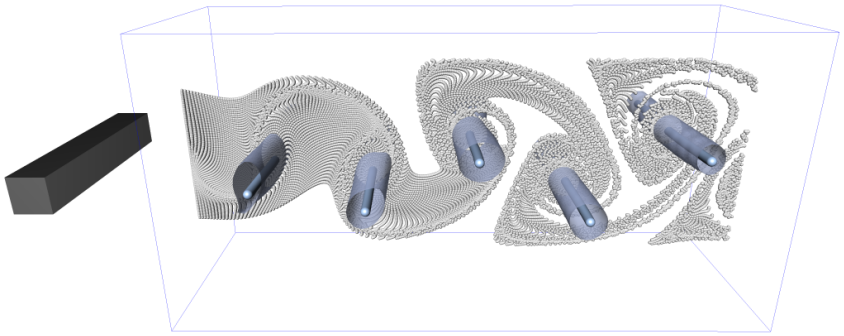
(a) Unsteady vector field $\mathbf{u}(\mathbf{x}, t)$, shown using LIC (streamlines). Vortices are not apparent.



(b) Subtraction of the "right" ambient flow, reveals vortices, extracted with Sujudi-Haimes [SH95].



(c) Extracted vortex corelines in (b) are indeed centers of rotating particle motion.



(d) *Galilean invariant* methods find vortices *without* subtraction of ambient flow. Here, with λ_2 isosurfaces [JH95] and pathline cores [WSTH07].

Figure 6.2.: Overview of the two approaches to vortex tracking: subtracting a suitable ambient flow (b)–(c) or the invariance to certain types of vortex motion (d).

Chapter 7.

7 MCFTLE: Monte Carlo Rendering of Finite-Time Lyapunov Exponent Fields

The following chapter describes a progressive rendering approach for the visualization of finite-time Lyapunov exponent fields that avoids discretization errors and ray marching artifacts. The chapter is based on the publication:

T. Günther, A. Kuhn and H. Theisel
MCFTLE: Monte Carlo Rendering of Finite-Time Lyapunov Exponent Fields
Computer Graphics Forum (Proc. EuroVis) 35, 3 (2016).

Traditionally, Lagrangian fields such as finite-time Lyapunov exponents (FTLE) are precomputed on a discrete grid and are ray casted afterwards. This, however, introduces both grid discretization errors and sampling errors during ray marching. In this work, we apply a progressive, view-dependent Monte Carlo-based approach for the visualization of such Lagrangian fields in time-dependent flows. Our approach avoids grid discretization and ray marching errors completely, is consistent, and has a low memory consumption. The system provides noisy previews that converge over time to an accurate high-quality visualization. Compared to traditional approaches, the proposed system avoids explicitly predefined fieldline seeding structures, and uses a Monte Carlo sampling strategy named Woodcock tracking to distribute samples along the view ray. An acceleration of this sampling strategy requires local upper bounds for the FTLE values, which we progressively acquire during the rendering. Our approach is tailored for high-quality visualizations of complex FTLE fields and is guaranteed to faithfully represent detailed ridge surface structures as indicators for Lagrangian coherent structures (LCS), as demonstrated in Fig. 7.1.

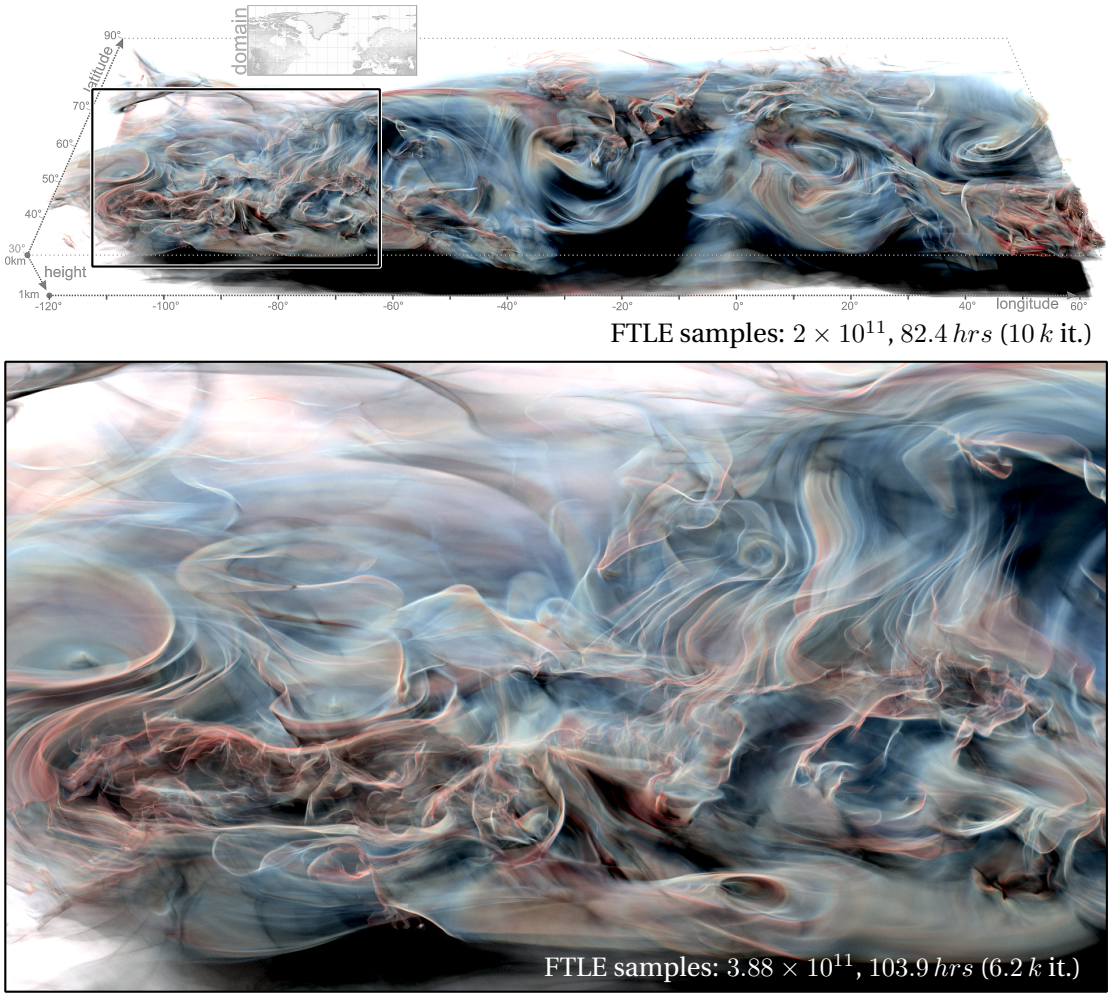


Figure 7.1.: High-quality FTLE visualization of a European Centre for Medium-Range Weather Forecasts (ECMWF) reanalysis simulation of the wind velocity field of the northern hemisphere at April, 10th in 2010. The three-dimensional FTLE field emphasizes the spatial turbulence structure in the wind field and illustrates differences above land masses of North America (left and detail region), the North Atlantic region (center) and Europe (right). The bottom image shows a close-up of the FTLE field above North America. Integration duration (GPU): $\tau = 10$, majorant extinction $\bar{\sigma}_t = 1$, FTLE values are in the range $R = [0.35,\ 0.61]$, resolution of acceleration grid ($\bar{\sigma}_t$ grid): $100 \times 50 \times 10$, image resolution top: 1600×600, bottom: 1400×800 pixels.

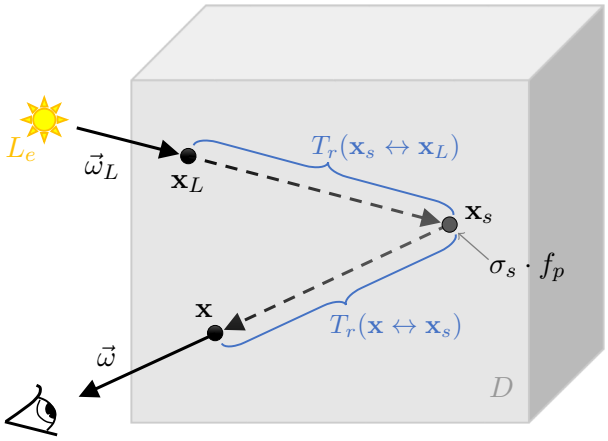


Figure 7.2.: Illustration of single-scattering light transport. A directional light emits radiance L_e in direction $\vec{\omega}_L$, which enters the domain D at $\mathbf{x}_L$. The transmittance $T_r(\mathbf{x}_s \leftrightarrow \mathbf{x}_L)$ accounts for the attenuation on the way toward $\mathbf{x}_s$, where it scatters with coefficient σ_s. The amount of light that is scattered toward the viewer in direction $\vec{\omega}$ is determined by $f_p(\mathbf{x}_s, \vec{\omega}_L \rightarrow \vec{\omega})$. Finally, the transmittance $T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s)$ attenuates toward the entry point $\mathbf{x}$ of the view ray.

7.1. Monte Carlo FTLE Rendering

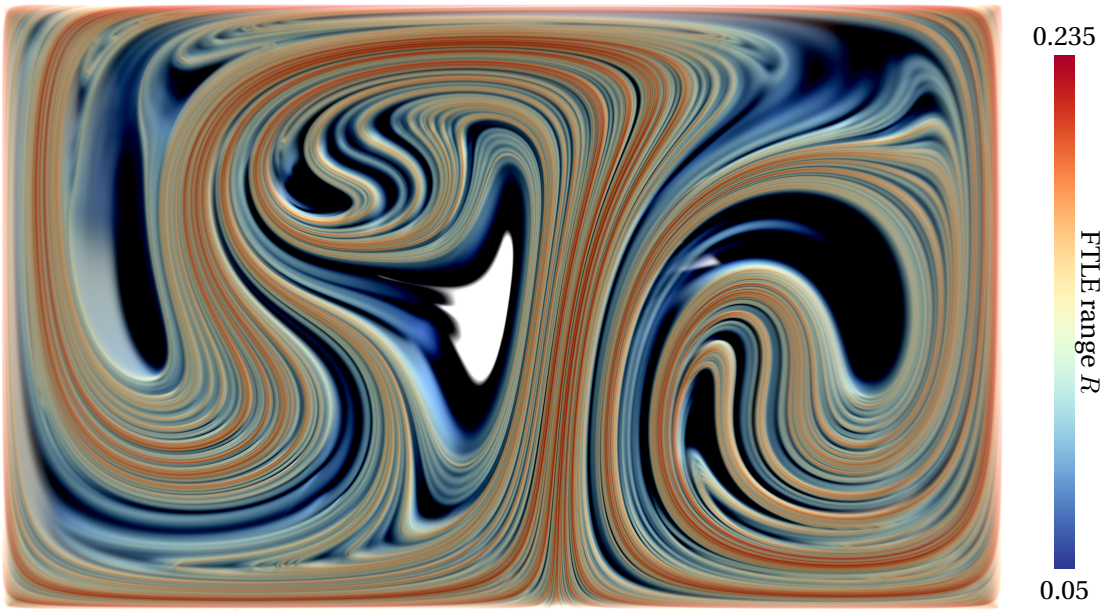
Our method is generally applicable to a progressive and consistent rendering of integration-based scalar fields. As such, we keep the theory and problem formulation general. Given is a scalar field $F(\mathbf{x}) : \mathcal{D} \rightarrow \mathbb{R}$ that is defined over the spatial domain $\mathcal{D} \subseteq \mathbb{R}^3$. In this chapter, we regard $F(\mathbf{x})$ as the FTLE value at $\mathbf{x}$ at a given time t and integration duration τ, see Eq. (6.1).

$$F(\mathbf{x}) = \mathrm{FTLE}(\mathbf{x}, t, \tau) \tag{7.1}$$

We drop the temporal arguments for brevity, as they remain fixed. Next, we explain the light transport model that we use and describe its progressive Monte Carlo-based computation.

7.1.1. Volume Rendering Equation

We employ a single-scattering model [Max95], as illustrated in Fig. 7.2. For this, we need to set an extinction coefficient $\sigma_t(\mathbf{x})$ and a scattering albedo color $\mathbf{c}(\mathbf{x})$, which are both derived via transfer functions from the scalar field $F(\mathbf{x})$. Fig. 7.3 shows the transfer function that we used throughout the chapter for color $\mathbf{c}(\mathbf{x})$. Extinction $\sigma_t(\mathbf{x})$ was mapped linearly on the same value range. The scattering albedo $\mathbf{c}$ is the ratio between scattering and extinction coefficients $\mathbf{c} = \sigma_s/\sigma_t$, and denotes the probability



FTLE samples: 4.75×10^{11}, $54.3\,hrs$ ($10\,k$ it.)

Figure 7.3.: DOUBLE GYRE flow on the GPU without $\bar{\sigma}_t$ grid (Section 7.2) at 4000×2000 pixels. Integration time: $T = [0,\ 40]$, $\bar{\sigma}_t = 100$, FTLE range $R = [0.05,\ 0.235]$.

of a scattering event at a certain location. Intuitively speaking, the color red means that all red photons scatter, whereas blue and green photons are absorbed.

The transmittance T_r is the fraction of light that reaches point $\mathbf{x}'$ if emitted from $\mathbf{x}$ (or vice versa), when traveling on a straight line in inhomogeneous media. It is defined as

$$T_r(\mathbf{x} \leftrightarrow \mathbf{x}') = e^{-\int_0^d \sigma_t(\mathbf{x}_s)\,\mathrm{d}s} \tag{7.2}$$

with $\mathbf{x}_s = \mathbf{x} + s\,\vec{\omega}$ and $s \in [0,\ d]$ parameterizes the ray in direction $\vec{\omega}$ from $\mathbf{x}$ to $\mathbf{x}'$. Note that by definition $0 \leq T_r \leq 1$.

The incoming radiance L at $\mathbf{x}$ with incoming direction $\vec{\omega}$ is governed by the *volume rendering equation* [Jar08, CPP*05]:

$$L(\mathbf{x} \leftarrow \vec{\omega}) = \int_0^d T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s)\ \sigma_s(\mathbf{x}_s)\ L_i(\mathbf{x}_s \leftarrow \vec{\omega})\,\mathrm{d}s \tag{7.3}$$

$$\approx \frac{1}{n}\sum_{i=1}^{n} \frac{T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s)\ \sigma_s(\mathbf{x}_s)\ L_i(\mathbf{x}_s \leftarrow \vec{\omega})}{p(\mathbf{x}_s)}\,. \tag{7.4}$$

Here, without emission and surface interaction, thus the integral only contains the in-scattered radiance L_i. Distributing samples $\mathbf{x}_s$ along the ray according to the probability

$$p(\mathbf{x}_s) = \sigma_t(\mathbf{x}_s)\ T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s) \tag{7.5}$$

importance samples the integrand with respect to high FTLE values (high extinction σ_t) and high transmittance T_r, which further reduces Eq. (7.4) to

$$L(\mathbf{x} \leftarrow \vec{\omega}) \approx \frac{1}{n} \sum_{i=1}^{n} \mathbf{c}(\mathbf{x}_s)\ L_i(\mathbf{x}_s \leftarrow \vec{\omega}) \tag{7.6}$$

Later in Section 7.1.2, we explain a method that determines the scattering location $\mathbf{x}_s$ of the photon according to probability $p(\mathbf{x}_s)$. The traveled distance from $\mathbf{x}$ to the scattering location $\mathbf{x}_s$ is called free flight distance $d_i(\mathbf{x} \rightarrow \mathbf{x}')$.

Assuming single-scattering from a directional light source with radiance L_e, incident direction $\vec{\omega}_L$, and $\mathbf{x}_L$ being the intersection of the light ray with the domain boundary, the in-scattered radiance L_i at a point $\mathbf{x}_s$ toward the viewer in direction ω is:

$$L_i(\mathbf{x}_s \leftarrow \vec{\omega}) = f_p(\mathbf{x}_s, \vec{\omega}_L \rightarrow \vec{\omega})\ T_r(\mathbf{x}_s \leftrightarrow \mathbf{x}_L)\ L_e \tag{7.7}$$

with $f_p(\mathbf{x}_s, \vec{\omega}_L \rightarrow \vec{\omega}) = 1/(4\pi)$ being an isotropic phase function (other choices are possible [AD16]), which describes how much light coming from $\vec{\omega}_L$ is scattered at $\mathbf{x}_s$ toward $\vec{\omega}$.

Given the scattering event at $\mathbf{x}_s$, we need to determine the fraction of light that arrives from the light source, i.e., $T_r(\mathbf{x}_s \leftrightarrow \mathbf{x}_L)$. Raab et al. [RSK08] and Szirmay-Kalos et al. [SKTM11] computed this for inhomogeneous participating media consistently as the expected value of a visibility experiment, in which they randomly emit photons at $\mathbf{x}_s$ and test if they reach $\mathbf{x}_L$:

$$\nu(\mathbf{x}_s) = \begin{cases} 1 & \text{if } d_i(\mathbf{x}_s \rightarrow \mathbf{x}_L) \geq \|\mathbf{x}_s - \mathbf{x}_L\| \\ 0 & \text{else} \end{cases} \tag{7.8}$$

Then, the final estimate of the incoming radiance becomes:

$$L(\mathbf{x} \leftarrow \vec{\omega}) \approx \frac{1}{4\pi\, n} \sum_{i=1}^{n} \mathbf{c}(\mathbf{x}_s)\ \nu(\mathbf{x}_s)\ L_e \tag{7.9}$$

Several improvements to the transmittance calculation have been proposed, including the separation of T_r into multiple parts and solving them separately [SKTM11, NSJ14], and estimating the visibility by multiple free path runs to reduce variance [JNT*11]. The more recent ratio tracking and residual tracking [NSJ14] lower the variance by weighting full paths, rather than taking them as binary estimates.

7.1.2. Free Path Sampling

Evaluating Eq. (7.9) requires a method to generate samples $\mathbf{x}_s$ along a ray according to probability $p(\mathbf{x}_s)$ in Eq. (7.5). The most commonly used method is the so-called

Woodcock tracking, which is also known as *free path sampling*. It probabilistically determines the free flight distance of a photon until it reaches a scattering event in inhomogeneous media. The method was invented in the 60s in the neutron transport community [WMHL65] and was picked up by [RSK08] in the rendering community, where it had great impact in practice [JNT*11, NSJ14].

The idea of the method is to fill up inhomogeneous media with virtual particles to obtain a homogeneous medium with a joint, spatially constant so-called *majorant extinction* $\bar{\sigma}_t$. It then randomly samples the free flight distance in the joint homogeneous medium along the ray and thereby generates tentative particle interactions, which are probabilistically selected as either being virtual or real. In case of a virtual interaction ($\sigma_t/\bar{\sigma}_t < rand()$), the random walk continues. As proven by Coleman [Col68], the algorithm is *unbiased* and generates samples according to probability $p(\mathbf{x}_s)$. The algorithm is listed in Alg. 3. Carter et al. [CCT72] and Galtier et al. [GBC*13] showed variants that handle non-bounding majorants, but at the cost of increased variance [NSJ14].

Input: Ray origin $\mathbf{x}_0$ and normalized direction $\vec{\omega}$, majorant extinction $\bar{\sigma}_t$ and a ray interval $(d_{min}, d_{max}]$ to evaluate.
Output: Free flight distance d.
$d \leftarrow d_{min} - \ln(1 - rand())/\bar{\sigma}_t$
while $d \leq d_{max} \wedge \sigma_t(\mathbf{x}_0 + d\vec{\omega})/\bar{\sigma}_t < rand()$ **do**
  $d \leftarrow d - \ln(1 - rand())/\bar{\sigma}_t$
end
Algorithm 3: This algorithm determines the free flight distance of a photon along a ray $(\mathbf{x}_0, \vec{\omega})$. It implements $d_i(\mathbf{x} \rightarrow \mathbf{x}')$ with $\vec{\omega}$ being the normalized direction vector from $\mathbf{x}$ to $\mathbf{x}'$, $d_{min} = \|\mathbf{x} - \mathbf{x}_0\|$, and $d_{max} = \|\mathbf{x}' - \mathbf{x}_0\|$. (Pseudo code adapted from [YIC*10].)

7.2. Acceleration by Spatially-Varying Extinction Bounds

The expected worst case number of samples a free path run performs on a ray interval $(d_{min}, d_{max}]$ is $\bar{\sigma}_t \cdot (d_{max} - d_{min})$, since the expected step size in the joint homogeneous medium is $1/\bar{\sigma}_t$. Thus, the runtime strongly depends on how tightly $\bar{\sigma}_t$ bounds the true extinction σ_t. This is especially crucial in "empty" areas, where a globally defined upper bound strongly over-estimates the extinction. Therefore, Yue et al. [YIC*10] proposed a kd-tree-based space partition of the domain, in which they stored a bounding majorant extinction $\bar{\sigma}_t$ per volume segment. A related approach was taken by Szirmay-Kalos et al. [SKTM11], who defined several segmentations over regular grids.

In this chapter, we follow one approach in [SKTM11] and use piecewise constant bounds stored on a coarse voxel grid, i.e., a constant value per voxel. In order to handle

transitions between volume segments correctly, they have shown that a free path run always has to start at the entry point in the next voxel. If a scattering event is found inside a voxel it is reported, otherwise the run proceeds in a DDA traversal with the next voxel until the ray leaves the domain, see Alg. 4.

Input: Ray origin $\mathbf{x}_0$ and normalized direction $\vec{\omega}$, and a ray interval $(d_{min}, d_{max}]$ to evaluate.
Output: Free flight distance d.
$\mathcal{V} \leftarrow$ locate voxel containing $\mathbf{x}_0 + d_{min}\,\vec{\omega}$
do
  $\hat{d}_{min} \leftarrow \mathcal{V}.\textit{entry}()$
  $\hat{d}_{max} \leftarrow \mathcal{V}.\textit{exit}()$
  $\bar{\sigma}_t \leftarrow \mathcal{V}.\textit{majorant}()$
  $d \leftarrow \textit{freePathRun}(\mathbf{x}_0, \vec{\omega}, \bar{\sigma}_t, \hat{d}_{min}, \hat{d}_{max})$
  if $d < \hat{d}_{max} \vee d \geq d_{max}$ **then break**;
while $\mathcal{V} \leftarrow \mathcal{V}.\textit{nextVoxel}()$;
Algorithm 4: Determines free flight distance along a ray $(\mathbf{x}_0, \vec{\omega})$ by DDA traversal in a voxel grid that contains the majorant extinction $\bar{\sigma}_t$. Alg. 3 implements *freePathRun*.

Finding Local Upper Bounds

In integration-based scalar fields a tight upper bound cannot be evaluated locally. Ridge features might become arbitrarily thin and trying to sample them is a costly endeavor. For this reason, we start with an (under-estimating) approximation of the true upper bound by taking a small number of k samples per voxel of the acceleration grid (typically, $k \approx 4$). This initial approximation causes the method to be *biased*. Over time, the free path runs take additional samples, at which we update the upper bounds. Since the free path sampling produces a continuous and unbiased sampling of the domain, the true upper bound is found in the limit. Thus, the bias converges to zero, and hence the method remains *consistent* (cf. definition of consistency in Section 6.4.1).

The initial upper extinction bound estimate of a voxel might be zero, even though a sharp ridge might exist in the voxel. In this case, all future free path runs would skip the voxel, and no further samples would be taken. The method would become inconsistent. To guarantee a certain minimum sampling rate, we clamp the extinction of each voxel to a lower bound, which adds additional tracing cost. The lower extinction bound is chosen such that a voxel with diagonal extent d has a $p_{\mathcal{V}}$ probability of being sampled by a free path run along the diagonal, i.e., the lower extinction bound is $p_{\mathcal{V}}/d$. Thereby, $p_{\mathcal{V}}$ is a user parameter that balances performance versus quality (convergence to true

upper bound), and we empirically set $p_{\mathcal{V}} = 0.1$: In practice, we found that setting $p_{\mathcal{V}} = 0$ gives visually similar results. In the ABC flow, the root mean squared error (RMSE) for $p_{\mathcal{V}} = 0$ after 1,000 iterations was reached with $p_{\mathcal{V}} = 0.1$ after 750 iterations already. For $p_{\mathcal{V}} = 0.1$ the runtime increases by about factor 1.22. Thus, the additional cost for assuring consistency amortizes.

7.3. Implementation

We implemented our method on the CPU (multi-threaded) and on the GPU using DirectCompute. The accompanying material of the corresponding paper [GKT16] contains C++ demo code that implements the Monte Carlo FTLE rendering in the DOUBLE GYRE flow on the CPU. The CPU code is easy to integrate into existing visualization frameworks, and may serve the purpose of generating ground truth data for other rendering methods.

We computed the flow map gradients by taking finite differences [HY00, Hal01]. Thereby, the separation distance was set to 10^{-6} to capture thin ridge structures. Generally, separation distances cannot be arbitrarily small, since for even smaller values numerical problems arise in the computation of the eigenvalues of the Cauchy-Green tensor in Eq. (6.1), see Kuhn et al. [KRWT12] for a study of this effect. The numerical issues in the underlying scalar field are independent of the rendering method, and thus do not make the Monte Carlo approach unbiased or inconsistent.

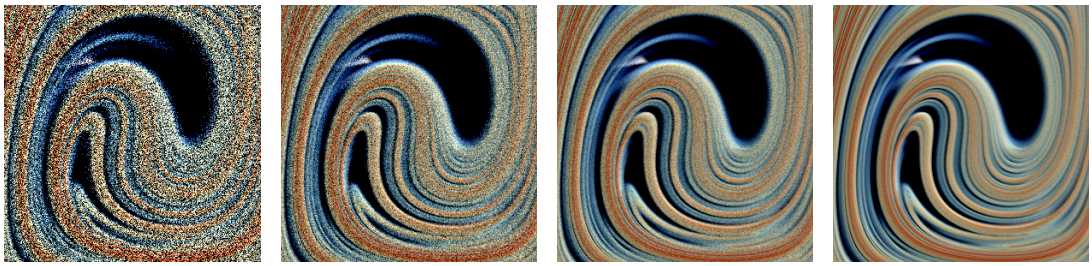
Alg. 5 computes the incoming radiance L of a pixel according to Eq. (7.9). Thereby, *generateViewRay* uniformly samples the pixel for anti-aliasing purposes and the method *intersectRayWithDomain* calculates the entry and exit distance of the view ray into and out of the domain. Note that d_{min} is set to 0 if the ray origin is inside the domain. Finally, Alg. 4 implements the method *freePathRunDDA*.

7.4. Results

We applied our method to a number of analytic and real-world vector fields. In all figures, we list the rendering time, number of iterations (rays per pixel), and *cost*, which refers to the total number of FTLE samples taken to compute the image. Note that we deliberately used a high number of iterations to produce high quality images. A convergence series, showing that early results are useful, too, is shown in the next section.

Input: Pixel coordinate $(p_x,\, p_y)$, number of iterations n.
Output: Incoming radiance L of pixel.
$L \leftarrow 0$;
for $i = 1$ **to** n **do**
  $(\mathbf{x}_0,\, \vec{\omega}) \leftarrow$ *generateViewRay*$(p_x,\, p_y)$
  $(d_{min},\, d_{max}) \leftarrow$ *intersectRayWithDomain*$(\mathbf{x}_0,\, \vec{\omega})$
  $d \leftarrow$ *freePathRunDDA*$(\mathbf{x}_0,\, \vec{\omega},\, d_{min},\, d_{max})$
  if $d < d_{max}$ **then**
    $\mathbf{x}_s \leftarrow \mathbf{x}_0 + d\,\vec{\omega}$
    $L \leftarrow L + \mathbf{c}(\mathbf{x}_s)\ \nu(\mathbf{x}_s)\ L_e$
  end
end
$L \leftarrow L/(4\pi\, n)$

Algorithm 5: Computes radiance according to Eq. (7.9).



(a) 1.1 secs, 3 iter. (b) 10 secs, 27 iter. (c) 1 min, 160 iter. (d) 10 mins, 1600 iter.

Figure 7.4.: Convergence series of a close-up in the DOUBLE GYRE at 300×300 pixels, using the GPU (without $\bar{\sigma}_t$ grid). The majority of the noise vanishes in a few minutes. Integration time: $T = [0,\, 40]$, $\bar{\sigma}_t = 100$, FTLE range $R = [0.05,\, 0.235]$.

Double Gyre. The DOUBLE GYRE [SLM05] is a periodic 2D unsteady vector field that is commonly used as a benchmark for FTLE computations. In this chapter, we define it in the temporal-periodic domain $D \times T = [0,\, 2] \times [0,\, 1] \times [0,\, 10]$ and use the parameterization

$$\mathbf{u}(x,\, y,\, t) = \begin{pmatrix} -0.1\pi\, \sin\left(f(x,\, t)\,\pi\right)\, \cos(y\,\pi) \\ 0.1\pi\, \cos\left(f(x,\, t)\,\pi\right)\, \sin(y\,\pi)\, \frac{\mathrm{d}}{\mathrm{d}x} f(x,\, t) \end{pmatrix} \tag{7.10}$$

with $f(x,\, t) = a(t)\, x^2 + b(t)\, x$ and $a(t) = 0.25 \sin(t\,\pi/5)$ and $b(t) = 1 - 0.5 \sin(t\,\pi/5)$. In order to create a 3D unsteady sequence, we set it constant along the z dimension. An example is shown in Fig. 7.3. Fig. 7.4 shows a convergence series, conveying an impression of what early results look like and at which rate the noise reduces. Further results on real-world data sets are shown at the end of this chapter in Figs 7.11–7.15.

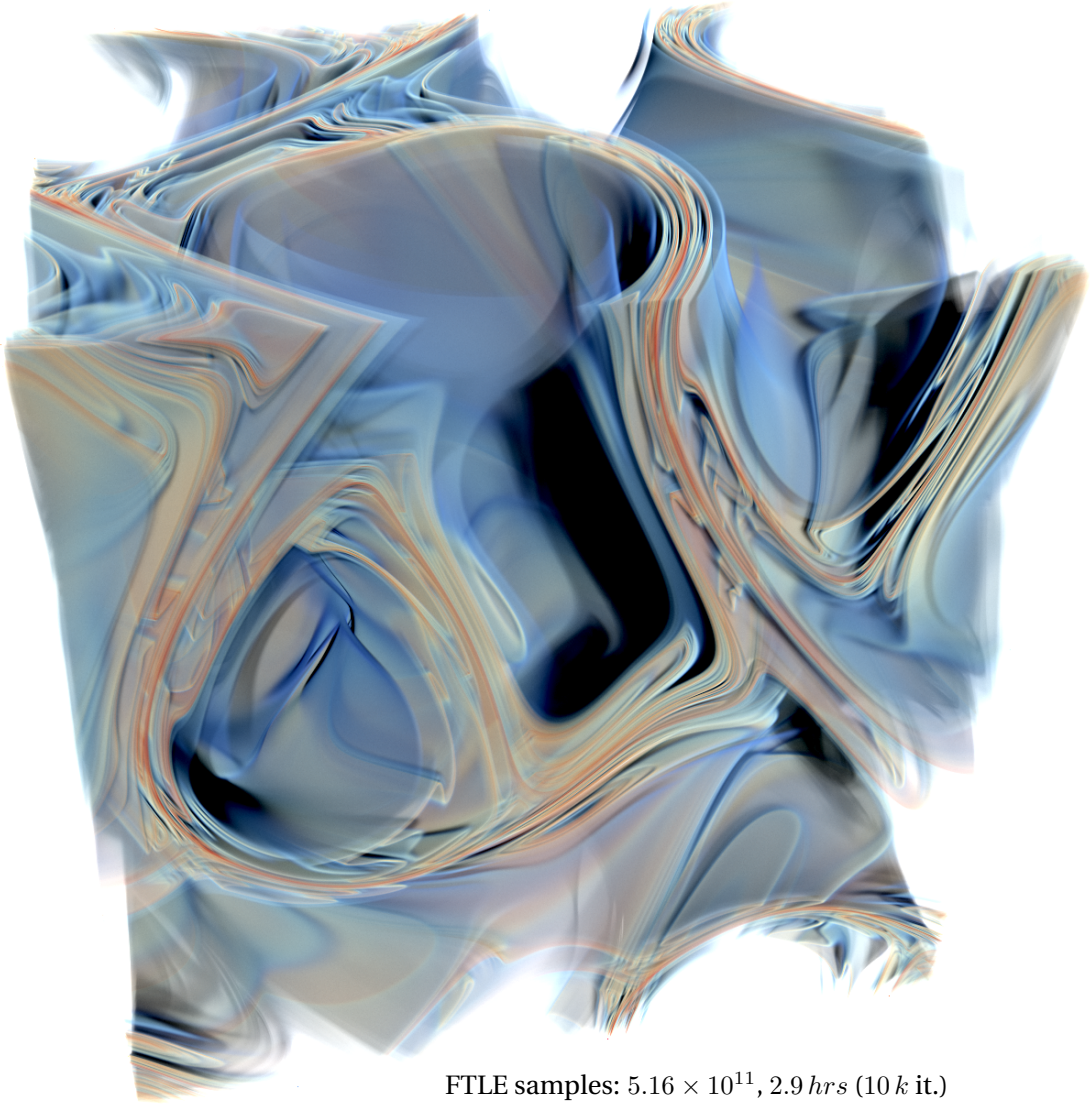


Figure 7.5.: ABC flow on the GPU without $\bar{\sigma}_t$ grid at 1500×1500 pixels, showing several sharp ridges. Integration time: $T = [5,\ 15]$, $\bar{\sigma}_t = 35$, FTLE range $R = [0.38,\ 0.92]$.

ABC Flow. The ABC (Arnold-Beltrami-Childress) flows are a class of parameterizable 3D unsteady flows that are often studied to assess turbulence. We used the following parameterization:

$$\mathbf{u}(x,\ y,\ z,\ t) = \begin{pmatrix} c(t)\ \sin(z) + \cos(y) \\ \sqrt{2}\ \sin(x) + c(t)\ \cos(z) \\ \sin(y) + \sqrt{2}\ \cos(x) \end{pmatrix} \tag{7.11}$$

with $c(t) = \sqrt{3} + (1 - e^{-0.1\,t})\ \sin(2\pi\,t)$. The flow is defined in the domain $D \times T = [0,\ 2\pi]^3 \times [0,\ 40]$. Fig. 7.5 gives an example of the particularly sharp ridges in this flow.

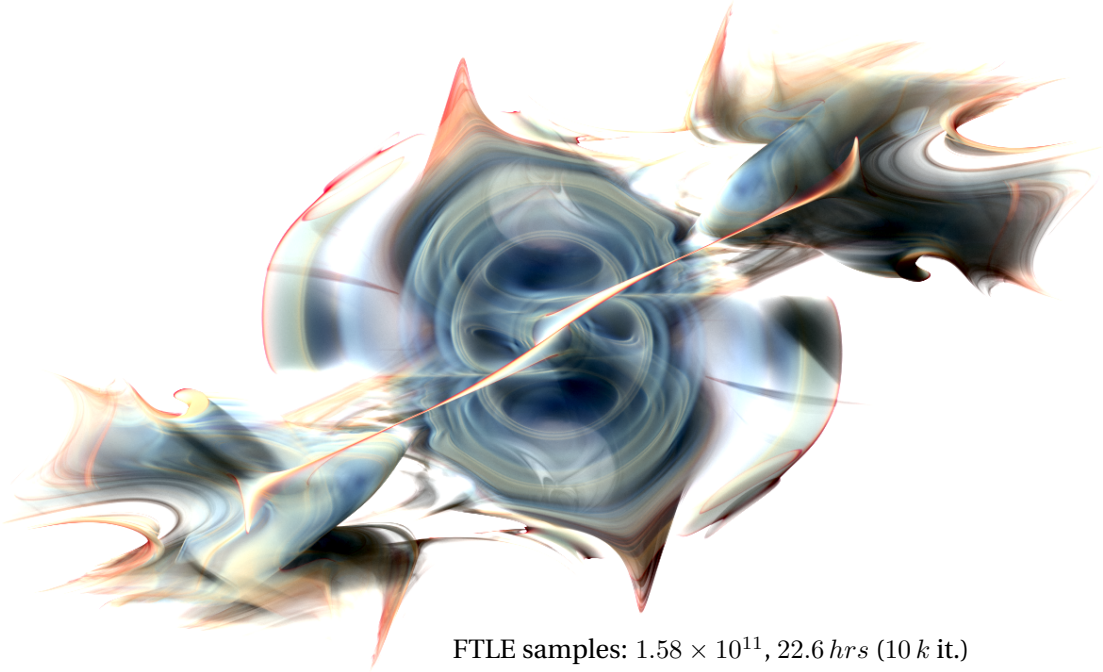


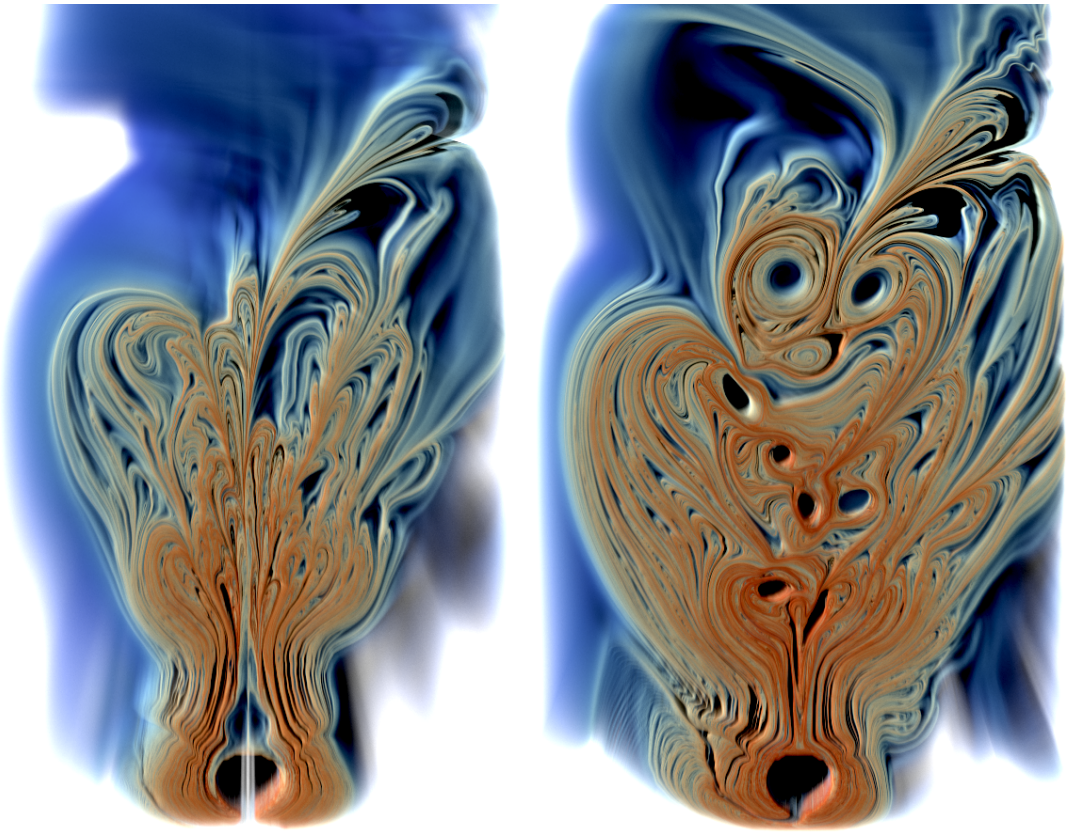
Figure 7.6.: RABINOVICH-FABRIKANT flow on the GPU with $\bar{\sigma}_t$ grid of $50 \times 20 \times 5$ at 1200×750 pixels. Integration duration $\tau = 20$, $\bar{\sigma}_t = 5$, FTLE range $R = [0,\ 0.47]$.

Rabinovich-Fabrikant Equations. The RABINOVICH-FABRIKANT equations describe a parameter-dependent 3D steady dynamical system that exhibits chaotic behavior for certain parameter configurations. We consider it in the domain $D = [-15,\ 15]^3$:

$$\mathbf{u}(x,\ y,\ z) = \begin{pmatrix} y(z - 1 + x^2) + \gamma\, x \\ x(3z + 1 - x^2) + \gamma\, y \\ -2z(\alpha + xy) \end{pmatrix} \tag{7.12}$$

here, with $\alpha = 0.98$ and $\gamma = 0.1$. This flow is difficult to analyze numerically, as traditional integrators reach different attractors depending on the step size [DC04]. In Fig. 7.6, we used a fourth-order Runge-Kutta integrator (cf. Section 2.1.2) with step size $h = 0.1$.

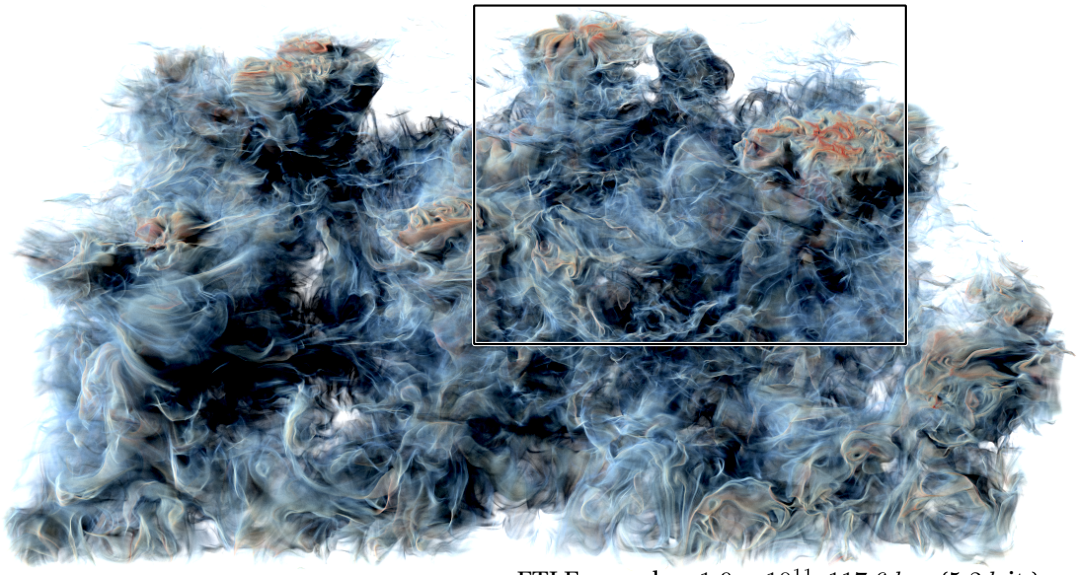
Boussinesq. The BOUSSINESQ flow contains a 2D unsteady convection simulation that develops around a heated cylinder. The flow was provided by Tino Weinkauf and was simulated with Gerris Flow solver [Pop04], using the Boussinesq-approximation to generate the turbulent vortex behavior. In Fig. 7.7, the start time of the integration is mapped to the third spatial dimension. The flow exhibits a high number of spatially close FTLE ridges.



FTLE samples: 8.57×10^{10}, $34.2\,hrs$ ($10\,k$ it.) FTLE samples: 7.64×10^{10}, $24.2\,hrs$ ($10\,k$ it.)

Figure 7.7.: BOUSSINESQ flow on the GPU with $\bar{\sigma}_t$ grid of $5 \times 10 \times 5$ at 600×1600 pixels. We set $\bar{\sigma}_t = 60$ and FTLE range $R = [0.02,\ 1]$. The flow is shown at different start times (front slice), left: $t_0 = 1$, right: $t_0 = 5$, both with $\tau = 10$.

Simulated Cloud-topped Boundary Layer (CTBL). The CTBL data set contains a cloud resolving boundary layer simulation (UCLA-LES, details in [Ste13]). A large eddy simulation (LES) was used on a finite size domain (longitudinal, latitudinal extend: $10\,km$ with 384 cells, height extend $3.2\,km$ with 130 cells) under idealized conditions: It uses double-periodic boundary conditions and homogeneous surface forcing, while large-scale information are taken from the COSMO-DE simulation model. The purpose of this model is to study the detailed cloud dynamics on high spatial and temporal resolutions for sub-scale parametrization of synoptic climate simulations. For this task, convective flow patterns are of central interest since they are tightly coupled with cloud production and interaction processes. FTLE fields are specifically useful to visualize the complex spatial structure and distribution of turbulent plumes produced by the UCLA-LES model. The results for our method are shown in Fig. 7.8.



FTLE samples: 1.9×10^{11}, $117.6\,hrs$ ($5.2\,k$ it.)

FTLE samples: 2.15×10^{11}, $98.1\,hrs$ ($4.4\,k$ it.)

Figure 7.8.: CLOUD-TOPPED BOUNDARY LAYER flow on the GPU with $40 \times 20 \times 20$ $\bar{\sigma}_t$ grid. Integration duration: $\tau = 30$, majorant extinction $\bar{\sigma}_t = 2$, FTLE range $R = [0.09,\ 0.21]$, left: 1200×640, right: 800×800 pixels.

ECMWF Reanalysis. The ECMWF data set shows a large scale ECMWF reanalysis simulation of the weather in the northern hemisphere (long. range: $-120°$ to $60°$ with 1200 cells, lat. range: $30°$ to $90°$ with 40 cells, height from $0\,m$ to $1050\,m$ with 90 cells) from April $10-19,\ 2010$. Similarly to the CTBL example, convective flow features in the wind velocity field are of central interest, since they are strongly related to the exchange of energy and transport of trace gases in the atmosphere. LCS features have been shown to characterize transport patterns and mixing behavior of atmospheric flows [Hal15]. In Fig. 7.1, FTLE highlights the characteristic structure of atmospheric features above the North American land surface (bottom left, detailed zoom), the North Atlantic ocean (center) and the European land mass (bottom right corner). Specifically the spatial turbulent structure of vortices (cyclones) and stream-like features are emphasized. Note that we used time-averaged velocity fields to compute FTLE fields.

7.5. Discussion

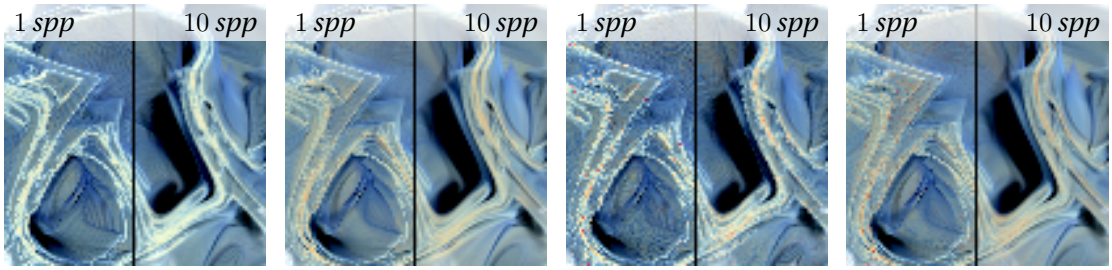
In the following, we compare our method with traditional ray casting and report timings, number of FTLE samples and memory consumption for the figures shown throughout the paper. Afterwards, we observe the convergence behavior of the Monte Carlo integration and discuss the limitations.

7.5.1. Comparison with Ray Casting

Next, we show the artifacts that grid discretization and ray marching entail. The results are obtained with our CPU code using a resolution of 100×100 pixels. For comparison, a GPU-based high resolution image is shown in Fig. 7.5.

In Figs. 7.9a and 7.9b, the scalar field was discretized onto a 200^3 or 500^3 regular grid, respectively, and was afterwards ray casted. Thereby, the separation distance of particles in the FTLE computation was set to the voxel size. The FTLE value range depends on the separation distance, same as the sharpness of the ridges that can be recovered. The tracing time is per iteration. In all remaining images, the separation distance in the FTLE computation was set to 10^{-6}.

Figs. 7.9c and 7.9d show the same setting as in (a)–(b), but with smaller separation distance. Here, the discretization onto the grid causes an under-sampling of the ridges. Since the discrete sampling does not preserve maxima, the transfer functions return more transparent and blueish values (color shift). We display the result with 1 sample per pixel (*spp*) to show the systematic bias and the average of 10 *spp*, i.e., 10 tracing iterations. Fig. 7.9e and Fig. 7.9f apply the ray casting directly to the FTLE scalar field

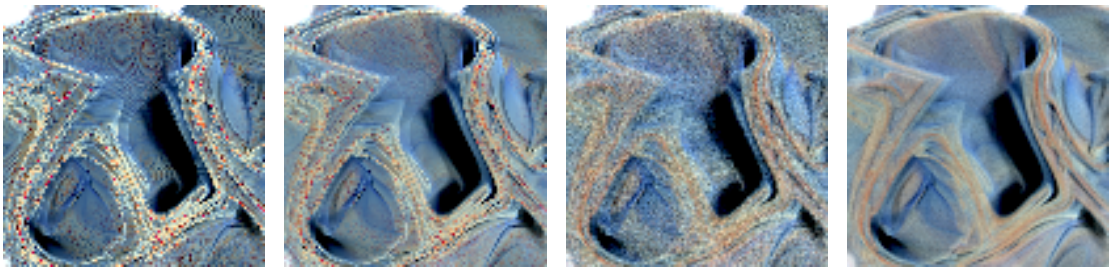


(a) Ray casting of discrete grid with 200^3 voxels ($30.5\,MB$), grid precomp.: $22\,s$, tracing: $1.3\,s$.

(b) Ray casting of discrete grid with 500^3 voxels ($476\,MB$), grid precomp.: $5\,min$, tracing: $7.2\,s$.

(c) Ray casting of discrete grid with 200^3 voxels ($30.5\,MB$), grid precomp.: $129\,s$, tracing: $1.3\,s$.

(d) Ray casting of discrete grid with 500^3 voxels ($476\,MB$), grid precomp.: $30\,min$, tracing: $7.2\,s$.



(e) Ray casting without grid (evaluate FTLE at each sample), $11.3\,min$, insufficient sampling rate (aliasing artifacts).

(f) Ray casting without grid (evaluate FTLE at each sample), $69.9\,min$, higher sampling rate than (e), some artifacts.

(g) Monte Carlo (ours), 70 iterations, $111.3\,s$, no discretization artifacts, contains noise.

(h) Monte Carlo (ours), 1055 iterations, $29.7\,min$, less noise.

Figure 7.9.: Comparison of Monte Carlo and ray casting.

without discretization. The step size discretization during ray marching results in Fig. 7.9e in heavy aliasing. We used the same step sizes as in (a)–(d) and show 1 *spp*. Although early ray termination was used, the computation time of ray casting is high.

Fig. 7.9g and Fig. 7.9h show results of our consistent Monte Carlo-based approach. The discretization artifacts (aliasing and color shifts) are avoided; instead the result contains the typical Monte Carlo noise. In a time comparable to Fig. 7.9c, 70 iterations could be computed per pixel in Fig. 7.9g. With Fig. 7.9d setting the reference time, 1055 iterations were computed in Fig. 7.9h. The precomputation of the initial $\bar{\sigma}_t$ grid (60^3, 4 samples per voxel) took $17\,s$. Our method takes multiple *spp*, while the initial $\bar{\sigma}_t$ grid is precomputed once per FTLE field.

7.5.2. Timings, Cost and Memory Consumption

We used an AMD Phenom II X6 1055T CPU, and an Nvidia GeForce GTX 970 GPU with 4 GB VRAM. The results in Fig. 7.9 were obtained on an Intel Core i7-2600K CPU.

Data set	CPU time (ms)		GPU time (μs)		cost	CPU cost	GPU cost	memory
	w/o acc.	w/ acc.	w/o acc.	w/ acc.	w/o acc.	w/ acc.	w/ acc.	(in MB)
DOUBLE GYRE, Fig. 7.3	2.41	**1.14**	**2.44**	8.25	5.94	2.79	5.20	122.07
ABC, Fig. 7.5	0.52	**0.13**	**0.46**	1.00	22.93	5.90	10.25	34.33
RABINOVICH, Fig. 7.6	1.82	**0.12**	11.40	**9.04**	91.31	5.73	17.56	13.75
BOUSSINESQ, Fig. 7.7 (left)	4.08	**0.13**	100.33	**12.83**	47.12	1.49	8.93	14.65
BOUSSINESQ, Fig. 7.7 (right)	1.73	**0.17**	36.58	**9.08**	24.74	2.09	7.96	14.65
CTBL, Fig. 7.8	14.10	**3.65**	250.83	**106.01**	62.99	16.09	47.55	11.78
CTBL, Fig. 7.8 (close-up)	20.94	**5.88**	220.00	**125.41**	93.38	25.77	76.49	9.83
ECMWF, Fig. 7.1	3.29	**0.54**	79.50	**30.90**	54.17	7.56	20.83	14.84
ECMWF, Fig. 7.1 (close-up)	6.82	**1.28**	121.03	**33.40**	111.56	17.62	34.64	17.28

Table 7.1.: Average time and average number of FTLE samples (average cost) per iteration per pixel, as well as the memory consumption for the respective figures throughout the chapter (single precision). Without acceleration the CPU and GPU have identical cost. Viewport resolutions are listed in the referenced figures.

Table 7.1 lists for all datasets the average time and average number of FTLE samples (cost) per iteration per pixel. The latter is hardware-independent and only depends on the dataset and view. The CPU timings are in milliseconds (ms), whereas GPU timings are in microseconds (μs). We conducted experiments with and without $\bar{\sigma}_t$ grids, and list the main memory required for rendering the images in the referenced figures. Our method stores four floats per pixel for the accumulated radiance (RGBA) and an optional (scalar) acceleration grid (both single precision). Thus, $16\,MB$ are required per megapixel plus $< 1\,MB$ for a $\bar{\sigma}_t$ grid, as we have used on the GPU. Among all examples, the DOUBLE GYRE was rendered at highest resolution, and has thus the largest memory consumption.

The GPU code is orders of magnitude faster than the CPU code, and especially on the CPU the acceleration greatly increases the performance. In the RABINOVICH case (Fig. 7.6), for instance, large portions of the domain are empty. Thus, here, the accelerated CPU version is about $15\times$ faster. The speedup obtained by the $\bar{\sigma}_t$ grid depends on the tightness of the upper extinction bounds. If large areas in the domain have low extinction, much can be gained compared to a global constant upper bound. This is visible in the BOUSSINESQ case in Fig. 7.7. The left image has more empty areas and here the time and cost savings are greater than in the right image.

On the GPU, arithmetic (ALU) calculations are much faster than memory accesses (e.g., lookups in the $\bar{\sigma}_t$ grid) which is why we generally recommend lower $\bar{\sigma}_t$ grid resolutions than for the CPU in order to reduce memory I/O pressure. Our manually-tuned grid resolutions are listed in Table 7.2. Tracing in analytic fields is entirely ALU bound and requires no memory access at all. For the two analytic examples (DOUBLE GYRE in Fig. 7.3, ABC in Fig. 7.5), the $\bar{\sigma}_t$ lookups are more expensive than the brute force evaluation of more FTLE samples, while the balance tips in favor of the $\bar{\sigma}_t$ grid for the

Data set	$\bar{\sigma}_t$ grid (CPU)	$\bar{\sigma}_t$ grid (GPU)
DOUBLE GYRE	$100 \times 50 \times 1$	$20 \times 10 \times 1$
ABC	$60 \times 60 \times 60$	$20 \times 20 \times 20$
RABINOVICH	$300 \times 300 \times 30$	$50 \times 20 \times 5$
BOUSSINESQ	$200 \times 200 \times 50$	$5 \times 10 \times 5$
CTBL	$200 \times 100 \times 100$	$40 \times 20 \times 20$
ECMWF	$200 \times 100 \times 100$	$100 \times 50 \times 10$

Table 7.2.: Manually-tuned resolutions of the $\bar{\sigma}_t$ grids.

RABINOVICH case. In general, the $\bar{\sigma}_t$ grid can be recommended for sampled real-world flows, as here the $\bar{\sigma}_t$ lookup cost amortizes much faster.

7.5.3. Convergence

In (bounded) Monte Carlo integration, the variance decreases asymptotically to zero. Plotting the root-mean-square error (RMSE) over time, i.e., the difference between the progressively estimated radiance from Eq. (7.9) and the ground truth radiance, in log-log scale thereby shows a linear convergence behavior in Fig. 7.10. The plotted RMSE for our data sets confirms that the method converges as expected. In this experiment, we used the GPU for the DOUBLE GYRE and ABC without $\bar{\sigma}_t$ grid and for the others with $\bar{\sigma}_t$ grid.

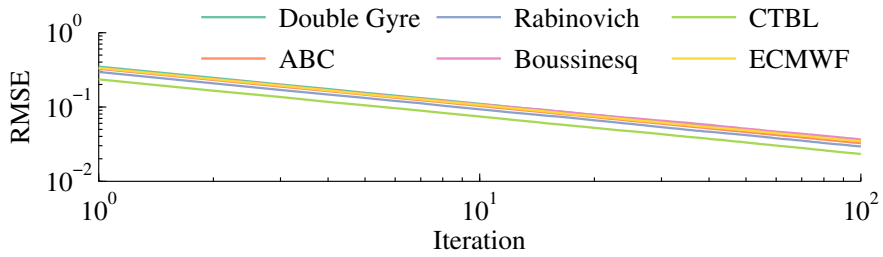


Figure 7.10.: Plotting the RMSE over time in log-log scale shows the linear convergence expected from MC methods.

7.5.4. Limitations

Our bottleneck is the pathline integration. Recently, interpolation of pathlines from a set of candidates rather than full numerical integration has been investigated [COJ15, AOGJ15]. Clearly, any interpolation-based approach introduces an error that makes the rendering *inconsistent*. Nevertheless, if faster previews are desired in an entirely Monte Carlo-based framework, this is a promising approach to take. Every Monte

Carlo-based approach essentially reduces variance on some quantity, which shows up as slowly reducing noise. In the rendering community, reconstruction filters and noise removal are applied to remove the residual noise in order to satisfy the often narrow time budgets. Out of that need, a large number of filters have been developed [ZJL*15] that could be applied to our technique as well.

7.6. Conclusions

In this chapter, we applied a consistent light transport simulation method to the rendering of Lagrangian scalar fields, focusing on FTLE. While previous approaches discretize the FTLE field onto a (possibly adaptive) grid [BGT12], our method avoids discretization errors completely and operates on a fixed and small memory bound. We used a progressive Monte Carlo sampling technique named Woodcock tracking for distributing the samples along the view ray, which obtains consistent solutions with iterative previews and is free of ray marching artifacts. With this, we obtained high quality visualizations of FTLE fields, which may serve as ground truth or are available for marketing applications. A technical novelty of this chapter is that we tailored the acceleration data structure to FTLE, by progressively converging to correct local upper extinction bounds, which are not available a-priori in Lagrangian scalar fields.

In the future, we would like to incorporate approximating techniques to provide faster previews, such as pathline interpolation methods [COJ15, AOGJ15]. Further, we would like to experiment with adaptive sampling and reconstruction filters [ZJL*15]. The recent residual tracking and ratio tracking [NSJ14] are alternative approaches to evaluate the transmittance, which could be evaluated in the context of Lagrangian scalar fields in future work. Using regular grids to find local bounds of the majorant extinction as in [SKTM11] requires to set the right voxel resolution manually to get the best performance. Yue et al. [YIC*10] avoided this by construction of a kd-tree with an adequate stopping criterion, which we would like to test. Our method is purely image-based. For further processing of the FTLE ridge surfaces, geometry extraction techniques are still a challenging topic.

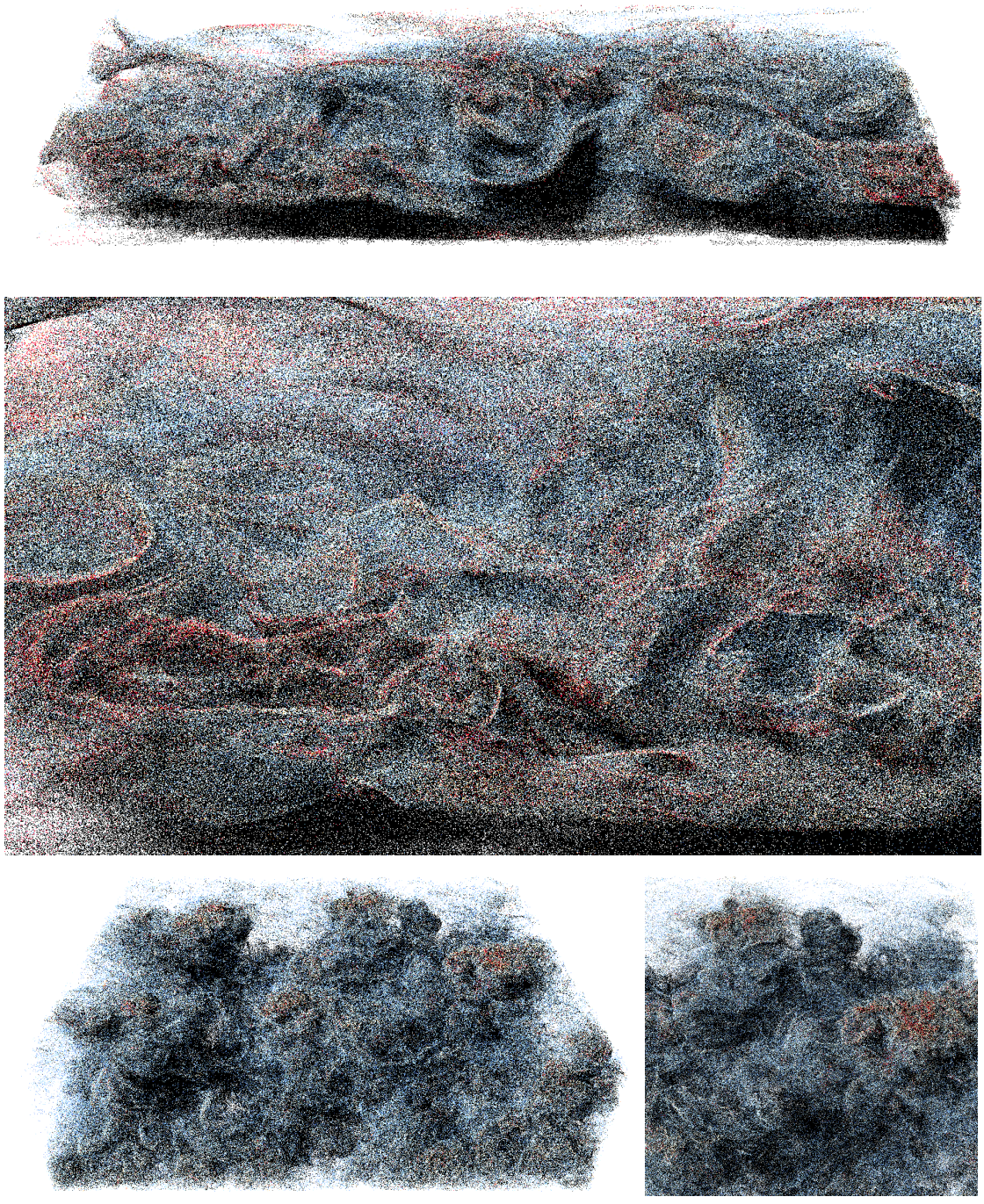


Figure 7.11.: MCFTLE results in ECMWF and CTBL after 1 iteration.

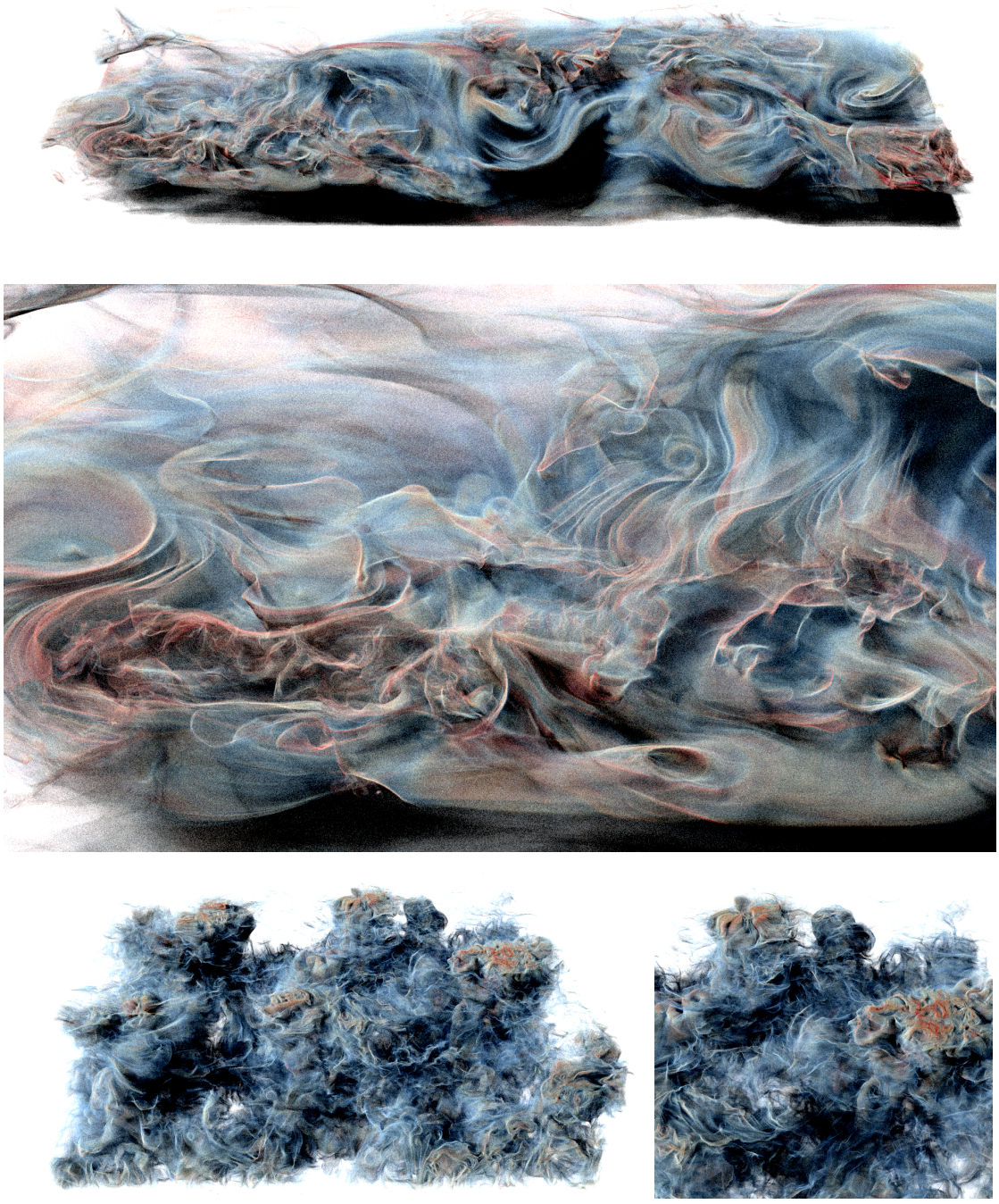


Figure 7.12.: MCFTLE results in ECMWF and CTBL after 100 iterations.

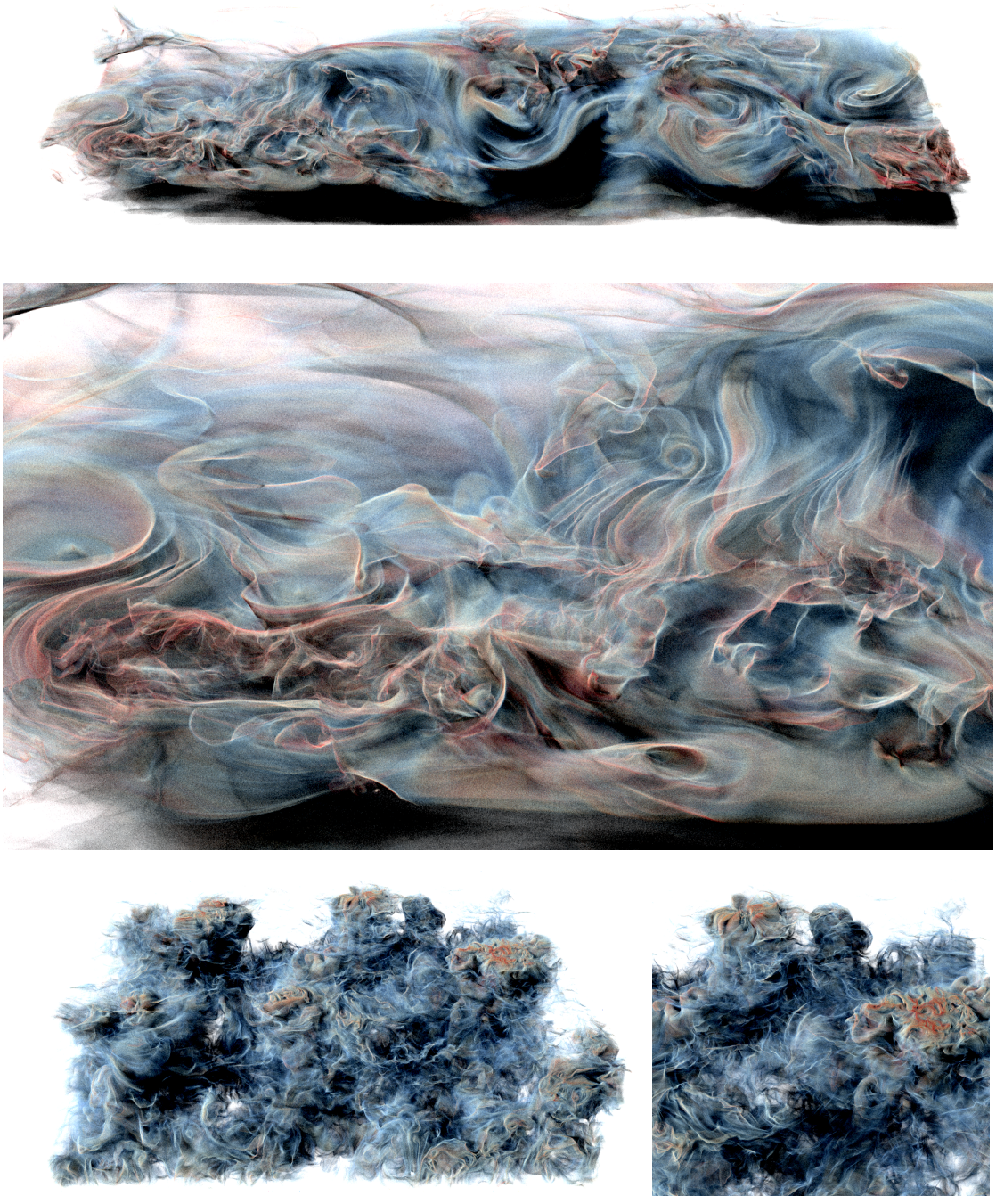


Figure 7.13.: MCFTLE results in ECMWF and CTBL after 200 iterations.

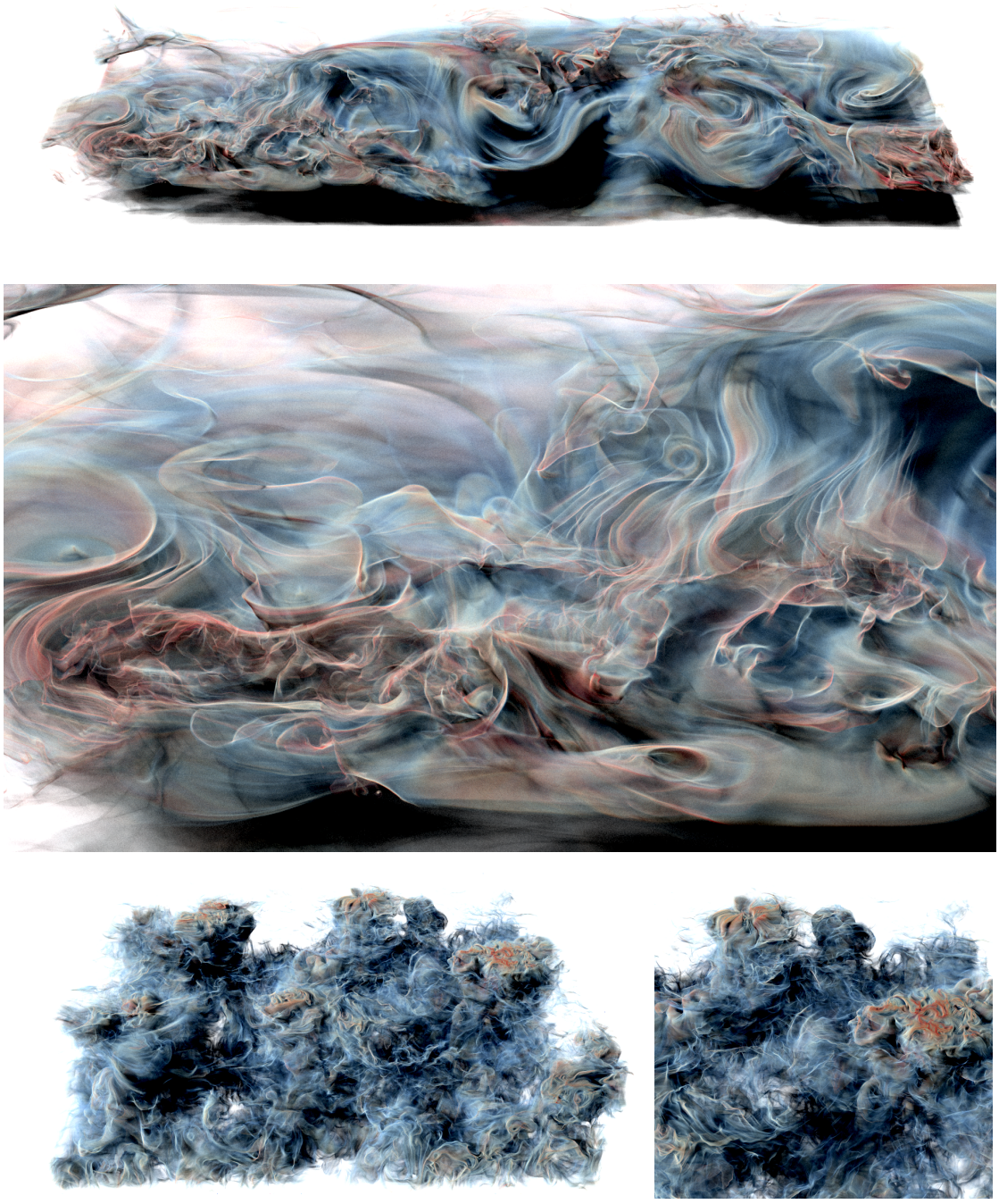


Figure 7.14.: MCFTLE results in ECMWF and CTBL after 1000 iterations.

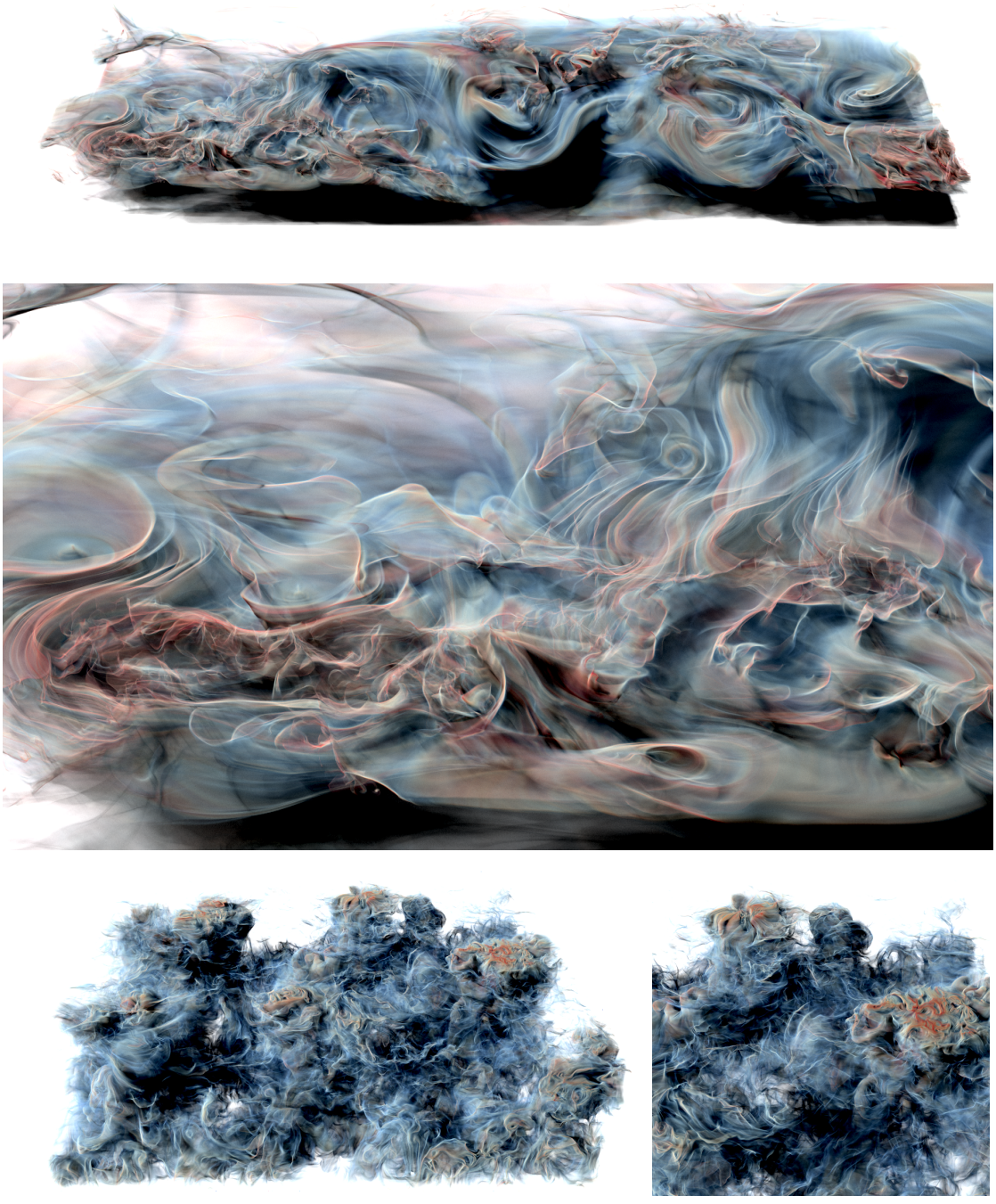


Figure 7.15.: MCFTLE results after $10\,k$ (ECMWF), $6.2\,k$ (ECWMF CLOSE), $5.2\,k$ (CTBL) and $4.4\,k$ (CTBL CLOSE) iterations.

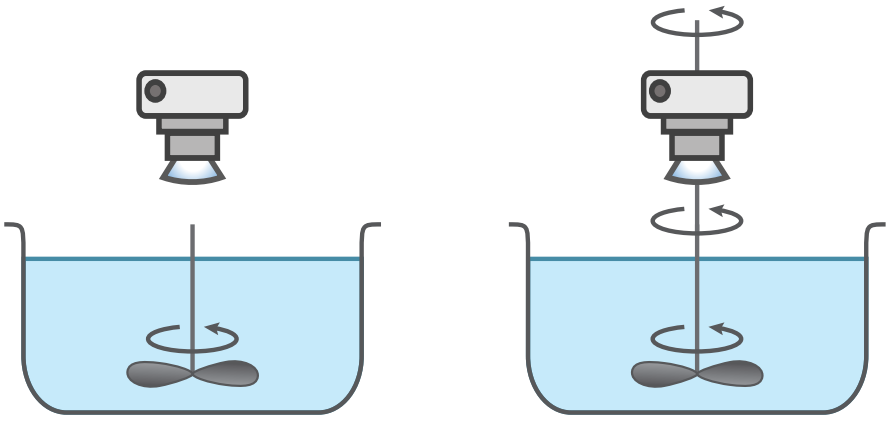
Chapter 8.

8 Rotation Invariant Vortices for Flow Visualization

In the following, we describe a general approach to obtain vortex concepts that are invariant under equal-speed rotations of the reference frame. The chapter is based on the publication:

T. Günther, M. Schulze and H. Theisel
Rotation Invariant Vortices for Flow Visualization
IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization 2015) 22, 1 (2016), 817–826.

This chapter focuses on vortices in flows that are obtained by rotating mechanical parts around a fixed and known axis. Examples are stirring devices, helicopters, hydrocyclones, centrifugal pumps, or ventilators. To illustrate the problem, consider a cylindrical container filled with a fluid and an inserted propeller, as shown in Fig. 8.1a. Rotating the propeller with constant angular speed induces a fluid motion. We are interested in its vortices. As depicted by a camera in Fig. 8.1a, the observer has a fixed position above the container, meaning that the observer sees the propeller rotating and the container standing still. In Fig. 8.1b, we have the same configuration, but now the observer is "sitting" on the propeller and rotating with it. This means that in the local coordinate system of the observer the propeller is standing still while the container is rotating. Computing vortices in the different reference frames usually gives different results, even though the applied vortex concepts are Galilean invariant. Which reference coordinate system should be used to get "correct" vortices? A first assumption may be that in areas close to the propeller configuration 8.1b gives better results because there the flow has a rotational speed similar to the propeller, and that close to the wall of the container configuration 8.1a is better if a no-slip boundary is assumed. This, however, raises the question how to treat the areas between propeller



(a) Fixed reference frame (b) Rotating reference frame

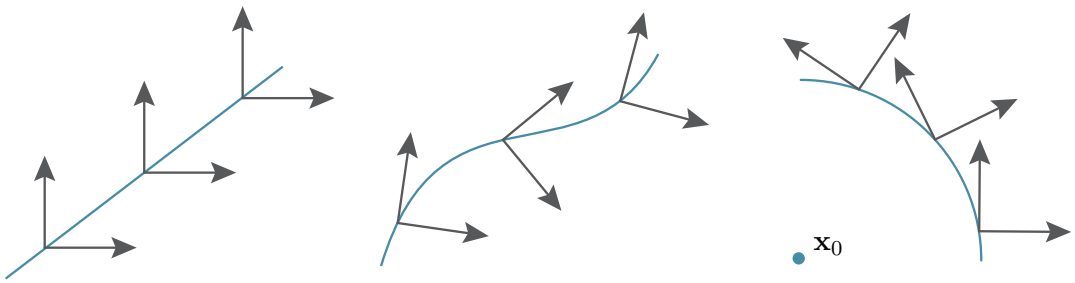
Figure 8.1.: Rotating flow with different positions of the observer; (a) observer has a fixed position and sees the propeller rotate; (b) observer is "sitting" on the propeller and sees the propeller standing still.

and wall. Because for those, the observer may have an optimal own rotation between zero and the propeller. In fact, the choice of the rotating reference frame of the observer has a great influence on the resulting vortices. How to choose it correctly?

This chapter presents a solution for this problem: we propose new vortex concepts for which the choice of the rotating reference frame does not matter, i.e., vortex concepts that are invariant under the relative rotation of the observer. We call them *rotation invariant vortices*. Starting with a formal definition of rotation invariance, we present a simple and generic approach to transform a Galilean invariant vortex concept to a rotation invariant one. In fact, it reduces to a slight modification of the Jacobian, i.e., adding of a simple closed-form matrix. We apply this to several standard vortex concepts: cores of swirling particle motion [WSTH07], λ_2 [JH95], and the Q criterion [Hun87]. We demonstrate our method on several rotating flows, showing that the rotation invariant methods give better results than their Galilean invariant counterparts.

8.1. Rotation Invariance

To give a formal definition of rotation invariance, we consider a fixed rotation center point $\mathbf{x}_0 = (x_0, y_0)^{\mathrm{T}}$ for $n = 2$ or a fixed rotation axis given by a point $\mathbf{x}_0$ and a normalized vector $\mathbf{n}$ for $n = 3$.



(a) Galilean invariance (b) Objectivity (c) Rotation invariance

Figure 8.2.: Invariance under different transformations of the coordinate frame (i.e., the camera): Galilean invariant (left), objective (middle), rotation invariant (right).

Definition 3 *A vortex measure is* rotation invariant *if it is invariant under any equal speed rotation of the reference system. For* $n = 2$ *by*

$$\mathbf{g}(\mathbf{x},t) = \begin{pmatrix} \cos(\omega\ t+\omega_0) & \sin(\omega\ t+\omega_0) \\ -\sin(\omega\ t+\omega_0) & \cos(\omega\ t+\omega_0) \end{pmatrix} (\mathbf{x}-\mathbf{x}_0)+\mathbf{x}_0 \tag{8.1}$$

and for $n = 3$ *by*

$$\mathbf{g}(\mathbf{x},t) = \mathbf{Q}_0^{-1} \begin{pmatrix} \cos(\omega\ t+\omega_0) & \sin(\omega\ t+\omega_0) & 0 \\ -\sin(\omega\ t+\omega_0) & \cos(\omega\ t+\omega_0) & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{Q}_0\ (\mathbf{x}-\mathbf{x}_0)+\mathbf{x}_0 \tag{8.2}$$

where $\mathbf{Q}_0$ *is the rotation matrix transforming* $\mathbf{n}$ *to the* z*-axis and* ω, ω_0 *are arbitrary scalars. Then, the following holds: the vortex measure classifies a point* $(\mathbf{x},t)$ *to be on the vortex in* $\mathbf{u}$ *iff the vortex measure classifies the point* $(\mathbf{g}(\mathbf{x},t),t)$ *to be on a vortex in the domain transformed field* $\mathbf{w}$.

Note that (8.1) and (8.2) describe rotations with constant angular speed of the observer around a fixed point/axis. Also note that up to now all introduced domain transformations (6.6), (6.10), (8.1) and (8.2) describe isometric deformations, i.e., they transform a square to a square of the same size. Fig. 8.2 illustrates the concepts.

Clearly, standard Galilean invariant vortex concepts like cores of swirling particle motion, λ_2, or Q are generally not rotation invariant. We describe now a simple generic approach to transform a Galilean invariant concept to a rotation invariant one. For $n = 2$, we introduce a non-isometric domain deformation $\mathbf{g}_p(\mathbf{x},t)$ and its inverse $\mathbf{h}_p(\mathbf{x},t)$

$$\mathbf{g}_p(\mathbf{x},t) = \begin{pmatrix} \arctan\frac{y-y_0}{x-x_0} \\ \|\mathbf{x}-\mathbf{x}_0\| \end{pmatrix}\ ,\ \mathbf{h}_p(\mathbf{x},t) = \mathbf{x}_0 + y\begin{pmatrix} \cos x \\ \sin x \end{pmatrix} \tag{8.3}$$

along with its domain transformed vector field $\mathbf{w}_p$. Eq. (8.3) describes a domain transformation to polar coordinates. Instead of applying a Galilean invariant vortex criterion to $\mathbf{u}$ at $(\mathbf{x}, t)$, we apply it to $\mathbf{w}_p$ at $\mathbf{g}_p(\mathbf{x}, t)$. In other words: we domain transform $\mathbf{u}$ to polar coordinates, do the vortex extraction there, and transform the resulting vortex structures back to the original Cartesian coordinates. Note that this gives in general different vortex structures than an application of the vortex measure in Cartesian coordinates because the non-isometric domain transformation induces a non-linear transformation of the Jacobian. Also note that this way we obtain a rotation invariant vortex measure if the vortex measure applied to $\mathbf{w}_p$ is Galilean invariant: an equal-speed rotation around $\mathbf{x}_0$ in the domain of $\mathbf{u}$ corresponds to an equal-speed translation of the coordinate system in the domain of $\mathbf{w}_p$, see Fig. 8.3. Hence, a Galilean invariant measure in $\mathbf{w}_p$ becomes a rotation invariant measure in $\mathbf{u}$.

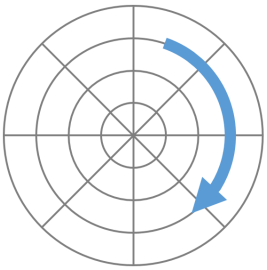
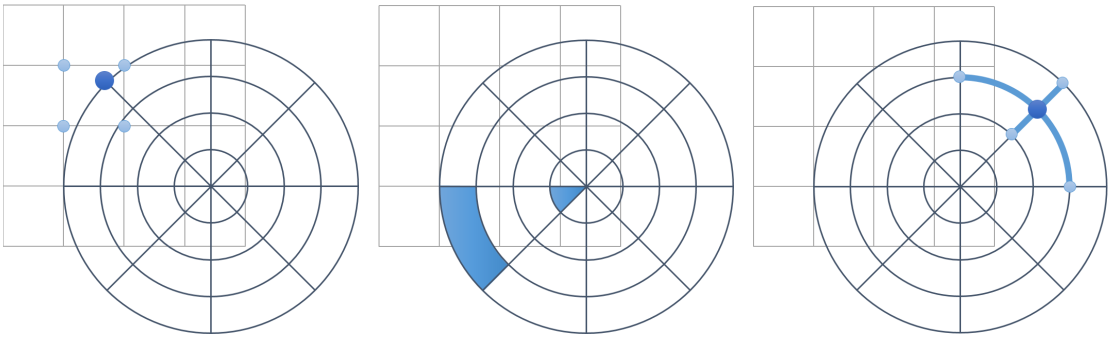


Figure 8.3.: Rotation in Cartesian frame equals translation in polar coordinates.

Although the domain transformation to polar coordinates is the theoretical solution of our problem, it creates new practical problems concerning the computation of the Jacobian in polar coordinates. In particular, it is unclear on which discretization to compute the Jacobian in polar coordinates since the domain transformation $\mathbf{g}_p$ maps cells with planar faces in Cartesian coordinates to cells of different size and density with non-planar faces in polar coordinates, see Fig. 8.4. To address this problem, we present an approach to compute the (modified) Jacobian in Cartesian coordinates only, without a transformation to polar coordinates. For this, we can directly use the grid discretization coming from the simulation. To introduce this, we keep in mind that most vortex measures are computed by a certain combination of $\mathbf{u}$, $\mathbf{u}_t$ and $\mathbf{J}$. This



(a) Interpolation/resampling (b) Varying cell sizes (c) Discretization

Figure 8.4.: Illustration of the problems related to the computation of a Jacobian in polar coordinates, when the flow data is provided on a regular grid.

means that after the domain transformation we have to consider $\mathbf{w}_p(\mathbf{g}_p(\mathbf{x},t),t)$ and its Jacobian $\mathbf{J}_p(\mathbf{g}_p(\mathbf{x},t),t) = \nabla(\mathbf{w}_p(\mathbf{g}_p(\mathbf{x},t),t))$. Instead of considering $\mathbf{w}_p$ and $\mathbf{J}_p$, we transform them back to Cartesian coordinates

$$\mathbf{u}_r(\mathbf{x},t) = (\nabla \mathbf{g}_p(\mathbf{x},t))^{-1} \cdot \mathbf{w}_p(\mathbf{g}_p(\mathbf{x},t),t) \tag{8.4}$$

$$\mathbf{u}_{rt}(\mathbf{x},t) = (\nabla \mathbf{g}_p(\mathbf{x},t))^{-1} \cdot \frac{\partial}{\partial t}\mathbf{w}_p(\mathbf{g}_p(\mathbf{x},t),t) \tag{8.5}$$

$$\mathbf{J}_r(\mathbf{x},t) = (\nabla \mathbf{g}_p(\mathbf{x},t))^{-1} \cdot \nabla(\mathbf{w}_p(\mathbf{g}_p(\mathbf{x},t),t)). \tag{8.6}$$

Fortunately, (8.4)–(8.6) have a simple closed-form solution:

$$\mathbf{u}_r = \mathbf{u} \tag{8.7}$$

$$\mathbf{u}_{rt} = \frac{\partial \mathbf{u}_r}{\partial t} = \mathbf{u}_t \tag{8.8}$$

$$\mathbf{J}_r = \mathbf{J} + \frac{1}{d}\,\mathbf{R}\,\mathbf{H}\,\mathbf{R}^{\mathrm{T}} = (\mathbf{u}_{xr}, \mathbf{u}_{y_r}) \tag{8.9}$$

with

$$d = \|\mathbf{x}-\mathbf{x}_0\| \quad , \quad \mathbf{r} = \frac{\mathbf{x}-\mathbf{x}_0}{d} \quad , \quad \mathbf{r}_p = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{r} \tag{8.10}$$

and the 2D matrices

$$\mathbf{R} = (\mathbf{r}_p, \mathbf{r}) \quad , \quad \mathbf{H} = \begin{pmatrix} -\mathbf{u}^{\mathrm{T}}\mathbf{r} & -\mathbf{u}^{\mathrm{T}}\mathbf{r}_p \\ \mathbf{u}^{\mathrm{T}}\mathbf{r}_p & 0 \end{pmatrix}. \tag{8.11}$$

See Appendix B for a derivation. Eqs. (8.7)–(8.9) have an interesting meaning: in order to transform a Galilean invariant vortex measure to a rotation invariant one, we only have to replace $\mathbf{J}$ by $\mathbf{J}_r$. Note that $\mathbf{J}_r$ is obtained by adding a closed-form matrix to $\mathbf{J}$, meaning that we do not have to worry about grid discretization in a transformation.

From the rotation invariant Jacobian, we directly get rotation invariant acceleration and feature flow field:

$$\mathbf{a}_r = \mathbf{J}_r\,\mathbf{u} + \mathbf{u}_t \quad , \quad \overline{\mathbf{a}}_r = \overline{\mathbf{J}}_r\,\overline{\mathbf{p}} \tag{8.12}$$

with

$$\overline{\mathbf{J}}_r = \begin{pmatrix} \mathbf{J}_r & \mathbf{u}_t \\ \mathbf{0}^{\mathrm{T}} & 0 \end{pmatrix}, \tag{8.13}$$

and

$$\overline{\mathbf{f}}_r = \begin{pmatrix} \det(\mathbf{u}_{y_r}, \mathbf{u}_t) \\ \det(\mathbf{u}_t, \mathbf{u}_{xr}) \\ \det(\mathbf{u}_{xr}, \mathbf{u}_{y_r}) \end{pmatrix} \quad , \quad \mathbf{f}_r = \frac{1}{\det(\mathbf{u}_{xr}, \mathbf{u}_{y_r})} \begin{pmatrix} \det(\mathbf{u}_{y_r}, \mathbf{u}_t) \\ \det(\mathbf{u}_t, \mathbf{u}_{xr}) \end{pmatrix}.$$

The case $n = 3$ is similar and a straightforward extension of the case $n = 2$. Instead

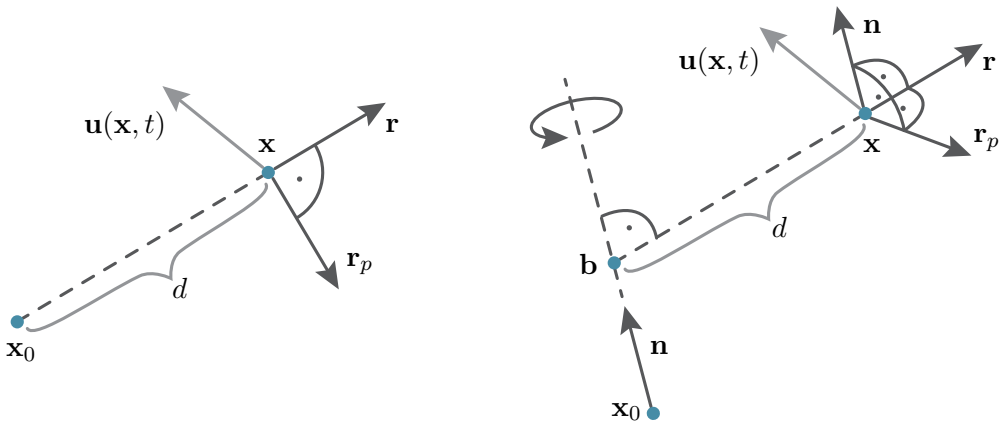


Figure 8.5.: Setup for computing the rotation invariant Jacobian for $n = 2$ (left) and $n = 3$ (right).

of a domain deformation to polar coordinates, we do a deformation to cylindrical coordinates around the axis given by $\mathbf{x}_0$ and $\mathbf{n}$. A transformation of the Jacobian back to Cartesian space gives the rotation invariant Jacobian

$$\mathbf{J}_r = \mathbf{J} + \frac{1}{d}\,\mathbf{R}\,\mathbf{H}\,\mathbf{R}^{\mathrm{T}} = (\mathbf{u}_{xr}, \mathbf{u}_{y_r}, \mathbf{u}_{zr})$$

with $\mathbf{b}$ being the point on the rotation axis with shortest distance to $\mathbf{x}$, i.e., $(\mathbf{x}-\mathbf{b})^{\mathrm{T}}\mathbf{n} = 0$, and

$$d = \|\mathbf{x} - \mathbf{b}\| \quad , \quad \mathbf{r} = \frac{\mathbf{x} - \mathbf{b}}{d} \quad , \quad \mathbf{r}_p = \mathbf{r} \times \mathbf{n} \tag{8.14}$$

and the 3D matrices

$$\mathbf{R} = (\mathbf{r}_p, \mathbf{r}, \mathbf{n}) \quad , \quad \mathbf{H} = \begin{pmatrix} -\mathbf{u}^{\mathrm{T}}\mathbf{r} & -\mathbf{u}^{\mathrm{T}}\mathbf{r}_p & 0 \\ \mathbf{u}^{\mathrm{T}}\mathbf{r}_p & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{8.15}$$

This gives for rotation invariant acceleration the same forms (8.12)–(8.13) as for $n = 2$, and for the rotation invariant feature flow field we have

$$\overline{\mathbf{f}}_r = \begin{pmatrix} -\det(\mathbf{u}_{y_r}, \mathbf{u}_{zr}, \mathbf{u}_t) \\ \det(\mathbf{u}_{zr}, \mathbf{u}_t, \mathbf{u}_{xr}) \\ -\det(\mathbf{u}_t, \mathbf{u}_{xr}, \mathbf{u}_{y_r}) \\ \det(\mathbf{u}_{xr}, \mathbf{u}_{y_r}, \mathbf{u}_{zr}) \end{pmatrix}, \; \mathbf{f}_r = \frac{1}{\det(\mathbf{u}_{xr}, \mathbf{u}_{y_r}, \mathbf{u}_{zr})} \begin{pmatrix} -\det(\mathbf{u}_{y_r}, \mathbf{u}_{zr}, \mathbf{u}_t) \\ \det(\mathbf{u}_{zr}, \mathbf{u}_t, \mathbf{u}_{xr}) \\ -\det(\mathbf{u}_t, \mathbf{u}_{xr}, \mathbf{u}_{y_r}) \end{pmatrix}.$$

Fig. 8.5 illustrates the setups for 2D and 3D.

8.2. Rotation Invariant Vortex Measures

We can now propose particular rotation invariant vortex measures by inserting $\mathbf{J}_r$ in the respective definitions of the previously Galilean invariant measures.

8.2.1. Rotation Invariant Cores of Swirling Particle Motion

For $n = 2$, any condition of (6.7) gives Galilean invariant cores of swirling particle motion. For rotation invariant cores of swirling particle motion we propose any of the following equivalent conditions:

$$\mathbf{a}_r = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{u} - \mathbf{f}_r = \mathbf{0} \quad \Leftrightarrow \quad \overline{\mathbf{f}}_r \parallel \overline{\mathbf{p}} \quad \Leftrightarrow \quad \overline{\mathbf{J}}_r\, \overline{\mathbf{p}} \parallel \overline{\mathbf{p}}. \tag{8.16}$$

For $n = 3$, the Galilean invariant conditions are in (6.8). In addition to this, we propose a new Galilean invariant vortex measure (which we already mentioned in Eq. (6.9)):

$$\nabla(\mathbf{u} - \mathbf{f}) \cdot (\mathbf{u} - \mathbf{f}) \parallel (\mathbf{u} - \mathbf{f}), \tag{8.17}$$

i.e., we apply Sujudi-Haimes to $\mathbf{u} - \mathbf{f}$. In case of vortices moving along an equal-speed translation, we can assume $\nabla \mathbf{f} = \mathbf{0}$, meaning that (6.8) and (8.17) show the same vortices. However, in the rotation invariant case, (8.17) performs better, as shown later in Section 8.4. We propose

$$\nabla(\mathbf{u} - \mathbf{f}_r) \cdot (\mathbf{u} - \mathbf{f}_r) \parallel (\mathbf{u} - \mathbf{f}_r) \tag{8.18}$$

for rotation invariant cores of swirling particle motion.

8.2.2. Rotation Invariant Region-based Methods

For the definition of rotation invariant region-based measures, we consider the decomposition of the rotation invariant Jacobian

$$\mathbf{J}_r = \mathbf{S}_r + \Omega_r \tag{8.19}$$

into the rate-of-strain tensor $\mathbf{S}_r = \frac{1}{2}(\mathbf{J}_r + \mathbf{J}_r^{\mathrm{T}})$ and the vorticity tensor $\Omega_r = \frac{1}{2}(\mathbf{J}_r - \mathbf{J}_r^{\mathrm{T}})$. Following [JH95], we define the rotation invariant λ_{2r} criterion by considering the second-largest eigenvalue of $\mathbf{S}_r^2 + \Omega_r^2$:

$$\lambda_{2r} = \lambda_2(\mathbf{S}_r^2 + \Omega_r^2) < 0. \tag{8.20}$$

Similarly, we follow [Hun87] and define the rotation invariant Q_r criterion

$$Q_r = \frac{1}{2}\left(\|\Omega_r\|^2 - \|\mathbf{S}_r\|^2\right) > 0, \tag{8.21}$$

which characterizes vortices as regions in which the Euclidean norm of the rotation invariant vorticity tensor dominates that of the rotation invariant rate-of-strain tensor.

8.3. On the Construction of Test Data

In the previous section, we added rotation invariant extractors to the tool box of local vortex extraction methods. In order to create test data, it is useful to understand, which options we have to construct rotating vector fields from a steady vector field.

Slice-by-Slice Rotation. The first construction method is to linearly rotate the steady vector field slice-by-slice, i.e, rotate both coordinates and vectors for each time slice individually. Here, the *reference frame remains fixed*, i.e., we watch the flow rotate from an outside perspective. Essentially, this approach rotates the streamlines of the vector field. Imagine a LIC slice rotating over time around the normal of the slice. The streamline vortices remain visible on the LIC slice, i.e., streamline cores are not vanishing. Observing rotating streamlines makes sense for instantaneous vector fields such as magnetic fields. In those, the methods of Bauer and Peikert [BP02], and Theisel et al. [TSW*05] can be used to extract the corelines.

Adding a Center. A different approach is to add a center with a certain angular velocity to a steady field. Here, the *reference frame rotates* with the angular velocity of the center. Thus, in this frame, the flow appears to be steady. Note that adding a center to a steady vector field might cancel out weak vortices (depending on the rotation direction and strength). The vortices that are visible in the rotating frame, i.e., the (near-)steady flow, are pathline cores (PC), and since we add a center they might cancel out.

It is important to note that slice-by-slice rotation and adding of a center result in the *same* rotating flow, but in *different* reference frames. In fact, we can convert between the two using Eqs. (6.4) and (6.5), by using Eq. (8.1) or (8.2) for 2D or 3D, respectively. This means, if we prescribe streamline cores (SC), i.e., add a center in the rotating frame, we might lose pathline cores (PC).

Moving Rotationally Symmetric Flows. For rotationally symmetric flows the slice-by-slice rotation is similarly achieved by simply moving the flow on a circle, as the rotation is freely selectable. The BEADS flow reported by Wiebel et al. [WCW*09], for instance, was approximated in Weinkauf et al. [WT10a] as a focus sink moving on a circle. Then, the streamline coreline tracks the location of the sink. Pathline cores, however, rotate around a different center. In fact, the analogous streamline core approach of adding a center in the rotating frame, gives us already a hint. Since cores can be canceled out due to addition of the center, pathline cores cannot be the same as streamline cores.

All three aforementioned approaches have in common that they construct a vector field by prescribing a rotation on the streamlines.

Linearly Rotate Reference Frame. For the fourth method, we let the given steady flow be our flow in the rotating reference frame, and we transform the vector field into the fixed reference frame, without adding a center. Therefore, this fourth approach must lead to a different unsteady flow than the three methods described before. As here no center was added in the rotating frame, the pathline cores remain visible and are not vanishing. Instead, streamline cores might vanish. Previously, no local method existed that was able to find the pathline core in rotating flows. In both unsteady flows (SC and PC) our rotation invariant extractor delivers the correct pathline cores, regardless of the chosen reference frame.

8.4. Results

In the following, we demonstrate the rotation invariant vortex measures. We use several synthetic data sets and study two real-world flows.

Beads. The BEADS FLOW was reported by Wiebel et al. [WCW*09] and has previously been among the most challenging benchmarks for vortex extraction. An analytic approximation to it was given in [WT10a]:

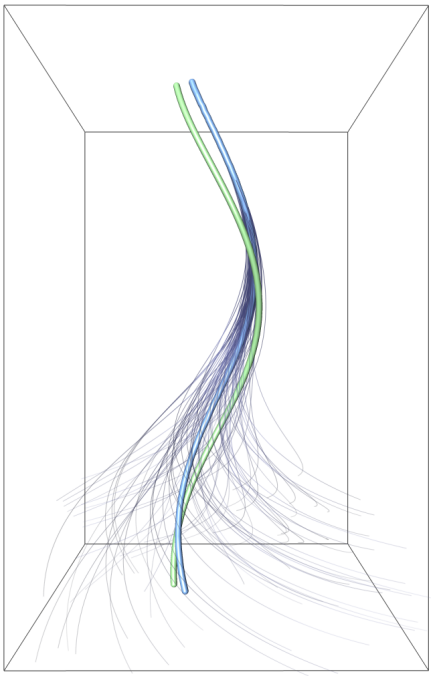
$$\mathbf{u}(x,y,t) = \begin{pmatrix} -(y - \frac{1}{3}\sin(t)) - (x - \frac{1}{3}\cos(t)) \\ (x - \frac{1}{3}\cos(t)) - (y - \frac{1}{3}\sin(t)) \end{pmatrix}. \tag{8.22}$$

The flow is defined in the domain $D{\times}T = [-2,2]^2{\times}[0,2\pi]$. Previously, all local vortex extractors failed and only integration-based methods could find the solution [WT10a], such as particle density estimation [WCW*09] or cores of swirling streaklines [WT10a]. As shown in Fig. 8.6a, our rotation invariant method is the first local method that finds the correct coreline, which was identified by Weinkauf et al. [WT10a] as the pathline $\mathbf{x}(t) = \frac{1}{3}(\sin(t) + \cos(t),\, -\cos(t) + \sin(t))^{\mathrm{T}}$.

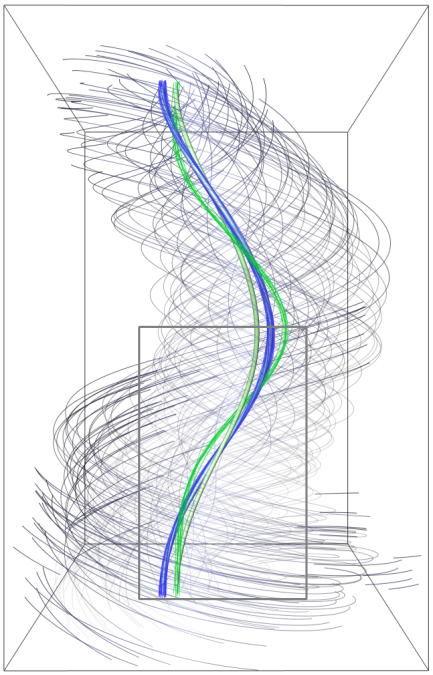
We go a step further and consider a modification of the BEADS FLOW that cannot be handled by integration-based particle density estimation, as it is *divergence-free*. Thus, neither standard local nor integration-based methods are applicable. That is, a center moving on a circle:

$$\mathbf{u}(x,y,t) = \begin{pmatrix} -2y + \frac{2}{3}\sin(t) \\ 2x - \frac{2}{3}\cos(t) \end{pmatrix}, \tag{8.23}$$

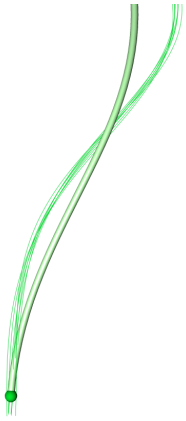
again defined in the domain $D{\times}T = [-2,2]^2{\times}[0,2\pi]$. Fig. 8.6b depicts the rotation invariant coreline and swirling pathlines as context, showing that our method delivers correct results in this data set, too. Here, the correct coreline is the pathline $\mathbf{x}(t) = \frac{2}{3}(\cos(t),\, \sin(t))^{\mathrm{T}}$.



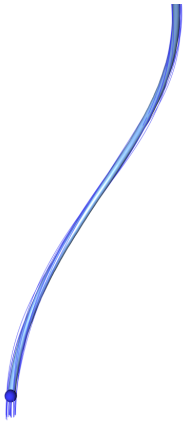
(a) Standard Beads flow.



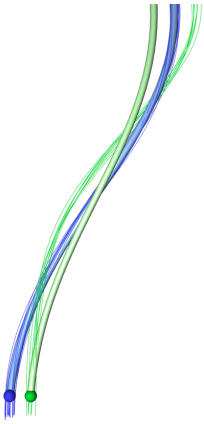
(b) Divergence-free Beads flow.



(c) Galilean invariant coreline with pathlines.



(d) Rotation invariant coreline with pathlines.



(e) Images (c) and (d) shown together.

Figure 8.6.: Local corelines in the standard BEADS flow in (a), and its divergence-free version in (b), both visualized in space-time where time denotes the vertical axis. Images (c)–(e) show close-ups of (b), with pathline seeds depicted by colored spheres. Green corelines are obtained by the Galilean invariant cores of swirling particle motion (wrong result), whereas blue corelines are computed with our rotation invariant extension (correct result). In image (b)–(e), pathlines are seeded on extracted corelines, showing that green pathlines (seeded at Galilean invariant coreline) spiral away, whereas blue pathlines stay at the rotation invariant coreline.

Four Rotating 2D Centers. The FOUR CENTERS (SC) data set is based on the scalar field

$$s(x,y) = 3\,x\,y \cdot e^{-x^2-y^2} \tag{8.24}$$

and is defined in the domain $D = [-2,\ 2]^2$. Its co-gradient vector field contains four centers. The abbreviation SC stems from *streamline core*, as we construct the unsteady flow by rotation of the streamlines, which prescribes the location of the streamline cores. That is, over time, we slice-by-slice rotate the underlying scalar field around $\mathbf{x}_0 = \mathbf{0}$, considering it in the temporal domain $T = [0,\ 2\pi]$:

$$\mathbf{u}(x,y,t) = \begin{pmatrix} \frac{\partial}{\partial y} s(x',y') \\ -\frac{\partial}{\partial x} s(x',y') \end{pmatrix} \tag{8.25}$$

with $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(t) & \sin(t) \\ -\sin(t) & \cos(t) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$. Equivalently, the flow can be constructed by adding a center with the angular velocity of the above rotation, and transforming the flow using Eqs. (6.5) and (8.1) from the rotating reference frame into the fixed reference frame. While the steady field contains four streamline vortex corelines, the rotating system exhibits only two pathline cores, as shown in Fig. 8.7. This cancellation of cores is due to the addition of the center in the analogous flow in the rotating reference frame. We use this flow to demonstrate the rotation invariance of our method in Fig. 8.8. A direct application of Galilean invariant pathline cores [WSTH07] (which are for 2D unsteady flows identical to Sujudi-Haimes [SH95] in space-time), as in

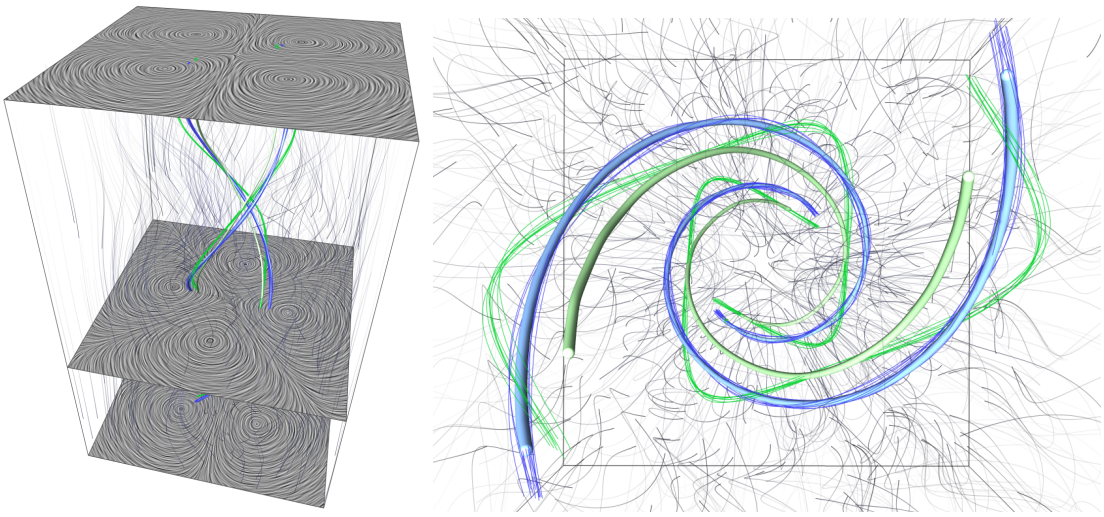
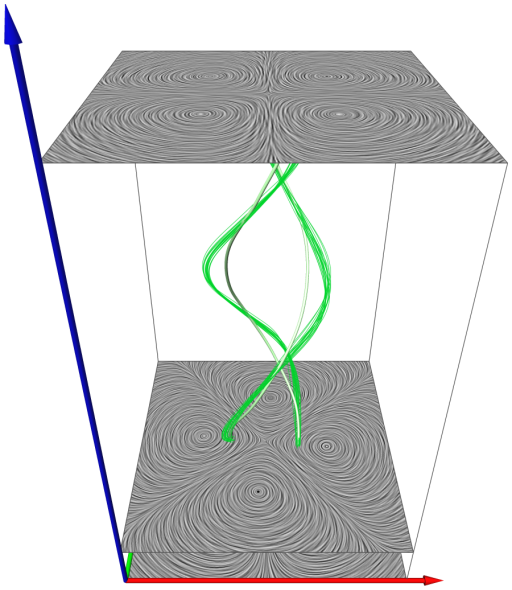
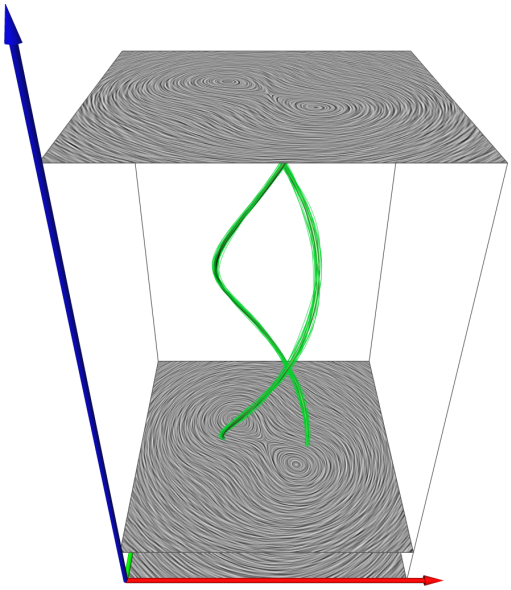


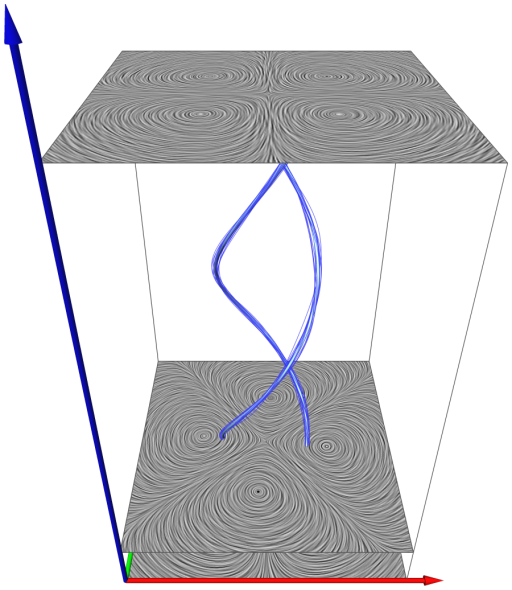
Figure 8.7.: Space-time visualization of the 2D unsteady FOUR CENTERS (SC) flow, with the camera viewing along the time axis and gray pathlines as context. Green corelines are obtained by cores of swirling motion, the blue corelines are their rotation invariant counterparts. As visible by the green and blue pathlines, only rotation invariant cores exhibit swirling motion in their vicinity.



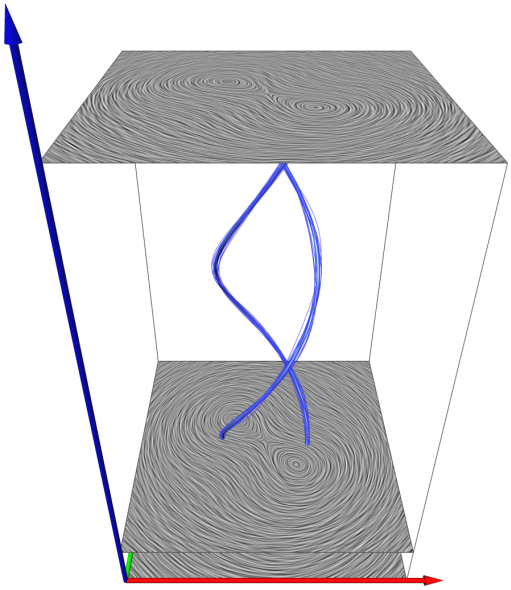
(a) Galilean invariant pathline cores (original reference frame)



(b) Galilean invariant pathline cores (modified reference frame)



(c) Rotation invariant pathline cores (original reference frame)



(d) Rotation invariant pathline cores (modified reference frame)

Figure 8.8.: The 2D unsteady FOUR CENTERS (SC) flow in space-time. Direct application of (Galilean invariant) pathline cores [WSTH07] in (a) does not reveal the correct corelines. It does produce correct corelines after an appropriate change of the reference frame (b). Our rotation invariant pathline cores in (c) and (d) find the centers of swirling motion in rotating systems regardless of the reference frame.

Fig. 8.8a does not find the correct rotation centers, as shown by green pathlines leaving the coreline. In this data set, a prior change to an appropriate reference frame, as in Fig. 8.8b, removes the rotational movement, making Galilean invariant pathline cores applicable. In this chapter, we extend existing Galilean invariant vortex core extractors to become *rotation invariant*, which extracts the correct rotation centers, regardless of the reference frame, see Figs. 8.8c and 8.8d.

Next, we construct another unsteady vector field named FOUR CENTERS (PC) by prescribing the flow in the rotating frame as:

$$\mathbf{w}(\hat{x},\hat{y},t) = \begin{pmatrix} -\hat{x}\cdot e^{-\hat{x}^2-\hat{y}^2}(2\hat{y}^2-1) \\ \hat{y}\cdot e^{-\hat{x}^2-\hat{y}^2}(2\hat{x}^2-1) \end{pmatrix} \tag{8.26}$$

with coordinates in the rotating frame $(\hat{x},\hat{y})^{\mathrm{T}} = \mathbf{g}(\mathbf{x},t)$ and rotate it using Eqs. (6.5) and (8.1) with $\omega = 1$ and $\omega_0 = 0$ into the fixed frame. With this, we prescribe the rotation of *pathline cores* (PC). In the fixed frame, the centers vanish, as visible in Fig. 8.9 (left). This time, we find four rotation invariant pathline cores, but only two (wrong) cores with the Galilean invariant counterpart. Our rotation invariant coreline extraction finds the correct cores of swirling pathlines in both unsteady flows (SC) and (PC).

Next, we compare the Galilean invariant region-based vortex criteria λ_2 and Q with their rotation invariant versions λ_{2r} and Q_r in the flow with prescribed pathline cores

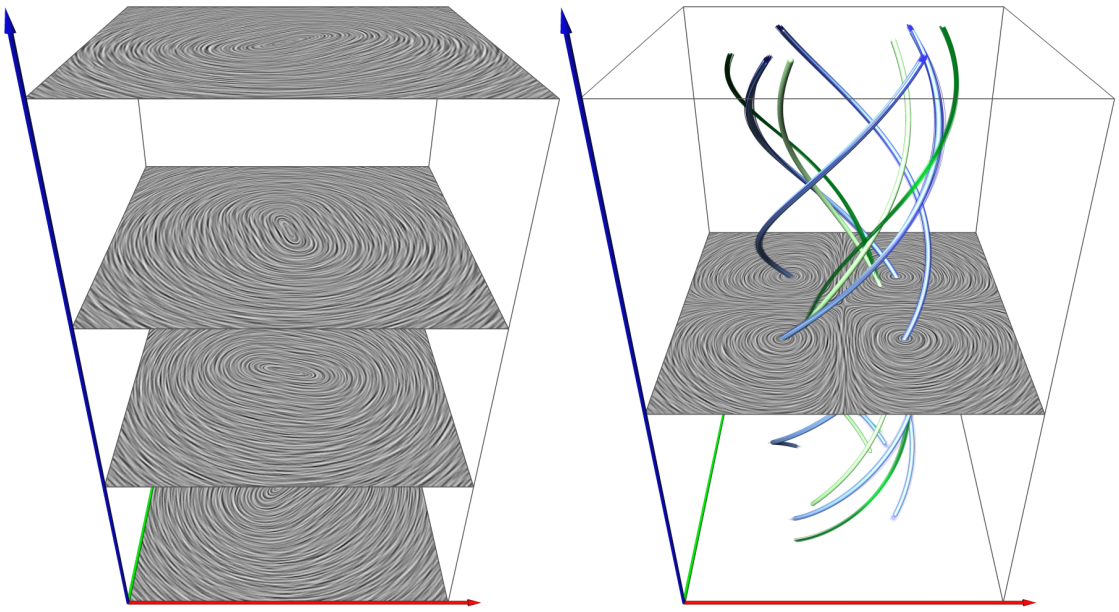


Figure 8.9.: FOUR CENTERS (PC) flow with prescribed centers in the rotating reference frame in space-time. In the fixed reference frame (left), the four centers partly vanished. Corelines were extracted in the fixed reference frame. The right image shows a LIC slice of the rotating frame, displaying the four centers.

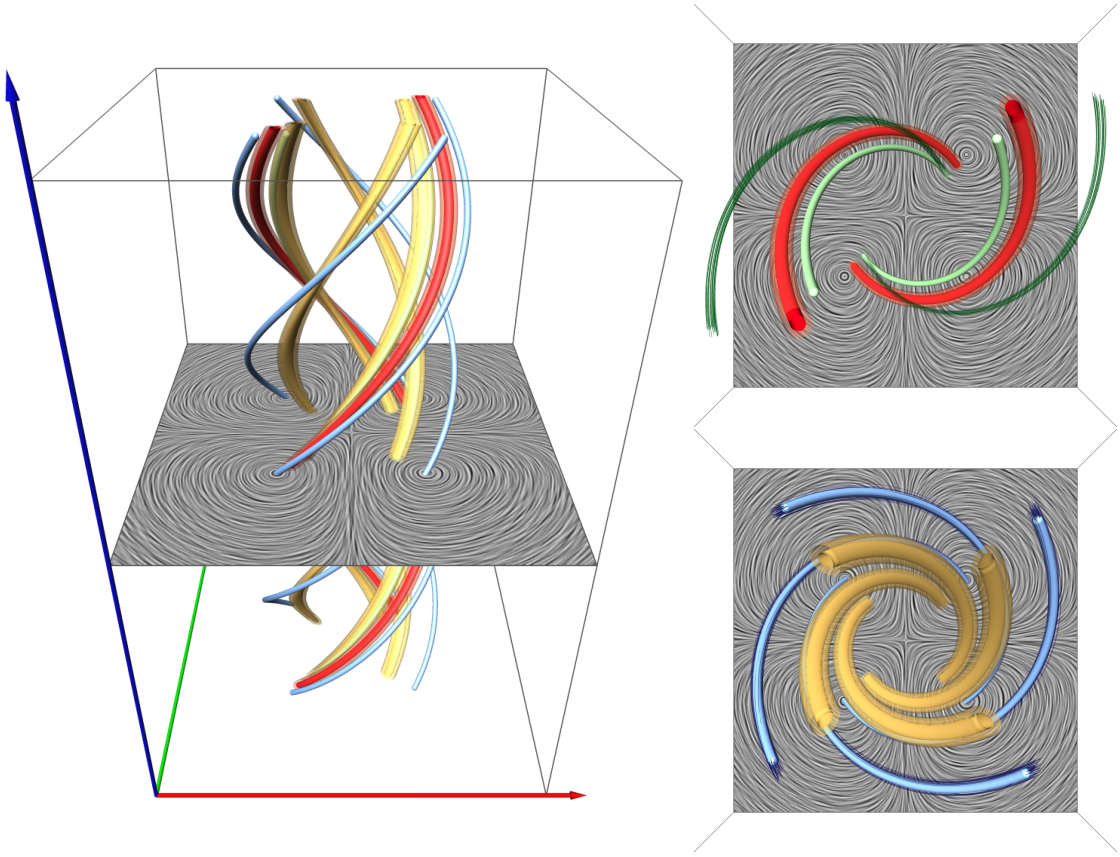


Figure 8.10.: Region-based methods in the FOUR CENTERS flow in space-time. Left: Overview image. Top right: Galilean invariant methods (two cores). Bottom right: Rotation invariant methods (four cores). Opaque isosurfaces show λ_2 (top) and λ_{2r} (bottom) and transparent isosurfaces depict Q (top) and Q_r (bottom). Here, Q and λ_2 deliver similar results. The same applies for Q_r and λ_{2r}.

(PC), i.e., we know that there are four vortices to expect. Fig. 8.10 (left) shows an overview of the methods, containing Galilean invariant pathline cores (green), rotation invariant pathline cores (blue), Galilean invariant region-based methods (red) and rotation invariant region-based methods (yellow). The top right image shows the Galilean invariant techniques, which detect only two cores. The green pathlines spiral away from the vortex corelines, showing that the classic Galilean invariant method delivers incorrect results. The bottom right image shows our rotation invariant extensions, which extract four vortex cores, as expected. The blue pathlines show that the rotation invariant vortex coreline is indeed the correct coreline.

Two Spiraling 2D Centers. A major strength of rotation invariance is that the angular speed of the underlying rotation must not be known in advance to obtain accurate extraction results. This is very helpful for non-linear rotations, in which the underlying

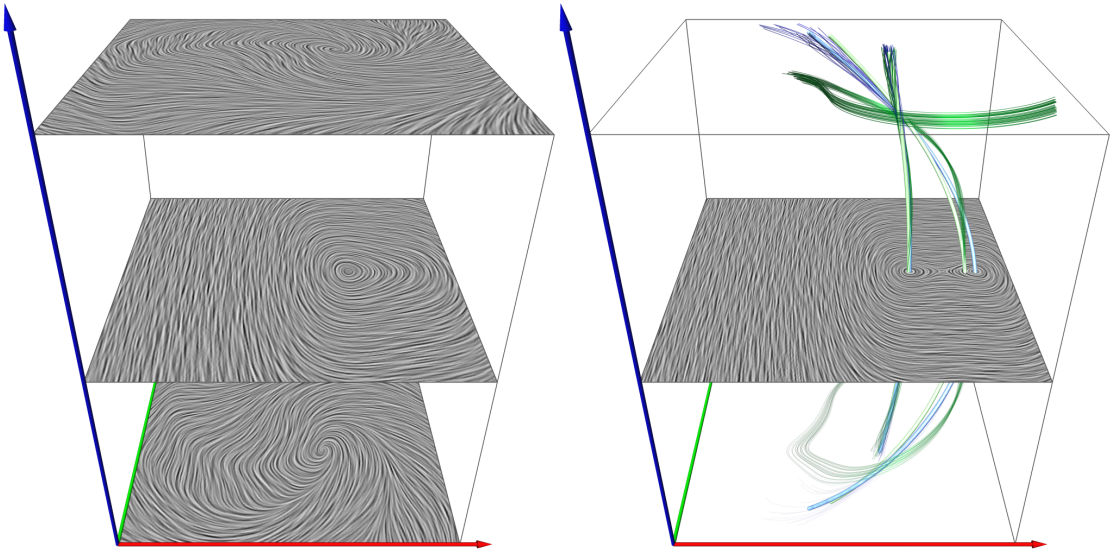


Figure 8.11.: TWO SPIRALING CENTERS in space-time with their center positions prescribed in the rotating reference frame. The left image visualizes the fixed reference frame, which was used to extract the corelines shown on the right. In the right image, pathlines were released from the corelines, and a LIC slice of the rotating reference frame is shown, displaying the two centers.

rotation is unknown or difficult to describe. An example is a system in which the angular speed decays or increases with distance to the center of rotation. We built a synthetic test data set to demonstrate such situation. We place centers at $(1,0)^{\mathrm{T}}$ and $(2,0)^{\mathrm{T}}$ in the rotating reference frame:

$$\mathbf{w}(\hat{x},\hat{y},t) = \begin{pmatrix} -2\hat{y} \\ 4\hat{x}^3 - 18\hat{x}^2 + 26\hat{x} - 12 \end{pmatrix} \tag{8.27}$$

with coordinates in the rotating frame $(\hat{x},\hat{y})^{\mathrm{T}} = \mathbf{g}(\mathbf{x},t)$ and transform it using Eqs. (6.5) and (8.1) with $\omega = 0.3(x^2+y^2)$ and $\omega_0 = 0$ into the fixed reference frame. We consider the flow in the domain $D \times T = [-2.5,\ 2.5]^2 \times [-\pi,\pi]$, as shown in Fig. 8.11. Note that in this flow the distance of a particle to the center of rotation remains constant, and thus a particle's angular speed does not change. It can be seen that with lower angular speed (closer to the global rotation center) the difference to cores of swirling particle motion becomes smaller, as the vector field is more steady, i.e., the temporal derivative $\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}$ is smaller. As shown by the pathlines, our rotation invariant method accurately extracts the vortex cores of swirling pathlines.

Rotating 3D Center. In the following, we construct a time-dependent 3D vector field. At time $t = 0$, it contains a center with its coreline through the points $(1,\ 0,\ 0)^{\mathrm{T}}$ and $(0,\ 1,\ 1)^{\mathrm{T}}$. (At $t = 0$, this coreline can be found using Sujudi-Haimes.) Over

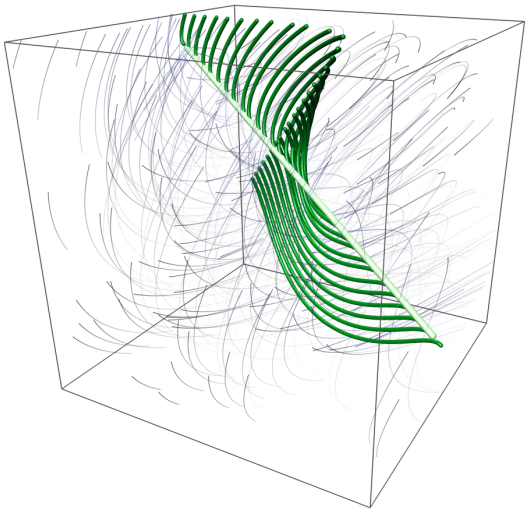
time, we rotate the center globally slice-by-slice around the z-axis, i.e., $\mathbf{x}_0 = \mathbf{0}$ and $\mathbf{n} = (0,\, 0,\, 1)^{\mathrm{T}}$. The resulting field in a fixed reference frame is defined in the domain $D \times T = [-5,\, 5]^3 \times [0, 2\pi]$ as:

$$\mathbf{u}(x,y,z,t) = \begin{pmatrix} \frac{\sqrt{3}}{3}(z\cos(t) + z\sin(t) - \cos(t) - y) \\ \frac{\sqrt{3}}{3}(z\cos(t) - z\sin(t) + \sin(t) + x) \\ -\frac{\sqrt{3}}{3}(x\cos(t) + y\cos(t) + x\sin(t) - y\sin(t) - 2) \end{pmatrix}$$

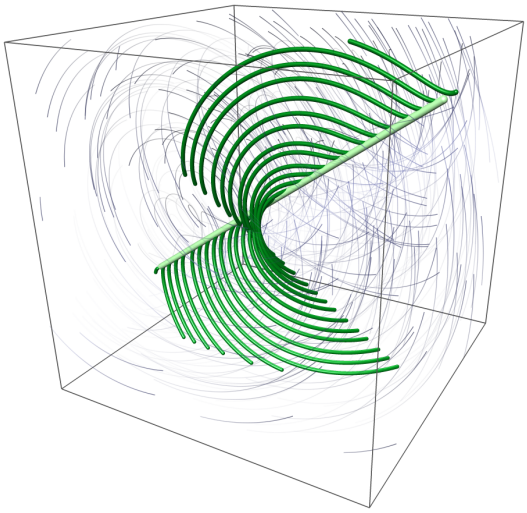
Fig. 8.12 depicts the rotating center and the vortex corelines therein. In Fig. 8.12a a streamline core is shown for $t = 1$ with pathlines seeded at the coreline and integrated up to $t = 4$. Similarly, Fig. 8.12b contains the streamline core of $t = 4$, with backwards integrated pathlines to $t = 1$. Comparing both, we see that pathlines seeded from one coreline do not reach the other coreline. Thus, the traced particles have not been seeded at a true vortex core. A direct extension of cores of swirling particle motion [WSTH07] does not result in the correct corelines, as shown in Fig. 8.12c. With our proposed generalization (8.18), the rotation invariant pathline cores can be found. This is demonstrated by the pathline ribbons that were seeded on one coreline and have reached the other. Further, their twist shows the swirling behavior of nearby particles. Note that this vector field has no Galilean invariant pathline cores, since $\det(\mathbf{J}) = 0$.

Centrifugal Pump. Our first real-world data set was subject of the IEEE Visualization Contest 2011. The data is courtesy of the Institute of Applied Mechanics, Clausthal University, Germany and was made available by Dipl. Wirtsch.-Ing. Andreas Lucius. We consider an unsteady 2D slice ($z = 0.01$) of the simulated flow based on the DES turbulence model, which is already provided in a rotating reference frame. Common practice is to track vortices in a reference frame rotating with the angular velocity of the blades. As we will show, this rotating frame is not ideal as the vortices are moving in the blank areas between the blades. In fact, our rotation invariant extractor finds cores of better quality. Fig. 8.13a depicts cores of swirling particle motion and our rotation invariant pathline cores.

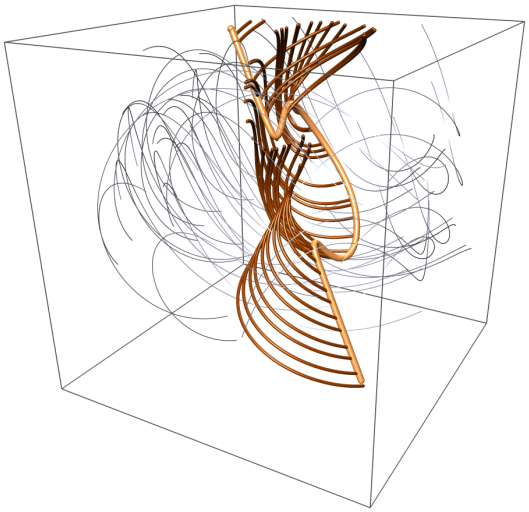
In the following, we explain how we evaluate the quality of the extracted corelines. Unfortunately, there is no ground truth location, since there is no universal vortex measure that can be considered best. There is, however, a rather commonly desired coreline property: Corelines should ideally be pathlines, since they would then represent particles that other particles swirl around, and hence, we get cores of swirling particle motion. We measure this property by the tangent alignment with the flow. We define the *tangent alignment* $\tau_{\mathbf{c}}(t)$ of a coreline $\mathbf{c}(t)$ as a scalar line attribute that is defined as the absolute value of the dot product between (space-time) vector field $\overline{\mathbf{p}}$



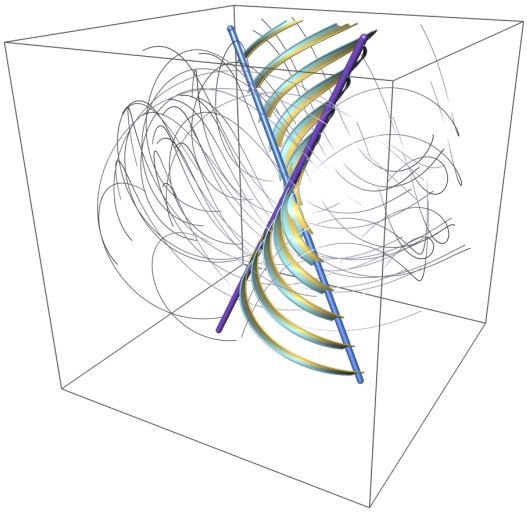
(a) Streamline core by Sujudi-Haimes at $t = 1$, streamlines in background.



(b) Streamline core by Sujudi-Haimes at $t = 4$, streamlines in background.



(c) A direct extension of [WSTH07], i.e., $\mathbf{J}_r(\mathbf{u}-\mathbf{f}_r) \parallel (\mathbf{u}-\mathbf{f}_r)$ is incorrect.

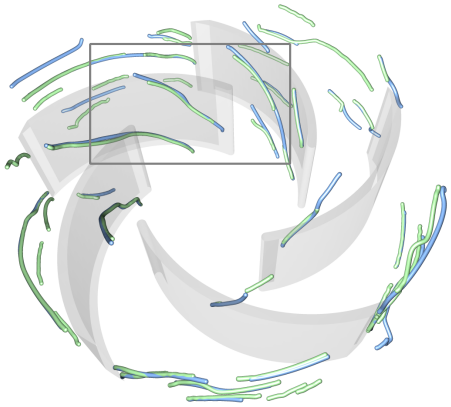


(d) Our correct rotation invariant pathline cores at $t = 1$ and $t = 4$.

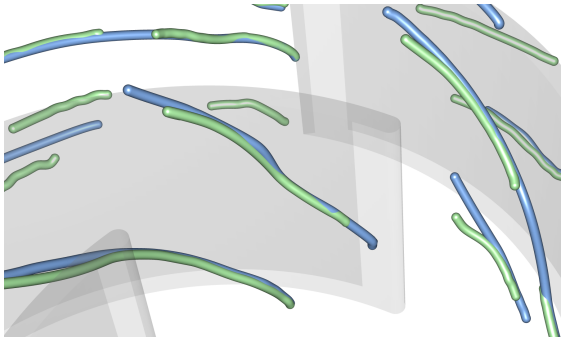
Figure 8.12.: Vortex cores in the ROTATING CENTER flow, with pathlines starting or ending at the corelines. Figs. (a) and (b) depict the flow at two instantaneous time steps, with streamlines in the background. A direct extension of cores of swirling particle motion yields wrong results (c). Image (d) shows the rotation invariant pathline cores at $t = 1$ (dark blue) and $t = 4$ (dark purple). Pathline ribbons from $t = 1...4$ connect the two corelines, showing both the coreline's movement path and the rotational movement of particles by the ribbon twist.

and the tangent of the space-time coreline $\overline{\mathbf{c}}(t) = \begin{pmatrix} \mathbf{c}(t) \\ t \end{pmatrix}$ (both normalized):

$$\tau_{\mathbf{c}}(t) = \left| \frac{\overline{\mathbf{p}}(\overline{\mathbf{c}}(t),t)}{\|\overline{\mathbf{p}}(\overline{\mathbf{c}}(t),t)\|} \cdot \frac{\mathrm{d}\overline{\mathbf{c}}(t)/\mathrm{d}t}{\|\mathrm{d}\overline{\mathbf{c}}(t)/\mathrm{d}t\|} \right| . \tag{8.28}$$

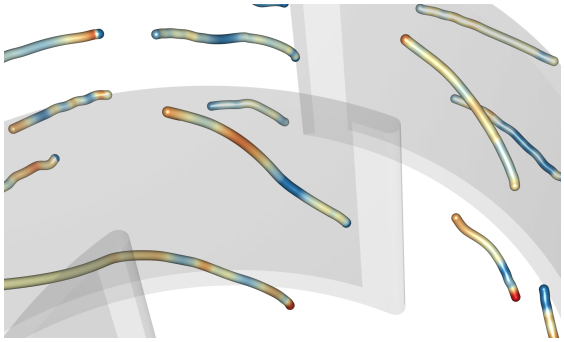


(a) Galilean (green) and rotation invariant (blue) cores.

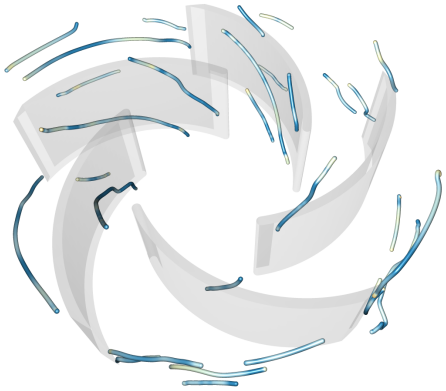


(b) Galilean (green) and rotation invar. (blue) cores.

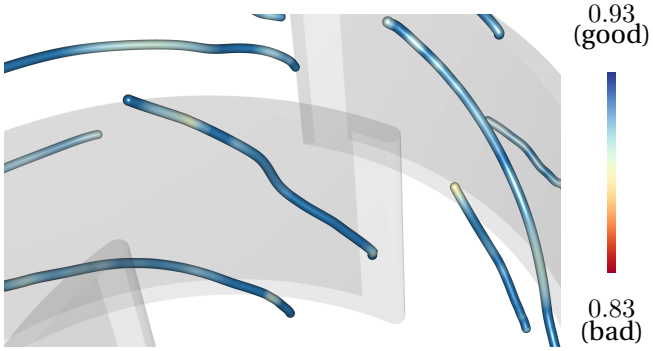
(c) Galilean invariant - Tangent alignment



(d) Galilean invariant - Tangent alignment

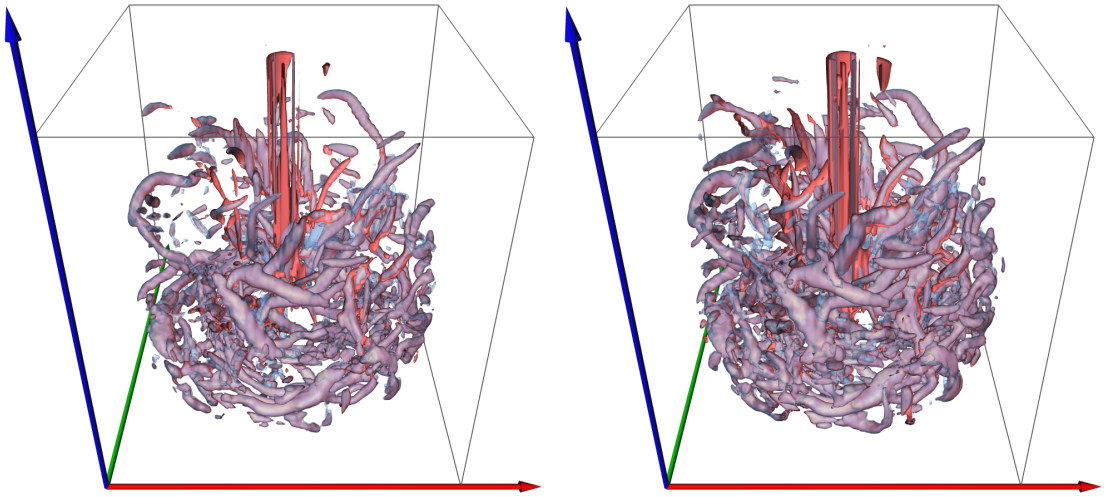


(e) Rotation invariant - Tangent alignment



(f) Rotation invariant - Tangent alignment

Figure 8.13.: Vortex cores of a CENTRIFUGAL PUMP in a rotating reference frame in space-time. While Galilean and rotation invariant cores are similar (a), the Galilean invariant cores (c) have poorer tangent alignment than our rotation invariant counterpart (e). The right column shows a close-up of the region highlighted in (a).



(a) isosurfaces $\lambda_2 = -60$ and $\lambda_{2_r} = -60$ (b) isosurfaces $Q = 40$ and $Q_r = 40$

Figure 8.14.: Region-based techniques in the ROTATING MIXER data set. Backfaces of the red isosurfaces depict the standard Galilean invariant λ_2 and Q criteria. The frontfaces of blue isosurfaces show our rotation invariant counterparts λ_{2_r} and Q_r.

As visible in Fig. 8.13c, Galilean invariant cores of swirling particle motion have a poorer tangent alignment than our rotation invariant pathline cores, shown in Fig. 8.13e.

Rotating Mixer. Our second real-world data set features a rotating mixer, which was provided by Gábor Janiga, who is with the Institute of Fluid Dynamics and Thermodynamics, University of Magdeburg, Germany. This vector field is given in a fixed reference frame. Fig. 8.14 displays isosurfaces of the λ_2, Q, λ_{2_r} and Q_r criteria, for the time step $t = 0.004$. The left image shows Galilean invariant (red) and rotation invariant (blue) isosurfaces for $\lambda_2 = -60$ and $\lambda_{2_r} = -60$, respectively. With the Q criterion we examined very similar results for $Q = 60$ and $Q_r = 60$. The right image depicts Galilean invariant (red) and rotation invariant (blue) isosurfaces for $Q = 40$ and $Q_r = 40$. Similarly, $\lambda_2 = -40$ and $\lambda_{2_r} = -40$ give similar results. For such turbulent real-world flows a ground truth is unknown, and thus it is hard to quantify which vortex extraction method works better. We observed that Galilean invariant and rotation invariant surfaces differed only slightly in their strength (isovalue). We found the largest differences in the proximity of the mixer geometry. Apparently, those areas are more influenced by rotating motion. The cylindrical mixer geometry itself becomes apparent too, but is largely a boundary artifact.

8.5. Implementation and Evaluation

Next, we provide details on the coreline extraction and elaborate on the performance.

8.5.1. Coreline Extraction and Filtering

As shown earlier, our rotation invariant corelines are expressed by the parallel vectors operator. Peikert and Roth [PR99] explained several methods for PV extraction, which can be applied here. For all examples shown in the chapter, we used the parallel vectors implementation of the visualization toolkit Amira [SWH05b]. The computation involves the setup of the vector fields to test for parallelism and a prior test for presence of complex eigenvalues in the rotation invariant Jacobian $\mathbf{J}_r$, which is a necessary swirling condition. After extraction, a tangent alignment filter using Eq. (8.28) is applied to remove lines that are not tangential enough to the flow, followed by subsequent joining of close line segments to fill gaps and a final thresholding of lines by length to remove the remaining noise. All these filter steps are considered as standard procedure in local coreline extraction methods. There are a number of alternatives for closed PV line extraction including the curve-following predictor-corrector method of Peikert and Roth [PR99], the PVsolve algorithm of van Gelder and Pang [VGP09], the use of stable feature flow fields as in Weinkauf et al. [WTGP11], or the extractor of Pagot et al. [POS*11] for higher-order data. For cell-based extractions, Ju et al. [JCWD14] recently proposed a robust parity test to determine the number of PV points per cell face.

8.5.2. Performance

Our modifications to enable rotation invariance are only local and rapidly computed. They are much faster compared to the actual parallel vectors extraction in the line-based techniques and the computation of eigenvalues in the region-based techniques. Thus, the performance is very similar to standard Galilean invariant techniques. Space-time grid sizes and extraction times are shown in Table 8.1 for our rotation invariant pathline cores. The timings are measured on a system with an Intel Core i7-2600K CPU with 3.4 GHz and 24 GB RAM. The computation time linearly depends on the number of voxels with complex-conjugate eigenvalues in the Jacobian. For instance, in the FOUR CENTERS (SC) flow more voxels could be skipped than in the BEADS flows, due to the prior test for swirling behavior. Timings for the region-based techniques are shown in Table 8.2. As the Q_r criterion comes almost for free with the computation of λ_{2r} because $Q_r = -\frac{1}{2}(\lambda_{1r} + \lambda_{2r} + \lambda_{3r})$, we list combined timings, i.e., the time required to compute the eigenvalues and from those λ_{2r} and Q_r.

Data set	Grid size	Extraction time
BEADS (STD)	$64 \times 64 \times 64$	20.7 sec.
BEADS (DIV.-FREE)	$64 \times 64 \times 64$	20.9 sec.
FOUR CENTERS (SC)	$64 \times 64 \times 64$	4.5 sec.
FOUR CENTERS (PC)	$128 \times 128 \times 128$	55.2 sec.
SPIRALING CENTERS	$128 \times 128 \times 128$	55.6 sec.
ROTATING CENTER	$64 \times 64 \times 64$	20.2 sec.
CENTRIFUGAL PUMP	$512 \times 512 \times 80$	7.5 min.

Table 8.1.: Total computation time of rotation invariant pathline cores.

Data set	Grid size	Extraction time
ROTATING MIXER	$256 \times 256 \times 128$	36.5 sec.
FOUR CENTERS (PC)	$256 \times 256 \times 256$	2.3 min.

Table 8.2.: Total computation time of rotation invariant λ_{2r} and Q_r.

8.6. Discussion and Limitations

Simplicity: Our technique to compute rotation invariant vortices is extremely simple: just add a closed form matrix to the Jacobian and feed the vortex extractor with it. There is virtually no computation overhead or performance and accuracy drop in comparison to the original methods.

Relation to objective vortices: Clearly, an objective vortex is also rotation invariant: objectivity is a much stronger property than rotation invariance. However, existing objective vortex measures can be "too strong" to find expected vortices in a standard data set. An example can be found in [Sah09], where the $\mathbf{M}_z$ criterion of [Hal05] is applied to a cylinder flow: the objective $\mathbf{M}_z$ does not detect the typical von Kármán vortex street behind the cylinder. In particular, we are not aware of an objective measure that detects the correct vortex in the divergence-free BEADS data set.

Restriction to rotating flows: Our technique makes only sense for flows that are induced by a rotating movement around an axis. We think that this class of flows is large enough to make our approach relevant. Note that moving the rotation center/axis to infinity lets $\mathbf{J}_r$ converge to $\mathbf{J}$ and therefore converges to a Galilean invariant technique. Another issue is that for our technique one must know the location of the rotation point/axis. We do not consider this as a strong restriction because this information usually comes with the data set.

No Galilean invariance: We stress again that our technique is not Galilean invariant anymore. We argue that, for the data considered here, rotation invariance is more important than Galilean invariance. The better vortex structures shown in this chapter confirm this assumption.

8.7. Summary

In this chapter, we made the following contributions:

- We identified rotation invariance as desirable property for vortex measures in flows that are induced by rotating parts.
- We gave a formal definition of rotation invariance.
- We have shown an extremely simple way to transform a Galilean invariant measure to a rotation invariant one: we simply have to add a closed form matrix to the Jacobian and feed the vortex extractor with it.
- We proposed rotation invariant versions of cores of swirling particle motion, λ_2 and Q.
- We have applied them to a number of data sets, showing that our technique gives better vortices than the Galilean invariant measures. In particular, our technique is – to the best of our knowledge – the first local method that finds the exact core in the standard and divergence-free BEADS flow.
- For $n = 3$, we proposed a slight variation of the cores of swirling particle motion [WSTH07] in (8.17). In fact, we propose to apply Sujudi-Haimes on $\mathbf{u} - \mathbf{f}$. While this does not affect the results in the Galilean invariant case, it improves the results for the rotation invariant case.

This concludes the second part of the thesis. We now move on to the visualization of finite-sized objects carried with flows. In this context, vortices will play a roll later on.

Part III.

Inertial Particles in Flow Visualization

Chapter 9.
9 Overview on Inertial Particles

The third part of this thesis focuses on the visualization of finite-sized particle dynamics. Finite-sized particles are commonly referred to as *mass-dependent* or *inertial particles* and they are used to model the motion of finite-sized objects in fluids. In this chapter, we start with an introduction into inertial particles, which is compiled from our previously published work [GKKT13, GT14, GT15, GT16a, GT16b]. Our mass-dependent particle studies include geometry-based techniques (Chapter 10), the visualization of mass separation (Chapter 11), the study of inertial steady vector field topology in 2D (Chapter 12), a novel approach to the source inversion problem (Chapter 13) and the extraction of inertial vortices (Chapter 14).

9.1. Introduction to Inertial Particles

Particle-based flow visualization has a long tradition in experimental and computational fluid dynamics, cf. McLoughlin et al. [MLP*10]. So far, the flow visualization literature only used massless particles, since the tangent of their trajectory directly correlates to the velocity direction in the vector field, making massless particles a meaningful tool to visualize the vector field, see Fig. 9.1(•) for example. However, in many applications, the mass of transported particles, and resulting inertia, has critical impact to observable phenomena and are yet inaccessible by the previous methods. The consideration of particle mass has fundamental implications on the numerical integration process, as a number of forces acts on finite-size particles in viscous fluids. These additional forces cause trajectories to be no longer tangential to the flow, as demonstrated in Fig. 9.1(•). The CFD

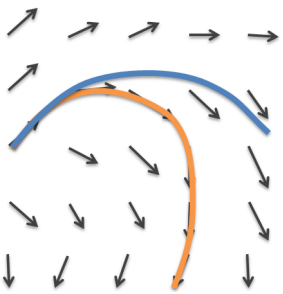


Figure 9.1.: Massless (•) and inertial (•) particle trajectories.

literature gives strong evidence that inertial particle dynamics vary considerably from the infinitesimal dynamics [OOG08, HS08]. The motion of solid inertial particles in gas or fluid flows is an important field of research, since they are involved in numerous scientific and industrial applications, such as sand saltation modeling [SL99], soiling of cars [RSBE01], visual obscuration in helicopter landing maneuvers [KGRK14], formation of rain [Bor11], jellyfish feeding [PD09], plant spores and pathogens carried by atmospheric flow [BH02], fuel injectors transporting droplets at supersonic speeds [MTP*02] and observations of charged particles in magnetic fields [BZ89]. Massless particles have been studied by the flow visualization community for decades, which created a rich pool of concepts that we can draw on to develop new visualizations that facilitate the assessment of inertial particle dynamics.

9.1.1. Inertial Equations of Motion

The trajectories of inertial particles are determined by their underlying equations of motion. Typically, they involve a number of forces that act on the particles, such as the force exerted by the flow itself, buoyancy, Stokes drag, the force exerted due to the mass of the fluid moving with the particle and the Basset-Boussinesq memory term, cf. Haller and Sapsis [HS08]. Today's most accepted form of these forces was described by Maxey and Riley [MR83] for small rigid spherical particles. The properties of the solutions to their equations of motion and the history of its corrections were recently documented by Farazmand and Haller [FH15]. Depending on the application, several assumptions can be made that simplify the equations of motion considerably. For the examples given throughout the thesis, we assume particles to be spherical and very small in size, which allows to assume Stokes flow due to the small particle Reynolds number. Also, the particle density is assumed to be far higher than the density of the surrounding fluid, which allows to neglect buoyancy. Further assuming dilute flow, the particle motion is dominated by drag forces, rather than particle-particle collision. Thus, we neglect collision handling and particle rotation, and assume one-way coupling, i.e., the particles do not affect the surrounding fluid. These simplifications are common and were used for instance in [SGL10, PSGC11a, KGRK14, CGP*10, BBC*09, BBC*11, SKK*15].

According to Crowe et al. [CST98], they lead to the following equations of motion:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{v}(t) \quad \text{with} \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{9.1}$$

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = \frac{\mathbf{u}(\mathbf{x}(t), t) - \mathbf{v}(t)}{r} + \mathbf{g} \quad \text{with} \quad \mathbf{v}(0) = \mathbf{v}_0 \tag{9.2}$$

where $\mathbf{u}(\mathbf{x}, t)$ is a time-dependent flow field, $\mathbf{v}$ is the current particle velocity, $\mathbf{g}$ is an (optional) gravity vector, $\mathbf{x}_0$ and $\mathbf{v}_0$ are the initial particle position and velocity, and

r is the particle response time[1]. The influence of the particle's mass on its trajectory is manifested in this response time r. For spherical particles in Stokes-type flows the response time r is characterized by the diameter d_p and density ρ_p of the particle, as well as the viscosity μ of the surrounding fluid (cf. Crowe et al. [CST98]):

$$r = \frac{d_p^2 \, \rho_p}{18 \, \mu} \tag{9.3}$$

Figuratively spoken, the response time is the time required for a particle released from rest in a gravity-free environment to acquire 63% of the velocity of the carrying fluid, cf. [CST98]. Note that these equations of motion hold for $d_p \ll \eta_k$, with η_k being the Kolmogorov length scale, i.e., the length scale of the smallest turbulent motion[2]. In the following, we speak at times of mass and particle diameter, rather than referring to the more general quantity response time, since mass and diameter are more accessible. All three directly relate to each other, since particle density and viscosity are kept constant.

The equations of motion (9.1) and (9.2) can be rendered autonomous by making particle position $\mathbf{x}$, particle velocity $\mathbf{v}$ and time t state variables:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\mathbf{x}\\ \mathbf{v}\\ t\end{pmatrix} = \begin{pmatrix}\mathbf{v}\\ \frac{\mathbf{u}(\mathbf{x},t)-\mathbf{v}}{r} + \mathbf{g}\\ 1\end{pmatrix} \quad \text{with} \quad \begin{pmatrix}\mathbf{x}\\ \mathbf{v}\\ t\end{pmatrix}(0) = \begin{pmatrix}\mathbf{x}_0\\ \mathbf{v}_0\\ t_0\end{pmatrix} \tag{9.5}$$

Viewing the problem in phase space, the state variables on the left hand side are the attributes stored per particle, i.e., position $\mathbf{x}$, velocity $\mathbf{v}$ and time t. Trajectories of inertial particles appear as phase lines, which are in fact tangent curves of the higher-dimensional vector field on the right hand side. Occasionally, we will refer to this higher-dimensional phase space as the *spatio-velocity(-time)* domain.

An alternative approach to the modeling of inertial particle motion is to add the material derivative $\mathrm{D}\mathbf{u}/\mathrm{D}t$ of the fluid to the particle equations in order to indirectly model gravity (and/or buoyancy) as external force. Thereby, fluid motion (in Eulerian frame) and particle motion (in Lagrangian frame) are made consistent, cf. Benzi et al. [BBC*09]. As we later synthetically alter the gravity to generate new test cases (without recomputing the fluid flow), we use equations of particle motion that model gravity explicitly [CST98]. This method is simpler in its modeling, but holds for the given

[1] In the fluid dynamics literature the particle response time is usually named τ_p, but to avoid confusions with the integration duration, we rename it to r.

[2] By means of the large scale flow features U and L, the Kolmogorov length scale can be estimated as [Pop00]:

$$\eta_k \approx \left(\frac{\nu^3 \, L}{U^3}\right)^{1/4}, \tag{9.4}$$

with U being the average speed of the flow, L being the length scale (domain size), and ν being the kinematic viscosity, here always assumed for air $\nu = 1.51 \cdot 10^{-5} \frac{m^2}{s}$.

assumptions. Another approach is the explicit consideration of buoyancy to model both aerosols and bubbles [HS08].

For further reading, we recommend a recent report on experimental and computational fluid dynamics of inertial particles in turbulent fluids by Balachandar and Eaton [BE10]. Of interest in the inertial fluid dynamics literature are problems such as direct numerical simulations of jets [PSGC11a], stirring of sand during helicopter maneuvers [SGL10], comparisons between experimentally determined and simulated trajectories [OOG08] (depicted by Euclidean distances over time), energy dissipation along trajectories [BBC*09], or the effects of inertial particles on the underlying flow (two-way coupling) [PSGC11b, TAB15]. The effect of inertia, resulting from finite-sized particles, has been studied by Cartwright et al. [CFK*10]. The motion dynamics of dust particles with mass in vortical flows has been analyzed by Angilella [Ang10].

9.1.2. Relation to Massless Particles

Inertial particles are not used to explore the properties of a vector field, but to assess the motion of inertial objects therein. Massless particles, on the other hand, are used for the visualization of the underlying vector field, as their trajectory is tangential to the flow. When approaching zero response time $r \to 0$, the inertial equations of motion approach the massless case. This is shown by rearranging Eq. (9.2) for $\mathbf{v}$ and substituting in Eq. 9.1, which yields in the limit tangent curves of $\mathbf{u}$:

$$\lim_{r\to 0} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{u}(\mathbf{x}(t),t) \underbrace{-r\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} + r\,\mathbf{g}}_{0}. \tag{9.6}$$

Further, a common measure for the inertia of particles is the Stokes number S_k:

$$S_k = r/r_\eta, \tag{9.7}$$

with r_η being the Kolmogorov time scale [LMC92]. Particles with very low Stokes numbers ($S_k \ll 1$) are known to follow pathlines of the flow closely. As shown in Eq. (9.7), the Stokes number S_k is proportional to the particle response time r, thus if the response time approaches zero due to zero mass a massless particle follows a pathline.

However, the smaller the response time becomes, the more numerical problems occur, since Eq. (9.5) contains a division by the response time. During integration with step size h the error in the position evolves at rate $O(h)$, while the error in the velocity evolves at $O(h/r)$, cf. [HS08]. Thus, for very small particles ($r \ll 1$), the error in the velocity becomes very high. For this reason, an adaptive higher-order Runge-Kutta integration is recommended that controls the error in the spatio-velocity domain (not

only in space). This so-called singular perturbation problem can be avoided by scaling the time step, cf. Haller and Sapsis [HS08]: Assuming, a very small response time $r = \epsilon \ll 1$. The usual approach to integrate Eq. (9.5) is to use:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\mathbf{x}\\ \mathbf{v}\\ t\end{pmatrix} = \begin{pmatrix}\epsilon\,\mathbf{v}\\ \mathbf{u}(\mathbf{x},t) - \mathbf{v} + \epsilon\,\mathbf{g}\\ \epsilon\end{pmatrix}. \tag{9.8}$$

Thereby, the integration of an inertial path can be very expensive due to the potentially small step size. In the practical use, the difference to massless trajectories becomes neglectable for very small particles. Thus, at some point (depending on particle diameter, particle density and viscosity), massless trajectories can be used instead.

9.2. Challenges with Inertial Particles

In the following, we explain several challenges that arise in the context of inertial particle visualization. In the subsequent chapters, we elaborate on these problems and present our solutions.

9.2.1. Inertial Integral Geometry

The equations of motion that define the behavior of the particles need to be modeled according to the assumptions that can be made in the respective application. Consequentially, different particle models were established as standards in certain CFD areas, making each application unique. In order to generalize this, we introduce a unified visualization framework for the geometry-based analysis of mass-dependent particles. The framework provides the following contributions:

- A unified description of mass-dependent particle trajectories independent of the application case, i.e., mass-dependent pathlines.
- In order to reveal temporal and spatial coherent structures, we derive mass-dependent streaklines and timelines.
- To analyze the influence of the mass on the trajectories, a new class of integral lines is introduced, called *masslines*.

The applications of mass-dependent particle visualizations are numerous and of great importance, e.g., in understanding particle separation due to density variations (for instance by hydrocyclones), or the analysis of the aerodynamics around a helicopter close to the ground, which will be the primary example in Chapter 10.

9.2.2. Mass Separation

Among the many unresolved problems that center around inertial particles is the search for an effective visualization of the differences between inertial particle trajectories, for instance to locate regions in which massless trajectories are similar enough to fall back on standard techniques. In the context of this work, the term *comparative visualization* comprises both the qualitative and quantitative comparison with reference particles, as well as the detection of critical masses that cause a separation of differently sized inertial particles. The latter plays a major role in rotorcraft engineering [SGL10], as sand particles dragged into a rotor disk cause mechanical wear and entrain further particles when being convected to the sediment bed, which severely hinders the pilot's view. For quantitative comparisons, common practice is to plot the Euclidean distance between differently sized inertial particles over time [Bor11], which however, suffers from occlusion and is unaware of the behavior of particle sizes in-between. Up until now, it was not possible to effectively explore where mass separation occurs, how distinct it is and which masses are separating. The difficulty of quantifying trajectory differences and the integration-based nature of separation detection make these problems challenging. We devote Chapter 11 to this problem.

9.2.3. Inertial Steady 2D Vector Field Topology

The motion of inertial particles is governed by an ODE, and their asymptotic behavior depends on the initial conditions, i.e., initial position (2D) and initial velocity (2D). Visualizing the possible outcome of any initial condition (4D) and understanding separatrices that divide the 4D spatio-velocity domain into compartments of homogeneous asymptotic inertial flow behavior are challenging tasks. Further, many concepts of vector field topology rely on the ability to integrate particles backwards. Extending these concepts to the inertial case is in part challenging, since inertial backward integration toward critical points exhibits a strong repelling behavior [HS08]. We explain this phenomenon in Chapter 12 by the use of differential vector field topology. Further, we devise a glyph that visualizes the asymptotic behavior of inertial particles that start with variable initial velocity.

9.2.4. Source Inversion in Inertial Flows

Even though inertial particle motion is theoretically well-defined by an ODE, inertial backward integration is problematic in practice. Given an observation from which a backward integration is started, the recovered initial position and velocity of the inertial particle heavily depend on the observed velocity that we started with. In

fact, a slight change leads to a completely different initial position, and most often the recovered initial velocity is extremely high and therefore implausible. Exploring different observed velocities until a plausible source is recovered is therefore a very tedious and impractical task. In Chapter 13, we address this problem. Instead of prescribing the observed position and the observed velocity, the idea of our approach is to prescribe the observed position and an *initial velocity*: we assume that particles can start from any spatial location but with a fixed initial velocity. Then, the set of possible initial positions that lead to the observation form a curve, which we extract as tangent curve of a derived vector field.

9.2.5. Inertial Vortex Cores

Another powerful concept from tradition flow visualization that we can draw on is the extraction of vortices. Vortices are among the most interesting structures in fluid flows, and frequently they are visually analyzed. In Section 6.4.2, we already discussion local and integration-based vortex extraction methods. Both classes extract their own subset of vortices. The local methods detect steady vortices or vortices that perform certain types of movements (e.g., equal-speed translations along straight lines are detected with Galilean invariant methods). The integration-based methods find only attracting corelines, though for every kind of vortex motion. From our experiments with inertial integral geometry in Chapter 10, we can expect that the location of vortex cores also depends on the mass of the considered particles. In Chapter 14, we extend well-established vortex concepts to the inertial case.

9.3. Related Work

Chapter 12 will deal with steady 2D vector field topology of inertial particles. Thus, we begin with a brief introduction into steady 2D vector field topology for the traditional massless case. Afterwards, we elaborate on attracting manifolds of inertial particles and inertial backward integration. We close this chapter with an outline of recent visualization work in the field of inertial particles.

9.3.1. Steady 2D Vector Field Topology

Topology-based visualization aims for compact descriptions of vector fields. This includes the extraction of features and the segmentation of the domain into regions of coherent asymptotic behavior. Topological methods received much attention and became a very active research area. Among others, they spurred work to smooth [WJE01],

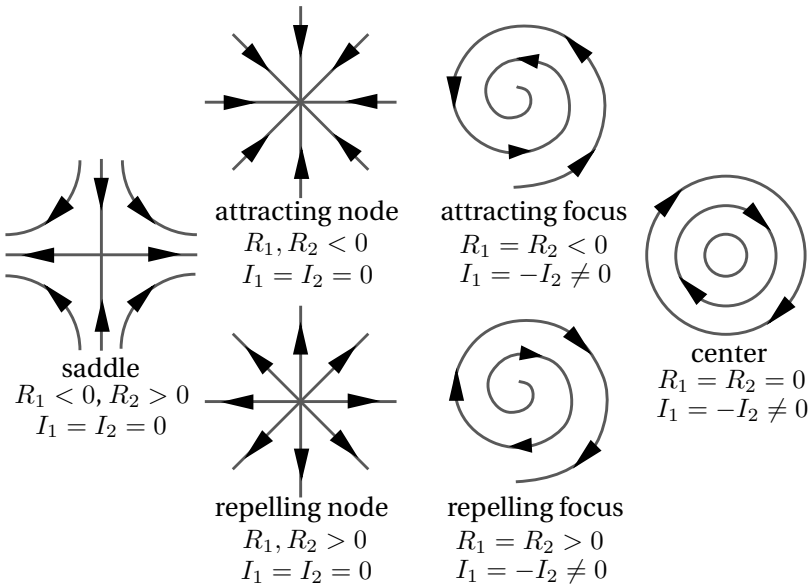


Figure 9.2.: Types of critical points in 2D steady vector fields (from [HH89]). R_1 and R_2 denote the real parts of the eigenvalues and I_1 and I_2 the imaginary parts.

compress [LRR00, TRS03] and model [The02, WTHS04b] or edit [CML*07] vector fields. For brevity, we will focus only on topology aspects relevant for the following chapters. We refer to the dissertation thesis of Tino Weinkauf [Wei08] for a comprehensive introduction into topology-based methods.

The most fundamental aspect of vector field topology is the extraction of first-order critical points. In the related dynamical systems literature (where phase spaces can be interpreted as vector fields), these points are also known as fixed points, stationary points or singularities. The term *critical point* was coined for vector fields by Helman and Hesselink in their seminal work [HH89], which introduced vector field topology to the visualization community.

Critical Points. Formally, a critical point $\mathbf{x}_0$ is a location at which the velocity vanishes:

$$\mathbf{u}(\mathbf{x}_0) = \mathbf{0}.$$

For their classification, we additionally require that the vector field is non-zero in the vicinity of the point $\mathbf{u}(\mathbf{x}_0 + \epsilon) \neq \mathbf{0}$, i.e., it is an *isolated* critical point. Further, for a first-order classification, we require that $\mathbf{u}$ is differentiable and that the Jacobian $\mathbf{J} = \nabla\mathbf{u}(\mathbf{x}_0)$ is non-singular. If this is fulfilled, the eigenvalues and eigenvectors of the Jacobian characterize the behavior of the flow around the critical point.

Fig. 9.2 gives an overview of the types of isolated first-order critical points that may arise in 2D steady vector fields. Each eigenvalue characterizes the behavior of the flow

in the direction of the corresponding eigenvector. If both eigenvalues have positive real parts, the critical point is repelling. If both are negative, it is attracting. If the real parts of the two eigenvalues have opposite signs, a saddle is present. If the eigenvalues are complex, they indicate a swirling motion. In this case, the critical point is called focus. A focus with zero real parts is called center.

Asymptotic Flow Behavior. Steady vector fields are not temporally bound and thus, it is possible to observe where a particle would go to after an infinite integration duration. This allows to segment the domain into regions that have the same asymptotic behavior. All locations in the domain $D \subseteq \mathbb{R}^2$ that reach the same isolated first-order critical point $\mathbf{p} \in D$ after infinite *forward* integration form a so-called ω-basin:

$$\omega(\mathbf{p}) = \{\mathbf{x}_q \in D : \phi^\tau(\mathbf{x}_q) \to \mathbf{p} \text{ as } \tau \to \infty\} \tag{9.9}$$

Likewise all locations in the domain $D \subseteq \mathbb{R}^2$ that reach the same isolated first-order critical point $\mathbf{p} \in D$ after infinite *backward* integration form a so-called α-basin:

$$\alpha(\mathbf{p}) = \{\mathbf{x}_q \in D : \phi^\tau(\mathbf{x}_q) \to \mathbf{p} \text{ as } \tau \to -\infty\} \tag{9.10}$$

The domain can be partitioned into disjoined ω-basins or into disjoined α-basins.

The boundaries of ω-basis and α-basins are called *separatrices*. In 2D flows, a separatrix is a streamline that connects a saddle with a source, sink or a saddle. In bounded domains additional separatrices arise that connect critical points with the domain boundaries at so-called *boundary switch points*, i.e., places on the boundary at which the inflow and outflow behavior switches.

Further Reading. Helman and Hesselink extended the topological methods to 3D [HH91], including the classification of first-order critical points, separatrices starting at saddles and attachment/detachment points at no-slip boundaries. Definitions of higher-order critical points [SKMR98], boundary switch points [dLvL99], closed separatrices [WS01], saddle connectors [TWHS03], boundary switch connectors [WTHS04a] and topology in two-parameter dependent vector fields [WTHS06] followed shortly later. Further examples of vector field topology in 3D can be found in [GLL91, LDG98, MBS*04]. For further reading we refer to the reports of Laramee et al. [LHZP07] and Pobitzer et al. [PPF*11], and the PhD thesis of Tino Weinkauf [Wei08].

9.3.2. Source Inversion, Attracting Manifolds and Backward Integration

In traditional massless vector field topology, many visualization concepts rely on the ability to integrate particles backward in time. In the context of dispersed fluids,

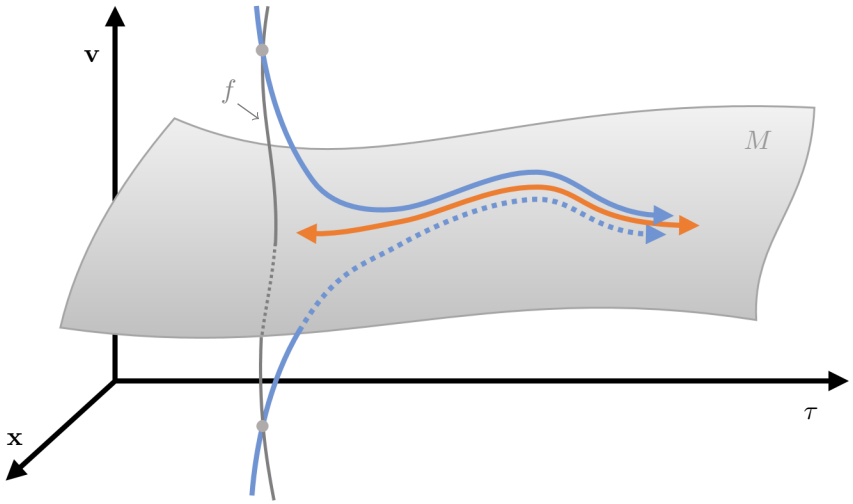


Figure 9.3.: Inertial particle trajectories (blue) are attracted by a manifold M, and repel away from it during backward integration. The union of initial conditions $(\mathbf{x}_0,\ \mathbf{v}_0)$ that converge to the same trajectory is called a fiber f. Haller and Sapsis [HS08] derived an ODE to move *on* the manifold M (orange) in order to integrate backward without repelling behavior. (Sketch is based on [HS08].)

backward integration is related to the *source inversion problem*, which is of interest in detecting localized sources of pollution in air or water. The problem can be formulated as follows: Given an observed airborne or waterborne dispersed pollutant, compute the source of the pollution. Typically, the transport involves a diffusion process [ABG*03, BRR05, BHDH05, CKC08], though even without it, source inversion is nontrivial. For this simplified case, Haller and Sapsis [HS08] provide an approximate solution. Their method is based on the observation that during forward integration, inertial particles are attracted toward manifolds in the spatio-velocity domain and cluster on them, cf. [MBZ06, HS08]. In fact, multiple initial conditions of $(\mathbf{x}_0,\ \mathbf{v}_0)$ might attract to the same spatio-velocity structure, as illustrated in Fig. 9.3. The union of initial conditions that converges to the same structure is called a fiber or a limit set in dynamical systems theory. Consequentially, a backward integration *repels away* from such an attracting manifold. In Chapter 12, we give another intuitive explanation why this repelling behavior occurs by observation and classification of the inertial critical points in the spatio-velocity domain. The extraction of topology of inertial flow becomes much more complicated than for traditional massless flow, since backward integration toward critical points is practically impossible due to the repelling behavior.

Haller and Sapsis [HS08] derived an ODE that allows to move robustly *on* such an attracting manifold in both forward and backward direction. They called this ODE the *inertial equation*. Applied to our model, a first-order approximation essentially boils

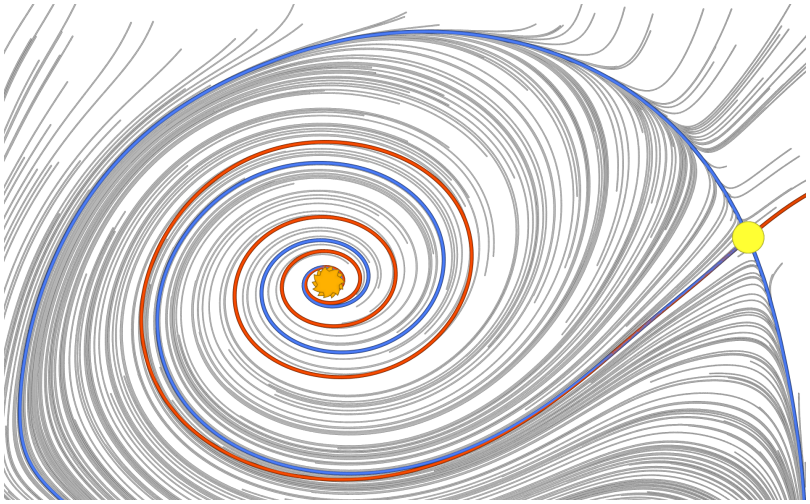


Figure 9.4.: The orange critical point acts as a source in Haller's inertial equation, and the red separatrix is found by backward integration from the saddle (yellow critical point on the right). It can be seen that inertial forward trajectories (gray) of particles released from rest for $d_p = 70\,\mu m$ cross the red line. Blue lines are actual trajectories of inertial particles that connect the critical points.

down to an integration in the vector field $\mathbf{u} + r\,\mathbf{g}$. While their findings have a strong theoretical merit, there are three problems with this approach when applied to source inversion. First, the inertial equation lives in the spatial domain and thus it cannot recover the initial velocity with which the pollutant entered the flow. Second, for larger particles, instabilities occur [SH09] that drive inertial particles away from the attracting manifold on which the inertial equation is valid. For their particle model, Sapsis and Haller [SH09] derived a threshold that characterizes when this happens, which they applied in [SPH11]. Third, the inertial equation operates on the limit assumption that particles have reached the manifold. The method does not consider the accumulating inertia effects of actual inertial trajectories. Fig. 9.4 shows for example that separatrices recovered from backward integration in $\mathbf{u}+r\,\mathbf{g}$ do not match the actual separatrices that are found via bisection and forward integration of the actual inertial flow of Eq. (9.5).

In Chapter 13, we present an approach to the source inversion problem by considering actual inertial trajectories and we require forward integration only. For this, we describe possible sources as tangent curves of a derived vector field, yielding so-called influence curves. Modeling topological features or integral curves as tangent curves of possibly higher dimensional vector fields is a common approach in flow visualization. Streaklines [WT10a] and timelines [WHT12] for instance have been modeled as tangent curves, same as the paths of critical points [TS03] or vortex corelines [WTGP11].

A problem related to source inversion is the computation of visitation maps. A visitation

map determines for a given initial position the visitation frequency of all locations that may be reached in an uncertain flow. Bürger et al. [BFMW12] computed instant visitation maps by a Monte Carlo simulation that traces massive particle sets. Monte Carlo particle tracing could be applied to the source inversion problem as well by releasing particles from all locations in the domain and testing, which of them reach the observation point. This, however, is very costly and does not provide a parametric description of the locations that reach the observation point.

9.3.3. Inertial Particles in Visualization

The observation of inertial particles is a relatively young field in the visualization community, which opened up new challenges. The third part of this thesis, centers around the extension of concepts that were originally devised for traditional massless flows in order to reach out to new application areas. In an early work, Roettger et al. [RSBE01] conducted sand particle simulations to determine the soiling of cars in a wind tunnel. They visualized the sand concentration on the cars via heat maps. Aside from that inertial particles have eluded the visualization community.

In the CFD literature, Lagrangian coherent structures (LCS) have also been extended to inertial particles. Haller [Hal01] has demonstrated that LCS can be extracted as ridges of the finite-time Lyapunov exponent (FTLE) field (cf. FTLE background in Section 6.1). Thus, repelling inertial Lagrangian coherent structures (ILCS or pLCS) have been defined in [SH09] and [PD09] as ridges of a forward FTLE computation based on the trajectories of inertial particles.

$$\mathrm{IFTLE}(\mathbf{x},\, t,\, \tau,\, r) = \frac{1}{|\tau|} \ln \sqrt{\lambda_{\max}\left(\frac{\mathrm{d}\phi_t^\tau(\mathbf{x}, r)}{\mathrm{d}\mathbf{x}}^{\mathrm{T}} \frac{\mathrm{d}\phi_t^\tau(\mathbf{x}, r)}{\mathrm{d}\mathbf{x}} \right)}$$

This scalar field measures the separation of nearby placed, equally-sized particles. Standard FTLE is the special case for massless particles, i.e., $r = 0$.

In order to be able to calculate backward inertial FTLE (attracting ILCS), Sapsis et al. [SPH11] calculated ILCS in the context of jellyfish feeding, based on trajectories of the inertial equation [HS08] (cf. Section 9.3.2). Since the inertial equation is valid on the attracting manifold only, it only holds well for very small particles. As it turns out, the dynamics of "larger"[3] inertial particles become instable in areas of high strain [SH08], causing particles to spin away from attracting ILCS due to their inertia rather than being attracted. We are not aware of a method that can reliably calculate attracting inertial LCS of large inertial particles.

[3] Here, "larger" refers to particles with sufficient inertia to overcome the forces of ILCS. In terms of scales, it is a matter of the particle Stokes number. A criterion for the identification of such instabilities was derived in [SH08] and was picked up again in [SPH11].

Raben et al. [RRV14] computed FTLE for both tracer particles and inertial particles from experimental trajectories. This way, they obtain trajectories without making assumptions in the equations of motion, which would usually be required for numerical computation of trajectories. On a related note, a scheme for FTLE computations from trajectories was provided by Kuhn et al. [KER*14]. More recently, Sudharsan et al. [SBR16] investigated the relationship between preferential particle concentrations and FTLE for both aerosols and bubbles. Inertial FTLE was not only calculated based on spatial separation, but also on the separation in the spatio-velocity domain, i.e., in the phase space in which inertial particles are tangent curves [GPPMn15].

10 Chapter 10. Mass-Dependent Integral Curves in Unsteady Vector Fields

In this chapter, we introduce a generalized concept to describe inertial particle trajectories and from that we define mass-dependent extensions of traditional integral curves, i.e., mass-dependent pathlines, mass-dependent streaklines and mass-dependent timelines, as well as a new class of integral curves that depicts the separation of masses: the masslines. The chapter is based on the publication:

T. Günther, A. Kuhn, B. Kutz and H. Theisel
Mass-Dependent Integral Curves in Unsteady Vector Fields
Computer Graphics Forum (Proc. EuroVis) 32, 3 (2013), 211–220.

10.1. Mass-dependent Flow Map

Given is a 3D time-dependent vector field $\mathbf{u}(\mathbf{x},t)$. For simplifying explanations we assume its domain – both spatial and temporal – to be infinite, which ensures that the integration of $\mathbf{u}$ never leaves the domain. (In real data sets, a boundary treatment is necessary.) In order to decouple the derivation of integral curves from the particle model, we generalize the particle model by extending the concept of *flow map* ϕ (cf. [Sha05]) to mass-dependent particles. This yields a new mass-dependent flow map $\phi_t^\tau(\mathbf{x},r,\mathbf{v}_0)$ that describes where a particle with response time r, seeded at location $(\mathbf{x},t)$, and with an initial velocity $\mathbf{v}_0$ moves to during integration of $\mathbf{u}$ over a time interval τ. The mass-dependent flow map can be understood as a black-box that comprises the application-dependent equations of motion. Thus, all concepts that are derived from the mass-dependent flow map (such as integral curves) are described independent of the underlying equations of motion. Note that the continuity of the

flow map's derivatives depends on the underlying particle model. If random events such as collisions are neglected, they are continuous everywhere in the domain.

In the following, we consider all particles to be released from rest, i.e., $\mathbf{v}_0 = \mathbf{0}$. For brevity, we therefore shortly denote the mass-dependent flow map for particles released from rest as $\phi_t^\tau(\mathbf{x}, r)$. Note that mass-dependent particles develop inertia, which influences the trajectory. In contrast to the massless flow map, the following property therefore does not generally hold in the shortened notation, as the velocity of the particle is assumed to be zero at time $t + \tau_1$:

$$\phi_{t+\tau_1}^{\tau_2}(\phi_t^{\tau_1}(\mathbf{x}, r), r) \neq \phi_t^{\tau_1+\tau_2}(\mathbf{x}, r).$$

The property, however, is fulfilled by considering the initial velocity when continuing the integration, as in the unshortened description:

$$\phi_{t+\tau_1}^{\tau_2}(\phi_t^{\tau_1}(\mathbf{x}, r, \mathbf{v}_0), r, \frac{\mathrm{d}\phi_t^{\tau_1}}{\mathrm{d}\tau}(\mathbf{x}, r, \mathbf{v}_0)) = \phi_t^{\tau_1+\tau_2}(\mathbf{x}, r, \mathbf{v}_0).$$

10.2. Mass-dependent Integral Curves

In this section, we introduce the four natural classes of integral curves for mass-dependent particles. They arise when varying one parameter of the mass-dependent flow map in turn.

Mass-dependent Pathline. Using the mass-dependent flow map $\phi_t^\tau(\mathbf{x}, r)$, the trajectory of a mass-dependent particle – for brevity called *mass-dependent pathline* – can be described as a parametric curve in the form:

$$\mathbf{p}(\tau) = \phi_t^\tau(\mathbf{x}, r),$$

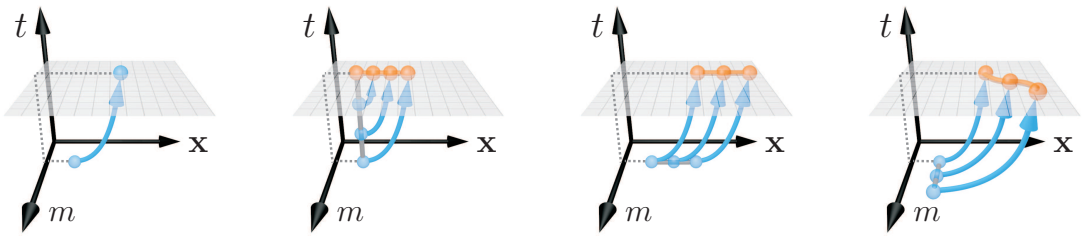
when seeding at position $\mathbf{x}$ at time t with response time r. Such pathlines can be used to visualize where particles of a certain mass are transported to over time. Fig. 10.1a illustrates a mass-dependent pathline in mass-space-time.

Mass-dependent Streakline. The temporal coherence between particles of equal mass can be visualized by *mass-dependent streaklines* $\mathbf{s}(\tau)$, see Fig. 10.1b, which can be described using:

$$\mathbf{s}(\tau) = \phi_\tau^{t-\tau}(\mathbf{x}, r).$$

Mass-dependent Timeline. *Mass-dependent timelines* $\mathbf{t}(u)$ are used to reveal spatial coherent structures, as illustrated in Fig. 10.1c, and are defined as:

$$\mathbf{t}(u) = \phi_t^\tau(\mathbf{c}(u), r).$$



(a) Pathline (•) (b) Streakline (•) (c) Timeline (•) (d) Massline (•)

Figure 10.1.: Illustrations of the integral lines in mass-space-time.

Massline. A new class of integral curves arises when varying the response time, while keeping the start location $\mathbf{x}$, start time t and integration duration τ constant. We name these curves *masslines* and define them as:

$$\mathbf{m}(r) = \phi_t^\tau(\mathbf{x}, r)$$

These lines connect all locations that can be reached at a certain time $t+\tau$, starting from one place $\mathbf{x}$ by using different masses, i.e., a massline connects the end points of mass-dependent pathlines, see Fig. 10.1d. Since a pathline and a streakline, seeded at the same position and time, share the end point, masslines are also the connections of the end points of mass-dependent streaklines. (We examine an example in Section 10.4.) On a further note, the length of masslines correlates with the separation introduced by particles released with different mass, which can be used as a descriptor for features. We elaborate on this later in Chapter 11. Similar to timelines, masslines have the limitation to not directly visualize the correspondence to the seeding structure.

10.3. Implementation

In the following examples, the integral curves are integrated on the GPU using geometry shaders. The capability of the geometry shader to dynamically insert vertex data is of use for the adaptive refinement of the integral curves. Streaklines, timelines and masslines are refined dependent on the length of the edge that connects neighboring particles. We chose the refinement scheme that is most commonly used on GPUs, i.e., to emit a new particle in the center of the edge that needs refinement. Unsteady vector field data often exceeds the video memory, which is why we asynchronously stream the data of the next time steps to a ring buffer on the GPU.

Integral lines were shaded with ISL shading by Zöckler et al. [ZSH96]. Depth cues were added by depth-dependent halos of Everts et al. [EBRI09].

10.4. Results

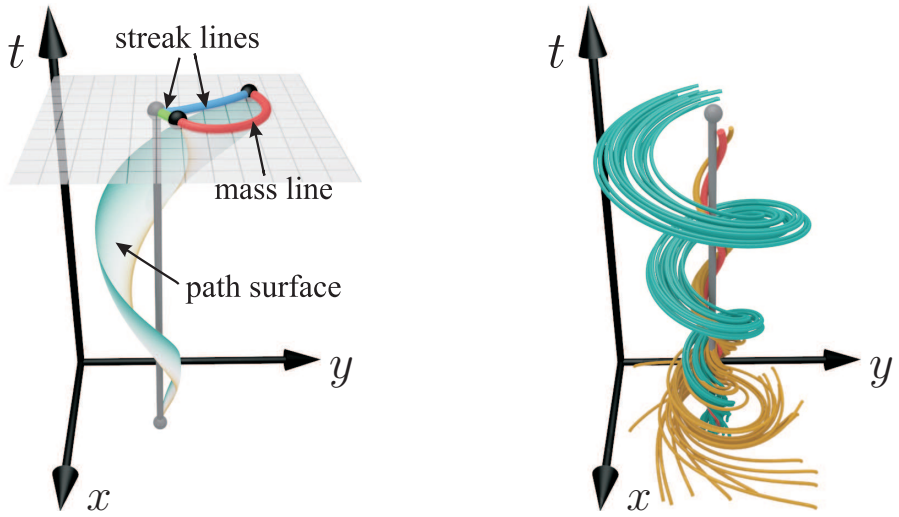
In this section, we apply the integral curves to three data sets, one analytic and two real-world. Furthermore, we present measurements of the performance of our interactive GPU implementation, captured in the helicopter data set.

In all experiments, the (dynamic) viscosity is set to $\mu = 1.81 \times 10^{-5}\ kg/(m\ s)$ (air), and the gravity is set to $\mathrm{g} = (0, -9.8065, 0)^{\mathrm{T}}\ m/s^2$. For the particles, we used the density of dry sand $\rho_p = 1600\ kg/m^3$. Since the particle diameter d_p varies between $d_p = 10\,\mu m$ and $d_p = 500\,\mu m$ the particle response time r is in the range $r \in [0.00049,\ 1.228]$.

Beads Flow. First, we investigate the BEADS flow, which was previously described in Section 8.4. Pathlines, streaklines and masslines are shown for this data set in space-time in Fig. 10.2. As illustrated, masslines connect the end points of both pathlines and streaklines, seeded with varying mass, see Fig. 10.2a. Releasing pathlines with continuously increasing mass assembles a surface. Furthermore, we show an example for the mass-dependence of topological structures, such as vortex cores, in Fig. 10.2b. While heavy particles diverge due to inertia, light particles can converge to a mass-dependent core line. The smaller the mass, the closer the massless pathline core, reported by Weinkauf and Theisel [WT10a], is approached. It can be seen that the mass-dependent particle model is coherent with traditional massless particles when approaching zero mass (for zero gravity).

Helicopter in Ground Proximity. When approaching the ground or hovering near it, the aerodynamics around a helicopter is significantly altered compared to the free flight. Especially the wake of the rotor is influenced, hindered and deflected by the ground. This flow regime is known to be highly unsteady and three-dimensional. Therefore, not all the mechanisms are fully understood [SGL10]. When this flight state occurs over a terrain with loose ground, such as sand or snow, the wake will lift up sediment particles, which will be entrained into the surrounding flow of the helicopter and lead to a dangerously obscured view capability of the pilot. This phenomenon is known as brownout or whiteout and may be hazardous to the flight crew as well as the ground personnel [MKW08]. Therefore, this flight state was subject of many studies in the past [Bet37, WT10b, TLKB11].

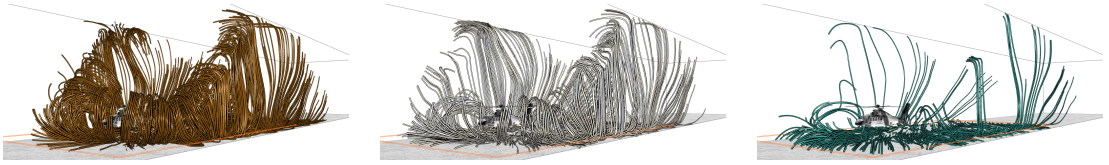
In the following, we illustrate brownout conditions around a helicopter in slow forward flight close to the ground. For this, we use the HELICOPTER HOVER sequence. The origin and simulation setting of this flow was described earlier in Section 3.3.2. We employ the equations of motion of Crowe et al. [CST98], which we stated earlier in Eq. (9.5). With this, we assume one-way coupling, i.e., the particles do not act back



(a) The massline (•) connects the end points of both pathlines ($d_p = 15\,\mu m$ (•) to $500\,\mu m$ (•)) and streaklines ($d_p = 15\,\mu m$ (•), $500\,\mu m$ (•)).

(b) Converging and diverging mass-dependent pathlines ($d_p = 300\,\mu m$ (•), $500\,\mu m$ (•)) and the massless pathline core (•), reported in [WT10a].

Figure 10.2.: Integral lines in the Beads flow, depicted in space-time. Gravity is set to zero.



(a) Mass-dependent pathlines, $d_p = 15\,\mu m$

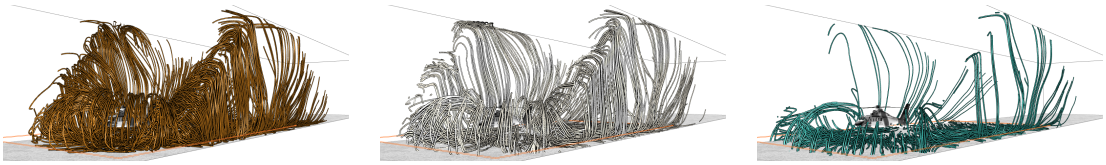
(b) Mass-dependent pathlines, $d_p = 50\,\mu m$

(c) Mass-dependent pathlines, $d_p = 90\,\mu m$

Figure 10.3.: Depictions of pathlines with different mass in the helicopter data set. Larger particles do not lift as high, due to gravity.

on the fluid. This assumption is in accordance with the current practice in CFD brownout engineering [TKB12]. The effects of inertial particles on the underlying flow (two-way coupling) [PSGC11b] are still actively under research. We stress that our mass-dependent flow map is a generalized description that allows to substitute the underlying particle model. The ability to use two-way coupled simulations in the future or to adapt the particle model to the application at hand are anticipated by design and our model is therefore generalized. The assumption of one-way coupling has a big advantage. While the underlying fluid flow simulation might take weeks to compute, inertial particles can be simulated and visualized interactively. This can be of useful to preview and steer parameters for long-term CFD simulations.

In Fig. 10.3 the trajectories of mass-dependent particles are shown for different masses.

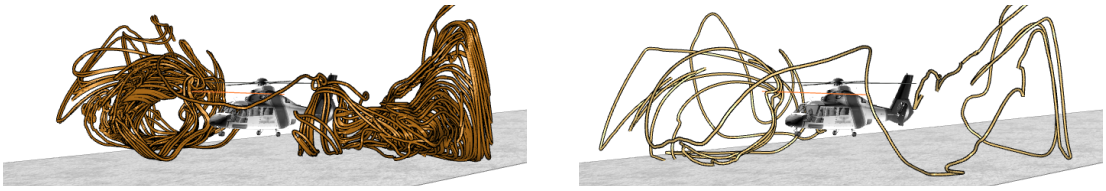


(a) Mass-dependent streaklines, $d_p = 15\,\mu m$

(b) Mass-dependent streaklines, $d_p = 50\,\mu m$

(c) Mass-dependent streaklines, $d_p = 90\,\mu m$

Figure 10.4.: Depictions of streaklines with different mass in the helicopter data set. A higher uplift threshold and gravity keep larger particles closer to the ground.



(a) Mass-dependent timeline, $d_p = 15\,\mu m$

(b) Mass-dependent timeline, $d_p = 30\,\mu m$

Figure 10.5.: Timelines seeded below the rotor disk reveal that lighter particles are more subject to turbulence.

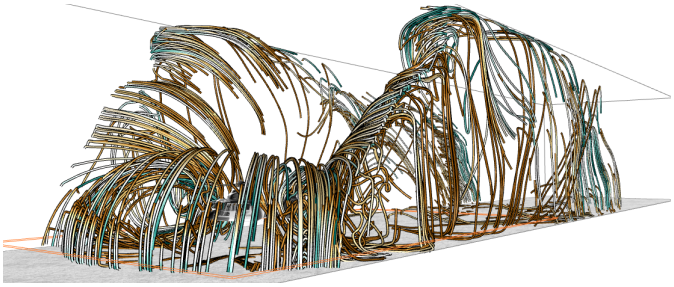


Figure 10.6.: Depiction of masslines with a varying diameter $d_p = 10\,\mu m..100\,\mu m$, color-coded from brown (●) to teal (●).

As a frame of reference we placed an Eurocopter AS365 Dauphin helicopter in the brownout scenes. (The CFD simulation only contained the rotor disk, not this particular model.) In Fig. 10.4 streaklines of different mass are shown. Note that the vortex in front of the helicopter uplifts smaller particles directly (Fig. 10.4a), whereas larger particles partially fall back to the ground and bounce off (Fig. 10.4c). Especially, to the left and right of the helicopter the vortex that traps the sand particles is highly unsteady and requires much refinement of the lines. Fig. 10.5 uses timelines to makes the influence of the mass apparent. Smaller particles are more subject to the unsteadiness of the vector field, thereby exhibiting more turbulent behavior.

Fig. 10.6 depicts masslines in the helicopter data set. Here, the response time is varied,

which is either achieved by a change of the diameter, the particle density or the viscosity. Either way, the same line is acquired, but with a different parameterization and other meaning. Modifying the diameter allows to observe the separation of differently sized particles, while a variation of the density allows to view differences between dry and moist sand. In all examples, we kept density and viscosity constant.

Our domain experts confirmed three use cases, in which the masslines in Fig. 10.6 are of particular help. First, when rotorcrafts approach the ground, ground personnel and also civilians (e.g., on rescue missions) are possibly exposed to forceful air blows containing sand particles of different sizes. These particles can cause injuries (eyes and lungs) and their ejection direction and strength varies around the rotorcraft. Masslines visualize for multiple sizes simultaneously in which directions particles are tossed and also what the critical sizes are that lead to uplift, giving leads which areas to avoid when rotorcrafts approach. (In the case of slow forward flight, this is a few meters in the frontal hemisphere.)

The second application concerns the rotor blades. Masslines depict which particles of a certain mass spectrum enter the rotor disk, causing blade erosion and mechanical wear. Fig. 10.6 shows that nearly the whole spectrum of masses enters the rotor from the frontal left and frontal right direction, as viewed from the pilot.

The third use case is called modified saltation bombardment and is concerned with particles that return to the sediment bed after a short uplift due to a nearby vortex. These impacting particles entrain smaller particles, thereby contributing to brownout. The masslines in Fig. 10.6 show the masses that take part in this process (which is dependent on the distance to the vortex), e.g., directly in front of the helicopter lower masses (i.e., smaller particles) fall back to the sediment bed than on the frontal left and right.

Hydrocyclone. Another example for the information conveyed by masslines is shown in Fig. 10.7 in the steady HYDROCYCLONE data set from Section 3.3.2. Here, the particle model includes particle-boundary interaction, i.e., the reflection of particles at the wall geometry. The employed particle model cannot yet describe interactions between different media, but gives reasonable results when observing the paths of only few particles (neglecting inter-particle forces).

In the hydrocyclone, inertial particles move on a helix trajectory downward. The helix-like movement depends on the mass, thus the exit location and direction are mass-dependent, causing a different mass-dependent exit distribution. The mass dependence of the particle trajectories is expressed by a swirling massline, which appears as a particle wave that swirls inside the bottleneck until it eventually exists

it. The fact that it expands to a visible line leads to the conclusion that particles with different masses pass the bottleneck at a different time. This is also visible when comparing the pathline images, in which lighter particles exit the bottleneck later, since they circled the bottleneck longer.

10.4.1. Performance

For all measurements, we used a machine equipped with an Intel Core I7-2600K CPU with 3.4 GHz and an Nvidia Quadro 6000 GPU with 6GB VRAM. All images were rendered at a resolution of 1280×916 pixels with $16\times$ coverage-sampling anti-aliasing (CSAA). For the individual classes of integral curves the performance is measured in Table 10.1. Due to the absence of refinement, pathlines are fast to integrate and render on the GPU. Streaklines and timelines adaptively generate more line geometry, which slows down integration and rendering (by factor 2). Masslines were shorter than the other line classes, therefore less geometry was created, which made both the integration and the rendering faster. Coincidentally, masslines produced less occlusion. Summarizing, direct particle visualizations usually require a high number of individual particles to reveal structures in the flow. The integral lines presented in this work, however, are a fast and compact alternative that can furthermore reveal temporal and spatial coherence, as well as mass separation.

(a) Pathlines with particle diameters of $15\,\mu m$ (●), $25\,\mu m$ (●), $35\,\mu m$ (●) and $50\,\mu m$ (●).

(b) Masslines ($d_p = 15..50\,\mu m$)

Figure 10.7.: Hydrocyclones are used to separate particles in liquid suspensions based on density variations. As visible in Fig. 10.7a, an increase of mass (and thereby inertia) causes particles to drift off the vortex core and to exit at different times. Masslines in Fig. 10.7b expand and swirl around the bottleneck, which shows that lighter particles reside longer in the bottleneck.

Table 10.1.: Timings for the interactive integration of $2\,k$ integral lines in the helicopter data set ($d_p = 15\,\mu m$, for masslines $d_p = 15..100\,\mu m$). The integration timings refer to the duration of one integration step. In our implementation advection and rendering are interleaved.

Pipeline step	**Pathlines**	**Streaklines**	**Timelines**	**Masslines**
Integration	1.2 ms	3.9 ms	3.4 ms	1.0 ms
Rendering	2.7 ms	6.0 ms	6.3 ms	1.7 ms
Total fps	202	100	92	340

10.5. Summary

In this chapter, we extended geometry-based flow visualization tools, usually used in massless particle-based visualizations, to mass-dependent particles. We defined a new flow map to abstract the application-dependent particle model, i.e., the assumptions made in the equations of motion. Based on that, we derived four classes of mass-dependent integral curves in unsteady vector fields: mass-dependent pathlines, mass-dependent streaklines, mass-dependent timelines, and a new class called *masslines* that visualizes the separation of particles with different response times, i.e., due to different size or density. We demonstrated the influence of the mass by the use of integral geometry in an artificial and two real-world data sets. Hereby, our main test case was the brownout simulation around a helicopter flying forward in ground proximity.

The processes involved in brownout are not yet fully understood, since it is difficult and expensive to validate the models, cf. Syal et al. [SGL10]. This is where we see much potential for future work, e.g., in the assistance in model validation, interactive previews for parameter steering of the long-term simulations, and most importantly, capturing the behavior of particles with mass in unsteady flow and revealing structures, e.g., vortices carrying heavy particles. But brownout simulations are just one application of mass-dependent particle visualization techniques. In the future, we plan to explore further applications.

11 Chapter 11. Finite-Time Mass Separation for Comparative Visualizations of Inertial Particles

The following chapter presents methods for the visualization of the mass-induced separation behavior of inertial particles and is based on the publication:

T. Günther and H. Theisel
Finite-Time Mass Separation for Comparative Visualizations of Inertial Particles
Computer Graphics Forum (Proc. EuroVis) 34, 3 (2015), 471–480.

Imagine we have a 2D time-dependent flow, as illustrated in Fig. 11.1. If we release at the very same seed point (here in black) two particles of different size – a small one in red and a big one in blue –, then they might separate due to inertia. The topic of this

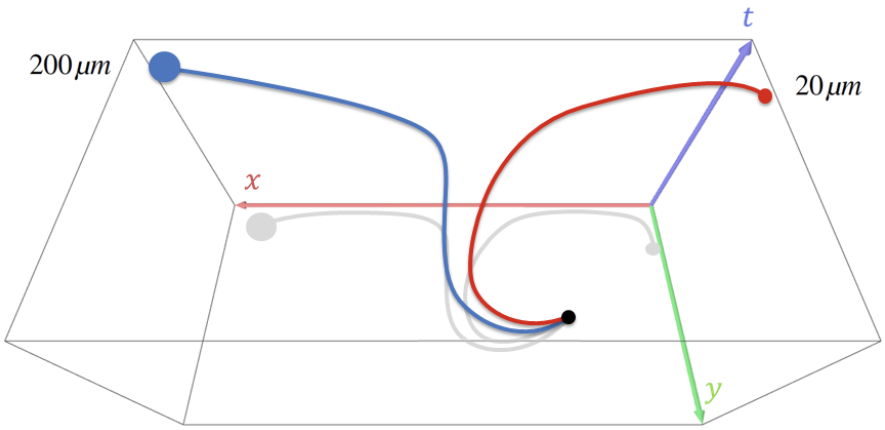


Figure 11.1.: The trajectories of two differently-sized particles separate in the 2D unsteady Double Gyre flow, which is here depicted in 3D space-time.

chapter is the exploration of this separation behavior, in order to determine where and when inertial particles separate due to their differences in mass. For this, we introduce a scalar field named *Finite-Time Mass Separation* (FTMS). Inspired by the inertial extension [SH09, PD09] of FTLE [Hal01, SLM05], the idea is to release inertial particles from the same location, but with slightly varied mass and to observe their separation behavior over time. Inertial FTLE, on the other hand, observes the separation due to spatial variation of the seed point, which can lead to different answers. Locating the mass that produces the *largest separation* in FTMS allows to extract and visualize critical masses. In addition, we complement the visualization by two coordinated views that provide detailed information on the mass-dependent particle trajectories released at a given seed point. In a *plot view*, we use a new distance measure between inertial particle trajectories that is occlusion-free and considers the actions of particle sizes in-between. This view allows to quantitatively measure absolute and relative differences, provides insights on how separation evolves over time and allows to quickly locate separation events. In a *domain view*, we depict inertial particle trajectories of unsteady 2D flows in 3D space-time, which embeds the trajectories in the domain and allows to see how they separate. By varying the mass, inertial particle trajectories assemble a surface, on which separation can be shown by visualizing its stretching. Fig. 11.2 gives an example. With this, we provide the first comprehensive study of the mass-dependent behavior of inertial particles that answers: *where* and *when* mass separation occurs, *which* masses are separating and *what* the separation looks like in the domain.

An application of this is the comparison of inertial particle trajectories with massless ones to see in which areas of the domain they make a difference and if it is possible to fall back to massless trajectories for efficiency. Rather specific to rotorcraft engineering is the question, which masses lift up from the ground and enter the rotor disk, cf. Syal et al. [SGL10]. In the wake of a helicopter, a large number of particles wash over the sediment bed, yet only some masses enter suspension. It is of interest to see, which mass is the one that induced the largest separation, i.e., smaller masses enter the disk whereas larger ones fall back to the sediment bed. Particles that enter the rotor disk cause mechanical wear of the blades and also, particles are convected downward by the rotordisk very rapidly. When they hit the sediment bed they stir up further particles, causing a chain reaction, i.e., brownout. Another application is jellyfish feeding [PD09, SPH11], in which prey motion was studied already with IFTLE.

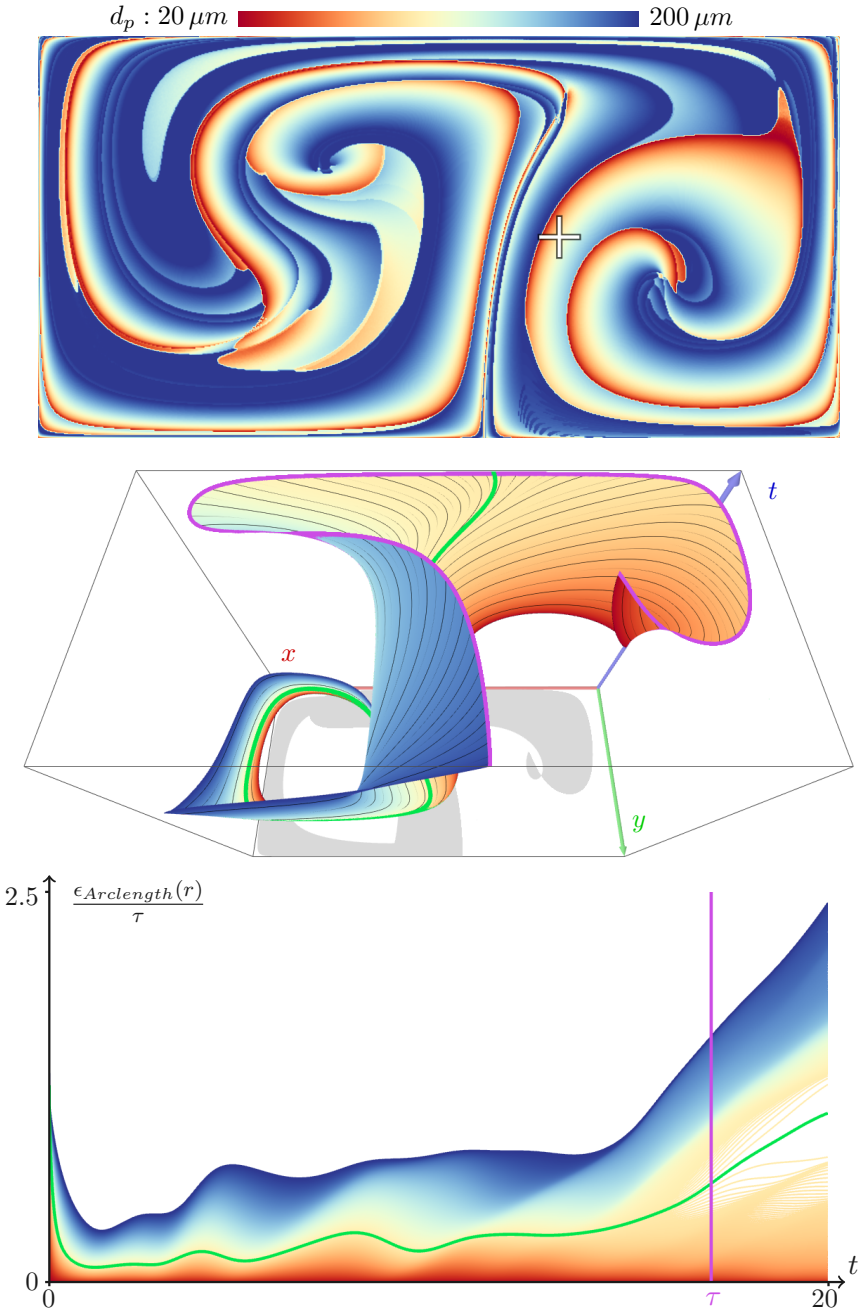


Figure 11.2.: Mass separation of inertial particles in the DOUBLE GYRE. Top image: Mass (color-coded) that separates strongest during integration over time $\tau = 17$. Center image: inertial particles of different mass are released at the cross in the left image. In space-time their trajectories assemble a surface, with the purple line being its front line – a so-called massline. The green line is the particle trajectory with strongest separation (here, with diameter $d_p = 99\,\mu m$). Bottom image: plot of difference to reference particle, here the smallest inertial particle (diameter $d_p = 20\,\mu m$), which makes the temporal evolution of the separation apparent.

11.1. Finite-Time Mass Separation

A so far unaddressed problem in flow visualization is the comparative visualization of inertial particles. This problem is quite challenging, as even from a single point in the domain an infinite number of inertial particles can be released that take different trajectories depending on their mass. Quantifying their differences requires an adequate distance measure and a study of the temporal evolution of these differences. This, however, first of all requires to set a seed point. Unfortunately, it is nearly impossible to predict ad-hoc *when* and *where* and for *which* mass interesting events occur in the domain. The key to the problem is to analyze the mass-induced separation behavior of inertial particles, as a separation entails the development of a difference. This insight leads us to the definition of the *Finite-Time Mass Separation* (FTMS) field. That is, we strive to quantify at every location in the space-time domain, how quickly inertial particles of slightly different mass separate if they are released from the *same* location. Inertial FTLE, in contrast, describes how inertial particles of the same mass separate if released in proximity, which does not produce insights on the relationship between different masses. Differences to inertial FTLE are discussed later in Section 11.4.1. FTMS is computed by considering the derivative of the flow map with respect to response time r[1], which is determined by the mass of the particle, cf. Eq. (9.3):

$$\mathrm{FTMS}(\mathbf{x},\, t,\, \tau,\, r) = \frac{1}{|\tau|} \ln \left|\left| \frac{\mathrm{d}\phi_t^\tau(\mathbf{x}, r)}{\mathrm{d}r} \right|\right| \tag{11.1}$$

This scalar field measures the separation of inertial particles that were released from the same location but with slightly different mass. Note that the field is derived from the flow map. Thus, it is independent of the particle model and the underlying equation of motions. The field can be equivalently computed from the magnitude of the first derivative of a massline[2], released at $(\mathbf{x},\, t)$, i.e., the massline's stretching:

$$\mathrm{FTMS}(\mathbf{x},\, t,\, \tau,\, r) = \frac{1}{|\tau|} \ln \left|\left| \frac{\mathrm{d}\mathbf{m}(r)}{\mathrm{d}r} \right|\right| \tag{11.2}$$

An example for an FTMS field is given in Fig. 11.3 for the DOUBLE GYRE [SLM05]. Similar to (inertial) FTLE, FTMS is defined only for particles that do not leave the domain.

[1] The response time linearly depends on diameter, density and viscosity. Since we keep density and viscosity constant, our experiments show diameter variation only. It would be possible to expand the definition of FTMS from response time to diameter. Defining FTMS via response time, however, is more general and thus we preferred it.

[2] Recalling Chapter 10, masslines are assembled by releasing particles from the same location at the same time, but with *varying mass*:

$$\mathbf{m}(r) = \phi_t^\tau(\mathbf{x},\, r)$$

Masslines are parameterized by particle mass, which is included in the particle response time r, and are continuous if the mass partial of the flow map is continuous.

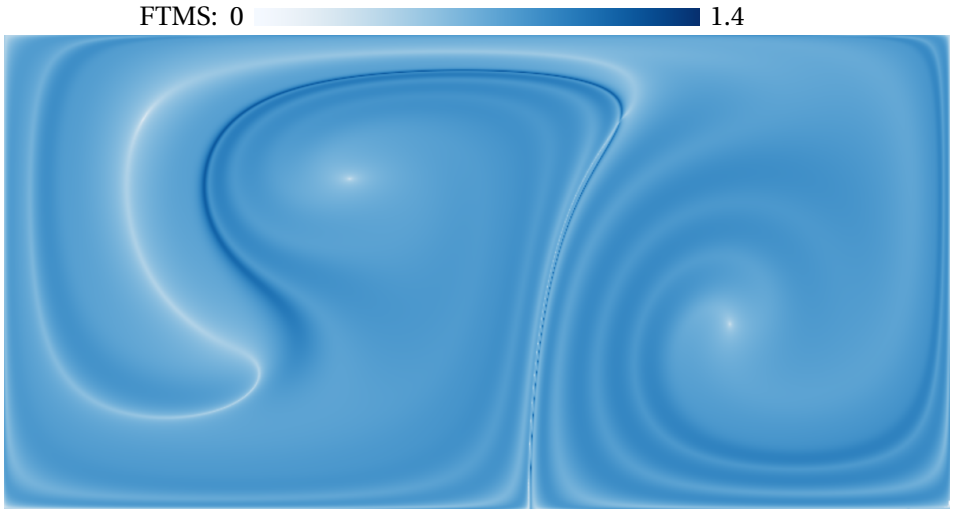


Figure 11.3.: Finite-Time Mass Separation (FTMS) of the DOUBLE GYRE for inertial particles with diameter $d_p = 99\,\mu m$ at $t = 0$ and $\tau = 9.95$.

At a certain time t, the parameter space of the FTMS scalar field is two-dimensional; it is spanned by response time r and integration duration τ. Due to the dependence on response time r the exploration of inertial particles in inherently more challenging than the exploration of traditional massless particles. Eventually, our task is to study the impact of r, rather than the selection of a particular value for it. In the following, we propose a guided way that enables the user to first slice in time to find an interesting starting point for the exploration of r. Once a starting point is found, the spatio-temporal evolution of inertial particles can be observed in additional views that are described in Section 11.2.

For an overview, it is interesting to identify for every space-time location, the response time r^* with largest separation, with $r^* \in [r_{\min}, r_{\max}]$:

$$r^* = \arg\max_r \mathrm{FTMS}(\mathbf{x}, t, \tau, r) \tag{11.3}$$

Fig. 11.4 shows its corresponding diameter. As an inset in the bottom right corner, we visualize the respective separation:

$$\mathrm{maxFTMS}(\mathbf{x}, t, \tau) = \mathrm{FTMS}(\mathbf{x}, t, \tau, r^*) \tag{11.4}$$

The response time interval $[r_{\min}, r_{\max}]$ is sampled by a Halton sequence [Hal60] to progressively refine the solution, as visualized in the accompanying video of the corresponding paper [GT15]. The computation is terminated by the user or after a certain number of iterations. The bounds of the interval are specified by the user as well or might be determined by the underlying equations of motion.

Depicting the maximal separation condenses the information into a visualization that only depends on duration τ, which can be explored by an animation. Of course,

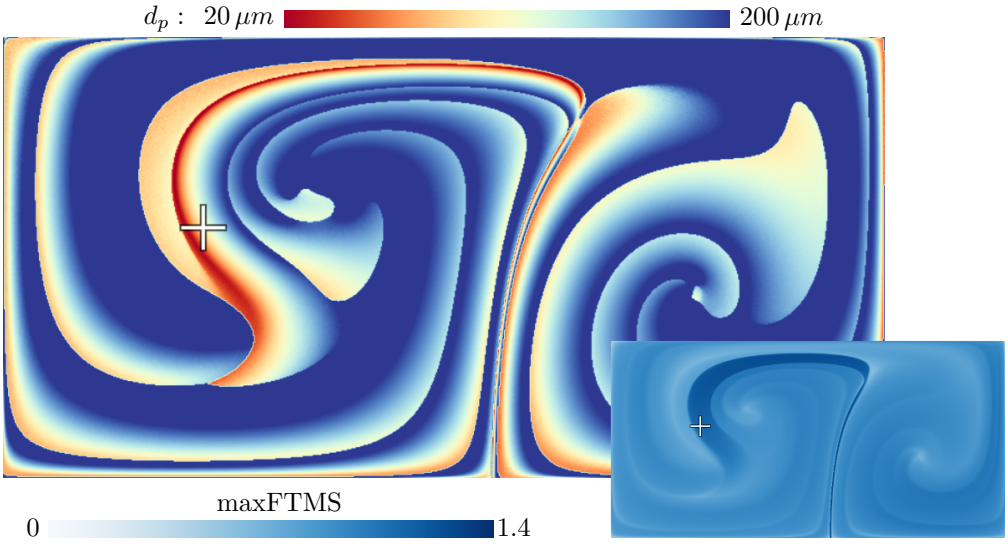


Figure 11.4.: Maximal Finite-Time Mass Separation (maxFTMS) in the DOUBLE GYRE at $t = 0$ after duration $\tau = 9.95$.

this only provides an overview and might not necessarily extract the most interesting separation event. At longer integration time several events might occur and if they exhibit similar separation strength it becomes less clear which to show, see later Section 11.4.5. For this reasons, maxFTMS cannot replace the detailed inspection of the two-dimensional parameter space. Providing the largest separation at a location, however, serves as a starting point for further investigations. Other separating masses can be observed in the views that are described next.

11.2. Comparative Visualizations

Once a spatio-temporal seed point is found, we investigate the behavior of differently-sized inertial particles released at this location.

11.2.1. Qualitative View

The mass-dependent flow map $\phi_t^\tau(\mathbf{x}, r)$, as introduced in Section 10.1, has four parameters. Varying one of them and keeping the others constant produces the four classes of integral curves that we have described in Chapter 10. Among those is the massline, which connects the locations that were reached by differently-sized inertial particles, i.e., particles with a varying response time r, chosen from an interval $[r_{\min}, r_{\max}]$. Varying two of the parameters similarly produces integral surfaces. In our visualization, we vary the response time r and integration duration τ to depict the temporal evolution

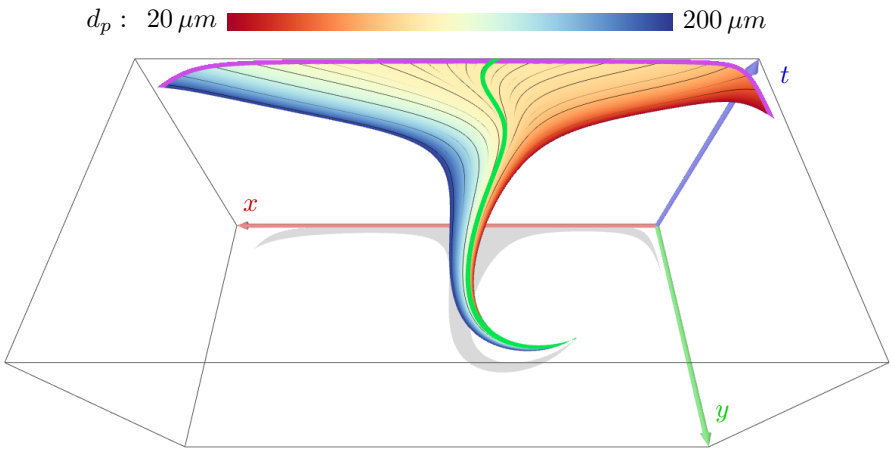


Figure 11.5.: Space-time domain of the DOUBLE GYRE in which inertial particle trajectories form a mass-path surface. The purple line is a massline, the green line depicts the trajectory of the most separating inertial particle, here with $d_p = 99\,\mu m$, $\tau = 9.95$ and seed point $(\mathbf{x}, t) = (0.45417, 0.5675, 0)$.

of the massline, which forms a *mass-path surface* in space-time as shown in Fig. 11.5. Thereby, the red and green axes are spatial dimensions and the blue axis corresponds to time.

We color-code the particle diameter on the surface and illustrate individual trajectories by an adaptive stripe pattern, as in [HGH*10]. The stripe spacing and the insertion of lines in between them are an indicator for separation. The front line of the surface is a massline (purple). It depicts the location of inertial particles after the currently selected integration duration. The user can further select a certain particle diameter for closer inspection. We highlight its corresponding particle trajectory by a green line. Color-coding, stripe patterns and highlighting selected lines are all implemented in the pixel shader of the surface. Alternatively, separation on a manifold could be visualized by space-time LIC as in [BSDW12].

11.2.2. Quantitative View

Another aspect is the mass distribution along a massline. Visualizing its temporal evolution can likewise show separation. Thus, in an additional view, we plot the integration duration τ of inertial particle trajectories vs. a specified distance measure, which measures the difference to a reference particle. Again, we color-code trajectories by their diameter. In these plots, a uniform sampling of the response time interval shows separation between the lines by their vertical distance.

Euclidean Distance: Bordás et al. [BBK*08, Bor11] measured the distance between experimentally traced particles by Euclidean distance, which was also used to compare experimental and simulated [OOG08] particle positions. Since experiments are conducted with a finite number of differently sized particles, there is no information on the behavior of particles with masses between them. Since our trajectories stem from a simulation, we can simulate particle sizes in-between. For each inertial particle, we measure the distance to the smallest inertial particle ($r = r_{\min}$):

$$\epsilon_{Euclidean}(r) = ||\phi_t^\tau(\mathbf{x}, r) - \phi_t^\tau(\mathbf{x}, r_{\min})|| \tag{11.5}$$

Fig. 11.6a gives an example for the space-time domain shown in Fig. 11.5, illustrating the split of particles that hit the domain boundary. It might not always be possible to attribute a separation in the plots to a single prominent event. Similar to FTLE, a strong separation might also occur due to accumulation of multiple (possibly smaller) events. After all, FTMS is an integration-based measure, too. However, if sudden changes occurs, they are likely worth to look at. While the Euclidean distance measure can capture periodicity of events, e.g., recurring proximity, it suffers especially at longer integration times from occlusion and cluttering, which is demonstrated later in Section 11.4.3.

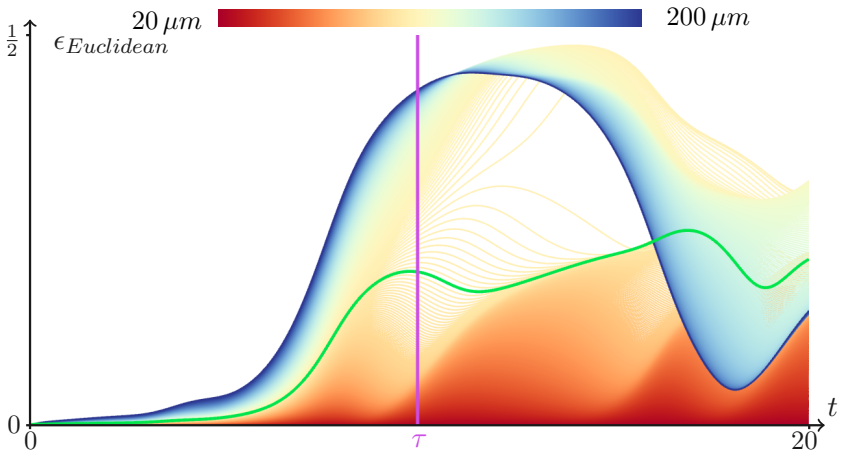
Arc length Distance: To overcome the limitations of the Euclidean distance measure, we propose to measure the arc length of the massline that connects the two particles.

$$\epsilon_{Arclength}(r) = \int_{r_{\min}}^{r} \left|\left| \frac{\mathrm{d}\mathbf{m}(s)}{\mathrm{d}s} \right|\right| \mathrm{d}s \tag{11.6}$$

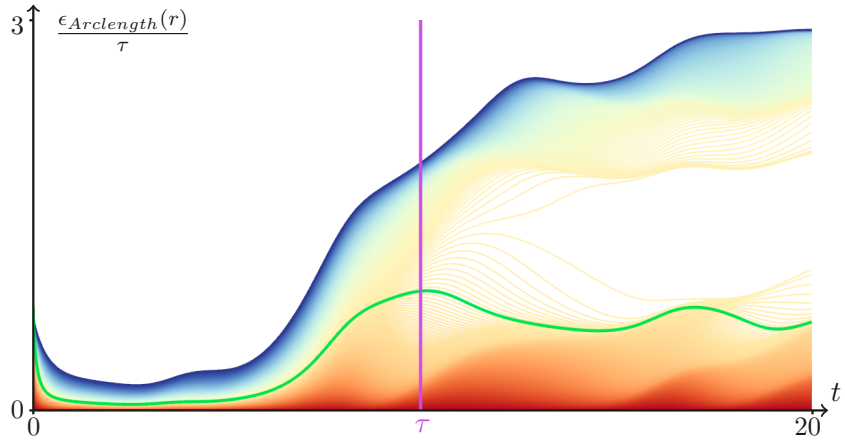
Such connecting massline exists only iff the particles were seeded at the same location at the same time. In contrast to Euclidean distance, this measure accommodates for the behavior of the masses in between and it monotonically increases with increasing response time, which avoids occlusion since no overlap occurs in the plot. For plotting this measure, we use two different normalizations. We either normalize the error by the integration duration $\frac{\epsilon_{Arclength}(r)}{\tau}$ to account for the growth of the line over time (as in Fig. 11.6b), or we use the relative position $\frac{\epsilon_{Arclength}(r)}{\epsilon_{Arclength}(r_{\max})}$ to display the evolution of the distribution of masses along the massline (see Fig. 11.6c).

11.3. Implementation

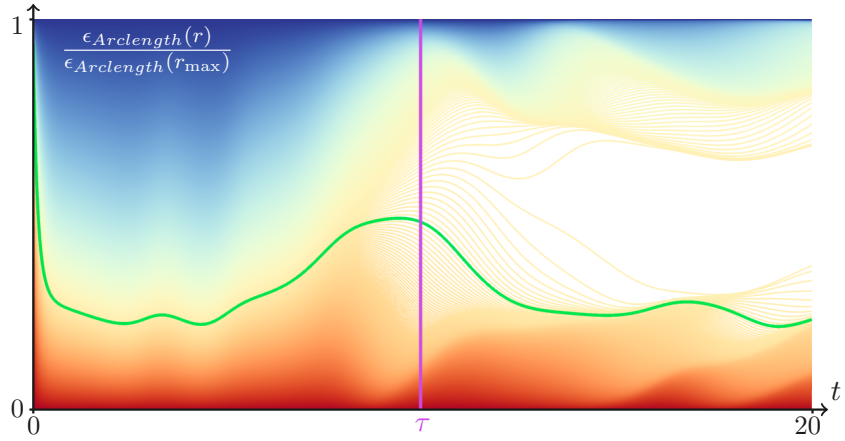
Interaction is a crucial aspect in exploration tasks, which requires fast updates on parameter changes. The integration of inertial particles according to Eq. (9.5), the computation of the mass-path surface in the domain view and the distance plots are



(a) Euclidean distance.



(b) Absolute arc length distance.



(c) Relative arc length distance.

Figure 11.6.: Distance plots over time in the DOUBLE GYRE, showing the temporal evolution of the difference to the smallest inertial particle ($r = 20\,\mu m$). As in Fig. 11.5, with selected diameter $d_p = 99\,\mu m$, $\tau = 9.95$ and the seed point $(\mathbf{x},\, t) = (0.45417,\, 0.5675,\, 0)$.

all implemented on the GPU, which comfortably achieves interactive frame rates. The $\mathrm{maxFTMS}$ field, however, is precomputed, which takes several minutes as listed later in Section 11.4.4. We discretize the space-time domain of the $\mathrm{maxFTMS}$ field onto a regular grid and progressively sample the mass range. The resolution of the regular grid is given later in Section 11.4.4. On the GPU, we therefore release inertial particles in the domain and integrate FTMS for the given time interval. In every time slice of the $\mathrm{maxFTMS}$ field, we check if the currently considered mass has larger separation than any previously processed mass and store it if this is the case. Finally, the field is stored as a 3D voxel grid (two spatial dimensions and time), with two entries per cell: the maximal mass and its separation. At runtime, the precomputed $\mathrm{maxFTMS}$ and FTMS fields can both be streamed to the GPU for interactive display.

As shown earlier, FTMS can be obtained from the magnitude of a massline's tangent. For computing the FTMS value, we do not have to integrate and refine a full massline to obtain a good estimate of its tangent. We have better control over the discretization error, if we stay local to the mass(es) that should be evaluated. Computing the response time partial in FTMS by first-order approximation (central differences) with step size h results in an error of order h. That is, we can place the computational power exactly where we need it rather than refining a full curve and only requiring tangents at several locations. For the distance plots, however, we use the arc length of a massline. Here, a refinement is clearly in order.

11.4. Results and Evaluation

We tested our method on a number of analytic and one real-world data set, which are briefly described in the following. In advance, one general note on the units: all domain sizes are assumed to be in meters and time is in seconds[3].

Throughout this chapter, the (dynamic) viscosity is set to $\mu = 1.532 \times 10^{-5}\ kg/(m\ s)$ (air), and we assume gravity-free environments $\mathbf{g} = \mathbf{0}$. For the particles, we used the density of dry sand $\rho_p = 1600\ kg/m^3$. The particle diameter d_p varies between $d_p = 10\,\mu m$ and $d_p = 500\,\mu m$. Given these parameters, particle response time r is in our experiments in the range $r \in [0.00058,\ 1.45]$.

Double Gyre. The unsteady 2D DOUBLE GYRE [SLM05] became a well-known benchmark data set for the study of particle separation by means of FTLE. We used the

[3] Checking if the length scale assumption $d_p \ll \eta_k$ holds in the analytic fields is an exercise of calculating the Kolmogorov length scale η_k, see Eq. (9.4). For the Duffing-based fields, we obtain η_k around $1 - 2\,mm$ and in the Double Gyre $\eta_k = 5\,mm$, which is larger than the particle sizes we used.

analytic form given in Section 7.4, here in the longer temporal domain $[0, 30]$. Examples are shown in Fig. 11.2 for $\tau = 17$ and throughout Sections 11.1 and 11.2 for $\tau = 10$.

Forced Duffing Oscillator. The FORCED DUFFING oscillator in an example for a dynamical system that experiences chaotic behavior [HS11]. It can be described and visualized as an unsteady 2D vector field of the form:

$$\mathbf{u}(x, y, t) = \begin{pmatrix} y \\ x - x^3 + 0.1 \sin t \end{pmatrix} \tag{11.7}$$

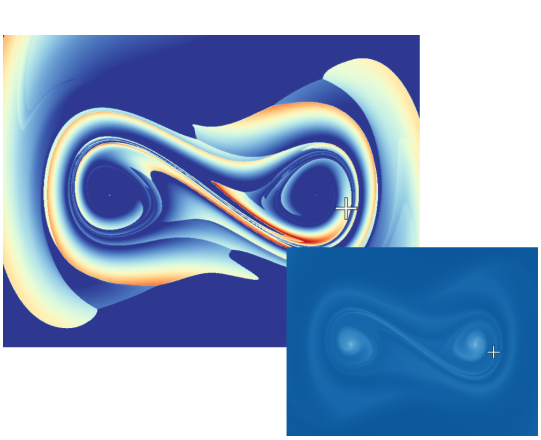
here, in the domain $D \times T = [-2, 2]^2 \times [0, 20]$. In Fig. 11.7a, we show the $\mathrm{maxFTMS}$ field and select a seed point. At this seed point, particles with diameter $d_p = 64\,\mu m$ show largest separation. For this diameter, we depict the FTMS field in Fig. 11.7b. In the space-time domain in Fig. 11.7c, a mass-path surface is released from the seed point. The trajectory with diameter $d_p = 64\,\mu m$ is highlighted and turns out to run toward a separation point, i.e., a saddle in the inertial dynamics that causes particle separation. The temporal evolution of the massline particles' distance to the smallest inertial particle is shown in Fig. 11.7d, indicating the separation event as well.

Forced-Damped Duffing Oscillator. A more general form of the oscillator is the FORCED-DAMPED DUFFING oscillator [HS11], which can likewise be described as an unsteady 2D vector field:

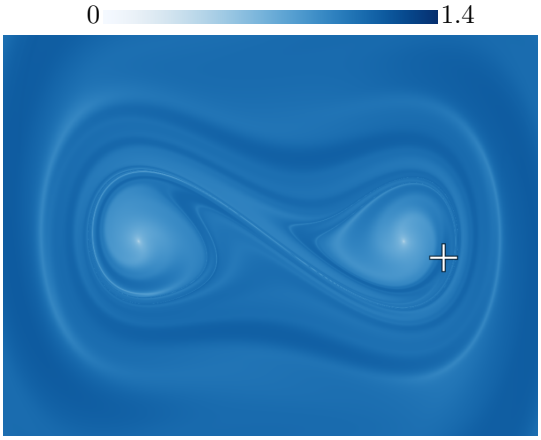
$$\mathbf{u}(x, y, t) = \begin{pmatrix} y \\ x - x^3 - 0.25y + 0.4 \cos t \end{pmatrix} \tag{11.8}$$

here, in the domain $D \times T = [-2.5, 2.5]^2 \times [0, 20]$. Fig. 11.8a depicts the $\mathrm{maxFTMS}$ field at $\tau = 10.1$[4]. At the selected point, a particle with diameter $d_p = 69\,\mu m$ shows the largest separation. The corresponding FTMS field is shown for this diameter in Fig. 11.8b. From the selected point inertial trajectories are released, which assemble in space-time a surface, see Fig. 11.8c. The trajectory with largest separation is shown on the surface, as well as in the distance plot in Fig. 11.8d.

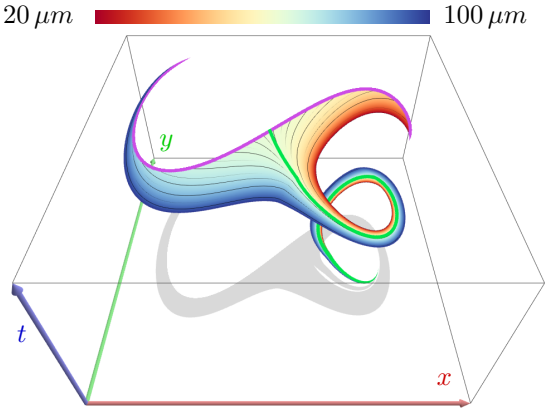
[4] A note on the choices of τ in the visualizations: As described in the implementation section, we precompute the $\mathrm{maxFTMS}$ sequences, i.e., we discretize the τ range. In the Figs. 11.7 and 11.8, the ranges were discretized slightly differently and we simply snapped to the time step closest to $\tau = 10$. For reproducibility, we named the actual value, even if $\tau = 10$ would look very similar. It might seem that we chose $\tau = 10$ rather frequently, but this is because the images often show purposefully different aspects of the very same setting, e.g., Figs. 11.3, 11.4, 11.5 and 11.6 all share time, integration duration and seed point, and illustrate the same data set. The integration duration was set to about 10 because in our examples this was often around the first strong separation event, which we wanted to inspect. The video of the corresponding paper [GT15] shows coherent animations of the integration duration and thereby other values.



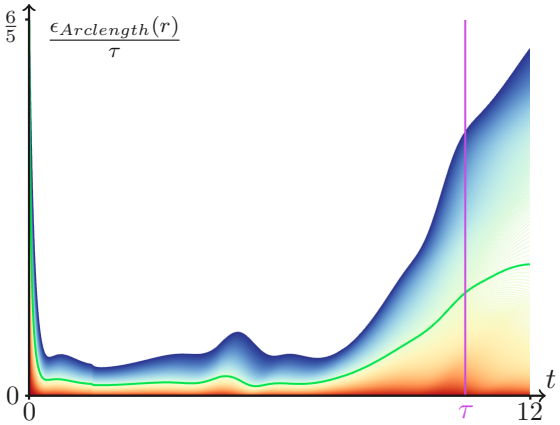
(a) At selected point, $d_p = 64\,\mu m$ has max. separation (maxFTMS).



(b) FTMS value for $d_p = 64\,\mu m$ is indeed high.



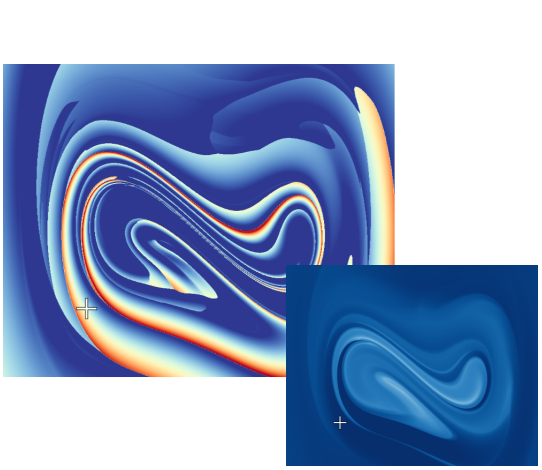
(c) Trajectory with maximal separation on the mass-path surface.



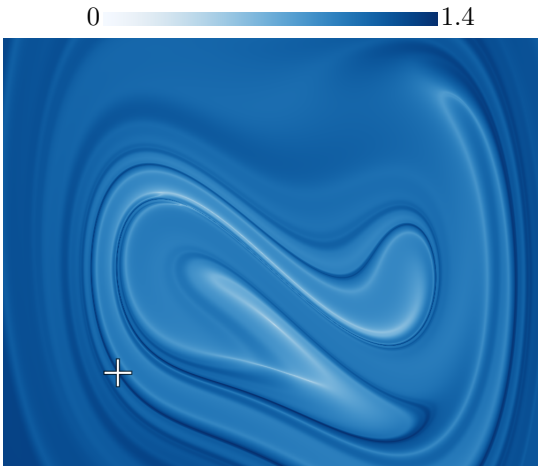
(d) Massline arc length-based distance plot to a reference particle.

Figure 11.7.: FORCED DUFFING data set at integration duration $\tau = 10.45$. The seed point is at $(\mathbf{x}, t) = (1.294, -0.22133, 0)$.

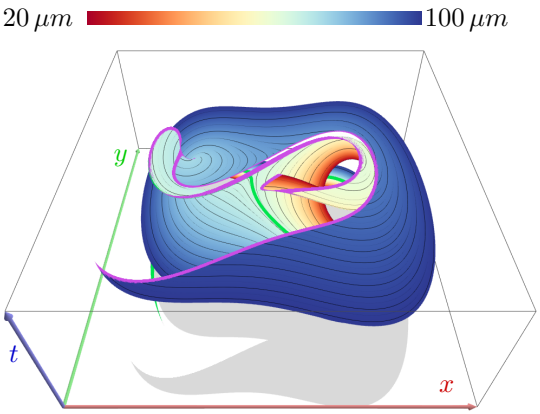
Square Cylinder. The SQUARE CYLINDER flow, as described in Section 4.4, is a 3D time-dependent flow around an obstacle. It is quite uniform along one dimension, which allows to select one slice and treat it as 2D unsteady flow. As visible in the maxFTMS field in Fig. 11.9a, large separation occurs for particles that flow toward the obstacle and for particles that enter the vortices of the von Kármán vortex street, since there, rotational movement amplifies the effects of inertia, leading to separation. The latter can be seen by the alternating and periodic separation fronts, running toward and past the obstacle. When selecting one seed point, both the space-time domain in Fig. 11.9b and the distance plot in Fig. 11.9c show multiple separations along the massline. After integration time $\tau = 64$, the strongest separation arises for a diameter $d_p = 380\,\mu m$, which is highlighted.



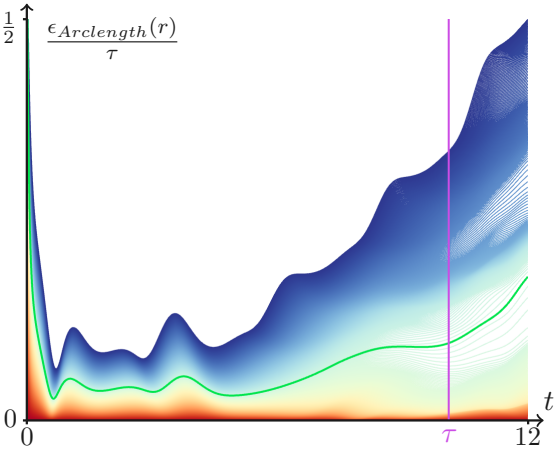
(a) At selected point, $d_p = 69\,\mu m$ has max. separation (maxFTMS).



(b) FTMS value for $d_p = 69\,\mu m$ is indeed high.



(c) Trajectory with maximal separation on the mass-path surface.

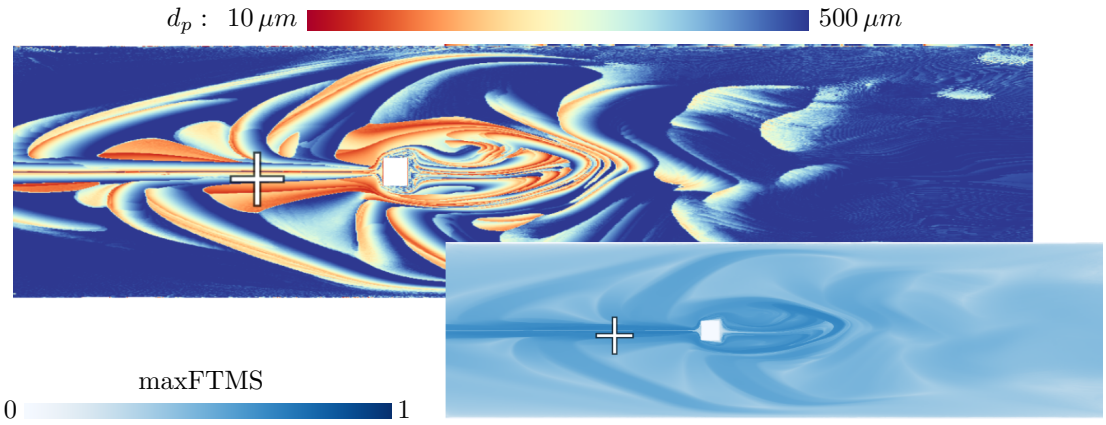


(d) Massline arc length-based distance plot to a reference particle.

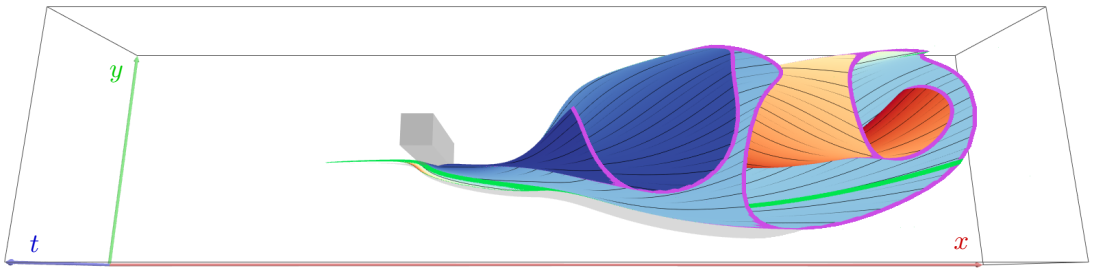
Figure 11.8.: FORCED-DAMPED DUFFING data at integration duration $\tau = 10.1$ and seed point $(\mathbf{x}, t) = (-1.42613, -1.41141, 0)$.

11.4.1. Comparison between Inertial FTLE and FTMS

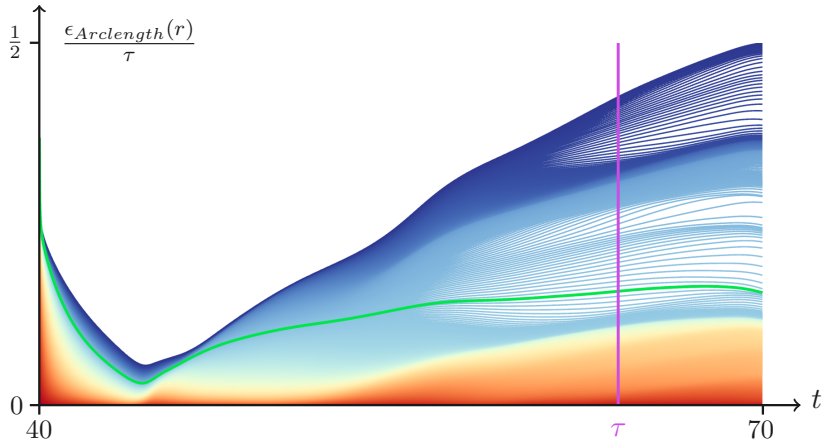
Inertial FTLE depicts the separation due to spatial variation of the seed point. The separation due to a varying mass is shown by FTMS. Frequently, both measures have a high correlation, but we observed distinct features that are visible in FTMS only. For instance, in Fig. 11.10, we see valley lines in FTMS, which do not appear in IFTLE. They may even cross an IFTLE ridge. This means, if a separation of nearby particles occurs, particles with different mass that were released from the same location may still stay together – and, vice versa. In fact, there are two sources of separation to consider: varying mass (FTMS) and variation of the seed point (IFTLE). When studying the differences between masses, FTMS is more suitable. Further, we found in this



(a) Maximal separation at selection for $d_p = 380\,\mu m$.



(b) Trajectory with maximal separation on the mass-path surface.



(c) Massline arc length-based distance plot.

Figure 11.9.: SQUARE CYLINDER data set at integration duration $\tau = 64$ and seed point at $(\mathbf{x}, t) = (7.664, 3.77067, 40)$.

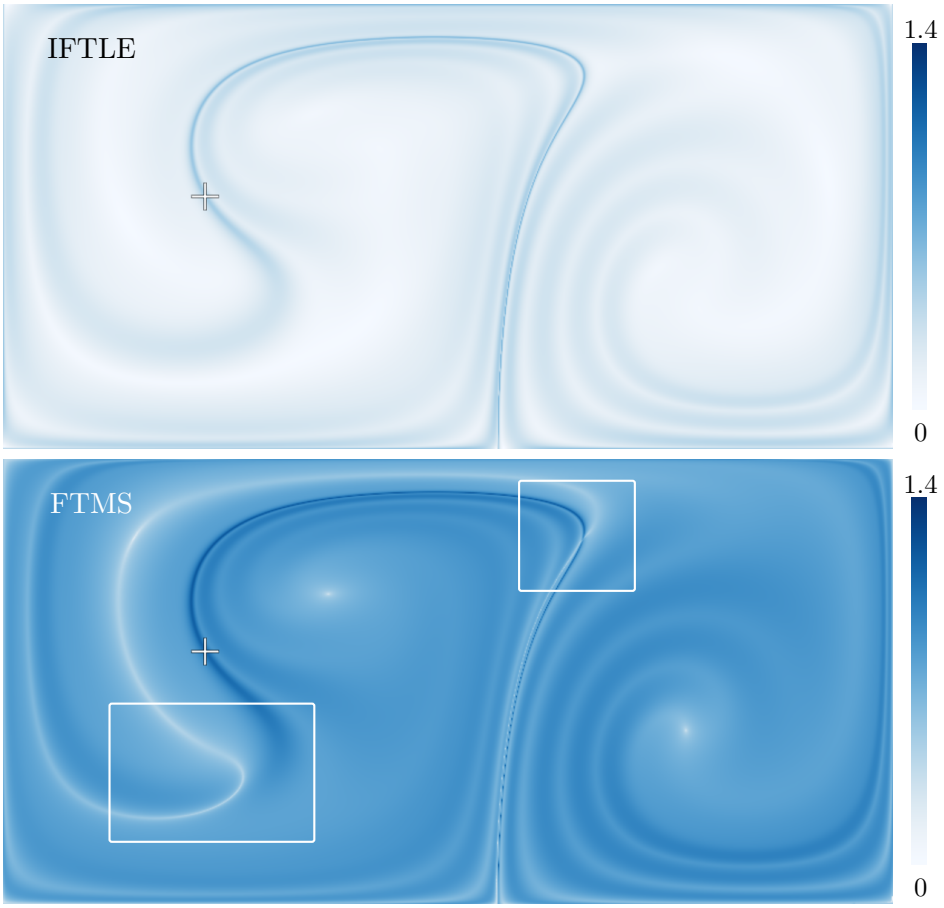
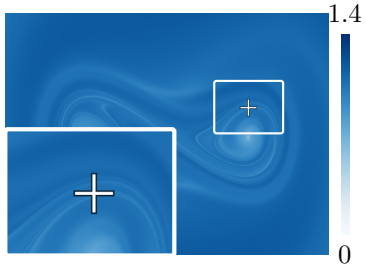


Figure 11.10.: Comparison between inertial FTLE (IFTLE) and FTMS for diameter $d_p = 99\,\mu m$ at $t = 0$ and $\tau = 9.95$. A corresponding point is selected, showing that ridges in IFTLE and FTMS frequently correlate. However, notice the absence of the white valley lines in IFTLE that even cross ridges.

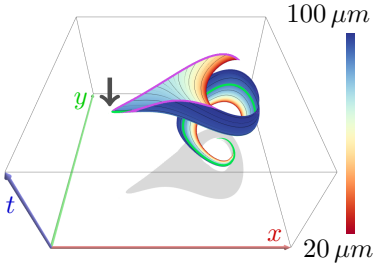
example that the separation due to varying mass is stronger, as visible by the color map. Generally, the computational complexity of FTMS is lower compared to IFTLE, since only halve the number of trajectories are integrated (or third in 3D).

11.4.2. Low and High Values in FTMS

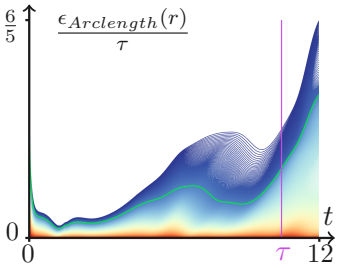
FTMS depends on the particle response time, which is determined by the particle size. In Fig. 11.11, we examine this dependence by observing FTMS for different particle sizes at the *same* seed point. For a particle with diameter $d_p = 89.9\,\mu m$ the seed point has a low FTMS value, i.e., we have low separation. In the space-time domain, this relates to a trajectory that is located in the fold of the mass-path surface. In this data set, this is a place of particularly low separation. In the difference plot, the low separation leads to a strong color gradient. For another diameter $d_p = 73.2\,\mu m$ the very same



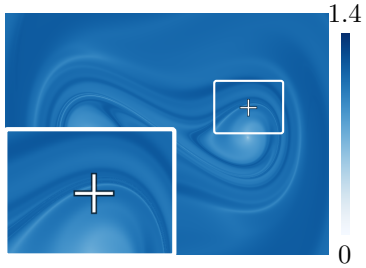
(a) Low FTMS value for $d_p = 89.9\,\mu m$.



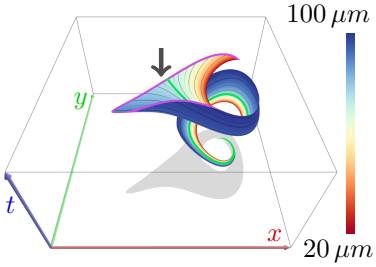
(b) Trajectory is in fold of mass-path surface.



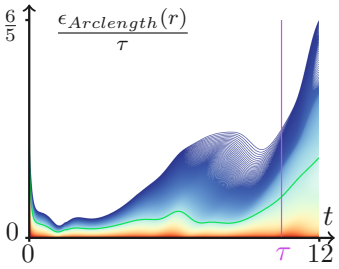
(c) Low FTMS has strong color gradient.



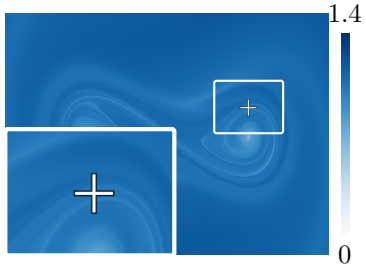
(d) High FTMS value for $d_p = 73.2\,\mu m$.



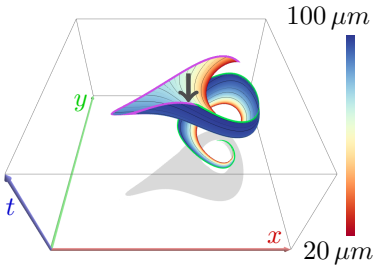
(e) Trajectory runs toward a saddle.



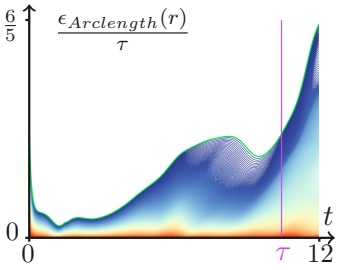
(f) High FTMS has flat color gradient.



(g) High FTMS value for $d_p = 100\,\mu m$.



(h) Separation due to shear.



(i) Similarily flat color gradient.

Figure 11.11.: Comparison of *different* masses that cause small and strong separation, released from the *same* seed point in the FORCED DUFFING data set at integration duration $\tau = 10.45$. The seed point is at $(\mathbf{x}, t) = (1.00496, 0.43565, 0)$.

seed point has a high FTMS value. In both space-time domain and difference plot the separation becomes apparent, as the selected trajectory runs toward a saddle, causing a separation. As shown by diameter $d_p = 100\,\mu m$, a high FTMS value does not necessarily relate to saddle-like separation. Similar to FTLE, a difference can also occurs due to shearing. Another example for a saddle-like separation was shown in Fig. 11.7.

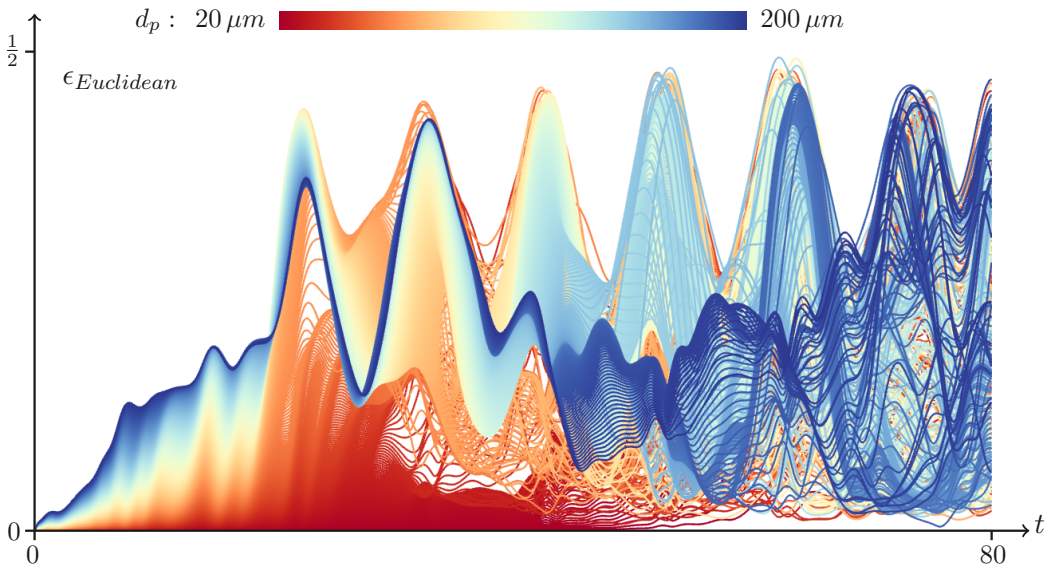
11.4.3. Comparison between Distance Plots

Section 11.2.2 introduced two distance measures between differently sized particles: Euclidean distance in Eq. (11.5), and the arc length of the connecting massline in Eq. (11.6). In the following, we compare them in more detail at longer integration time. In Fig. 11.12a, the Euclidean distance between differently-sized inertial particles and the location of a reference particle is shown in the DOUBLE GYRE sequence. In this data set, particles are confined in a closed domain and repeatedly flow against a wall, which causes them to split up. These periodic separation events, as well as the recurring particle proximity can be seen in the Euclidean distance plot. However, the longer particles are integrated the more chaotic the distance plot becomes, due to massive occlusion and clutter.

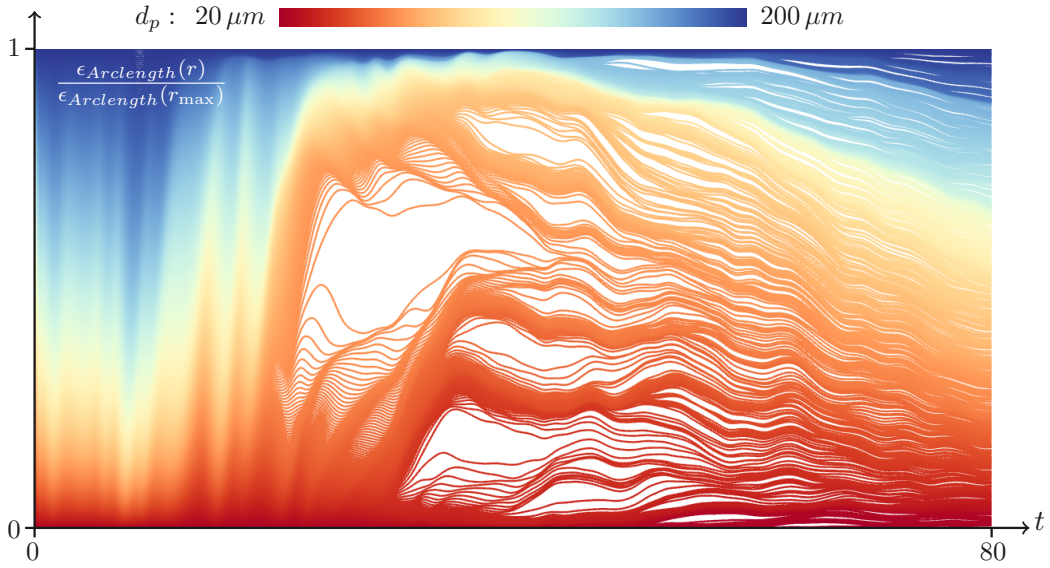
For this reason, we proposed the usage of the arc length distance along the connecting massline to measure the distance to a reference particle, here, the smallest inertial particle. Fig. 11.12b depicts the temporal evolution of the relative position of a mass along the massline. This plot shows separation events clearer, because occlusion cannot occur. But, as arc length increases over time, separation events become relatively less prominent. At a certain time, multiple masses can experience separation, which is likewise visible.

11.4.4. Performance

We used an Intel Core i7-2600K CPU with 3.4 GHz and 24 GB RAM, and an Nvidia Quadro K5000 GPU with 4 GB VRAM. The computation time of the $\mathrm{maxFTMS}$ field is in the order of minutes and depends on spatial resolution, integration duration and the number of sampled masses. We precompute the sequence on the GPU (see Section 11.3) and store it to disk. Preprocessing timings of the $\mathrm{maxFTMS}$ time series are listed in Table 11.1. For the DOUBLE GYRE the preprocessing of the full time and mass range took less than two minutes. The FORCED DUFFING had twice the spatial resolution and thus took roughly twice as long. In the FORCED-DAMPED DUFFING, we increased the number of samples in the mass dimension. In our implementation, the sampling of masses is serialized, which linearly increases the number of com-



(a) With Euclidean distance periodic events are visible. But, it becomes cluttered with longer integration duration.



(b) The arc length difference appears more organized and splitting events are better visible.

Figure 11.12.: Periodic events for a long time range in the DOUBLE GYRE: Euclidean (top) clutters eventually. Integration duration τ is in $[0, 80]$, seed point at $(0.83087, 0.28715, 0)$.

Data set	X	Y	T	M	maxFTMS ($min.$)
DOUBLE GYRE	800	400	200	128	1.68
FORCED DUFFING	800	800	200	128	4.75
F-DAMPED DUFFING	800	800	100	512	18.26
SQUARE CYLINDER	800	400	200	256	61.20

Table 11.1.: Space-time resolution of the maxFTMS grid $X \times Y \times T$, number of sampled masses M and the computation time of maxFTMS in minutes.

pute kernel invocations. Thus, with four times more masses, the computation time increases by factor four. It can further be seen that the smaller temporal resolution had computationally almost no effect. This is because it only reduced the number of max() operations due to fewer time slices. The dominating factor is the fourth-order Runge-Kutta integration of Eq. (9.5), which took in both Duffing oscillators roughly the same time. The integration duration was given in the data set descriptions in Section 11.4. The SQUARE CYLINDER flow was the slowest, since here flow vector values are sampled from a 3D texture (2D unsteady), rather than being evaluated analytically. In this case, the runtime is bound by fetches from texture memory.

11.4.5. Limitations

For long integration times, strong separation may occur for multiple masses, as shown in Section 11.4.3. The maxFTMS field depicts only the mass with strongest separation. If multiple masses have an equally large separation, an effect comparable to z-fighting [AMHH08] occurs, see Fig. 11.13. The longer the integration, the more likely this artifact becomes.

Also, the mass with maximum separation might not necessarily be the mass that is most interesting, because separation might also happen due to shearing – an artifact known from FTLE. However, while further local maxima of other masses might be hidden behind the global one, our method allows to rule out areas in which no separation occurs at all.

11.5. Summary

In this chapter, we presented an approach to a comparative visualization of inertial particle trajectories. For this, we introduced the Finite-Time Mass Separation (FTMS), a scalar field that measures the separation of inertial particles that were released from the same location but with slightly different mass. By extracting for every location in

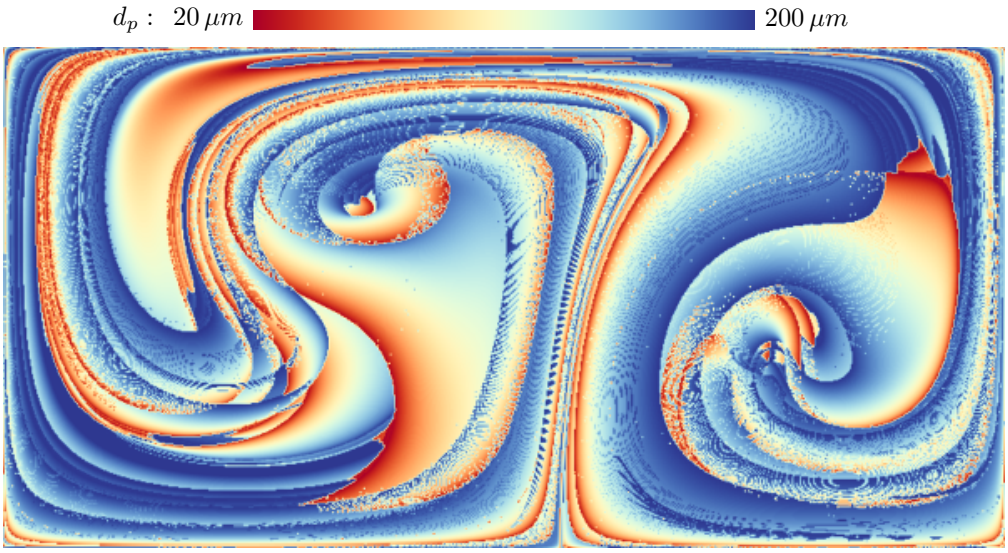


Figure 11.13.: Multiple masses exhibit a similarly strong separation after integration duration $\tau = 30$ in the DOUBLE GYRE.

space-time the mass that induced the largest separation, we provided a guiding tool that allows to quickly locate seed points and masses at which differences in inertial particle trajectories can be observed. In additional coordinated views, we displayed inertial particle trajectories at a specified seed point in space-time by integral curves and surfaces. Moreover, we plotted differences to a reference particle over time to observe the mass distribution and the temporal evolution of possibly further separation events.

The next step for future work is the third spatial dimension. Conceptually, the definition of FTMS and the plot view directly carry on into 3D. In the domain view, time cannot be visualized by a spatial axis anymore, which relates to the problem of visualizing path surfaces in 3D, i.e., occlusion and self-intersection become more prominent visualization problems. Also, the depiction of the maxFTMS field needs an extension. It probably reduces to a volume rendering problem, in which two attributes should be shown: the maximum separation and the corresponding mass. In 3D, the computation of the maxFTMS field would be quite expensive. An acceleration of its computation therefore seems a worthwhile topic for future work.

As for IFTLE, we focused on the conceptual differences to FTMS and at times referred to it to show where the two are conceptually similar. We have indicated that visual differences exist mainly to spur further research in this direction. Developing a deep understanding of the relations to IFTLE opens up further questions. For instance, in which ways do they agree/disagree from a physical point of view, and can the agreement be calculated?

Studying the relation with IFTLE possibly includes the definition of a unified separation measure that accounts for both spatial and mass variation. Currently, we see this rather as a long-term goal. If we see a separation in a combined measure, we would still like to understand which of the two caused the separation: mass variation or spatial variation. Before combining the two concepts, we should probably first understand the two components separately. IFTLE is known to the CFD community for six years and was already applied in practice [SPH11]. Yet still, even for IFTLE there are still open questions, e.g., there is no method that reliable finds attracting ILCS (requires backward integration) for "larger" particles. In this chapter, we introduced mass separation (FTMS) and took a first step toward the relations between the two. Of course, there is much more to study on both IFTLE and FTMS, and a combination of the two measures is certainly part of it.

Another aspect to study is the comparison of trajectories that were computed from different particle models or other equations of motion to see if certain simplifying assumptions can actually be made. Also the comparison of simulated and measured trajectories is an important task. Both require different distance measures, as for those, there are no connecting masslines.

12 Chapter 12. Inertial Steady 2D Vector Field Topology

The following chapter extends traditional steady 2D vector field topology to the inertial case and visualizes the asymptotic behavior of inertial particles for varying initial conditions. It is based on the publication:

T. Günther and H. Theisel
Inertial Steady 2D Vector Field Topology
Computer Graphics Forum (Proc. Eurographics) 35, 2 (2016), 455–466.

Vector field topology is a powerful tool to study asymptotic flow behavior and compactly describe fluid dynamics. Traditional vector field topology in flow visualization considers the motion of massless tracer particles of zero size in a fluid, and not the motion of finite-sized objects immersed therein. In this chapter, we utilize that the behavior of inertial particles is governed by a higher dimensional vector field, in which inertial trajectories appear as tangent curves (cf. Eq. (9.5)). To capture the asymptotic behavior of inertial particles, we conduct a topological analysis of the high-dimensional flow and show that the location of the critical points can be efficiently computed in 2D. Among others, we found that sources cannot exist in inertial flows, which explains the strong repelling behavior during backward integration. To visualize the asymptotic behavior of inertial particles, we propose a glyph visualization that encodes for every initial position and initial velocity the critical point that is reached in the limit. Fig. 12.1 gives an example. To obtain interactive rates, we compute the glyphs progressively and adaptive to the current view. With this, we present a first step to extend traditional vector field topology to the inertial case, and present a visualization that encodes the possible outcome of varying initial conditions to an ODE. We apply our extraction and visualization to a number of synthetic and real-world 2D steady flows, including the potential field of a benzene molecule and magnetic fields of decaying magnetic rings.

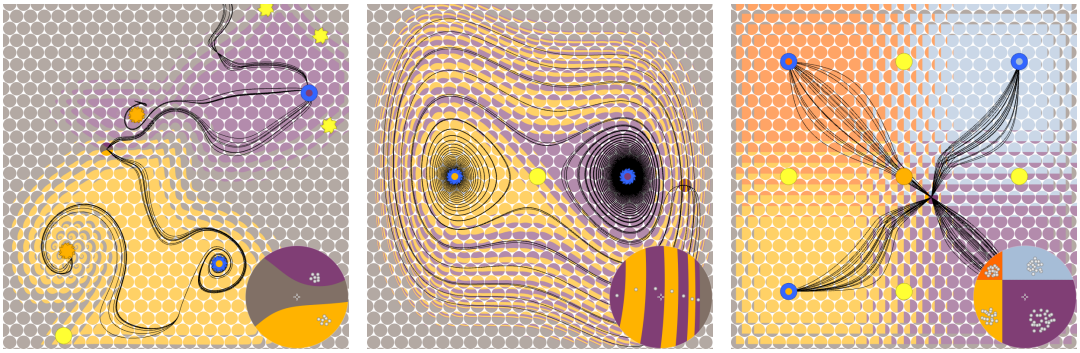


Figure 12.1.: Visualization of inertial critical points and a glyph-based visualization of the asymptotic behavior of inertial particles. The sink (blue critical point) that is reached by an inertial particle depends on its initial position $\mathbf{x}_0$ and the initial velocity $\mathbf{v}_0$. The glyph location encodes $\mathbf{x}_0$ and the glyph disc color-codes the sink that is reached for a certain $\mathbf{v}_0$. The glyph center represents $\mathbf{v}_0 = \mathbf{0}$ and the selected glyph is enlarged in the bottom right corner, showing the seeds of inertial particle trajectories. Left: BORROMEAN RINGS with particle diameter $d_p = 70\,\mu m$ and maximal velocity $v_{max} = 10\,m/s$ (occurs at glyph boundary), middle: DUFFING OSCILLATOR with $d_p = 100\,\mu m$, $v_{max} = 5\,m/s$, and right: NINE CP with $d_p = 70\,\mu m$, $v_{max} = 10\,m/s$. Inertial critical point types are listed in Table 12.1.

Notation: Since for inertial particles the direction of movement does not only depend on the vector field $\mathbf{u}$ but also depends on the current particle velocity $\mathbf{v}$, we denote 4D points/vectors as $\tilde{\mathbf{x}}, \tilde{\mathbf{u}}$ where the first 2 components refer to the spatial locations and the last 2 refer to the current particle velocity. Further, $\mathbf{0}_{2,2}$ denotes the 2×2 zero matrix, $\mathbf{I}_2$ is the 2×2 identity matrix, and $\mathbf{0}_2$ and $\mathbf{0}_4$ are the 2D and 4D zero vector, respectively. For brevity, we omit the index when it can be inferred from context. We denote the imaginary unit i, and $Re(z)$ and $Im(z)$ the real and imaginary part of a complex number z.

12.1. Inertial Topology

We begin the topological analysis of the asymptotic behavior of inertial particles by consideration of inertial critical points and separatrices.

12.1.1. Inertial Critical Points

In Chapter 9, we described the trajectories of inertial particles as tangent curves of a high-dimensional vector field, which models both the rate of change of particle

position $\mathbf{x}$ and particle velocity $\mathbf{v}$. For a steady underlying 2D flow $\mathbf{u}(\mathbf{x})$, the governing 4D vector field becomes:

$$\tilde{\mathbf{u}}(\mathbf{x},\mathbf{v}) = \frac{\mathrm{d}}{\mathrm{d}\tau}\begin{pmatrix}\mathbf{x}\\ \mathbf{v}\end{pmatrix} = \begin{pmatrix}\mathbf{v}\\ \frac{\mathbf{u}(\mathbf{x})-\mathbf{v}}{r}+\mathbf{g}\end{pmatrix} \text{ with } \begin{pmatrix}\mathbf{x}\\ \mathbf{v}\end{pmatrix}(0) = \begin{pmatrix}\mathbf{x}_0\\ \mathbf{v}_0\end{pmatrix} \tag{12.1}$$

where $\mathbf{g}$ is a gravity vector, $\mathbf{x}_0$ and $\mathbf{v}_0$ are the initial particle position and velocity, and r is the particle response time. The 4×4 Jacobian field of $\tilde{\mathbf{u}}(\mathbf{x})$ is:

$$\tilde{\mathbf{J}}(\mathbf{x},\mathbf{v}) = \begin{pmatrix}\mathbf{0}_{2,2} & \mathbf{I}_2\\ \frac{1}{r}\mathbf{J} & -\frac{1}{r}\mathbf{I}_2\end{pmatrix} . \tag{12.2}$$

We search for isolated first-order critical points of the inertial 4D flow from Eq. (12.1) as locations where $\tilde{\mathbf{u}}(\mathbf{x},\mathbf{v}) = \mathbf{0}_4$. This gives $\mathbf{v} = \mathbf{0}_2$ as the solution for the velocity components and yields the following 2D condition for retrieving the location in the spatial subspace:

$$\mathbf{u}(\mathbf{x}) + r\,\mathbf{g} = \mathbf{0}_2 . \tag{12.3}$$

Note that this is consistent with the massless case, since for $r = 0$ it retrieves locations where $\mathbf{u}(\mathbf{x}) = \mathbf{0}_2$. For the special case of gravity-free environments, i.e., $\mathbf{g} = \mathbf{0}_2$, we have a one-to-one relation between the critical points of $\tilde{\mathbf{u}}$ and $\mathbf{u}$: a critical point $\mathbf{x}_c$ of $\mathbf{u}$ corresponds to the critical point $(\mathbf{x}_c, \mathbf{0}_2)$ of $\tilde{\mathbf{u}}$. In order to classify the critical points of $\tilde{\mathbf{u}}$, we conduct an eigenanalysis of $\tilde{\mathbf{J}}$ at the critical points. The four eigenvalues $f_{1,1}$, $f_{1,2}$, $f_{2,1}$ and $f_{2,2}$ of $\tilde{\mathbf{J}}$ can be expressed in terms of the eigenvalues e_1, e_2 of $\mathbf{J}$ as:

$$f_{j,1} = \frac{-1-\sqrt{1+4\,r\,e_j}}{2r}\ ,\ f_{j,2} = \frac{-1+\sqrt{1+4\,r\,e_j}}{2r} \tag{12.4}$$

for $j \in \{1,2\}$. Note that the eigenvalues $f_{j,1}$ and $f_{j,2}$ that stem from the eigenvalue e_j are not necessarily complex conjugates. Using the common convention that the square root of a number has a non-negative real part, we have $Re(f_{j,1}) \le Re(f_{j,2})$. The proof of Eq. (12.4) is in a Maple sheet in the accompanying material of the corresponding paper [GT16a]. Eq. (12.4) gives a simple relation between the eigenvalues of $\tilde{\mathbf{J}}$ and $\mathbf{J}$, which allows a complete classification of $f_{j,k}$ based on e_j. From Eq. (12.4) follows

$$\frac{f_{j,1}+f_{j,2}}{2} = -\frac{1}{2r} < 0 \tag{12.5}$$

which entails interesting properties. It means that each pair of eigenvalues $f_{j,1}$, $f_{j,2}$ is located diametrically opposite around the real-valued constant center $-1/(2r)$ in the complex plane. Since, the mean of two eigenvalues is negative, it follows

$$Re(f_{j,1}) < 0. \tag{12.6}$$

Hence, every inertial critical point in $\tilde{\mathbf{u}}$ *must exhibit* in at least two directions attracting behavior (one for each pair). For $Re(f_{j,2})$, both a positive or a negative sign is possible. Here the condition is

$$Re(f_{j,2}) = 0 \iff Re(e_j) = -r\ Im(e_j)^2 \tag{12.7}$$

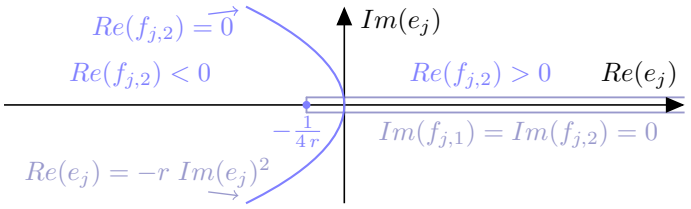


Figure 12.2.: A classification of the critical points in 4D. Each eigenvalue e_j of the massless Jacobian $\mathbf{J}$ leads to two eigenvalues $f_{j,1}$, $f_{j,2}$ of the 4D inertial Jacobian $\tilde{\mathbf{J}}$. Thereby, $Re(f_{j,1}) \leq Re(f_{j,2})$ and $Re(f_{j,1}) < 0$.

as shown in Appendix C. Thus, the locations at which the sign flips describe a parabola in the complex plane. The remaining question is when $f_{j,k}$ have imaginary parts, i.e., indicate a swirling behavior. Eq. (12.4) gives that for $Im(f_{j,k}) = 0$ the radicand must be real and non-negative. This gives

$$Im(f_{j,k}) = 0 \iff Re(e_j) \geq -\frac{1}{4r} \text{ and } Im(e_j) = 0 \tag{12.8}$$

for $k \in \{1, 2\}$. Fig. 12.2 illustrates the classification of $f_{j,1}, f_{j,2}$ depending on e_j in the complex plane. From this and the fact that e_1 and e_2 are either real-valued (on the abscissa) or complex conjugates, we get a complete geometric classification of all possible 4D inertial critical points, as shown in Table 12.1. There, the left-hand side shows all possible 2D critical points with the following coding for the eigenvalues: $+/-/\circ$ means positive/negative/zero real part, while i symbolizes the presence of an imaginary part. The right-hand side depicts all possible 4D critical points that can be deduced from the 2D points. Table 12.1 also shows the glyphs that we designed for each of the 2D and 4D critical points, respectively. (The glyph design of 2D critical points follows the common choices in the literature [Wei08].) Attracting nodes and foci in 4D have an empty interior that is used later for color-coding an additional property. Fig. 12.3 contains an extended classification overview of the inertial critical points, including illustrations of the eigenvalues in the complex plane. Table 12.1 reveals a number of observations of the behavior of the inertial critical points of $\tilde{\mathbf{u}}$:

- $\tilde{\mathbf{u}}$ does not have sources (repelling critical points). All critical points have at least two eigenvalues with negative real parts. Thus, during backward integration an inertial particle cannot converge to a critical point, which explains topologically the strong repelling behavior of inertial backward integration.
- The 4D counterpart of a 2D saddle is always a 1/3 saddle, i.e., a saddle with 1 repelling direction and 3 eigenvalues denoting attracting behavior. This observation is consistent with findings in related CFD literature [HS08], namely that saddles stay saddles, though there eigenvalues have not been classified so far.
- For $\mathbf{g} = \mathbf{0}$, every 4D sink corresponds to a 2D sink, but not reversely: a 2D

massless CP (2D)		→		inertial CP (4D)
rep. node	$\begin{matrix}+ & + \\ & \end{matrix}$	→	$\begin{matrix}+ & - & + & - \\ & & & \end{matrix}$	2/2 saddle
rep. focus	$\begin{matrix}+ & + \\ i & i \end{matrix}$	→	$\begin{matrix}+ & - & + & - \\ i & i & i & i \end{matrix}$	2/2 saddle
center	$\begin{matrix}\circ & \circ \\ i & i \end{matrix}$	→	$\begin{matrix}+ & - & + & - \\ i & i & i & i \end{matrix}$	2/2 saddle
attr. focus	$\begin{matrix}- & - \\ i & i \end{matrix}$	→	$\begin{matrix}+ & - & + & - \\ i & i & i & i \end{matrix}$	2/2 saddle
		→	$\begin{matrix}- & - & - & - \\ i & i & i & i \end{matrix}$	attr. focus
attr. node	$- -$	→	$\begin{matrix}- & - & - & - \\ i & i & i & i \end{matrix}$	attr. focus
		→	$\begin{matrix}- & - & - & - \\ & & i & i \end{matrix}$	attr. focus
		→	$- - - -$	attr. node
saddle	$+-$	→	$+ - - -$	1/3 saddle
		→	$\begin{matrix}+ & - & - & - \\ & & i & i \end{matrix}$	1/3 saddle

Table 12.1.: Full classification of first-order inertial critical points in 2D flows. Left: shows for all possible 2D critical points whether eigenvalues have positive/negative/zero real part ($+/-/\circ$), and whether eigenvalues are complex (i). Right: all possible 4D critical points that can be deduced from the 2D points. Zags at glyph boundaries denote the presence of 2 or 4 complex eigenvalues, respectively.

attracting focus corresponds in 4D to either a sink or a 2/2 saddle. An example of the latter is shown later in Fig. 12.4.

12.1.2. Inertial Separatrices

Similar to the massless case (cf. Section 9.3.1), asymptotic behavior can be studied for inertial particles. All locations in the spatio-velocity domain $D \subseteq \mathbb{R}^2 \times \mathbb{R}^2$ that reach the same isolated first-order critical point $\mathbf{p} \in D$ after infinite forward integration form a so-called ω-basin, which in dynamical systems is frequently referred to as stable set of $\mathbf{p}$:

$$\omega(\mathbf{p}) = \left\{ \begin{pmatrix} \mathbf{x}_q \\ \mathbf{v}_q \end{pmatrix} \in D : \phi^\tau(\mathbf{x}_q, r, \mathbf{v}_q) \to \mathbf{p} \text{ as } \tau \to \infty \right\} \tag{12.9}$$

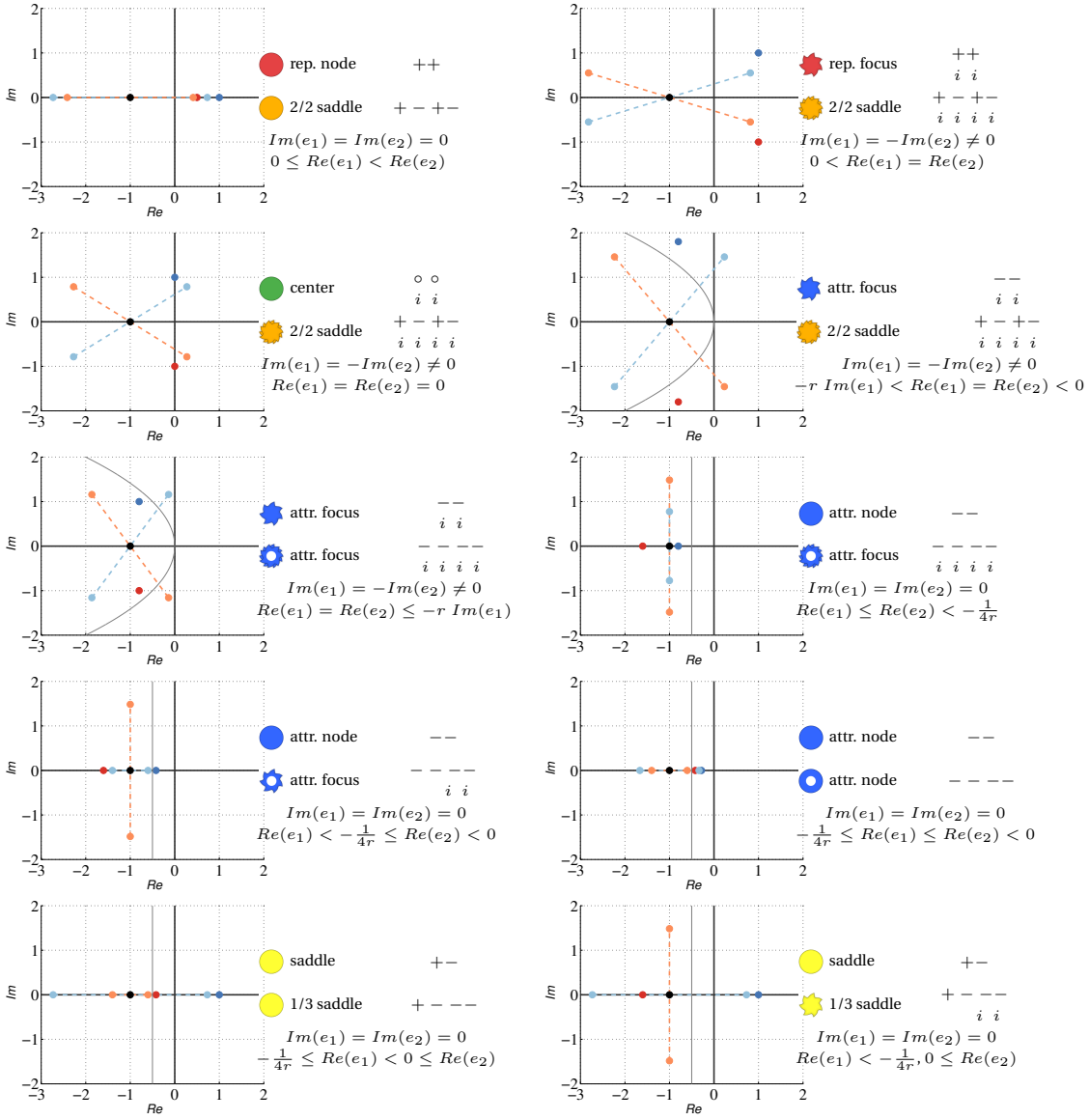


Figure 12.3.: Classification of inertial critical points based on the eigenvalues e_1, e_2 of $\mathbf{J}$. W.l.o.g., we assume that $Re(e_1) \leq Re(e_2)$. The eigenvalue e_1 (•) belongs to the eigenvalues $f_{1,1}, f_{1,2}$ (•) of $\tilde{\mathbf{J}}$, and the eigenvalue e_2 (•) belongs to the eigenvalues $f_{2,1}, f_{2,2}$ (•). Each pair of eigenvalues $f_{i,1}, f_{i,2}$ is located diametrically opposite around the real-valued constant center $-1/(2r)$ (•), here shown for $r = 0.5$. For each possible case, we show the eigenvalues in the complex plane, the type of critical point in the massless and inertial case, and the eigenvalue conditions.

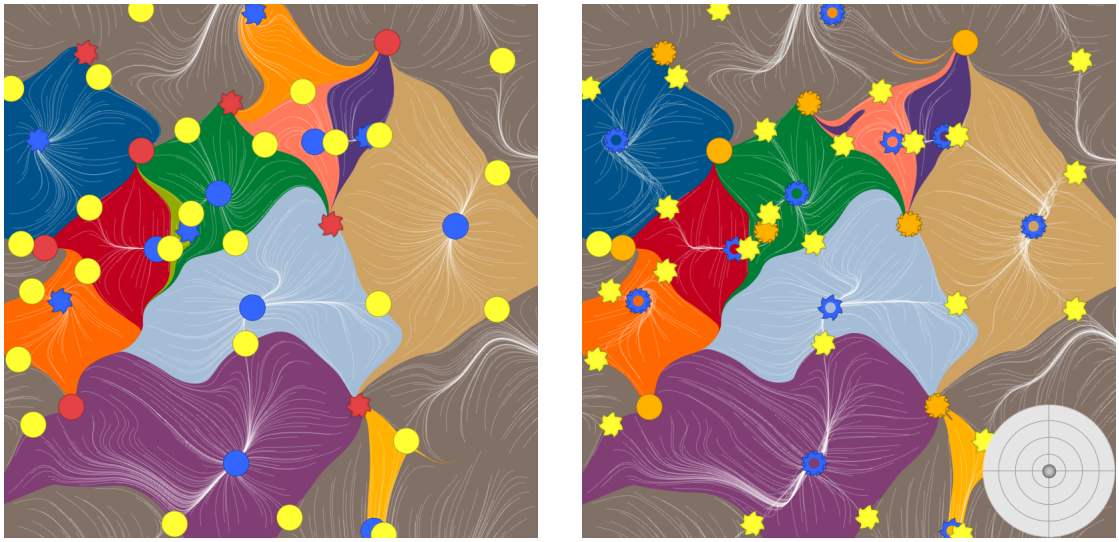


Figure 12.4.: ω-basins of the underlying flow field $\mathbf{u}(\mathbf{x})$ (left) and projections of the 4D ω-basins of inertial flow onto the spatial subspace for $d_p = 75\,\mu m$ (right). Note that in the projection, the basins of inertial flow may consist of multiple pieces and that in this example an attracting focus turned into a 2/2 saddle.

Since there are no sources in the inertial flow, backward integration has zero probability to terminate in a critical point.

We are interested in the separatrices of the 4D inertial flow. They are 3-manifolds starting at 1/3 saddles which follows from a direct generalization from the 2D and 3D case: in 2D, separatrices are 1-manifolds (curves) starting from 1/1 saddles, while in 3D the separatrices are 2-manifolds starting from 1/2 saddles.

In 4D, the 2/2 saddles also indicate a separation but are topologically of less interest because they only separate the particles converging towards the saddle (building a 2-manifold, i.e., a random particle has a zero probability to be on this manifold) vs. particles passing by the saddle (an example for this is in Fig. 9.4). While the ω-basins are connected components in 4D, their projection to the 2D spatial space can consist of several unconnected components. Fig. 12.4 compares the resulting ω-basins of the underlying 2D flow $\mathbf{u}(\mathbf{x})$ with the projections of 4D basins of the inertial flow with initial velocity $\mathbf{v}_0 = \mathbf{0}$ for a gravity-free environment. Aside from response time r and gravity $\mathbf{g}$, the 4D basins depend on the initial velocity $\mathbf{v}_0$, as demonstrated in Fig. 12.5. We would like to capture the impact of $\mathbf{v}_0$ in a single visualization, hence we design a glyph that is explained in the next section.

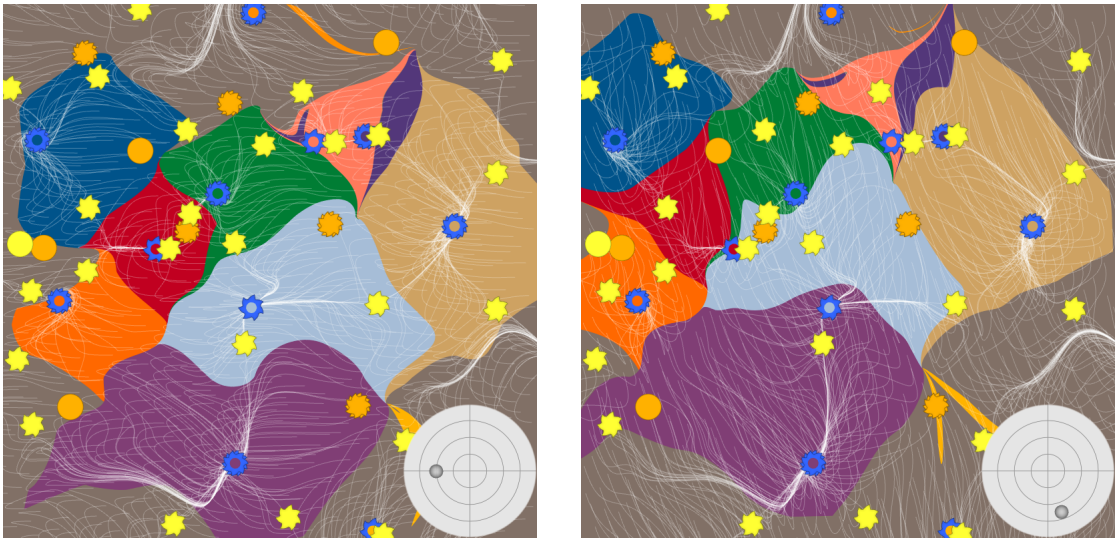


Figure 12.5.: Projections of the 4D ω-basins of inertial flow onto the spatial subspace for $d_p = 75\,\mu m$, $v_{max} = 10\,m/s$ and different initial velocities. Left: $\mathbf{v}_0 = (-5,\, 0)^{\mathrm{T}}$ and right: $\mathbf{v}_0 = (2,\, -6)^{\mathrm{T}}$.

12.2. Glyph-based Visualization of Asymptotic Behavior

We set out to solve the following problem: For every possible location $\mathbf{x}_0$ in the domain: show for every possible initial velocity $\mathbf{v}_0$ (up to a certain maximal speed), which sink an inertial particle enters that is released at $(\mathbf{x}_0,\, \mathbf{v}_0)$. Essentially, we need to visualize a 4D scalar field (2D position + 2D velocity) that is computed by particle integration. Even for a moderate discretization, a full precomputation is not feasible, since that would exhaust too much computation time and memory.

Glyph Design. To avoid this problem, we design an interactive glyph visualization that is rendered progressively and adaptive to the current view. Glyphs are frequently applied in flow visualization, e.g., in [HLNW11, HSJW14]. See the recent reports on glyph design by Ward [War08] and Borgo et al. [BKC*13] for an overview. The idea of our glyph is as follows: Every glyph represents a single spatial position $\mathbf{x}_0$. The glyph is placed at its respective spatial position and we densely cover the domain with glyphs to obtain a discrete sampling of the spatial domain. With this, we limit the computation to this set of discrete spatial points, which reduces the 4D domain considerably.

Each glyph is disc-shaped and represents the $\mathbf{v}_0$ plane. Thereby, each pixel of a glyph represents a discrete sample of the velocity domain. The angle represents the direction and the distance to the center the magnitude, up to some upper bound v_{max}, which can be adjusted interactively. The glyph concept is illustrated in Fig. 12.6. Each glyph covers

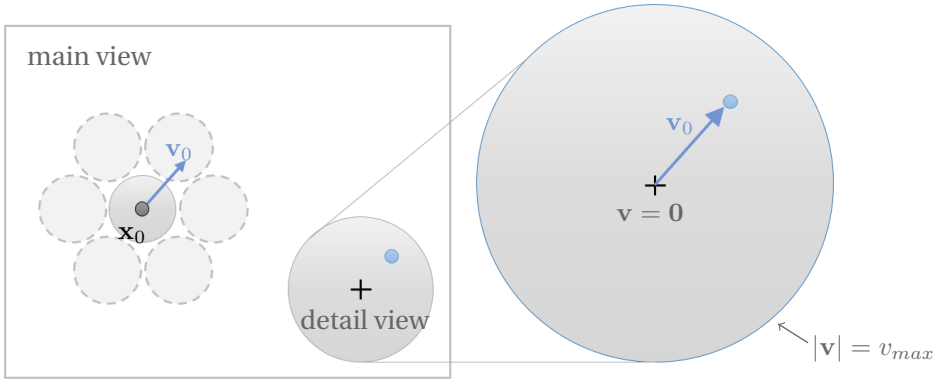


Figure 12.6.: Visualization concept. The location of a glyph in the domain corresponds to an initial position $\mathbf{x}_0$. A point inside a glyph corresponds to an initial velocity $\mathbf{v}_0$. Thereby, the glyph center corresponds to $\mathbf{v} = \mathbf{0}$ and the boundary has $|\mathbf{v}| = v_{max}$.

only a certain number of pixels, which reduces the set of velocities to consider again considerably. We progressively add additional samples in every pixel for anti-aliasing purposes over the course of multiple frames, but a rather good preview is obtained with one sample already. The color of a glyph pixel encodes the sink that is reached after integration from $\mathbf{x}_0$ with initial velocity $\mathbf{v}_0$. As color table, we used Kelly's colors of maximum contrast [GA10]. The size of the glyphs is specified in screen space and is interactively adjustable. In the examples throughout the chapter, we set a typical glyph radius of 15–20 pixels. We layout the glyphs on hexagonal grids, which delivers the densest possible packing and reaches 90% screen space coverage. We set a glyph spacing of one pixel to make the circles easier distinguishable.

Detail View. To support the exploration process, the user can select a glyph, for which a close-up is presented in the bottom-right corner of the screen. In this detail view, seed points of inertial trajectories can be added and removed by clicking on the close-up. The clicked coordinate determines the initial velocity $\mathbf{v}_0$, and the selected glyph itself represents the initial position $\mathbf{x}_0$. From there, inertial trajectories are integrated and displayed in the main view. Fig. 12.7 gives an example. In addition, hovering over a seed point in the detail view highlights the corresponding trajectory in the main view.

12.3. Implementation

In order to enable interactive response in the exploration of the 4D space of possible initial conditions, we devise an efficient GPU-based computation of the glyph visualization. In the following, we elaborate on the memory layout and the progressive computation under constraint time budgets per frame.

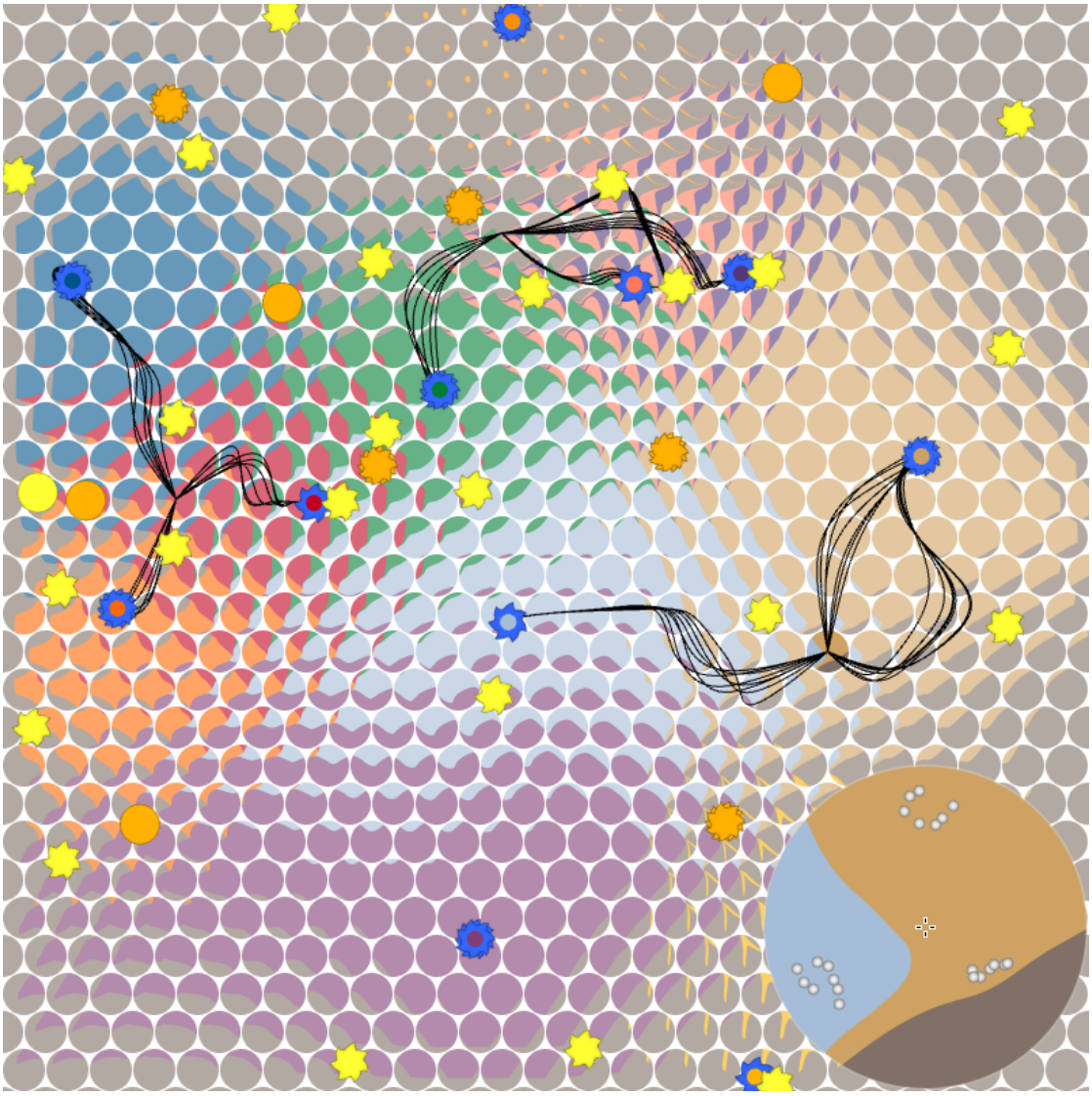


Figure 12.7.: Glyphs show the asymptotic behavior of inertial particles that were released with $d_p = 75\,\mu m$, $v_{max} = 10\,m/s$, and varying $\mathbf{x}_0$ and $\mathbf{v}_0$. Note that sink glyphs contain the color of the corresponding ω-basin. The user is provided detail views of selected glyphs in which seed points of inertial particle trajectories can be specified.

12.3.1. Glyph Memory Layout and Camera Navigation

We begin with glyph data layout on the GPU, and its update during camera navigation.

Data per Glyph Pixel. Each glyph covers a certain number of pixels and for each of those pixels, we progressively generate subsamples that correspond to initial conditions $(\mathbf{x}_0, \mathbf{v}_0)$. We evaluate their color based on the classification of the sink that they reach in the limit, and progressively accumulate the evaluated colors. Thus, each pixel stores a floating point vector with four components, containing the accumulated color (RGB) and the number of samples (A) generated so far. If a subsample is outside the glyph, the color evaluates to the background color in order to anti-alias the glyph boundary.

Texture Atlas and Glyph Data. We maintain a memory pool of glyphs that is organized as a texture atlas. Thereby, each entry stores the aforementioned glyph pixel data of one glyph. In addition, we store in a buffer for each glyph its spatial position in the domain ($\mathbf{x}_0$), the location of its contiguous pixel data in the texture atlas and a counter that accounts for the number of pixels that have been processed in the glyph so far. During rendering, we simply place the portion of the texture atlas that belongs to a glyph on the screen at the respective domain location. Note that the texture atlas needs to be recomputed whenever the size of the glyphs or a simulation parameter (particle diameter, particle density, viscosity or gravity) is varied by the user.

Camera Navigation. Interaction requires a swift navigation of the camera, including panning and zooming. During panning, we move the camera pixel-wise, so that a pixel in the texture atlas always covers a full pixel on the viewport. This allows to make use of the full potential of the texture atlas, because the accumulated pixel data of the glyphs can be reused in the new view. Glyphs that leave the viewport are freed and their texture atlas space can be assigned to newly appearing glyphs. (We compute the placement of glyphs on the CPU and map the updated glyph data to the GPU.) During zooming, we maintain the size of the glyphs. In order to preserve the dense covering we scale the underlying coordinates. Zooming requires to reset the texture atlas, since the initial position $\mathbf{x}_0$ of each glyph changes.

12.3.2. Progressive Computation

In the following, we explain how the progressive computation can be efficiently deferred over multiple frames, and how previews can be generated.

Processing Order. As the integration-based evaluation of an initial condition is computationally expensive, only a certain number of glyph pixels can be processed each frame to maintain interactive response. The order in which the pixels of a glyph are processed is pre-determined and stored in a randomized pixel permutation vector, in order to obtain a roughly uniform sampling of the glyph. (The permutation repeats when all pixels were processed to progressively add further samples for anti-aliasing.) The procedure for each GPU thread goes as follows:

- Select the glyph (determines $\mathbf{x}_0$) and its next pixel to process (lookup in pixel permutation vector).
- Take a low-discrepancy (quasi-)random sample inside the pixel to progressively anti-alias the pixel. The location is determined via Halton sequences on the CPU side and is for all pixels that are processed in an iteration the same.
- Convert the subpixel location $\mathbf{p}_s$ in the disk to the initial velocity as $\mathbf{v}_0 = v_{max} \cdot (\mathbf{p}_s - \mathbf{p}_d)/r_d$, with $\mathbf{p}_d$ being the center of the velocity disc and r_d its radius.
- Integrate an inertial trajectory until it enters a sink (4D fourth-order Runge Kutta) and lookup the color of the sink. If a trajectory leaves the domain or does not terminate in a critical point, the kernel evaluates to gray.

Hole Filling. Since only a certain number of pixels can be processed each frame, we need to fill the holes to generate adequate previews. In this work, we implemented a Shepard interpolation among the already processed pixels on the GPU. For this, we dispatch one thread group per glyph and load the already processed pixels in blocks to shared memory, iterate block by block the pending pixels and sum up the weighted distances to the processed pixels in shared memory. This method is simple and fast enough. Alternative approaches would be multi-scale push-pull, (truncated) Gaussian kernels or anisotropic diffusion (possibly heuristically in direction of the flow). A more sophisticated and general sampling and reconstruction framework was described by Frey et al. [FSME14]. Fig. 12.8 demonstrates results of our hole filling. In this example, an iteration took $29\,ms$ with additional $6\,ms$ on hole filling (on average). Here, after 6 frames each pixel was processed once. The following frames add further subsamples for anti-aliasing.

Detail View. In the detail view, the number of pixels is small enough to process each pixel in the first iteration, which leaves no holes. Afterwards, we process in each frame only a fraction of the pixels, which adds further subsamples for anti-aliasing. We found that the extra time for generating more samples in the detail view is better spent on filling holes in the main view. The time it takes to process a pixel depends on the data

1	2	3	6	10	15	reference
35 ms	70 ms	105 ms	190 ms	298 ms	432 ms	

Figure 12.8.: Close-up on the selected glyph in Fig. 12.10a. Results of the hole filling after a certain number of frames and total passed time (below the images). Here, for a glyph with a radius of 25 pixels (2133 pixels in total), we process 400 pixels per frame.

set (average time to reach a sink) and the integration step size. In our experience it typically takes less than 1.5 seconds to generate a sample for every glyph pixel on the viewport. Further timings are listed later in Section 12.4.1.

Additional Query Type. Figs. 12.4 and 12.5 illustrate an additional query type, in which we select one certain $\mathbf{v}_0$. In this case, no glyphs are needed and we compute for every pixel on the screen, which sink is reached in the limit, when releasing an inertial particle at the pixel. In contrast to the glyph visualization, here, a pixel depicts the spatial domain. Again, we progressively generate more samples in each pixel to fight aliasing. Technically, we split the domain into tiles of 8×8 pixels, determine a pixel processing permutation, and update every iteration a certain number of pixels. In the hole filling step, we perform the Shepard interpolation per tile.

12.4. Results

We applied our glyph-based visualization to several synthetic and real-world data sets. The real-world data sets also serve as synthetic testing ground, in which we assume that the equations of motion hold. Generally, we assume all domain sizes to be in meters and time in seconds. For all examples, we used as particle density ρ_p the density of dry sand, i.e., $\rho_p = 1600\,kg/m^3$. If not mentioned otherwise, we assume gravity-free environments. The diameter d_p varies between $d_p = 70\,\mu m$ and $d_p = 200\,\mu m$. The surrounding medium was assumed to be air, thus the viscosity was set to $\mu = 1.532{\cdot}10^{-5}\,kg/(m{\cdot}s)$. Given these parameters, response time r is in our experiments in the range $r \in [0.028,\, 0.232]$.

While the extraction and classification of inertial critical points is tailored to the underlying equations of motion, our glyph-based visualization of asymptotic behavior is independent of the equations of motion, as it is based on the abstract inertial flow

map. This means, the glyph concept can be applied to any particle model.

Borromean Rings. Fig. 12.1 (left) shows a slice of the 3D BORROMEAN RINGS sequence from Section 3.4. It contains a simulation of a magnetic field with field lines initially interlocked in the shape of Borromean rings. Over time the rings decay and topologically reconnect to two smaller magnetic knots that drift apart. For the purpose of generating test data, we selected one slice that contains both swirling and non-swirling inertial critical points, and show the trajectories of several manually seeded inertial particles (black).

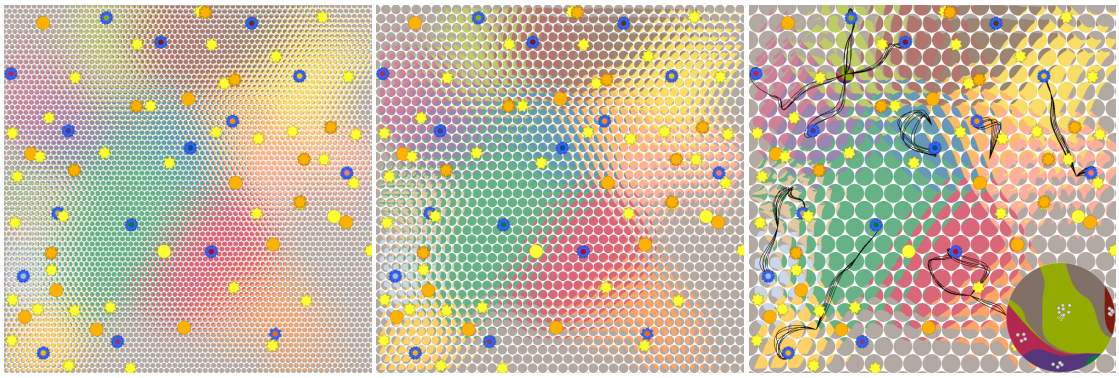
Forced-Damped Duffing Oscillator. The FORCED-DAMPED DUFFING oscillator is a dynamical system that experiences chaotic behavior. Its analytic expression was given in Section 11.4. The system served as test data for studying the separation of inertial particles in terms of inertial FTLE (separation due to spatial variation) [HS11] and in terms of FTMS (separation due to mass variation) in Chapter 11. As we observe steady vector fields, we select a particular time slice $t = 0$ in the spatial domain $[-2,\, 2]^2$. As shown in Fig. 12.1 (middle), the flow contains two attracting foci, and a non-swirling saddle. Dependent on the initial position and initial velocity, an inertial particle reaches one focus or the other. In this example, we released seven inertial particles from the same initial position, but with varying initial velocity. Their initial velocity is depicted in the glyph detail view, and their trajectories are drawn in the main view. The glyph tells for the respective initial conditions the asymptotically coherent regions distinctly apart.

Nine Critical Points. The NINE CP data set is an analytic 2D flow that contains nine non-swirling critical points: four sinks at $(\pm 1, \pm 1)$, four saddles between them at $(0, \pm 1)$ and $(\pm 1, 0)$, and a source at $(0, 0)$. The flow is given as:

$$\mathbf{u}(x, y) = \begin{pmatrix} x\,(1-x)(1+x) \\ y\,(1-y)(1+y) \end{pmatrix} \tag{12.10}$$

and we observe it in the domain $[-1.5,\, 1.5]^2$. For $\mathbf{g} = \mathbf{0}$, the inertial critical points stay at the same locations, but their type changes, as shown in Fig. 12.1 (right). The domain contains four ω-basins, in which we released several inertial particles at the same location but with varying initial velocity.

Random Flows. We constructed several conservative flows as gradient vector fields of a random scalar field. For this, we placed random scalars in $[-1,\, 1]$ on a 12×12 (Figs. 12.4, 12.5, 12.7) or 14×14 (Fig. 12.9) regular grid in the spatial domain $[-1,\, 1]^2$,



(a) 5313 glyphs with a radius of $r = 5$ pixel.

(b) 1694 glyphs with a radius of $r = 10$ pixel.

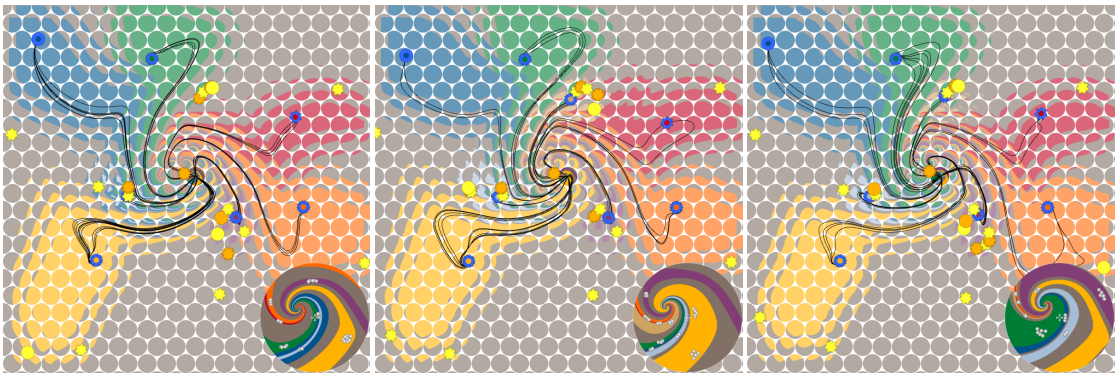
(c) 559 glyphs with a radius of $r = 20$ pixel.

Figure 12.9.: Another random data set, containing the various types of inertial critical points. Here, shown for varying glyph sizes, $d_p = 70\,\mu m$ and $v_{max} = 10\,m/s$. Several inertial trajectories were interactively placed in the right image.

and bilinearly interpolated values in between. The flows exhibit a variety of critical points and serve as explanatory examples, including a showcase for varying glyph sizes in Fig. 12.9.

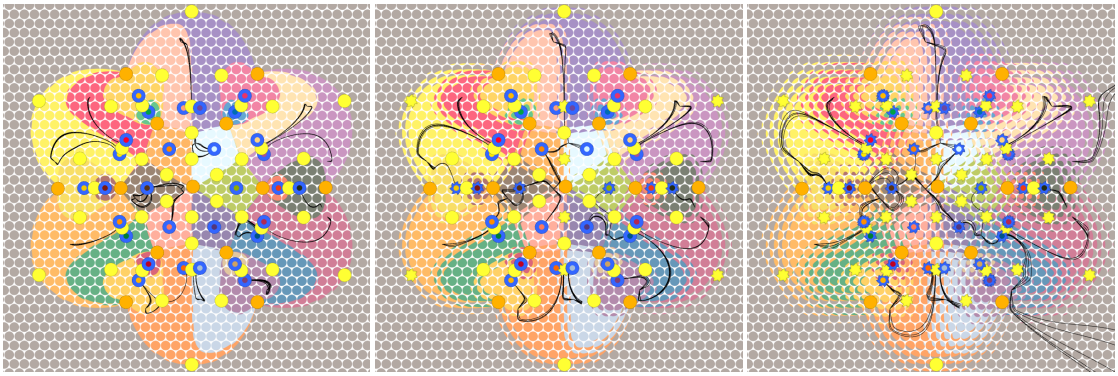
Trefoil Knot. The TREFOIL KNOT data set is another 3D magnetic field and is courtesy of Candelaresi and Brandenburg [CB11]. It contains three interlocked magnetic rings that decay over time. For demonstration purposes, we selected a 2D slice that shows swirling patterns. It contains seven swirling and non-swirling attracting critical points and their respective ω-basins are winding in the middle of the domain around a swirling saddle. In Fig. 12.10, we seeded inertial trajectories at a glyph near the aforementioned swirling saddle, and examine the influence of the gravity vector. We selected initial velocities in the detail view of the selected glyph, which eventually reach their respective attracting node or focus.

Benzene Molecule. The BENZENE data set contains the central z-slice of the analytic approximation of a 3D potential field of a benzene molecule [ZSH96]. The vector field is shown in Fig. 12.11 for varying d_p and contains a large number of symmetrical inertial critical points. For reference, we placed inertial particle trajectories in the domain, and it can be seen that the inertia has great influence on their paths. With increasing diameter, more inertial critical points exhibit swirling behavior, and the stronger is the impact of the initial velocity on the critical point that is reached in the limit.



(a) $\mathbf{g} = (0,0)^{\mathrm{T}}$: 1 , 1 , 5 , 3 , 10 , 1 , 5 (b) $\mathbf{g} = (0,-1)^{\mathrm{T}}$: 1 , 1 , 6 , 2 , 12 , 0 , 6 (c) $\mathbf{g} = (-1,0)^{\mathrm{T}}$: 1 , 1 , 6 , 3 , 11 , 2 , 4

Figure 12.10.: In the TREFOIL data set, spiral patterns occur in the glyphs, showing that the magnitude of an initial velocity has great impact. Here, shown for varying gravity, with $d_p = 70\,\mu m$ and $v_{max} = 5\,m/s$. Note that a change in gravity causes critical points to move, change their type and appear/disappear.



(a) $d_p = 100\,\mu m$: 30 , 42 , 13 (b) $d_p = 150\,\mu m$: 24 , 6 , , 8 , 13 (c) $d_p = 200\,\mu m$: 24 , 6 6 , 34 , 36 , 13

Figure 12.11.: The BENZENE data set, contains a large number of inertial critical points. Here, shown for varying d_p and $v_{max} = 20\,m/s$. The inertial trajectories (black) were released in all three experiments with the same initial position and initial velocity.

12.4.1. Performance

We used an Intel Core i7-2600K CPU with 3.4 GHz and 24 GB RAM, and an Nvidia GeForce GTX 970 GPU with 4 GB VRAM. All results were rendered at a resolution of 800×800 pixels with $4\times$ MSAA. Table 12.2 reports the numbers of extracted inertial critical points for our test data sets, shows the glyph parameters that we set and lists the computation timings. The glyph parameters include the number of glyphs on the

Data Set	Inertial Critical Points							Glyph Parameters			Computation Time		
	# attr.			# 1/3 sad.		# 2/2 sad.		# glyphs on viewport	# pixels per glyph	# glyph pixels per it.	CP extr. [in ms]	iteration [in ms]	hole filling [in ms]
Borromean, Fig. 12.1	1	0	1	1	3	0	2	907	801	50	0.49	50.6	5.36
Duffing, Fig. 12.1	0	0	2	1	0	0	0	907	801	32	0.47	51.7	6.37
Nine CP, Fig. 12.1	4	0	0	4	0	1	0	907	801	200	0.29	25.1	4.11
Random, Fig. 12.9c	0	1	15	2	38	10	8	559	1385	500	0.72	27.7	6.13
Trefoil, Fig. 12.10a	1	1	5	3	10	1	5	559	1385	400	1.28	27.6	5.90
Benzene, Fig. 12.11b	24	6	0	34	8	13	0	1655	373	373	0.74	20.3	—

Table 12.2.: This table summarizes the extraction results of inertial critical points, reports the glyph parameters that we set (number of glyphs on the viewport, number of pixels per glyph, and number of pixels per glyph that are processed each frame), and lists the computation timings for the extraction of inertial critical points, the time it takes to process one frame and the maximal time needed for the hole filling.

viewport and the number of pixels per glyph. Naturally, the larger the glyphs the less glyphs fit on the viewport. The third entry in the glyph parameters column reports the number of glyph pixels that we process each frame per glyph. This number balances the quality of the preview for performance and was chosen so that roughly $20 - 50$ *fps* were reached. Its setting depends on the data set, mainly on the average time it takes an inertial particle to enter an attracting node or leave the domain, and the chosen numerical step size. In the FORCED DUFFING for example, inertial particles follow a rather long path, compared to the BENZENE data set, in which each glyph pixel could be processed every frame. Here, hole filling was not even necessary. Hole filling takes places until every glyph pixel was processed at least once. In the other data sets, this was achieved after $3 - 25$ frames.

The extraction of inertial critical points takes almost the same time as for the massless case, since only the field $r\,\mathbf{g}$ is added to $\mathbf{u}(\mathbf{x})$ prior to extraction. The extraction is a root finding problem, for which we used a recursive subdivision of the spatial domain up to an accuracy of $1 \cdot 10^{-10}$. The hole filling cost grows with the number of already processed pixels from close to zero up to the value given in the table, as the number of pixels grows that are taken into account during Shepard interpolation. Reconstruction kernels with local support can reduce this cost.

12.4.2. Limitations

We encoded the mapping of sink glyphs to their respective ω-basins by color. The number of easily distinguishable colors is limited and our current table only supports up to 20 colors (excluding black and white) [GA10]. For the BENZENE data set in Fig. 12.11, we repeated the colors at 30% more brightness. Alternatively, textures

could be used to generate further distinguishable patterns. Currently, we use a random permutation to determine the order in which the pixels of a glyph are processed. Frey et al. [FSME14] used a multi-resolution capacity-constrained sample distribution, which could be the basis for a more evenly distributed pixel processing permutation. Flows with thousands or critical points are generally perceptually difficult for all topological methods. Topological simplification addresses this problem, see Chen et al. [CML*07] and the references therein.

12.5. Conclusions

The chapter presented – to the best of our knowledge – the first approach to a topology-based visualization of inertial flows. We conducted a full classification of critical points in 2D steady inertial flow. We studied the relation between eigenvalues of the massless 2D Jacobian and the inertial 4D Jacobian. Among others, we made the observation that in inertial flow, sources cannot exist. This explains the strong repelling behavior of backward integration from a topological point of view, and is consistent with the findings in related CFD literature [HS08]. In order to study the asymptotic behavior of inertial particles, we developed a glyph that encodes for both varying initial position and initial velocity, which sink is reached in the limit. We see potential use of this in general dynamical systems theory. The motion of inertial particles is a subclass of this large area, and studying topologically the possible outcome of initial conditions in general dynamical systems (ODEs) seems very useful.

In future work, we look forward to extensions to 3D, i.e., a full classification of the 6D critical points and the visualization of their separation manifolds. The 3D case poses significant additional challenges concerning the rendering and visual representation of the segmentation. As for an extension to unsteady flows, the same issues arise as in the traditional massless case: due to the limited time range, asymptotic behavior can usually not be observed. As an alternative, finite-time approaches have been proposed, both in the massless and the inertial case. Aside from extensions to more general particle models, we also look forward to adaptations that allow for spatially and temporally varying response times. Both will allow to reach out to further applications. The glyph design can be evaluated by the principles of Borgo et al. [BKC*13]. Finally, there are many more concepts from traditional vector field topology that can be extended to the inertial case, including saddle connectors, boundary switches, closed orbits etc. The extraction of some of these features could be quite challenging, since backward integration toward critical points is practically not possible.

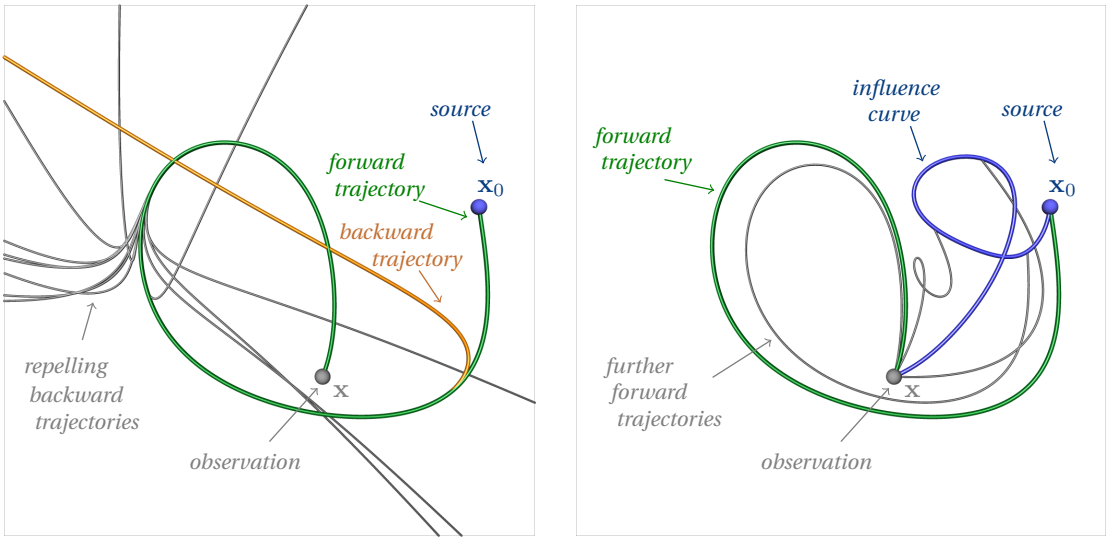
13 Chapter 13. Source Inversion by Forward Integration in Inertial Flows

The following chapter presents an approach to the source inversion problem and is based on the paper:

T. Günther and H. Theisel
Source Inversion by Forward Integration in Inertial Flows
Computer Graphics Forum (Proc. EuroVis) 35, 3 (2016).

Some problems in the area of inertial particle dynamics can be approached by direct extension of methods from the massless case. Other problems and phenomena, however, are unique to inertial flows. A particular example is the so-called *source inversion problem*, which arises in the source recovery of airborne or waterborne pollutions based on the observation of dispersed pollutants. While these are typically modeled along with diffusion processes as in Akçelik et al. [ABG*03], Boano et al. [BRR05], El Badia et al. [BHDH05] or Chow et al. [CKC08], the source inversion problem is nontrivial even when diffusion is neglected. So far – even with this simplification – a solution to the source inversion problem could only be approximated [HS08], and it was not possible to recover the initial velocity with which the pollutant entered the flow.

Given an observation from which a usual backward integration is started, the recovered initial position and velocity of the inertial particle heavily depend on the observed velocity that we started with. In fact, a slight change leads to a completely different initial position. Fig. 13.1 illustrates the problem. Most often the recovered initial velocity is extremely high and therefore implausible. Instead of prescribing the observed position $\mathbf{x}$ and the observed velocity $\mathbf{v}$, the idea of our approach is to prescribe the observed position and an *initial velocity* $\mathbf{v}_0$: we assume that particles can start from any spatial location but with a fixed initial velocity $\mathbf{v}_0$. Such assumption is justified, for instance, when the pollutant is released from rest.



(a) Inertial backward integration exhibits strong repelling behavior.

(b) Our *influence curves* avoid this problem.

Figure 13.1.: Source inversion in left half of the DOUBLE GYRE for $\tau = 9$ in (a): Starting from a source, forward integration takes an inertial particle to an observation. A backward integration, however, repels and does not reach the source (forward and backward curves are for numerical reasons not identical). Slight variations of the observation (in the order of 10^{-8}) lead to sooner repelling behavior. Backward integrated curves exhibit extreme velocities (particles exit the domain on rather straight tracks) and are therefore *implausible*. In (b), our *influence curve* connects all locations that lead during forward integration to the observation, which allows to recover the source.

We formulate the following two problems to be solved in this chapter:

1 Given an observation point $\mathbf{x}$, from which points $\mathbf{c}$ in the spatial domain will an inertial particle integration with initial velocity $\mathbf{v}_0$ starting at time t pass through $\mathbf{x}$ after finite integration time τ?

2 With which velocities $\mathbf{v}$ can such particle pass through $\mathbf{x}$?

We show that for both problems the solutions are one-parametric sets of points (Problem 1) or vectors (Problem 2). In fact, we show that all solutions of Problem 1 form a parametric curve $\mathbf{c}(\tau)$, which we call the *influence curve*. It is the collection of all points $\mathbf{c}$ in the spatial domain that have influence on $\mathbf{x}$ in the sense that inertial particles starting from $\mathbf{c}$ pass through $\mathbf{x}$. The main theoretical contribution of this chapter is to show that *all* influence curves can be described as pathlines of *one* time-dependent vector field $\mathbf{h}(\mathbf{x}, t)$, i.e., they can be extracted by a simple massless integration. Furthermore, the computation of $\mathbf{h}$ is based on forward integrated inertial particle trajectories only. Therefore, the computation is much more stable than a backward integration.

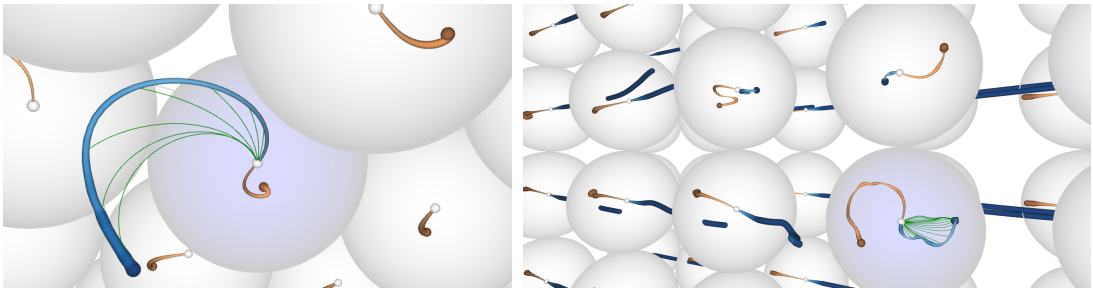


Figure 13.2.: Visualization of the initial positions (blue curves) of inertial particle trajectories (green curves) that lead to the same location (small white sphere). At this location, a glyph is placed that depicts the velocities (orange curves), with which the inertial particles arrive. We solve the source inversion problem for left: TREFOIL and right: WALL-MOUNTED CYLINDER.

The solution of Problem 2 is also a one-parametric family of vectors $\mathbf{v}(\tau)$. This means that for a given observation point $\mathbf{x}$, inertial particles passing through it can have only "a few" possible velocities.

For the visualization, we systematically place observation points in the spatial domain and prescribe a user-defined initial velocity. After extraction, we interactively visualize the influence curves and display the velocities observed at the observation points using glyphs. As inertial particle trajectories converge to attracting manifolds in the spatio-velocity domain [MBZ06, HS08], we additionally encode the rate of change of the observed velocity along the influence curves, which gives a notion for the convergence to the terminal velocity in the attracting manifold. With this, we present the first solution to the source inversion problem that considers actual inertial trajectories. We apply our method to a number of steady and unsteady flows in both 2D and 3D domains. An example is given in Fig. 13.2.

Inertial Flow and Flow Map

In Chapter 9, we described the trajectories of inertial particles as tangent curves of a high-dimensional vector field, which models both the rate of change of particle position $\mathbf{x}$ and particle velocity $\mathbf{v}$. Recalling for an unsteady underlying flow $\mathbf{u}(\mathbf{x}, t)$ in n-D with $n = \{2, 3\}$, the autonomous governing m-D vector field, with $m = 2n + 1$, becomes (cf. Eq. (9.5)):

$$\hat{\mathbf{p}} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\mathbf{x}\\ \mathbf{v}\\ t\end{pmatrix} = \begin{pmatrix}\mathbf{v}\\ \frac{\mathbf{u}(\mathbf{x},t)-\mathbf{v}}{r} + \mathbf{g}\\ 1\end{pmatrix} \text{ with } \begin{pmatrix}\mathbf{x}\\ \mathbf{v}\\ t\end{pmatrix}(0) = \begin{pmatrix}\mathbf{x}_0\\ \mathbf{v}_0\\ t_0\end{pmatrix} \tag{13.1}$$

where $\mathbf{g}$ is a gravity vector (if not mentioned otherwise we set $\mathbf{g} = \mathbf{0}$), and $\mathbf{x}_0$, $\mathbf{v}_0$ and t_0 are the initial particle position, velocity and time. Throughout the following chapter, we set as particle density ρ_p the density of dry sand, i.e., $\rho_p = 1600\,kg/m^3$. The diameter d_p was set to $d_p = 300\,\mu m$ if not mentioned otherwise. The surrounding medium was assumed to be air, thus the viscosity was set to $\mu = 1.532{\cdot}10^{-5}\,kg/(m{\cdot}s)$. Given these parameters, r is in our experiments in the range $r \in [0.058,\ 0.928]$.

In Chapter 10, an inertial n-D flow map ϕ was introduced that maps a given initial condition (position, time, velocity) to the reached spatial location after integration. In this chapter, we require flow map derivatives of the full inertial phase-space, thus we consider the full m-D flow map of $\hat{\mathbf{p}}$ as

$$\hat{\phi}(\mathbf{x}, \mathbf{v}, t, \tau) = \begin{pmatrix} \phi(\mathbf{x}, \mathbf{v}, t, \tau) \\ \psi(\mathbf{x}, \mathbf{v}, t, \tau) \\ t + \tau \end{pmatrix} \tag{13.2}$$

where ϕ denotes the location and ψ the velocity of an inertial particle after integration duration τ when starting the integration at location $\mathbf{x}$ at time t with initial velocity $\mathbf{v}$. The gradient of $\hat{\phi}$ is a $m \times m$ matrix

$$\nabla\hat{\phi}(\mathbf{x}, \mathbf{v}, t, \tau) = \begin{pmatrix} \phi_{\mathbf{x}}(\mathbf{x}, \mathbf{v}, t, \tau) & \phi_{\mathbf{v}}(\mathbf{x}, \mathbf{v}, t, \tau) & \phi_t(\mathbf{x}, \mathbf{v}, t, \tau) \\ \psi_{\mathbf{x}}(\mathbf{x}, \mathbf{v}, t, \tau) & \psi_{\mathbf{v}}(\mathbf{x}, \mathbf{v}, t, \tau) & \psi_t(\mathbf{x}, \mathbf{v}, t, \tau) \\ \mathbf{0}_n^{\mathrm{T}} & \mathbf{0}_n^{\mathrm{T}} & 1 \end{pmatrix} \tag{13.3}$$

where $\phi_{\mathbf{x}}$, $\phi_{\mathbf{v}}$, $\psi_{\mathbf{x}}$, $\psi_{\mathbf{v}}$ are $n \times n$ matrices describing the partial derivatives of $\hat{\phi}$ with respect to $\mathbf{x}$, $\mathbf{v}$, and ϕ_t, ψ_t being the start time partials.

13.1. Influence Curves

In the following, we define influence curves, describe a method to their efficient extraction and propose a visualization that depicts both the curves and the observable velocities.

13.1.1. Definition

We introduce the concept of the *influence curve* of an observation point $\mathbf{x}$ as a curve containing all points from which inertial integration with initial velocity $\mathbf{v}_0$ and start time t_0 ends in $\mathbf{x}$ after a certain integration duration τ. That is, we prescribe the *observed location* and the *initial velocity*. If not mentioned otherwise we set $\mathbf{v}_0 = \mathbf{0}$. We define all influence curves as a family of parametric curves $\mathbf{c}(\mathbf{x}, \tau)$ so that

$$\phi(\mathbf{c}(\mathbf{x}, \tau), \mathbf{v}_0, t_0, \tau) = \mathbf{x} \tag{13.4}$$

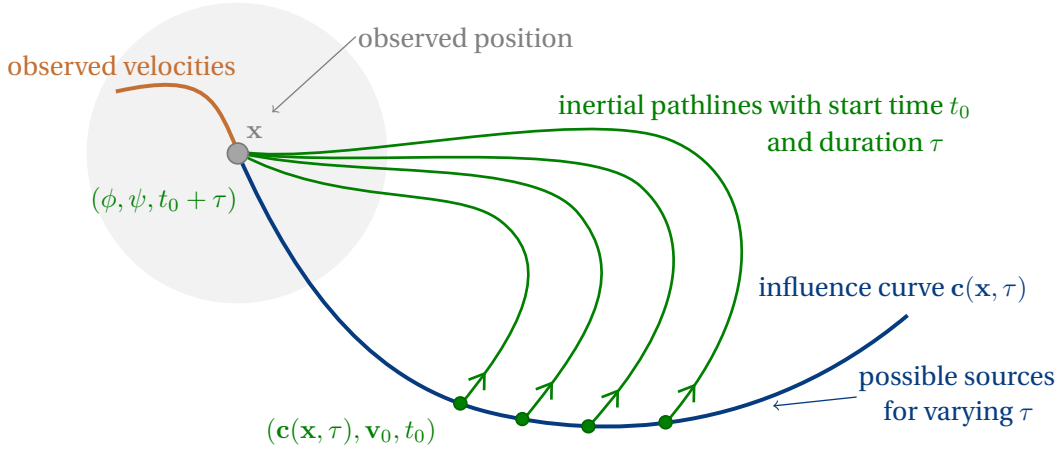


Figure 13.3.: Given an observation $\mathbf{x}$, the influence curve is the union of all locations that lead an inertial pathline to the observation $\mathbf{x}$. The velocity of arriving particles (orange) is shown in a glyph at $\mathbf{x}$. The glyph design is described later in Section 13.1.3.

for all $\mathbf{x}$, curve parameterization τ and the family parameters $\mathbf{v}_0, t_0$. Given a curve $\mathbf{c}$, we can compute the observable velocity of particles arriving at $\mathbf{x}$ as $\psi(\mathbf{c}(\mathbf{x}, \tau),\ \mathbf{v}_0,\ t_0,\ \tau)$ for any integration duration τ. Fig. 13.3 illustrates influence curves.

13.1.2. Extraction

The family of all influence curves can be characterized as pathlines of an n dimensional unsteady vector field $\mathbf{h}(\mathbf{x}, t)$, starting at $t = t_0$:

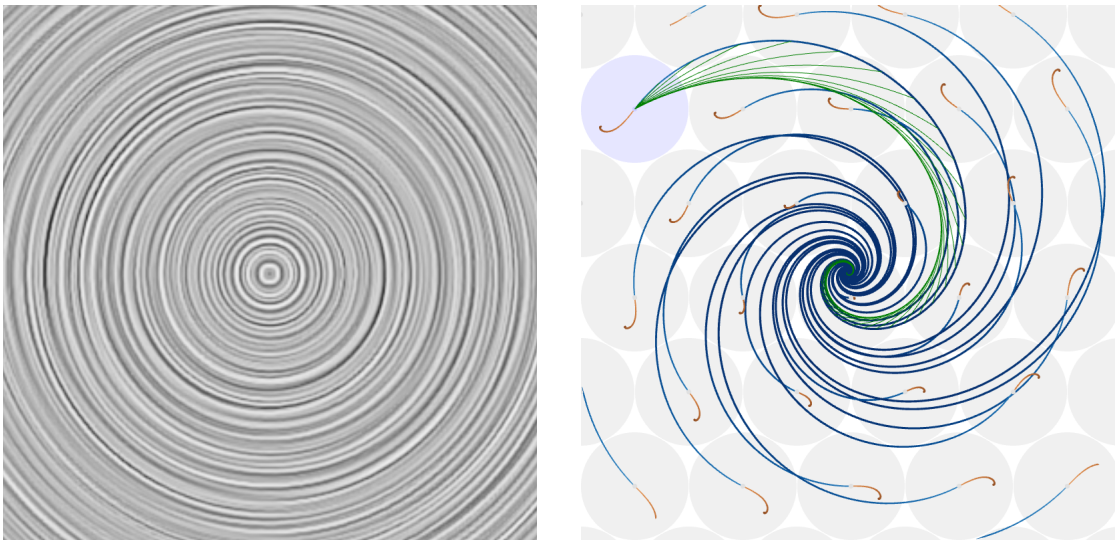
$$\frac{d\,\mathbf{c}}{d\,\tau} = \mathbf{h}(\mathbf{c}(\mathbf{x}, \tau),\ t_0 + \tau). \tag{13.5}$$

Fortunately, vector field $\mathbf{h}$ has a simple form:

$$\mathbf{h}(\mathbf{x}, t) = -\ {\phi_\mathbf{x}}^{-1} \left[\phi_\mathbf{v} \left(\frac{\mathbf{u}(\mathbf{x}, t_0) - \mathbf{v}_0}{r} + \mathbf{g} \right) + \phi_t \right] - \mathbf{v}_0 \tag{13.6}$$

where $\phi_\mathbf{x} = \phi_\mathbf{x}(\mathbf{x}, \mathbf{v}_0, t_0, t - t_0)$, $\phi_\mathbf{v} = \phi_\mathbf{v}(\mathbf{x}, \mathbf{v}_0, t_0, t - t_0)$ and $\phi_t = \phi_t(\mathbf{x}, \mathbf{v}_0, t_0, t - t_0)$ are inertial flow map derivatives that can be computed by forward integration, cf. Eq. (13.3). Note that Eq. (13.6) is the main theoretical contribution of this chapter. See Appendix D for a derivation of Eq. (13.6). With this, influence curves can be extracted via pathline integration in $\mathbf{h}$.

The definition of $\mathbf{h}$ raises another question: How far can we integrate forward in $\mathbf{h}$? Integration in $\mathbf{h}$ stops, when inertial flow maps or the influence curve itself leaves the domain or when the influence curve enters a critical point in $\mathbf{h}$. The latter is only expected for steady underlying flows $\mathbf{u}$. Consider a steady 2D center $\mathbf{u}(x, y) = (-y, x)^\mathrm{T}$ as shown in Fig. 13.4. While massless particles perfectly follow the circular motion,



(a) LIC of $\mathbf{u}(x, y) = (-y, x)^{\mathrm{T}}$.

(b) Influence curves in a center.

Figure 13.4.: Inertial particles in a center flow, showing influence curves, inertial pathlines (seeded from selected curve), as well as velocity glyphs (see Section 13.1.3).

inertial particles spiral outward due to inertia. In fact, inertial particles that move for a longer period of time will be carried further outward. Consequently, influence curves must spiral inward, since all inertial particles released from the influence curve must reach the same observation point. Eventually, influence curves converge toward a critical point of $\mathbf{h}$. An example of this behavior in 3D is shown later in a steady flow in Fig. 13.12.

13.1.3. Visualization

For computing an influence curve via pathline integration in Eq. (13.6), we have to specify an *observation point* in the spatial domain from which the integration in $\mathbf{h}$ starts. In our visualizations, we systematically place the observation points and compute their corresponding influence curves. In 2D, the observation points are laid out in a hexagonal pattern in order to achieve the densest possible packing. In 3D, we decided to place them in a regular grid layout, as this makes the placement of axis-aligned clip planes easier.

We visualize the influence curves in the spatial domain (in blue) for a certain time range $[0, \tau]$, which is reported later in Table 13.1 for all figures. Additionally, we place a glyph at their observation point to depict the velocities with which the particles arrive (in orange). The glyphs have a circle shape in 2D and a sphere shape in 3D. They visualize the observed velocities in polar coordinates, with $\psi = \mathbf{0}$ in the center and

up to a certain user-specified upper velocity magnitude v_{max} at the boundary of the circle/sphere. The velocity of an arriving inertial particle is a point in the glyph and the union of velocities of all arriving particles forms a curve. The 2D case has similarities with the glyph designed in Chapter 12, though with the difference that here a glyph encodes the observed velocity, whereas in Chapter 12 it depicts the initial velocity. In the 3D case, we cull the front faces of the spheres to allow looking inside them.

For both the influence curve in the spatial domain and the corresponding observed velocities in the glyph, we encode the change of the observed velocity with respect to the curve parameter. In particular, we map $1/(1 + \partial\psi/\partial\tau)$ to line width and brightness. The change in the observed velocity gives a notion for the convergence of the particle velocity to the terminal velocity. The thicker and darker a curve segment the more likely has the observed velocity converged to the terminal velocity, since over time convergence gets slower. Note that this is only meant to give a notion for the convergence. Inertial particles may shortly be driven away from attracting (spatio-velocity) manifolds [SH09].

We allow the user to select a glyph by picking. For the selected glyph, we release a user-specified number of inertial pathlines (in green) from the selected influence curve. Then, all trajectories have in common that they reach the observation point. This pathline depiction is related to the eyelet particle tracing of Wiebel and Scheuermann [WS05], as they displayed pathlines that pass through a single point at different times for massless flows in order to capture the unsteady behavior of a time-dependent flow in a single steady image. In some cases, influence curves may clutter the view, especially in unsteady or in 3D flows. Thus, we allow the user to either show the influence curve of the selected glyph only (Fig. 13.8, left) or of all glyphs (Fig. 13.8, right).

13.2. Implementation

In the following, we share insights and observations we made about the numerical integration of the influence curves. Fig. 13.5 gives an example of the influence curve vector field in the DOUBLE GYRE, precomputed at $256 \times 128 \times 128$ in 100 minutes. In this flow, (massless) tracer particles stay inside the domain; inertial particles, however, do not. Due to inertia they might be dragged outside, depending on the particle response time, i.e., diameter, density and viscosity. This means, flow maps might leave the domain, and thus the influence curve field might not be defined at those places. Undefined areas are shown in Fig. 13.5 in black. Thus in these areas, influence curve integration stops and no further sources can be found. Also, influence curves themselves might leave the domain. We further elaborate on source inversion in bounded domains in Section 13.4.2.

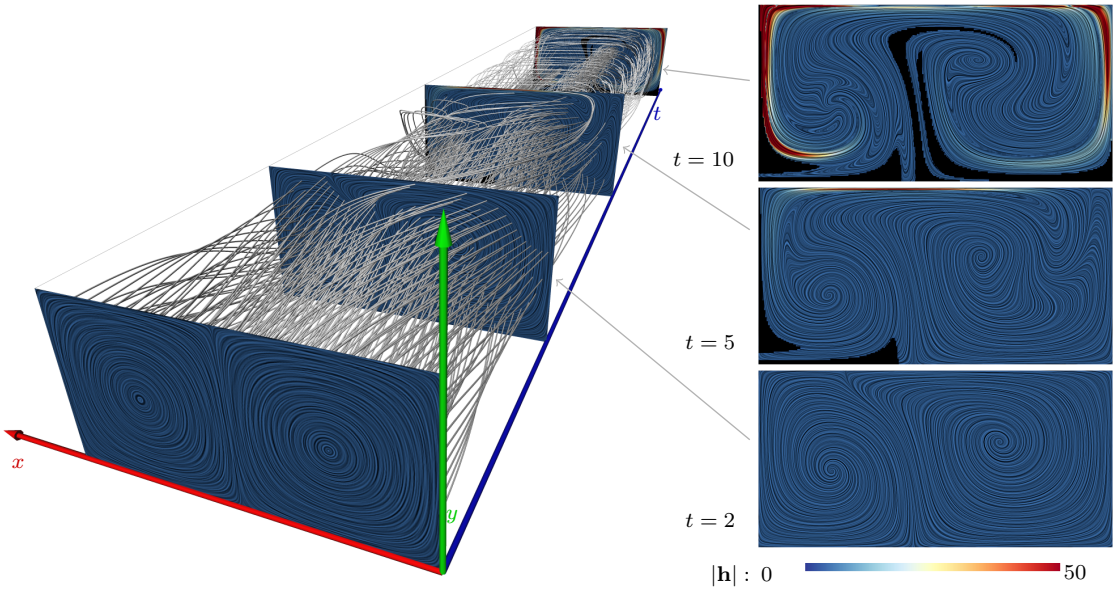


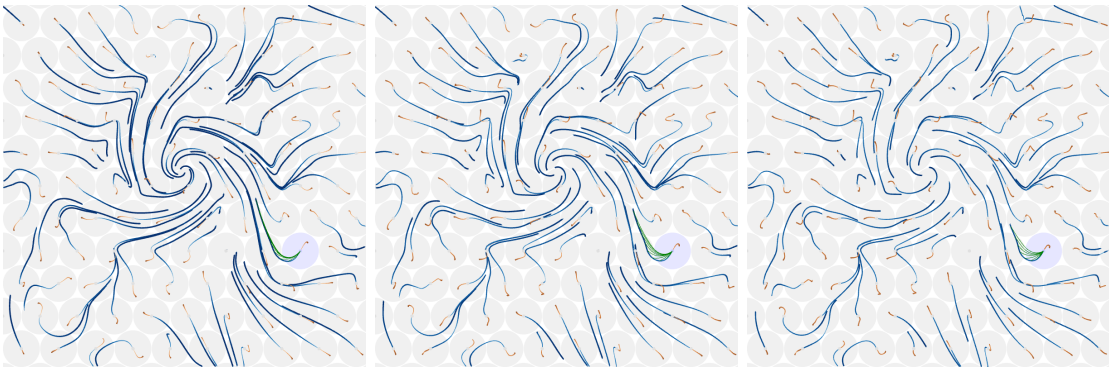
Figure 13.5.: Influence curve vector field $\mathbf{h}(x, y, t)$ in the DOUBLE GYRE. Left: Space-time visualization of the influence curve vector field, containing several pathlines (i.e., influence curves). Right: Selected LIC slices, showing areas where $\mathbf{h}$ is undefined, since particles left the domain (black) and where the magnitude is high (red).

Another observation we made is that even though the DOUBLE GYRE flow has a rather moderate variation in the velocity magnitude, the influence curve field might have places with extremely high magnitude (variation of up to factor 3000). Fig. 13.5 (right) visualizes the magnitude by color-coding. In this example, high magnitudes occurred at later integration times ($\tau > 9$) close to the domain boundary, which causes numerical difficulties. We found that an adaptive integrator is mandatory in order to integrate through these areas. Thus, the influence curve integration in $\mathbf{h}$ is done using the adaptive Runge-Kutta-Fehlberg method (RK45).

If the domain should be densely covered with influence curves, the vector field $\mathbf{h}$ can be precomputed similar to [WT10a]. If only few influence curves are shown, like in our glyph examples, a precomputation of $\mathbf{h}$ for all space-time locations does not pay off.

13.3. Results

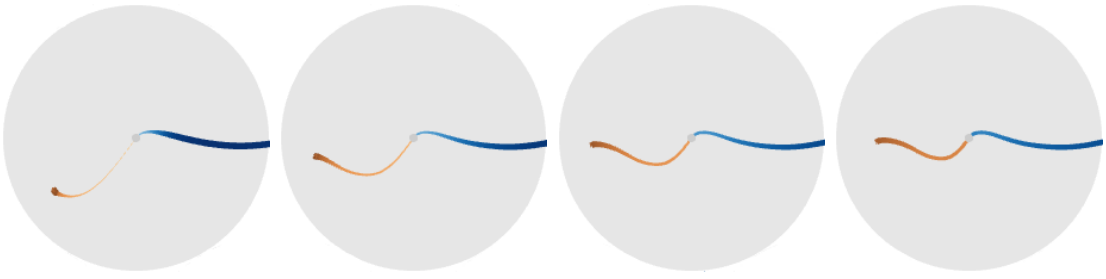
We applied our method to a number of data sets. Note that some of them were studied in related CFD literature [HS11, SBR16] and that the remaining mainly serve as synthetic testing ground for source inversion problems, and that in all flows the spatial units are assumed to be in meters and time in seconds. The visualization parameters (v_{max}, t_0 and τ) are given for all figures later in Table 13.1.



(a) $d_p = 200\,\mu m$ (b) $d_p = 300\,\mu m$ (c) $d_p = 400\,\mu m$

Figure 13.6.: Visualization of all locations (blue curves) that reach the observation points (centers of gray circles). Here, we show the impact of particle size d_p in the TREFOIL 2D data set, together with inertial pathlines (green curves) for a selected glyph.

Trefoil. Our first data set is the TREFOIL from Section 12.4, which contains a simulation of the decay of magnetic rings. In Fig. 13.6, we selected one particular 2D slice (steady, with velocity vectors projected into the 2D slice plane) and computed the influence curves for particles with different diameters. In this example, we make several observations that are in accordance with the behavior that is expected. First, we see that the smaller the particles are, the faster they align with the flow. This can be seen by the thickness and color of the lines, which represents the rate of change in the final velocity. Small particles converge to the terminal velocity faster, which is visible by the "drop shape" of the orange curves in the velocity domain, see Fig. 13.7. For small particles, the orange lines begin very thin and have a noticeable blob at the end, which represents the near-terminal velocity which almost all incoming inertial particles have reached. Further, we see that inertia effects get stronger the larger the



$d_p = 100\,\mu m$ $d_p = 200\,\mu m$ $d_p = 300\,\mu m$ $d_p = 400\,\mu m$

Figure 13.7.: Typically, the observed velocities (orange) are distributed in a characteristic drop shape inside the glyphs.

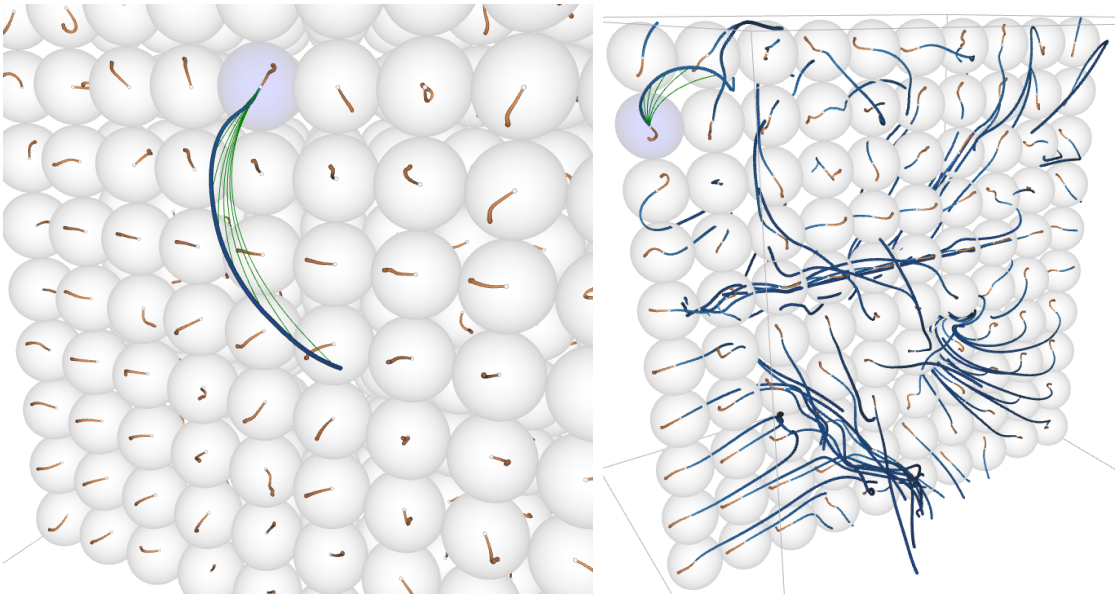
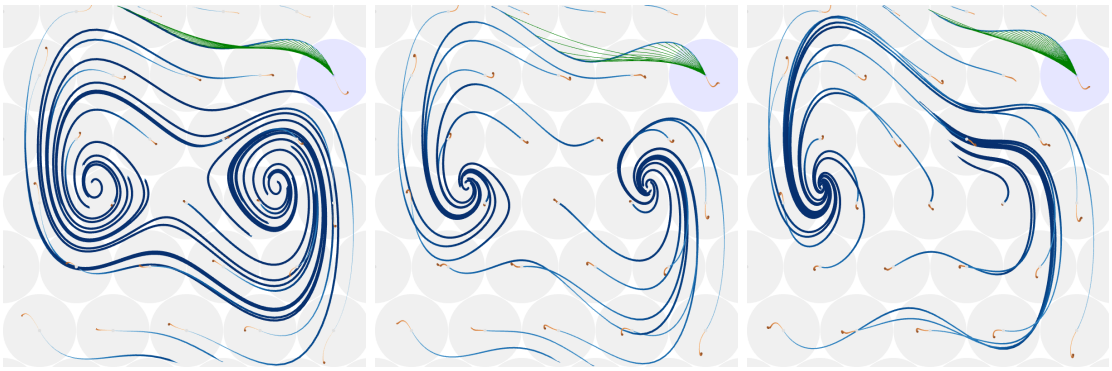


Figure 13.8.: A dense sampling of a 3D domain, such as in the TREFOIL flow, might lead to cluttered influence curves. Left: only one influence curve is shown, right: all are shown.

particles are, as the inertial pathlines (green) deviate stronger from the influence curve (blue). This is expected, since larger particles have a higher response time, thus they take longer to respond to changes in the underlying flow.

In Fig. 13.8, we applied our method to the same data set in the original 3D domain. In the right image, patterns of coherent behavior emerge which relate to the underlying magnetic ring structures. However, line depictions can get cluttered, thus we display in the left image only a selected influence curve. Fig. 13.2 (left) contains a detail view on one of the glyphs.

Forced-Damped Duffing. Next, we used the FORCED-DAMPED DUFFING oscillator from Section 11.4, again, in the spatial domain $D = [-2,\ 2]^2$. Fig. 13.9 depicts influence curves for particles of different size and with varying gravity for the steady slice $t = 0$. It can be seen that influence curves of smaller particles are longer than those of bigger particles. This is due to the smaller response time of small particles. All particles were released from rest and due to its larger response time, a large particle takes longer to accelerate and thus travels in the same time a shorter distance, compared to a smaller particle. An increase in gravity affects the course of inertial particles and thereby the influence curves.



(a) $d_p = 200\,\mu m$, $\mathbf{g} = \mathbf{0}$ (b) $d_p = 400\,\mu m$, $\mathbf{g} = \mathbf{0}$ (c) $d_p = 400\,\mu m$, $\mathbf{g} = (0, -0.5)^{\mathrm{T}}$

Figure 13.9.: Influence curves in the phase space of the FORCED-DAMPED DUFFING oscillator for different particle sizes d_p and gravity g. The larger the particles, the slower their response to changes in the underlying flow. Thus, they take longer to accelerate and as shown by the orange curves (observed velocity), they enter the observation points with less speed than the smaller particles. With increased gravity (downward), influence curves bend upward to accommodate the downward motion of inertial particles.

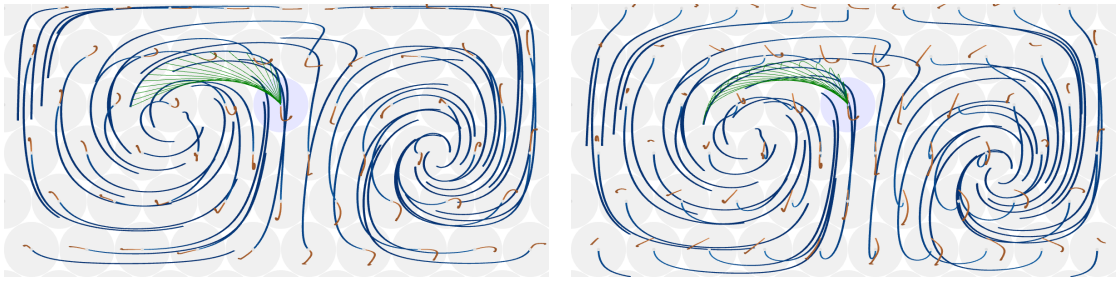


Figure 13.10.: Influence curves for different initial velocities in the DOUBLE GYRE. Left: $\mathbf{v}_0 = (0, 0)^{\mathrm{T}}$, right: $\mathbf{v}_0 = (0, 0.15)^{\mathrm{T}}$. When inertial particles are not released from rest (right image), orange curves (observed velocities) no longer start at the glyph center.

Double Gyre. In Fig. 13.10, we extracted influence curves for different initial velocities in the DOUBLE GYRE flow from Section 7.4. It can be seen that for $\mathbf{v}_0 \neq \mathbf{0}$, the orange line in the velocity glyph (depicting the observed velocities) does not start at the glyph center. The impact of the initial velocity on the particle trajectories can be seen by the green inertial pathlines.

Tornado. Fig. 13.11 depicts the synthetic TORNADO data set from Section 3.4 for two different viewpoints. In 3D domains, glyphs occlude each other. Therefore, we allow the user to place clip planes in order to explore inner structures. In this example, we clipped away some of the glyphs to get a better view. The underlying flow has

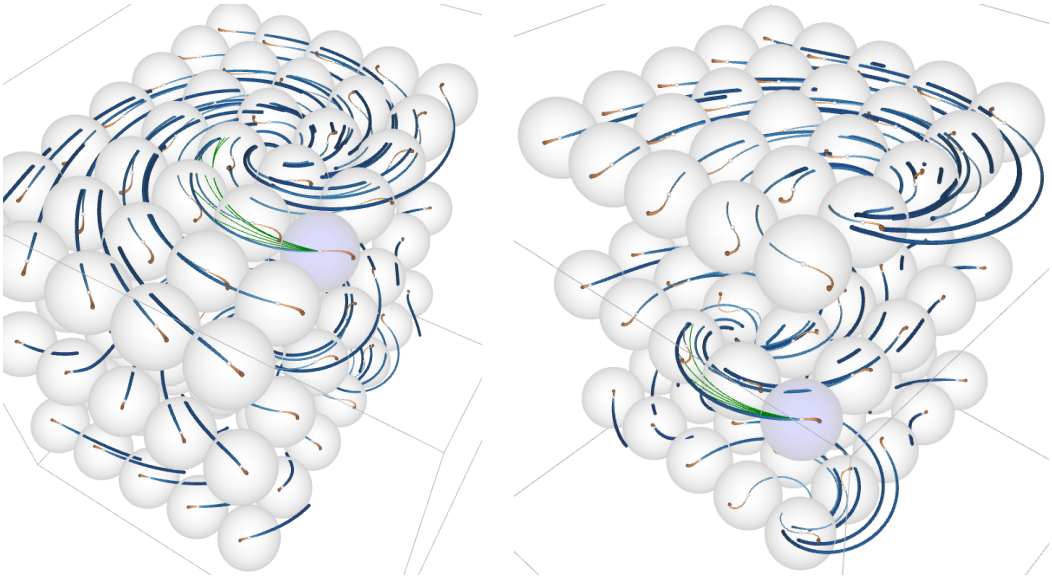


Figure 13.11.: Influence curves in a steady TORNADO flow.

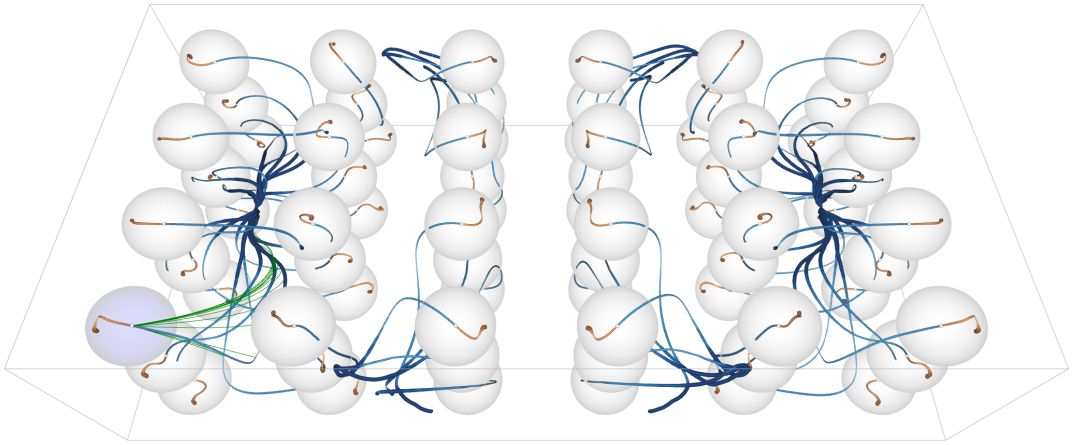


Figure 13.12.: Influence curves in a symmetric RAYLEIGH-BÉNARD convection are separated into convection cells.

a relatively simple structure that is visible in the influence curves (blue) and their observed velocities (orange).

Rayleigh-Bénard. Fig. 13.12 shows the steady RAYLEIGH-BÉNARD convection from Section 3.4, i.e., a thin layer of liquid that is heated from below. In this flow, separate convection cells are formed that rotate. Our visualization captures the symmetry of the domain and it can be seen that at least for the chosen observation points, influence curves are separated into the left and right halve of the domain. With ongoing influence curve integration, the curves reach critical points in $\mathbf{h}$, which are here inside the left-most and right-most convection cells.

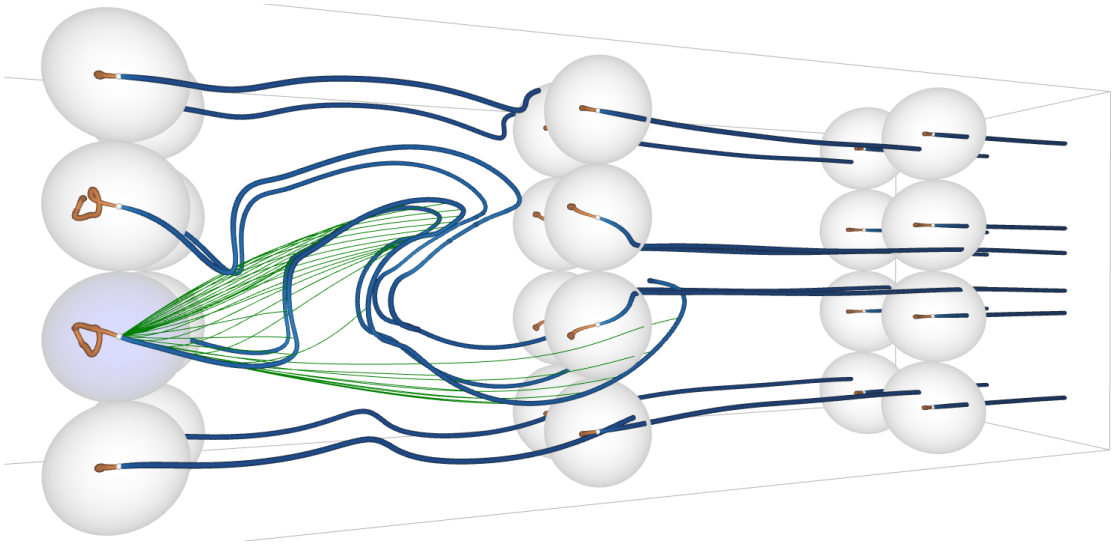


Figure 13.13.: Influence curves (blue) in the unsteady wake of the 3D SQUARE CYLINDER sequence.

Square Cylinder. The SQUARE CYLINDER flow was earlier described in Section 4.4. Fig. 13.13 shows influence curves in the unsteady wake of the square cylinder. The two left-most glyphs in the middle show relatively strong variation in the observed velocity. The display of inertial pathlines (green) confirms the correctness of the extracted influence curves (blue).

Wall-mounted Cylinder. In Fig. 13.14, influence curves are shown in the time-dependent 3D air flow around the WALL-MOUNTED CYLINDER from Section 3.3.2. This flow contains a laminar layer in the background and a turbulent layer in the foreground. In laminar areas, influence curves are rather straight, whereas especially in turbulent areas, observed velocities vary strongly. Still, curves are formed that can be extracted by our method. A close-up on the selected glyph in Fig. 13.14 is shown in Fig. 13.2 (right).

13.4. Discussion

In this section, we discuss the corresponding solution in the massless case, source inversion in bounded domains, an optional predictor-correct extension, the application of influence curves to another problem, and we provide performance measurements.

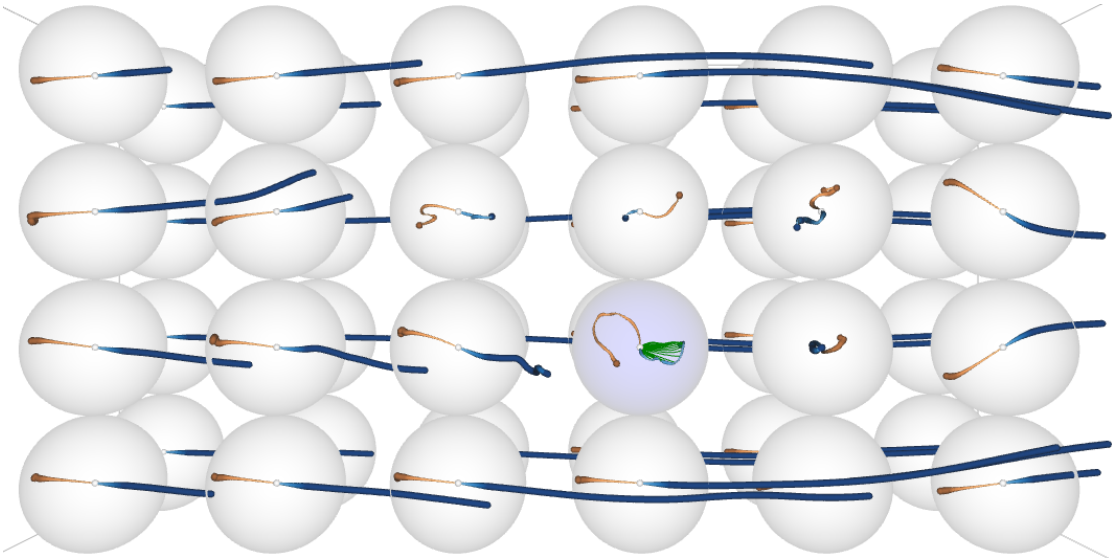


Figure 13.14.: Unsteady flow around WALL-MOUNTED CYLINDER. Behind the obstacle, the flow is slow and turbulent.

13.4.1. Relation to Massless Case

In the massless case, our influence curve is equivalent to a backward-integrated streakline. In the inertial case, a backward-integrated streakline also originates from a point $\mathbf{x}$, i.e., starting inertial pathlines from the streakline in forward direction will end up (spatially) at $\mathbf{x}$. We can, however, prescribe either of two things: the initial velocity (when starting from the streakline) or the observed velocity at $\mathbf{x}$ as in Chapter 10. In this chapter, we prescribe the initial velocity. The other is often problematic, as it requires inertial backward integration, which is difficult due to strong repelling behavior in the high-dimensional spatio-velocity domain, cf. [HS08] and Chapter 12.

13.4.2. Source Inversion in Bounded Domains

For the argumentation, we use the analogy to massless streaklines. Since streaklines represent individually advancing particle fronts, parts of a streakline might exit the flow domain, letting the streakline decay into disconnected pieces. If the streakline is integrated as a tangent curve (as in [WT10a]), the integration terminates when the streakline leaves the domain and thus the disconnected pieces are not found. This means in our context that influence curves cannot recover further sources if the influence curve leaves the domain (even though it might reenter at a later time). In the future, we would like to investigate how points can be efficiently found at which an influence curve reenters the domain so that the integration can be continued. For the integration in $\mathbf{h}$, we can directly measure the integration error as $E(\mathbf{c}(\mathbf{x},\tau)) =$

$\|\mathbf{x} - \phi(\mathbf{c}(\mathbf{x},\tau), \mathbf{v}_0,\, t_0,\, \tau)\|^2$, since an inertial pathline that was released from influence curve point $\mathbf{c}(\mathbf{x},\tau)$ should reach the observation point $\mathbf{x}$ exactly. Based on this, we would like to search the domain boundaries for reentry points.

13.4.3. Accumulating Errors

We described influence curves as tangent curves of a derived vector field. Generally, tracing curves this way is subject to accumulating numerical errors. With influence curves, however, this error can be measured (see Section 13.4.2), which allows for a predictor-corrector approach. After each integration step, the error can be reduced using Newton iterations. For the examples shown in the paper, the adaptive Runge-Kutta-Fehlberg (RK45) produced small enough errors. Nevertheless, they could be further reduced if the need arises.

13.4.4. Reachable Subspaces of the Spatio-Velocity Domain

In Chapters 10–12, the full phase space of possible particle positions and velocities was considered. In this case, the observation of inertial particles becomes a mass-dependent and time-dependent 4D or 6D problem, which poses challenges on both the efficiency of the extraction methods and the visualization. Limiting the phase space allows to reduce the typically high dimensionality of computations. Assuming for instance that inertial particles are released from rest, we found in this chapter that particles may reach a certain location only with a limited set of possible velocities. Further, we have shown that the set of initial positions that lead to the same location form a curve. Sudharsan et al. [SBR16] pointed out that the reachable subspace of the phase space decreases the smaller inertial particles become. This is plausible, since small inertial particles approach the terminal velocity faster and exhibit thus less variation than the larger particles. By a dense sampling of the domain with glyphs, we can visualize the spatio-velocity subspace that can actually be reached by inertial particles. Of course, a dense sampling with glyphs raises perceptual challenges, especially in 3D domains, when occlusion plays a role.

13.4.5. Performance and Parameters

We used an Intel Core i7-2600K CPU with 3.4 GHz. The number of glyphs, the extraction time of the influence curves (in seconds) and the respective parameters for v_{max} (maximal velocity shown in the velocity glyph), start time t_0 (if unsteady) and integration duration τ (parameterizing the curve in $[0,\, \tau]$) are listed for all data sets in Table 13.1.

Data Set	Glyphs	Time	v_{max}	t_0	τ
CENTER, Fig. 13.4	70	4.1	1	–	20
TREFOIL 2D, Fig. 13.6	180	3.1	1.2	–	2
TREFOIL 3D, Fig. 13.8	500	15.6	1.2	–	3
FORCED-D. DUFFING, Fig. 13.9	70	0.8	3.5	–	8
DOUBLE GYRE, Fig. 13.10	120	1.0	0.36	0	4
TORNADO, Fig. 13.11	147	5.4	5	–	3
RAYLEIGH-BÉNARD, Fig. 13.12	72	5.4	50	–	2
SQUARE CYLINDER, Fig. 13.13	8	14.9	2	10	14
WALL-M. CYLINDER, Fig. 13.14	72	69.9	11	0	10

Table 13.1.: Numbers of glyphs, extraction time (in seconds) and parameters used in the figures, shown throughout the chapter: maximal velocity shown in the glyphs v_{max}, start time t_0 (for unsteady flows) and integration duration τ, parameterizing the curve in $[0, \tau]$.

The extraction time is determined by the efficiency of the inertial pathline integration, which depends on the data set. In the analytic and simpler flows the influence curves were computed in a few seconds. In unsteady 3D flows, the extraction took up to 70 seconds. Generally, the further we integrate in $\mathbf{h}$, the more expensive the evaluation of an integration step becomes, since the pathlines involved in the flow map gradients have longer integration time τ. After extraction, the scene can be interactively explored and inertial pathlines can be interactively released from selected glyphs.

13.5. Conclusions

In this chapter, we considered the source inversion problem of inertial dispersed particles. Given an observation point and a certain initial velocity, we recovered all locations that an inertial particle could originate from. We have shown that these locations form a curve, which we called *influence curve* and we extracted them by massless pathline integration in a derived time-dependent vector field. Thereby, all computations were based on inertial forward integration only. After extraction, we interactively displayed the influence curves in the spatial domain and placed glyphs at the observation points, illustrating the velocities with which the inertial particles arrived. With this, we presented the first solution to the source inversion problem that considers actual inertial trajectories rather than operating on attracting manifolds only [HS08].

Future work includes the search for filtering methods that allow to limit computations (such as inertial vortex detection) to regions in the spatio-velocity domain that are

reachable by inertial flow. Dependent on the application, seeding regions and possibly varying initial velocities (modeled as probability distributions) play a role [GKKT13] that could be applied to further filter influence curves. Further, we would like to determine reentry points at which influence curves return back into the flow domain.

14 Chapter 14. Vortex Cores of Inertial Particles

The following chapter describes algorithms for the extraction of vortex corelines of swirling inertial particles. The chapter is based on the publication:

T. Günther and H. Theisel
Vortex Cores of Inertial Particles
IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization) 20, 12 (2014), 2535–2544.

For the purpose of understanding the swirling behavior of inertial particles, we extend the local Sujudi-Haimes [SH95], its unsteady extension [WSTH07], and the integration-based particle density estimation [WCW*09] to the inertial case. While for massless particles, the particle position only depends on the vector field, inertial particles have their own velocity vector, which is affected over time by inertia. The integration therefore requires to model position, current velocity and optionally time as state variables, rendering the system 6D or 7D. Detecting the swirling behavior within this high-dimensional space is accompanied by a large computation effort – at least for the brute force search that follows from a direct extension of the Sujudi-Haimes method from the massless case. However, we show that the search can actually be reduced to a 3D (or 4D) parallel vectors operation, if additional knowledge is carefully included. In consequence, we found a method that is computationally as expensive as an extraction of massless corelines, but can answer the same question for inertial particles. Moreover, due to the rich set of long known extraction methods via parallel vectors [PR99], it is easily integrated into existing visualization tools.

As a second strategy for finding core lines, we apply an integration-based particle density estimation after a short integration span to inertial particles, in order to find attracting structures that the local methods cannot correctly extract (not even for the massless case). In the remainder of this chapter, we introduce the extraction methods

and conduct a qualitative evaluation on a number of benchmark data sets.

Notation: Since for inertial particles the direction of movement does not only depend on the vector field $\mathbf{u}$ but also depends on the current particle velocity $\mathbf{v}$, we denote 6D points/vectors as $\tilde{\mathbf{x}}, \tilde{\mathbf{u}}$ where the first 3 components refer to the spatial locations and the last 3 refer to the current particle velocity. Similar considerations for time-dependent flow fields give 7D points/vectors $\hat{\mathbf{x}}, \hat{\mathbf{u}}$ where the additional last component refers to time.

14.1. Cores of Inertial Swirling Particles

Even for massless particles there is no universal coreline extraction or vortex criterion. Several approaches exist, all having their benefits and problems. The *local methods* only work for static corelines or corelines that move along straight lines. *Integration-based methods*, on the other hand, only detect attracting behavior. Though, at present, these are the best-established approaches. However, none is directly applicable to inertial particles, which is why we extend both approaches to the inertial case. In this section, we will show that the extraction of vortex corelines of inertial particles reduces to a parallel vectors operation in space (3D) for steady or space-time (4D) for unsteady flows.

14.1.1. Local Methods in a Nutshell

Given a (steady or unsteady) flow field $\mathbf{u}$ with its spatial Jacobian matrix $\mathbf{J}$, the (necessary) local conditions for corelines are summarized in Table 14.1. Here, r is the response time from Eq. (9.3), $\mathbf{g}$ is the gravity vector, and $\mathbf{f} = -\mathbf{J}^{-1}\,\mathbf{u}_t$ (cf. Eq. (2.4)) is the feature flow field. Note that Table 14.1 only gives necessary conditions: the extracted line structures have to be filtered by the presence of swirling motion, i.e., the

	massless	inertial
steady	$\mathbf{u} \parallel \mathbf{J}\mathbf{u}$	$(\mathbf{u} + r\,\mathbf{g}) \parallel \mathbf{J}(\mathbf{u} + r\,\mathbf{g})$
unsteady	$(\mathbf{u} - \mathbf{f}) \parallel \mathbf{J}(\mathbf{u} - \mathbf{f})$	$(\mathbf{u} - \mathbf{f} + r\,\mathbf{g}) \parallel \mathbf{J}(\mathbf{u} - \mathbf{f} + r\,\mathbf{g})$

Table 14.1.: Conditions for the presence of a core of swirling particle motion for massless [SH95, WSTH07] and inertial particles: $\mathbf{u}$ denotes the vector field, $\mathbf{J}$ its spatial Jacobian, $\mathbf{f}$ the feature flow field, r the response time, $\mathbf{g}$ the gravity vector. Note that in addition, the extracted line has to approximately align with the parallel vectors and that $\mathbf{J}$ must contain complex-conjugate eigenvalues.

existence of complex eigenvalues of $\mathbf{J}$ and the tangent of the extracted coreline should approximately align with the parallel vectors. In Table 14.1, the condition for the steady massless case is the Sujudi-Haimes condition [SH95]. Its extension from [WSTH07] is the condition for the unsteady massless case. The two conditions for the inertial case (right-hand column of Table 14.1) are the main theoretical contributions of this chapter. Fortunately, they do not depend on the current particle velocity $\mathbf{v}$, leading to a simple 3D PV extraction which can be done by standard methods. Although the conditions for inertial particles are extremely simple, their derivations are not. They are presented in the subsequent sections 14.1.2 and 14.1.3. To extract attracting corelines we apply an integration-based method that works in both steady and unsteady flows, as described in Section 14.1.4.

14.1.2. Local Method for Steady Case

For steady flows, we generalize the well-established Sujudi-Haimes approach [SH95] to inertial particles. Following (9.5), we consider the 6D steady vector field, in which inertial particle trajectories are tangent curves:

$$\tilde{\mathbf{u}}(\mathbf{x},\mathbf{v}) = \begin{pmatrix} \mathbf{v} \\ \frac{\mathbf{u}(\mathbf{x})-\mathbf{v}}{r} + \mathbf{g} \end{pmatrix}. \tag{14.1}$$

Its Jacobian is the 6×6 matrix

$$\tilde{\mathbf{J}} = \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{I}_3 \\ \frac{1}{r}\mathbf{J} & -\frac{1}{r}\mathbf{I}_3 \end{pmatrix}. \tag{14.2}$$

where $\mathbf{0}_{3,3}$ is the zero matrix, $\mathbf{I}_3$ the identity matrix, and $\mathbf{J}$ the 3×3 Jacobian of $\mathbf{u}$[1].

The Jacobian $\tilde{\mathbf{J}}$ characterizes the inertial particle behavior in both the spatial and the velocity domain. Extending the Sujudi-Haimes approach to 6D, we search for all 6D locations where $\tilde{\mathbf{u}}$ is parallel to a real eigenvector of $\tilde{\mathbf{J}}$, i.e., we search for 6D locations with

$$\tilde{\mathbf{u}} \parallel \tilde{\mathbf{J}}\tilde{\mathbf{u}}. \tag{14.3}$$

[1] Eq. (14.2) is shown by computing the spatio-velocity gradient of Eq. (14.1) and simplifying:

$$\begin{aligned} \nabla\tilde{\mathbf{u}}(\mathbf{x},\mathbf{v}) = \begin{pmatrix} \frac{\partial\tilde{\mathbf{u}}(\mathbf{x},\mathbf{v})}{\partial\mathbf{x}} & \frac{\partial\tilde{\mathbf{u}}(\mathbf{x},\mathbf{v})}{\partial\mathbf{v}} \end{pmatrix} &= \begin{pmatrix} \frac{\partial\mathbf{v}}{\partial\mathbf{x}} & \frac{\partial\mathbf{v}}{\partial\mathbf{v}} \\ \frac{1}{r}\frac{\partial\mathbf{u}(\mathbf{x})}{\partial\mathbf{x}} - \frac{1}{r}\frac{\partial\mathbf{v}}{\partial\mathbf{x}} + \frac{\partial\mathbf{g}}{\partial\mathbf{x}} & \frac{1}{r}\frac{\partial\mathbf{u}(\mathbf{x})}{\partial\mathbf{v}} - \frac{1}{r}\frac{\partial\mathbf{v}}{\partial\mathbf{v}} + \frac{\partial\mathbf{g}}{\partial\mathbf{v}} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{I}_3 \\ \frac{1}{r}\mathbf{J} - \frac{1}{r}\mathbf{0}_{3,3} + \mathbf{0}_{3,3} & \frac{1}{r}\mathbf{0}_{3,3} - \frac{1}{r}\mathbf{I}_3 + \mathbf{0}_{3,3} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{I}_3 \\ \frac{1}{r}\mathbf{J} & -\frac{1}{r}\mathbf{I}_3 \end{pmatrix} = \tilde{\mathbf{J}} \end{aligned}$$

with $\frac{\partial\mathbf{v}}{\partial\mathbf{x}} = \mathbf{0}_{3,3}$, $\frac{\partial\mathbf{v}}{\partial\mathbf{v}} = \mathbf{I}_3$, $\frac{\partial\mathbf{u}(\mathbf{x})}{\partial\mathbf{x}} = \mathbf{J}$, $\frac{\partial\mathbf{u}(\mathbf{x})}{\partial\mathbf{v}} = \mathbf{0}_{3,3}$, and since $\mathbf{g}$ is constant $\frac{\partial\mathbf{g}}{\partial\mathbf{x}} = \frac{\partial\mathbf{g}}{\partial\mathbf{v}} = \mathbf{0}_{3,3}$.

Using the abbreviation

$$\mathbf{w} = \frac{\mathbf{u}-\mathbf{v}}{r} + \mathbf{g} \tag{14.4}$$

we get[2]

$$\tilde{\mathbf{u}} = \begin{pmatrix} \mathbf{v} \\ \mathbf{w} \end{pmatrix} \quad , \quad \tilde{\mathbf{J}}\,\tilde{\mathbf{u}} = \begin{pmatrix} \mathbf{w} \\ \frac{1}{r}\left(\mathbf{J}\mathbf{v} - \mathbf{w}\right) \end{pmatrix}. \tag{14.5}$$

From the parallelity (14.3) follows the parallelity in all projections to subspaces. Projecting (14.3) into the spatial subspace gives

$$\mathbf{v} \parallel \mathbf{w} \tag{14.6}$$

which simplifies under the consideration of (14.4) to

$$\mathbf{v} \parallel \mathbf{u}(\mathbf{x}) + r\,\mathbf{g}\ . \tag{14.7}$$

At the moment, this condition requires a particle velocity $\mathbf{v}$ at position $\mathbf{x}$, which is problematic. We cannot evaluate the condition without a brute-force simulation of a large number of particles in order to observe the different velocities of particles passing location $\mathbf{x}$. We circumvent the problem of not knowing $\mathbf{v}$ by introducing another condition: we project the condition (14.3) into the $\mathbf{v}$ subspace. Considering (14.5), this gives the condition

$$\mathbf{w} \parallel \frac{1}{r}(\mathbf{J}\mathbf{v} - \mathbf{w}). \tag{14.8}$$

By inserting (14.6) into (14.8) we get

$$\mathbf{v} \parallel \frac{1}{r}(\mathbf{J}\mathbf{v} - \mathbf{v}) \tag{14.9}$$

which further simplifies to

$$\mathbf{v} \parallel \mathbf{J}\mathbf{v}. \tag{14.10}$$

Finally, we combine the parallelity conditions of both subspaces by inserting (14.7) into (14.10), which gives the final condition

$$(\mathbf{u} + r\,\mathbf{g}) \parallel \mathbf{J}(\mathbf{u} + r\,\mathbf{g}). \tag{14.11}$$

Note that (14.11) does not contain $\mathbf{v}$ any more: it can be solved in the spatial domain only by applying standard 3D PV techniques.

For massless particles, the response time is $r = 0$. In this case, (14.11) becomes $\mathbf{u} \parallel \mathbf{J}\mathbf{u}$, which is the parallel vectors expression of Sujudi-Haimes. Thus, the swirling of massless particles is a special case of our generalized parallel vectors condition.

[2] The expression in Eq. (14.5) for $\tilde{\mathbf{J}}\,\tilde{\mathbf{u}}$ is shown by expanding and simplifying:

$$\tilde{\mathbf{J}}\,\tilde{\mathbf{u}} = \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{I}_3 \\ \frac{1}{r}\mathbf{J} & -\frac{1}{r}\mathbf{I}_3 \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{w} \end{pmatrix} = \begin{pmatrix} \mathbf{0}_{3,3}\,\mathbf{v} + \mathbf{I}_3\,\mathbf{w} \\ \frac{1}{r}\mathbf{J}\,\mathbf{v} - \frac{1}{r}\mathbf{I}_3\,\mathbf{w} \end{pmatrix} = \begin{pmatrix} \mathbf{w} \\ \frac{1}{r}\left(\mathbf{J}\mathbf{v} - \mathbf{w}\right) \end{pmatrix}$$

.

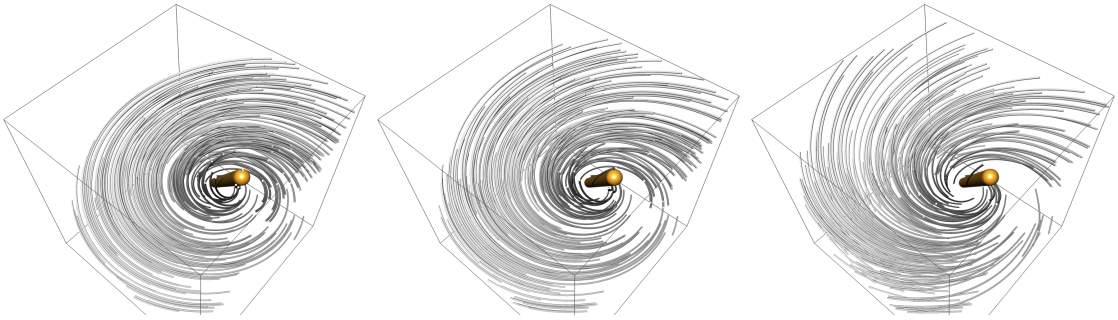


Figure 14.1.: For $\mathbf{g} = \mathbf{0}$ inertial particles of any size have the same coreline. Left to right: $d_p = 0\,\mu m$ (massless), $d_p = 100\,\mu m$ and $d_p = 200\,\mu m$.

Condition (14.11) also reduces to $\mathbf{u} \parallel \mathbf{J}\mathbf{u}$ if there is no gravity, i.e., $\mathbf{g} = \mathbf{0}$. This means that all inertial particles, regardless of their response time (i.e., diameter or density), swirl in gravity-free environments around the same coreline. We demonstrate this effect in Fig. 14.1. The trajectories of inertial particles change with a varying particle diameter, but the coreline remains the same.

14.1.3. Local Method for Unsteady Case

For unsteady flows, we extend the cores of swirling particle motion of [WSTH07] to inertial flows. For this, we consider the 7D inertial flow field (cf. Eq. (9.5))

$$\hat{\mathbf{p}}(\mathbf{x}, \mathbf{v}, t) = \begin{pmatrix} \mathbf{v} \\ \frac{\mathbf{u}(\mathbf{x},t)-\mathbf{v}}{r} + \mathbf{g} \\ 1 \end{pmatrix}, \tag{14.12}$$

Its Jacobian is the 7×7 matrix[3]

$$\hat{\mathbf{J}} = \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{I}_3 & \mathbf{0}_3 \\ \frac{1}{r}\mathbf{J} & -\frac{1}{r}\mathbf{I}_3 & \frac{1}{r}\mathbf{u}_t \\ \mathbf{0}_3^{\mathrm{T}} & \mathbf{0}_3^{\mathrm{T}} & 0 \end{pmatrix} \tag{14.13}$$

[3] Eq. (14.13) is shown by computing the gradient of Eq. (14.12) and simplifying:

$$\begin{aligned} \nabla\hat{\mathbf{p}}(\mathbf{x},\,\mathbf{v},\,t) &= \begin{pmatrix} \frac{\partial\hat{\mathbf{p}}(\mathbf{x},\mathbf{v},t)}{\partial\mathbf{x}} & \frac{\partial\hat{\mathbf{p}}(\mathbf{x},\mathbf{v},t)}{\partial\mathbf{v}} & \frac{\partial\hat{\mathbf{p}}(\mathbf{x},\mathbf{v},t)}{\partial t} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial\mathbf{v}}{\partial\mathbf{x}} & \frac{\partial\mathbf{v}}{\partial\mathbf{v}} & \frac{\partial\mathbf{v}}{\partial t} \\ \frac{1}{r}\frac{\partial\mathbf{u}(\mathbf{x},t)}{\partial\mathbf{x}} - \frac{1}{r}\frac{\partial\mathbf{v}}{\partial\mathbf{x}} + \frac{\partial\mathbf{g}}{\partial\mathbf{x}} & \frac{1}{r}\frac{\partial\mathbf{u}(\mathbf{x},t)}{\partial\mathbf{v}} - \frac{1}{r}\frac{\partial\mathbf{v}}{\partial\mathbf{v}} + \frac{\partial\mathbf{g}}{\partial\mathbf{v}} & \frac{1}{r}\frac{\partial\mathbf{u}(\mathbf{x},t)}{\partial t} - \frac{1}{r}\frac{\partial\mathbf{v}}{\partial t} + \frac{\partial\mathbf{g}}{\partial t} \\ \frac{\partial 1}{\partial\mathbf{x}} & \frac{\partial 1}{\partial\mathbf{v}} & \frac{\partial 1}{\partial t} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{I}_3 & \mathbf{0}_3 \\ \frac{1}{r}\mathbf{J} & -\frac{1}{r}\mathbf{I}_3 & \frac{1}{r}\mathbf{u}_t \\ \mathbf{0}_3^{\mathrm{T}} & \mathbf{0}_3^{\mathrm{T}} & 0 \end{pmatrix} = \hat{\mathbf{J}} \end{aligned}$$

with $\frac{\partial\mathbf{v}}{\partial\mathbf{x}} = \mathbf{0}_{3,3}$, $\frac{\partial\mathbf{v}}{\partial\mathbf{v}} = \mathbf{I}_3$, $\frac{\partial\mathbf{v}}{\partial t} = \mathbf{0}_3$, $\frac{\partial\mathbf{u}(\mathbf{x},t)}{\partial\mathbf{x}} = \mathbf{J}$, $\frac{\partial\mathbf{u}(\mathbf{x},t)}{\partial\mathbf{v}} = \mathbf{0}_{3,3}$, $\frac{\partial\mathbf{u}(\mathbf{x},t)}{\partial t} = \mathbf{u}_t$, and since $\mathbf{g}$ is constant $\frac{\partial\mathbf{g}}{\partial\mathbf{x}} = \frac{\partial\mathbf{g}}{\partial\mathbf{v}} = \mathbf{0}_{3,3}$, and $\frac{\partial\mathbf{g}}{\partial t} = \mathbf{0}_3$

where $\mathbf{0}_3$ is the 3D zero (column) vector and $\mathbf{u}_t = \frac{\partial \mathbf{u}}{\partial t}$ is the t-partial of $\mathbf{u}$. Following [WSTH07], we use the *coplanar vector operator* for detecting vortex structures: we search for 7D locations where we can find two real eigenvectors $\hat{\mathbf{e}}_i, \hat{\mathbf{e}}_j$ of $\hat{\mathbf{J}}$ such that $\hat{\mathbf{e}}_i, \hat{\mathbf{e}}_j, \hat{\mathbf{p}}$ are coplanar, i.e.,

$$\hat{\mathbf{p}} = \beta\, \hat{\mathbf{e}}_i + \gamma\, \hat{\mathbf{e}}_j \tag{14.14}$$

for certain scalars β, γ. The eigenvectors of $\hat{\mathbf{J}}$ are

$$\begin{pmatrix} \tilde{\mathbf{e}}_1 \\ 0 \end{pmatrix}, ..., \begin{pmatrix} \tilde{\mathbf{e}}_6 \\ 0 \end{pmatrix}, \hat{\mathbf{e}}_7 = \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \\ 1 \end{pmatrix} \tag{14.15}$$

where $\tilde{\mathbf{e}}_1, ..., \tilde{\mathbf{e}}_6$ are the eigenvectors of $\tilde{\mathbf{J}}$, and $\hat{\mathbf{e}}_7$ is the eigenvector belonging to the eigenvalue 0, i.e., the solution of $\hat{\mathbf{J}}\ \hat{\mathbf{e}}_7 = \mathbf{0}_7$. The explicit solution of this equation gives the vector $\mathbf{f}$ as defined in (14.1.1). Note that it has been used elsewhere [WSTH07] to track critical points in 3D unsteady vector fields. Considering the last components of (14.15) and (14.14) gives $j = 7$ and $\gamma = 1$ in (14.14). Applying this to the first 6 components of (14.14) gives the condition[4]

$$\tilde{\mathbf{u}} - \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix} \parallel \tilde{\mathbf{e}}_i \tag{14.16}$$

where $\tilde{\mathbf{e}}_i$ is a real eigenvector of $\tilde{\mathbf{J}}$. Eq. (14.16) means that we have reduced the 7D coplanar vectors problem to a 6D PV problem. Similar to Section 14.1.2, this is now further reduced to a 3D PV condition which can be searched in 3D space only. Eq. (14.16) can be rewritten as

$$\tilde{\mathbf{u}} - \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix} \parallel \tilde{\mathbf{J}} \left(\tilde{\mathbf{u}} - \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix} \right). \tag{14.17}$$

[4] Setting $\gamma = 1$ and subtracting $\hat{\mathbf{e}}_7$ on both sides reduces the seventh dimension to $0 = 0$. Thus, the 7D coplanarity condition reduces to a 6D parallel vectors operation:

$$\begin{aligned} \hat{\mathbf{p}},\ \hat{\mathbf{e}}_i,\ \hat{\mathbf{e}}_7 \text{ are coplanar} &\Leftrightarrow \begin{pmatrix} \tilde{\mathbf{u}} \\ 1 \end{pmatrix} = \beta \begin{pmatrix} \tilde{\mathbf{e}_1} \\ 0 \end{pmatrix} + \gamma \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \\ 1 \end{pmatrix} \qquad \text{set: } \gamma = 1 \\ &\Leftrightarrow \tilde{\mathbf{u}} - \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix} = \beta\, \tilde{\mathbf{e}}_i \\ &\Leftrightarrow \tilde{\mathbf{u}} - \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix} \parallel \tilde{\mathbf{e}}_i \end{aligned}$$

Since[5]

$$\tilde{\mathbf{J}}\left(\tilde{\mathbf{u}}-\begin{pmatrix}\mathbf{f}\\ \mathbf{0}_3\end{pmatrix}\right)=\begin{pmatrix}\mathbf{w}\\ \frac{1}{r}\left(\mathbf{J}(\mathbf{v}-\mathbf{f})-\mathbf{w}\right)\end{pmatrix} \tag{14.18}$$

with $\mathbf{w}$ defined in (14.4), condition (14.17) can be projected into their $\mathbf{x}$ and $\mathbf{v}$ subspace respectively:

$$\mathbf{v}-\mathbf{f} \parallel \mathbf{w} \tag{14.19}$$

$$\mathbf{w} \parallel \frac{1}{r}(\mathbf{J}(\mathbf{v}-\mathbf{f})-\mathbf{w}). \tag{14.20}$$

From multiplying (14.4) by r and substituting in (14.19) we get

$$(\mathbf{v}-\mathbf{f}) \parallel (\mathbf{u}-\mathbf{v}+r\,\mathbf{g}) \tag{14.21}$$

To obtain a condition in which one side is independent of $\mathbf{v}$, we add the left-hand side $(\mathbf{v}-\mathbf{f})$ to the right-hand side of (14.21)

$$(\mathbf{v}-\mathbf{f}) \parallel (\mathbf{u}-\mathbf{f}+r\,\mathbf{g}). \tag{14.22}$$

Inserting (14.19) into (14.20) and multiplying the right-hand side by r gives

$$(\mathbf{v}-\mathbf{f}) \parallel (\mathbf{J}(\mathbf{v}-\mathbf{f})-\mathbf{w}). \tag{14.23}$$

Considering (14.19) again, we see that the $\mathbf{w}$ on the right of (14.23) is parallel to the left-hand side and can thus be removed

$$(\mathbf{v}-\mathbf{f}) \parallel \mathbf{J}(\mathbf{v}-\mathbf{f}). \tag{14.24}$$

Inserting (14.22) into (14.24) gives the final condition

$$(\mathbf{u}-\mathbf{f}+r\,\mathbf{g}) \parallel \mathbf{J}(\mathbf{u}-\mathbf{f}+r\,\mathbf{g}). \tag{14.25}$$

Note that (14.25) does not contain $\mathbf{v}$ any more: it can be solved by applying a standard PV extractor in 3D space and subsequently filtering the resulting lines by swirling motion, i.e., the presence of imaginary eigenvalues of $\mathbf{J}$.

For massless particles, i.e., $r=0$, or in a gravity-free environment, i.e., $\mathbf{g}=\mathbf{0}$, the generalized Eq. (14.25) reduces to the parallel vectors condition of cores of swirling, massless pathlines: $(\mathbf{u}-\mathbf{f}) \parallel \mathbf{J}(\mathbf{u}-\mathbf{f})$, which was described in [WSTH07]. This is similar to the steady case, shown in Section 14.1.2.

Appendix E contains a proof of the Galilean invariance of our method.

[5] Eq. (14.18) is shown by expanding and simplifying:

$$\tilde{\mathbf{J}}\left(\tilde{\mathbf{u}}-\begin{pmatrix}\mathbf{f}\\ \mathbf{0}_3\end{pmatrix}\right)=\begin{pmatrix}\mathbf{0}_{3,3} & \mathbf{I}_3\\ \frac{1}{r}\mathbf{J} & -\frac{1}{r}\mathbf{I}_3\end{pmatrix}\begin{pmatrix}\mathbf{v}-\mathbf{f}\\ \mathbf{w}\end{pmatrix}=\begin{pmatrix}\mathbf{0}_{3,3}(\mathbf{v}-\mathbf{f})+\mathbf{I}_3\mathbf{w}\\ \frac{1}{r}\mathbf{J}(\mathbf{v}-\mathbf{f})-\frac{1}{r}\mathbf{I}_3\mathbf{w}\end{pmatrix}=\begin{pmatrix}\mathbf{w}\\ \frac{1}{r}\left(\mathbf{J}(\mathbf{v}-\mathbf{f})-\mathbf{w}\right)\end{pmatrix}.$$

14.1.4. Integration-based Method

Attracting corelines can be found for both steady and unsteady flows in a number of different, yet quite similar ways: all have in common that they are integration-based. One option is to apply the particle density estimation of Wiebel et al. [WCW*09] to inertial particles, which works as follows: To extract the attractors at time t, seed a large number of particles at time $t - \tau$ on a regular grid that spans the entire spatial domain. Note that inertial corelines are found at the accuracy of the resolution of the seeding grid. Integrate all particles for a duration τ up to the time t. During this time, the particles will converge to the attracting structures, thus their density will increase near them and will eventually obtain its maximum directly on them. The density can be obtained by rasterization of the particles into a second grid, which is used for counting rasterized particles. Local maxima are found by inspecting each density grid cell in turn in order to search for cells that have a higher density than all adjacent cells. Optionally, the density field can be smoothed prior to the maxima extraction if too many local extrema exist. For the tracking of attractors over time, the procedure is repeated for a series of observation times. For each time step, attractors are extracted and are connected over time by lines by using a correspondence heuristic, e.g., by joining with the closest attractor of the previous observation time.

Another variant of the density estimation is to seed at each cell of a regular grid a small group of particles and to await their contraction: if the particles within a group get very close to each other, e.g., the maximum pairwise distance falls below a threshold, an attractor is found at subgrid accuracy. Moreover, the full coreline can be found by continuing the integration until the particles leave the (temporal) domain, since they will stay on the attractor. Afterwards, identical lines have to be removed. In this approach, the integration duration and the extent of the group's seed structure should be chosen carefully. The closer particles are seeded, the shorter the duration until they contract. If they are seeded too far apart, the temporal domain might not be long enough for the particles to collapse. For time-periodic fields, this is not a problem, since the integration can be continued until an attractor is found. This is the approach that we followed in this chapter.

A third option that was successfully applied to the massless case is to consider the cores of swirling streaklines [WT10a]. For this a derived vector field is constructed, in which streaklines are tangent curves. Then, the swirling in this derived vector field is observed, which – for the massless case – reduced to a parallel vectors operation. This strategy has a high computational effort, but an extension to the inertial case is worth a study in future work.

14.2. Results and Discussion

In the following, we show extraction results for a number of test data sets, as we alter the gravity that acts on the inertial particles. Note that the gravity we choose in the test cases is not the same as was used in the computation of the given vector field. They need to be consistent in order to draw meaningful conclusions for the application. By varying the gravity only in the particle motion, we obtain a different flow setting that, given the gravity, coincides after running a new fluid simulation with the flow field that was originally given. Thus, it is not possible to infer statements on the original field. However, it allows us to generate more synthetic test data, in which we seed inertial particles and extract their cores. The goal of our qualitative evaluation is not to come up with new insights for the applications, but to show that given a flow and a gravity: we can compute the cores of inertial particles.

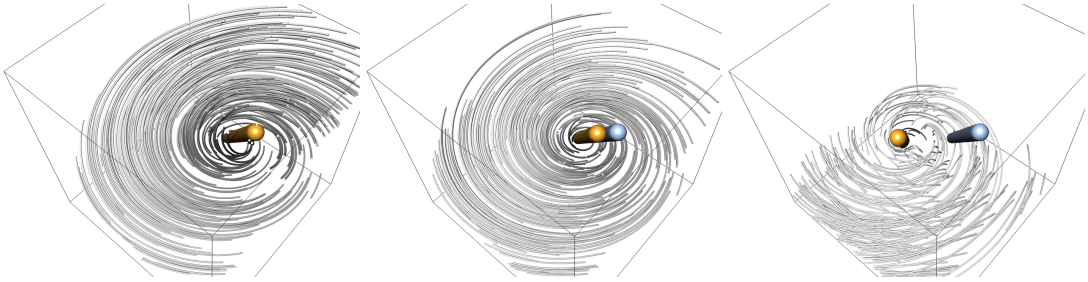
Throughout the experiments, the (dynamic) viscosity is set to $\mu = 1.532 \times 10^{-5}\ kg/(m\,s)$ (air). For the particles, we used the density of quartz glass $\rho_p = 2650\ kg/m^3$. Since the particle diameter d_p varies between $d_p = 20\,\mu m$ and $d_p = 300\,\mu m$ the particle response time r is in the range $r \in [0.0038,\ 0.865]$.

Moving Center (2D unsteady). Weinkauf et al. [WSTH07] used an analytic unsteady 2D field to demonstrate their extraction of the centers of swirling (massless) pathlines. They illustrated its construction, which is a linear interpolation over time between two 2D centers. The flow is defined on the domain $D \times T = [-2,2]^2 \times [0,4]$, with centers located at $(1,-1,0)^{\mathrm{T}}$ and $(-1,1,4)^{\mathrm{T}}$:

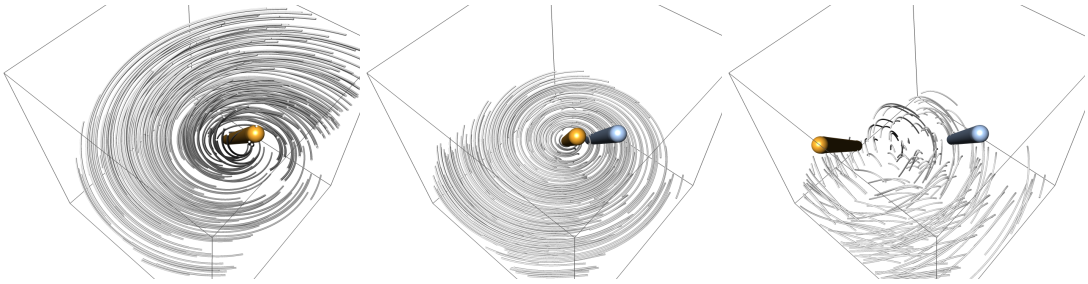
$$\mathbf{u}(x,y,t) = \begin{pmatrix} -y + \frac{t}{2} - 1 \\ x + \frac{t}{2} - 1 \end{pmatrix}. \tag{14.26}$$

In this data set, the massless coreline is a line that travels on a straight line, thus Sujudi-Haimes is known to work perfectly [WSTH07]. By using it, we find for this simple vector field the family of corelines analytically as $\mathbf{c}(a,s) = ((a-s+2)/2, (a+s-2)/2, s)^{\mathrm{T}}$, with s being the parameter along the coreline, and a being the z-component of the field. Setting $a = 0$ and $a = 1$ produces the cores reported in [WSTH07]. In the inertial case, we have $a = 1 + r\,\mathbf{g}_z$. When varying gravity or mass, corelines might move in an unanticipated direction. For this particular data set it turns out that both gravity $\mathbf{g} = (0,0,\mathbf{g}_z)^{\mathrm{T}}$ and coreline tangent $\partial\mathbf{c}/\partial s = (-\frac{1}{2}, \frac{1}{2}, 1)^{\mathrm{T}}$ are perpendicular to the movement direction $\partial\mathbf{c}/\partial a = (\frac{1}{2}, \frac{1}{2}, 0)^{\mathrm{T}}$ that arises when varying particle diameter or gravity. Such perpendicularity is not generally the case.

In Fig. 14.1 we have shown that for no gravity, i.e., $\mathbf{g} = \mathbf{0}$, all inertial particles regardless of their size, swirl around a common coreline. If a certain gravity acts, the position



(a) $\mathbf{g} = (0, 0, -4)^{\mathrm{T}}$, left to right: $d_p = 0\,\mu m$, $d_p = 100\,\mu m$, $d_p = 200\,\mu m$



(b) $\mathbf{g} = (0, 0, -8)^{\mathrm{T}}$, left to right: $d_p = 0\,\mu m$, $d_p = 100\,\mu m$, $d_p = 200\,\mu m$

Figure 14.2.: If a gravity force acts on the inertial particles, the position of the vortex coreline (•) changes when varying the mass. The coreline of massless particles (•) is shown as a reference.

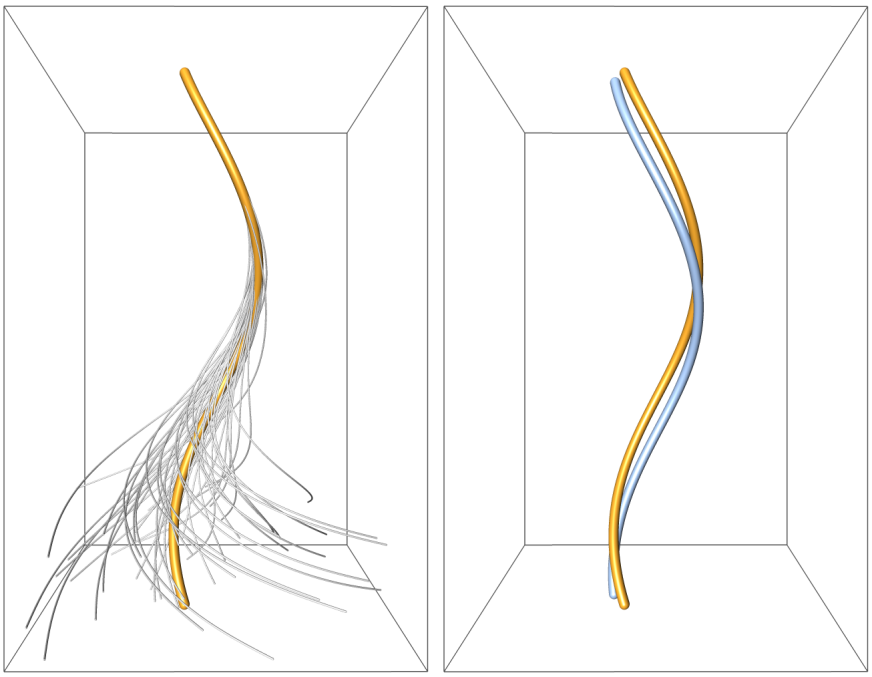
of the coreline depends on the mass, as demonstrated in Fig. 14.2. Similarly, the coreline moves when the mass is constant, but the gravity is varied, as apparent when comparing columns of Figs. 14.1 and 14.2.

Beads Problem (2D unsteady). A benchmark data set for vortex coreline extraction is the BEADS FLOW, which was described in Section 8.4. In this flow, no standard visualization tool is able to detect the attracting coreline; neither LIC, pathlines, vector field topology nor FTLE. It was shown that neither the tracking of the cores of swirling streamlines over time nor the cores of swirling pathlines match the attractor [WT10a]. Only integration-based methods such as particle density estimation [WCW*09] and the cores of swirling streaklines [WT10a] are able to deliver the attracting line. Consequentially, our local inertial methods from Sections 14.1.2 and 14.1.3 are not applicable, here. Instead, we will use the integration-based method, introduced in Section 14.1.4.

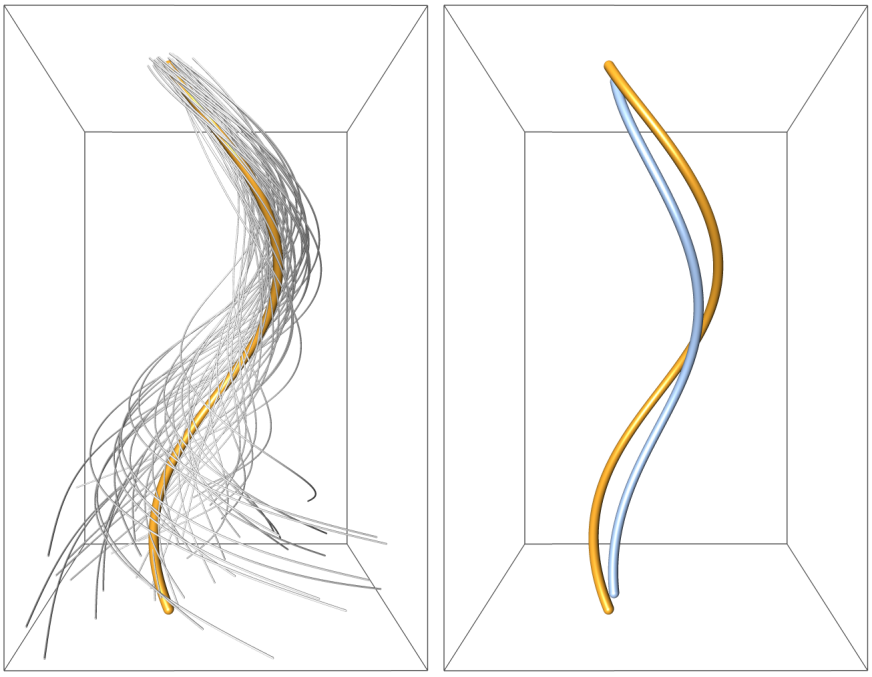
An analytic approximation to this data set with similar properties was given in [WT10a]. It is defined in the domain $D \times T = [-2, 2]^2 \times [0, 2\pi]$:

$$\mathbf{u}(x, y, t) = \begin{pmatrix} -(y - \frac{1}{3}\sin(t)) - (x - \frac{1}{3}\cos(t)) \\ (x - \frac{1}{3}\cos(t)) - (y - \frac{1}{3}\sin(t)) \end{pmatrix}. \tag{14.27}$$

In Chapter 10, we observed that inertial particles exhibit in the BEADS FLOW a mass-



(a) Inertial particle trajectories with $d_p = 100\,\mu m$.



(b) Inertial particle trajectories with $d_p = 200\,\mu m$.

Figure 14.3.: In the BEADS FLOW, the integration-based particle density estimation (●) successfully detects the attractor of the ascending and thereby converging inertial particles, whereas the local method (●) fails.

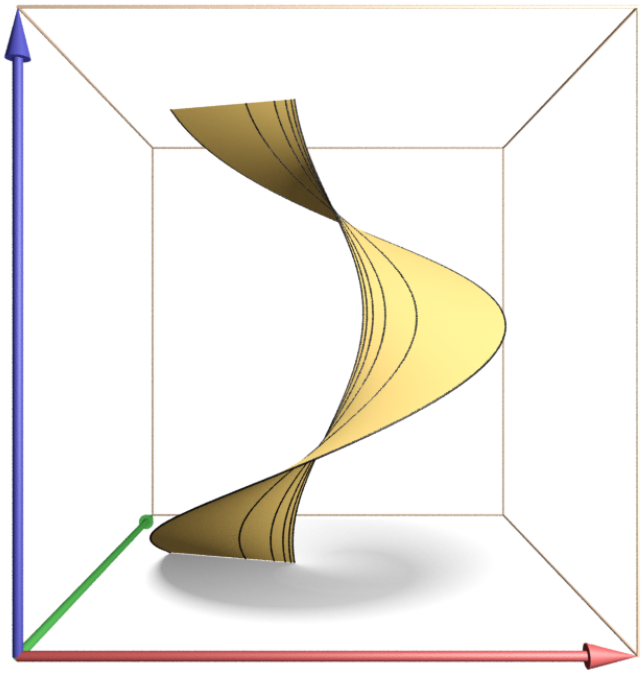


Figure 14.4.: The attracting corelines that were found in the BEADS FLOW by our integration-based method are mass-dependent and assemble a surface. Here, this surface is shown for $d_p = 100 \ldots 300\,\mu m$ in the extended domain $D \times T = [-3,3]^2 \times [0, 2\pi]$.

dependent swirling behavior. In Fig. 14.3a, we depict swirling inertial particles with $d_p = 100\,\mu m$ and the coreline they attract to. It is shown that the core of swirling inertial pathlines (our local method of Section 14.1.3) does not match the attractor. Next to it, in Fig. 14.3b, we show the same for $d_p = 200\,\mu m$. Since $\mathbf{g} = \mathbf{0}$, the locally extracted coreline (•) is the same for every mass. However, it can be seen that the integration-based coreline (•) is mass-dependent and that for larger particles, the particles converge slower onto the attracting coreline. More specifically, the attracting corelines have a different, mass-dependent radius. In fact, the coreline changes continuously with varying mass, which assembles a surface, shown in Fig. 14.4. On this surface, isolines are depicted for $d_p = 100,\ 150,\ 200,\ 250,\ 300\,\mu m$ (from inside to outside). It is apparent that the radius grows exponentially with increasing diameter.

Helicopter Descent (3D steady). The HART-II test by Yu et al. [YTvdW*02] was a major experimental cooperation program of rotorcraft engineers from Germany (DLR), the Netherlands (ONERA) and the United States (NASA). Its main objective was to study the analytical modeling capabilities of noise and vibrations that arise due to the interaction of rotor blades with wake vortices. The velocity field data was acquired experimentally by particle image velocimetry (PIV) in the German-Dutch Windtunnel (DNW).

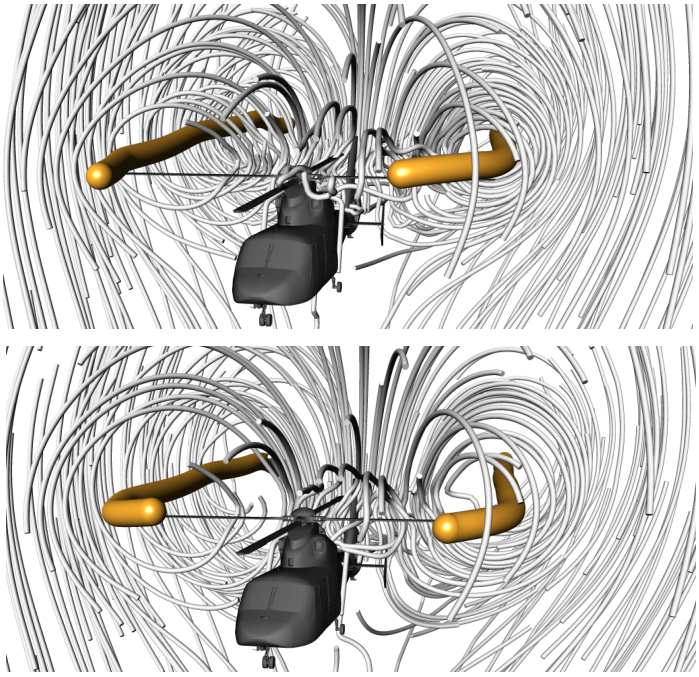


Figure 14.5.: Vortices detaching from a helicopter in descent. Smaller particles get closer to the corelines, whereas larger particles move on a larger orbit due to inertia. Top $d_p = 0\,\mu m$ (massless), bottom: $d_p = 100\,\mu m$.

We used and modified this data to better suit our visualization purposes. First, the inflow velocity $(0, -0.11, 0)^{\mathrm{T}}$ was subtracted in order to make the wake vortices apparent. (The y-axis is the forward flight direction.) A Galilean invariant method would find the vortices with and without the inflow velocity, though in the original state they are not visible to the eye. Consequentially, the velocity field had to be scaled (by factor 300) in order to get the wind up to speed again, so that the gravity $\mathbf{g} = (-9.8065, 0, 0)^{\mathrm{T}}$ is not acting overly strong. In Fig. 14.5, we released particles with $d_p = 0\,\mu m$ (left) and $d_p = 100\,\mu m$ (right). The resulting corelines differ only marginally, though the inertial trajectories exhibit a behavior that we have similarly experienced in the previous BEADS FLOW. Larger particles appear to swirl around the coreline at a larger distance than smaller or massless particles. This seems reasonable due to the acting inertia.

Delta Wing (3D steady). The DELTA WING data set is a simulation of a triangular surface in upstream flow. As described in Section 5.5, the flow contains two large wake vortices. The gravity is set to $\mathbf{g} = (0, -9.8065, 0)^{\mathrm{T}}$. Originally, this vector field exhibits very high wind speeds. Thus, the trajectories of inertial particles are nearly identical to the massless case. In order to create benchmark data that showcases inertial corelines that differ from the massless case, we scaled the vector field by 0.006. With this, gravity is proportionally stronger. Pathlines and their corelines are depicted in Fig. 14.6.

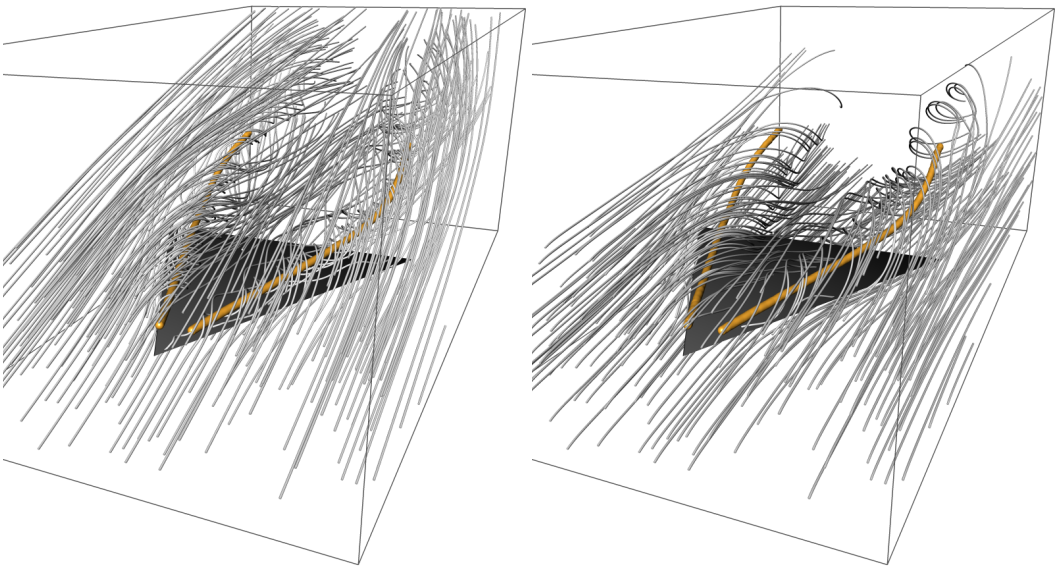


Figure 14.6.: Two large vortices detach from the DELTA WING surface. Due to gravity inertial particles are forced into a lower altitude. The inertial corelines vary only marginally. Top $d_p = 0\,\mu m$, bottom: $d_p = 100\,\mu m$.

Hurricane Bonnie (3D steady). In August 1998, a tropical wave emerged off the coast of Africa and turned into a major hurricane, named Bonnie. Zhu et al. [ZZW02, ZZW04] conducted a weather simulation of the northern hemisphere and we picked one particular region of interest that features the hurricane. For this data set, we explored what happens, if gravity is exaggerated. Therefore, we increased the gravity up to $\mathbf{g} = (0, -40, 0)^{\mathrm{T}}$, whereas the velocity field was scaled by 0.01, so that small particles of different mass follow trajectories that are apparently different. In Fig. 14.7, the hurricane is shown from the *same camera view* for two different masses, i.e., for $d_p = 0\,\mu m$ (massless) and $d_p = 20\,\mu m$. The increased gravity in the right image, forces the inertial particles faster to the ground. The extracted corelines are shown, and it is apparent that close to the ground the corelines still match. At higher altitudes, however, they vary considerably.

Stuart Vortex (3D unsteady). Another analytic data set that was used in [WSTH07] for the extraction of cores of swirling pathlines is the following variant of the STUART VORTEX:

$$\bar{\mathbf{u}}(x, y, z, t) = \begin{pmatrix} \sinh(y)/(\cosh(y) - \frac{1}{4}\cos(x-t)) + 1 \\ -\frac{1}{4}\sin(x-t)/(\cosh(y) - \frac{1}{4}\cos(x-t)) \\ z \\ 1 \end{pmatrix} \tag{14.28}$$

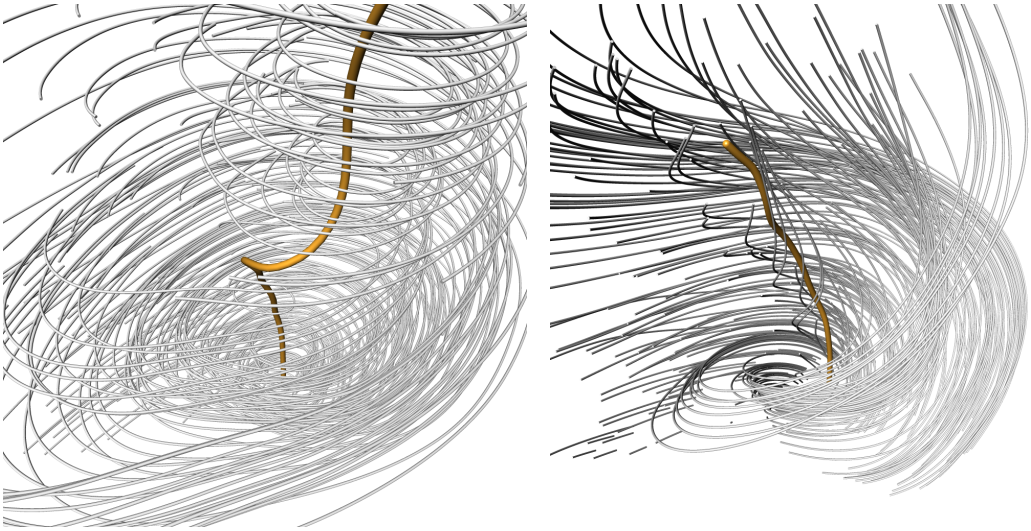


Figure 14.7.: Corelines for different sizes of a particles. The difference is large to due the high gravity. Left $d_p = 0\,\mu m$ (massless), right: $d_p = 20\,\mu m$.

The vortex is moving constantly over time in x-direction to the right and the vector $(1,0,0)^{\mathrm{T}}$ is superimposed on the spatial coordinates. We consider the domain $D \times T = [-4,4] \times [-2,2] \times [-1,1] \times [0,1]$. For the inertial particles, gravity acts with $\mathbf{g} = (0, 9.8065, 0)^{\mathrm{T}}$ (the y-axis goes down).

When only considering the streamlines of one time step the swirling cannot be faithfully observed. In Fig. 14.8 for instance, a center is visible that is not the core of swirling particle motion, seen in Fig. 14.9a. However, taking the temporal derivatives into account allows to extract the blue coreline, shown in Fig. 14.9a. This massless pathline core can be found by standard PV extraction methods on the field $\mathbf{u} - \mathbf{f}$, as demonstrated in Section 14.1.3. Fig. 14.9b shows a LIC of this field, revealing the core centers. The inertial particles are subject to gravity and inertia, which affects the trajectories considerably. In fact, a number of particles exit the domain because of them. First the particles, released at the left hand side of the image, fall down, and are then uplifted by the vortex above. The inertial particles exhibit a swirling behavior later down the flow, which can be seen by the particles in Fig. 14.9c. The common center of rotation is apparent by the translated coreline, found in the derived flow field in Fig. 14.9d. It is noticeable that the massless and the inertial coreline differ.

The difference is clearer in the next time step, shown in Fig. 14.10. In the moving STUART VORTEX flow, the vortices advance in x-direction. Since the flow is periodic, a new coreline enters from the left in Fig. 14.10c. While the rotation of the newly entering vortex is not very prominent in the particle visualization (as it is located shortly behind the seeding line), the LIC visualization of the derived vector field $\mathbf{u} - \mathbf{f} + r\,\mathbf{g}$ in Fig. 14.10d is clearly showing the streamline core that our method extracts.

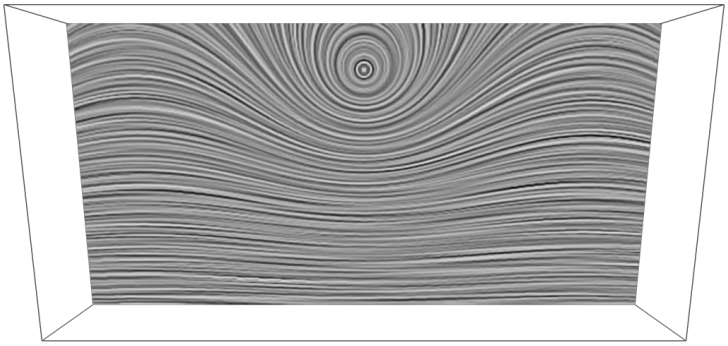
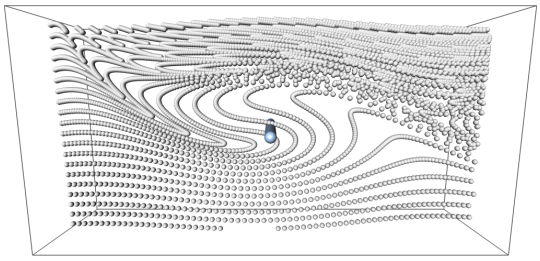
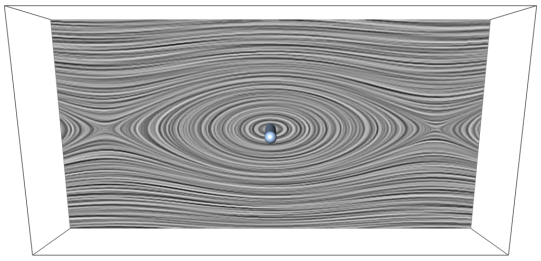


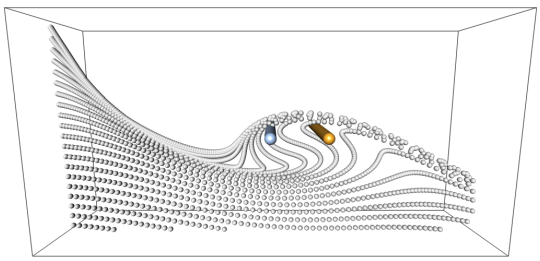
Figure 14.8.: Line integral convolution of the STUART VORTEX at $t = 0$. Streamlines show a center at the top of the domain.



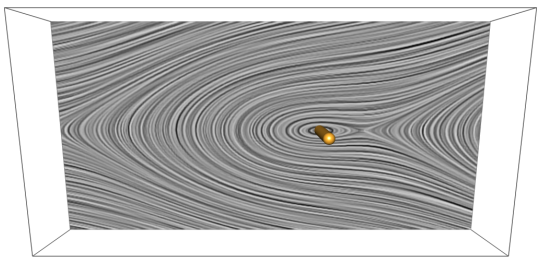
(a) Particles with $d_p = 0\,\mu m$ (massless) swirl around the massless pathline core (•). It can be extracted with [WSTH07] and is also a special case of our method.



(b) In this image a LIC is utilized to illustrate the vector field $\mathbf{u} - \mathbf{f} + r\,\mathbf{g}$, in which we extract the coreline (•) using the standard PV operator. Note that for $d_p = 0\,\mu m$ the response time is $r = 0$, according to Eq. (9.3). Thus, for the massless case we find standard pathline cores in $\mathbf{u} - \mathbf{f}$.

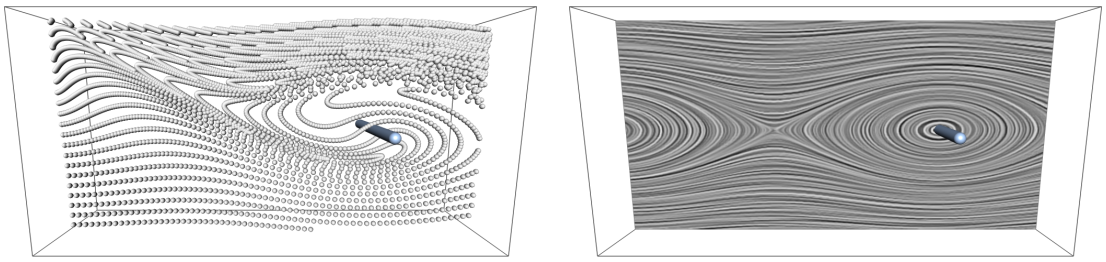


(c) Inertial particles with $d_p = 50\,\mu m$ first fall due to gravity and are then lifted up by the vortex. The common center of rotation is at the inertial coreline (•). The massless pathline core (•) is shown for reference. It is clearly off.



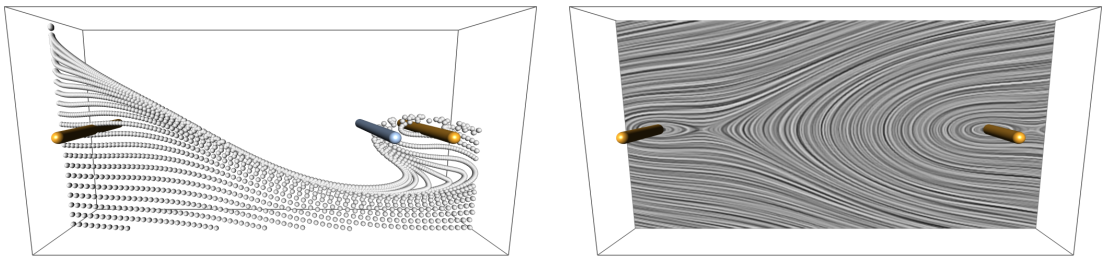
(d) Here, LIC is used to depict the vector field $\mathbf{u} - \mathbf{f} + r\,\mathbf{g}$ for $d_p = 50\,\mu m$. Note that inertial particles have their common coreline (•) later down the flow.

Figure 14.9.: The STUART VORTEX at time $t = 0$. Particles are seeded repeatedly at the left border of the image. Essentially, this resembles a particle-based streakline construction. Streaklines proved useful for the observation of swirling behavior of particles [WT10a].



(a) Particles with $d_p = 0\,\mu m$ (massless) and their massless pathline core (•).

(b) LIC illustration of the vector field $\mathbf{u} - \mathbf{f} + r\,\mathbf{g}$ for $d_p = 0\,\mu m$ (massless).



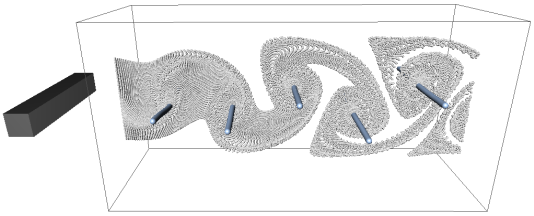
(c) For inertial particles with $d_p = 50\,\mu m$ a second inertial coreline appears.

(d) The second vortex is apparent in the LIC of the vector field $\mathbf{u} - \mathbf{f} + r\,\mathbf{g}$ for $d_p = 50\,\mu m$.

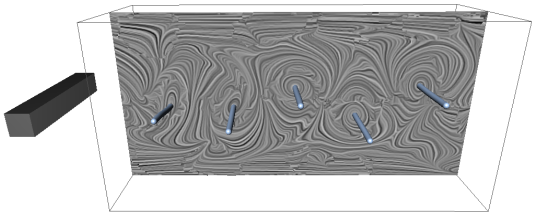
Figure 14.10.: The STUART VORTEX at time $t = 1$. All particles and their vortices move to the right over time. As the field is periodic, other vortices will enter from the left.

Square Cylinder (3D unsteady). The SQUARE CYLINDER sequence shows the development of a von Kármán vortex street, i.e., a periodic shedding of vortices in the unsteady flow around an obstacle. In this case, the obstacle is a confined square cylinder that is positioned symmetrically between two parallel walls. The origin of the data set was described earlier in Section 4.4.

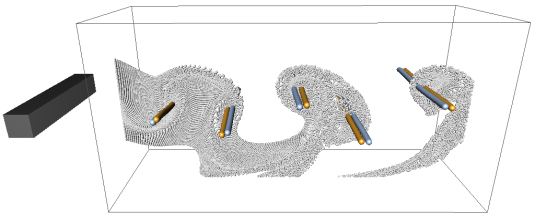
Fig. 14.11a shows the von Kármán vortex street for massless particles. The pathline cores are extracted by the parallel vectors operator on the field $\mathbf{u} - \mathbf{f}$, which is shown via LIC in Fig. 14.11b. As shown in Fig. 14.11c, the inertial particles with $d_p = 50\,\mu m$ are pulled down by the gravity with $\mathbf{g} = (0, -9.8065, 0)^{\mathrm{T}}$ and as a consequence, some of them exit the domain at the bottom. The remaining particles are lifted up and are trapped by the shed vortices. Their centers of rotation are shown in Fig. 14.11d, together with the massless cores for reference. It can be seen that the inertial corelines differ from the massless case. The inertial cores were extracted on the field $\mathbf{u} - \mathbf{f} + r\,\mathbf{g}$, which is visualized in Fig. 14.11d.



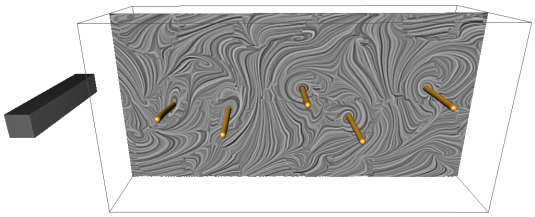
(a) Massless particles form a von Kármán vortex street in the wake of the square cylinder. The corelines of the shed vortices (•) are shown for $d_p = 0\,\mu m$.



(b) For the massless case, pathline cores (•) are extracted by the PV operator in the flow $\mathbf{u} - \mathbf{f}$, which is depicted here using LIC.



(c) Due to gravity, inertial particles, here with $d_p = 50\,\mu m$, do not form a perfectly symmetric vortex street. Their pathline cores (•) are shown to differ slightly from the massless case (•).



(d) The cores of swirling inertial particles (•) are extracted by applying the PV operator to the flow $\mathbf{u} - \mathbf{f} + r\,\mathbf{g}$. This flow is shown by LIC.

Figure 14.11.: Extraction results in the SQUARE CYLINDER sequence at time $t = 88$. Particles are released on the left hand side of the domain.

14.3. Implementation and Evaluation

14.3.1. Coreline Extraction and Filtering

In Section 14.1, we have shown that the extraction of inertial pathline cores can be reduced to a parallel vectors operation. Therefore, any of the existing PV extraction methods can be used, see Peikert and Roth [PR99] for a comprehensive overview. Because of this, our method can be easily integrated into standard visualization libraries. In fact, we implemented the extraction process in the visualization toolkit Amira [SWH05b] by using existing standard modules. Only the integration of inertial particle trajectories was precomputed in a separate tool, for which we used a fourth-order Runge-Kutta integrator.

Peikert and Roth [PR99] have shown an analytic parallel vectors solution for triangles, which is implemented in Amira and works as follows. First, a uniform grid is placed in the domain. Its resolution is shown for our data sets in Table 14.2. Following the recommendation of Weinkauf and Theisel [WSTH07], we used the resolution of the data sets. For every grid vertex that has complex eigenvalues, i.e., exhibits swirling behavior, the two vector fields are computed according to Table 14.1. Then, each face

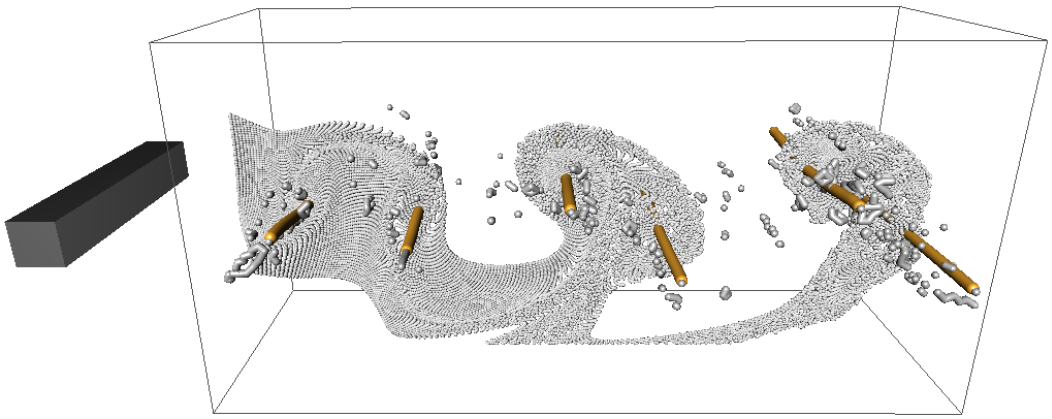


Figure 14.12.: The coreline extraction by parallel vectors creates a number of false positives (•), which are usually filtered. In this image, the filtering by approximate tangent alignment with the parallel vectors is disabled. The corelines are shown (•) in the SQUARE CYLINDER for $t = 94.2$.

of the grid cells is subdivided into two triangles at which the locations of parallelity are computed analytically. In a next step, the line segments of adjacent cells are merged if the tangent is aligned and the end points are identical.

Afterwards, the resulting line set is further filtered to only contain line segments that are tangential to the parallel vectors, which is standard procedure for Sujudi-Haimes. Then, to fill in gaps, lines search near their endpoints for nearby segments to join with. Eventually, lines are filtered by length by a threshold given in Table 14.2. An unfiltered result is shown in Fig. 14.12.

Data Set	Grid Resolution	Filter Length
MOVING CENTER	64×64×64	—
BEADS FLOW	64×64×128	—
HELI DESCENT	192×262×309	13.5
DELTA WING	250×125×100	4.3
HURRICANE	64×64×64	8.0
STUART VORTEX	128×64×64	—
SQUARE CYLINDER	192×64×48	1.0

Table 14.2.: Grid resolutions used for the parallel vector operator and the used filter lengths, given in the unit of the data set scale.

14.3.2. Performance

Our test system is equipped with an Intel Core i7-2600K CPU with 3.4 GHz and 24 GB RAM. Table 14.3 lists the timings for the three steps in the extraction process. Note that these steps are the same as for the massless Sujudi-Haimes [SH95] using parallel vectors [PR99] and its unsteady variant [WSTH07]. For unsteady data, first the feature flow field $\mathbf{f}$ is computed, which took for the grid resolutions that we used less than two seconds. Then, the pathline cores are extracted in our derived vector field, resulting in the inertial pathline cores. For this, the standard parallel vectors module of Amira is used. In summary, this took less than a minute on all our test data sets, most of them in a span of just a few seconds. The runtime depends not only on the grid resolution, but also on the presence of swirling. The DELTA WING for instance, has a high resolution, but only few voxels exhibit swirling behavior, i.e., have complex-conjugate eigenvalues in the Jacobian, thus large portions of the domain can safely be skipped. In the MOVING CENTER and BEADS FLOW, on the other hand, every voxel has swirling behavior. For these two data sets, the runtime linearly scales with the resolution. The extraction can be significantly accelerated, if parallel vectors are computed in a region of interest, which can be found by manual inspection of the parallel vector fields. In the SQUARE CYLINDER, for instance, this reduced the extraction time from $13.67\,s$ to $1.58\,s$. Finally, it is common to filter the extracted line geometry to only keep lines that are approximately tangential to the parallel vectors. This step is in the order of milliseconds. If the filtering was not required for a particular data set, we left the entry in the table blank.

In general, the overhead for the consideration of inertial properties is compared to the massless case *close to zero*. In its essence, only the constant vector field $r\,\mathbf{g}$ is added prior to the standard extraction. The overhead of adding the field linearly depends on the number of voxels. For our smallest data set (MOVING CENTER) the overhead is 0.327 milliseconds. In the largest data set (HELI DESCENT) the overhead goes up to 21.747 milliseconds. (The values are averaged over 500 measurements.) This corresponds to 0.002%–0.08% of the total extraction time (dependent on the data set).

Since all operations are local, the computation of the PV operator can be done iteratively per cell without memory overhead. In fact, the memory scales linearly with the size of the output coreline set.

For the time-periodic BEADS FLOW, the integration-based method took about 1.1 seconds. Since all particles are attracted by the same coreline, we only had to integrate one group of 5 particles. Once they converged, we continued the integration by one time period to record the complete coreline.

Data Set	Field $\mathbf{f}$	Coreline	Filtering
MOVING CENTER	—	15.21 s	—
BEADS FLOW	—	29.68 s	—
HELI DESCENT	—	57.18 s	2.27 ms
DELTA WING	—	6.43 s	0.48 ms
HURRICANE	—	1.49 s	2.49 ms
STUART VORTEX	1.87 s	7.79 s	—
SQUARE CYLINDER	1.08 s	13.67 s	4.30 ms

Table 14.3.: Timings for the individual computation steps: Computation of the feature flow field $\mathbf{f}$ (only for unsteady data), the extraction of the corelines in Amira, and the runtime of the additionally applied filtering, if it was necessary.

14.3.3. Limitations

Even for the massless case, the existing vortex coreline extractors are not able to handle all scenarios. The local cores of swirling particle motions method [WSTH07] only finds corelines that are moving along straight lines (or not moving at all), as it is Galilean invariant. Weinkauf and Theisel [WT10a] have shown that their extension of Sujudi and Haimes to unsteady flows [WSTH07], does not reveal the apparent attractor in the BEADS FLOW. The integration-based methods [WCW*09, WT10a], on the other hand, only detect attracting corelines. A simple data set in which none of the methods will work is a center moving along an ellipse. As our techniques are extensions of existing methods from the massless to the inertial case, they do not fill these gaps.

When considering time sequences, the coreline is extracted for each frame individually. A problem of standard local PV extractors is that they do not explicitly enforce frame coherence, which is apparent in the slightly oscillating corelines of the SQUARE CYLINDER sequence in the accompanying video of the corresponding paper [GT14]. In fact, we inherit this property from the massless case. If it is desired, a temporal smoothing can be applied in a post-process. However, in this work, we preferred to display the results as they were extracted.

Also Sujudi-Haimes cannot extract a coreline if a real eigenvalue is zero, i.e., $\det \mathbf{J} = 0$. This happens, for instance, in a center without movement in the non-swirling direction: $\mathbf{u}(x, y, z) = (-z, 0, x)^{\mathrm{T}}$. This case can be detected and then handled with 2D techniques, since in the swirling plane, spanned by the eigenvectors to the complex-conjugate eigenvalues, the velocity magnitude will vanish to zero at the coreline.

The advantage of the Sujudi-Haimes method in terms of computation effort is that it is

a local method. As a consequence, it detects corelines without regarding if particles could actually appear in a certain region of the flow. This, in turn, would actually require an integration of a large number of particles from the inlet zones to see if any of them is reaching the region in question. This has an implication for inertial vortex cores. Particles of a certain size might never reach higher altitudes due to gravity. Though, since our method is based on the local Sujudi-Haimes, vortex cores might be extracted that are never visited by particles of a certain size. This means, an additional (integration-based) filtering is in order. However, this filtering process is completely orthogonal to the coreline extraction and the performance advantages of being local are considerable.

14.4. Summary

In this chapter, we extended an established local vortex coreline extractor for massless particles to the inertial case. The problem, actually in 6D or 7D (space, velocity and optional time), reduces to a 3D or 4D parallel vectors operation, which can be solved efficiently using standard implementations. The method is therefore easily added to existing flow analysis tools. In fact, except for the integration of inertial particles, we could implement all components in Amira [SWH05b] by using existing modules. Moreover, we demonstrated the use of an integration-based extractor that detects attracting corelines.

The particle model we used is designed for the prediction of small particles. If the velocity magnitude is high compared to the acting gravity, the resulting inertial corelines are quite similar for the masses considerable by the simplified inertial equations of motion. In fact, differences are observable in particular for low wind speeds. For this reason, we synthetically altered some of our test data to showcase the extraction of mass-dependent corelines. Further sources for differences are additional forces, such as buoyancy or drag, which are typically neglected for small particles. The local vortex extraction methods we derived are *dependent* on the underlying equations of motion. A derivation of a coreline extraction for more general particle models or finding even a description that is *independent* of the underlying equations of motion are an avenue for future research. Also the visual analysis of where and why differences occur is interesting; not only for the analysis of vortex cores, but also for mass-dependent integral curves. The approximations made by the underlying equations of motion lead to a certain level of uncertainty. Considering the uncertainty of vortices and other features of inertial flow in the spirit of [OT12] is also an interesting topic. A more quantitative evaluation, including inertial particles and fluids of different densities might help to better understand the underlying physical effects. So far, we extended the

local Sujudi-Haimes and the integration-based particle density estimation. Another possibility to explore is to apply the streakline-based extraction method of Weinkauf and Theisel [WT10a].

15 Chapter 15. Conclusions

This thesis contributed a number of approaches to geometry-based and feature-based / topology-based flow visualization.

In the first part, we formulated the line selection problem in 3D flows as an optimization problem that balances the presentation of relevant information and occlusion avoidance. Starting with an initial set of steady lines that covers the domain, all lines are rendered with a varying opacity, which is subject to the minimization of a bounded-variable least-squares problem. This way, we obtain view-dependent opacities for the line segments, allowing a real-time free navigation while minimizing the danger of missing important structures in the visualization. Since most approaches to line data visualization focus on steady line fields, we extended our approach to sets of line fields such that significantly larger sets of line representatives can be handled. Furthermore, time coherence was addressed for animations, making this the first approach that solves the line selection problem for 3D time-dependent flow. In surface-based flow visualization, integral surfaces rapidly tend to expand, fold and produce vast amounts of occlusion. While silhouette enhancements and local transparency mappings proved useful for semi-transparent depictions, they still introduce visual clutter when surfaces grow more complex. Thus, we extended the concept of opacity optimization to surfaces to obtain the first global approach to the occlusion problem. Starting with a partition of the surfaces into patches, we compute per-patch opacity and interpolate it for final rendering across the surfaces. There are a number of aspects that deserve further research. At present, all our methods operate on a precomputed set of lines or surfaces. Thus, when zooming into the domain, no further detail is added. The scalability can still be improved and an inherent problem to all opacity-based approaches is the limited depth perception, which could be treated by adding further visual cues. Another unsolved problem is the selection of meaningful surfaces in unsteady flows.

The second part of the thesis was dedicated to novel rendering techniques for the

visualization of unsteady flows. In this area, ridges in finite-time Lyapunov exponent fields (FTLE) are frequently used as indicators for Lagrangian coherent structures, i.e., separators that divide the domain into sets of similar flow behavior. The ridges of Lagrangian scalar field may become arbitrarily thin, even though the underlying vector field might be discretized. Typically, FTLE fields are discretized onto grids and afterwards raycasted, which both introduce bias. We presented a progressive, view-dependent Monte Carlo-based technique that avoids grid discretization and ray marching errors completely, is consistent, and has a low memory consumption. Our approach is an application of rendering techniques for light transport in inhomogeneous participating media. In the rendering literature, a number of acceleration strategies exist that are applicable in our context as well. Another task related to unsteady flows is the tracking of vortices. Instead of the frequently used Galilean invariance, which is useful for translating vortices, we enforced a rotation invariance, i.e., the invariance of a vortex under a uniform-speed rotation of the underlying coordinate system around a fixed axis. We formulated a general approach to turn a Galilean invariant vortex concept into a rotation invariant one by simply adding a closed-form matrix to the Jacobian. In particular, we presented rotation invariant versions of the well-known Sujudi-Haimes, λ_2, and Q vortex criteria. An open challenge is the automatic selection of the rotation center, which we currently have selected by hand.

In the third part of the thesis, we studied the motion of finite-sized objects carried by fluids. Using the inertial flow map, we described mass-dependent trajectories in an abstract way, which was used to extend common geometry-based concepts, such as pathlines, streaklines, and timelines to the inertial case. Furthermore, we introduce a new class of curves, called masslines, which effectively visualizes the separation of masses. To further assess mass separation in 2D flows, we defined *finite-time mass separation*, a scalar field that measures at each point in the domain how quickly inertial particles separate when released from the same location but with slightly different mass. Extracting and visualizing the mass that induces the largest separation provides a simplified view on the critical masses. By using complementary coordinated views, we additionally visualized corresponding inertial particle trajectories in space-time by integral curves and surfaces. For a quantitative analysis, we plotted Euclidean and arc length-based distances to a reference particle over time, which allows to observe the temporal evolution of separation events. Using the fact that the trajectories of inertial particles can be described as tangent curves of a higher dimensional vector field, we presented a first approach to extend vector field topology to the inertial case. We conducted a full classification of the first-order critical points of this higher dimensional flow, and devised a method to their efficient extraction. Further, we interactively visualized the asymptotic behavior of finite-sized particles by a glyph visualization that encodes the outcome of any initial condition of the governing ODE,

i.e., for a varying initial position and/or initial velocity. We found that in steady 2D inertial flows, sources cannot exist, since at least two eigenvalues of the 4D Jacobian have negative real parts. This has implications for the source inversion problem, which is concerned with the recovery of the source of airborne or waterborne pollutions. If an inertial particle is observed in space, it is very difficult to recover where it came from, since inertial backward integration is extremely sensitive to slight changes in the observed velocity. Assuming that inertial particles are released with a given initial velocity (e.g., from rest), particles may reach a certain observation location only with a limited set of possible observable velocities. We showed that the set of initial positions that lead to the same location form a curve, which we extracted as tangent curve of a derived vector field. Most importantly, the derived vector field only involves forward integrated flow map gradients, which are much more stable to compute than backward trajectories. Finally, we focused on the vortical behavior of inertial particles. We developed two strategies for the extraction of the corelines that inertial particles swirl around. The first is to deduce the local swirling behavior from the autonomous inertial motion ODE, which eventually reduces to a parallel vectors operation. For the second strategy, we use particle density estimation to locate inertial attractors. With this, we were able to extract the cores of swirling inertial particle motion for both steady and unsteady 3D vector fields.

Our research on the source inversion problem led us to the notion of influence curves, which provided us a method to integrate inertial particles backward, though not quite as a traditional initial value problem. Instead, we prescribed the terminal value of the backward integration and constructed the path toward it. We intend to apply this sort of backward integration to conduct a segmentation of the domain into regions of coherent asymptotic behavior in backward direction. Further, we would like to experiment with more general particle models. Some of our proposed visualization concepts are independent of the underlying particle model, as they were formulated based on the abstract flow map. The other techniques would have to be revised for every new model. Using our proposed backward integration, we would like to study the correlation of inertial backward FTLE with particle settling in forward direction. In the CFD literature, there is evidence [SBR16] that the asymptotic behavior of inertial particles differs among aerosols and bubbles. In this thesis, we only extended a few of the many massless concepts that the visualization community has proposed over the years. There are many more concepts to extend in the future, including dense / texture-based methods or the extension of our rotation invariant vortices to the inertial case.

Appendix

A Appendix A.

Reformulation of Energy into Matrix Notation

This section describes the reformulation of our energy minimization into a standard quadratic programming problem of the form, as used in Section 3.2.4:

$$\text{minimize} \quad E = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x} + \text{const} \tag{A.1}$$

$$\text{subject to} \quad 0 \leq \mathbf{x}_i \leq 1 \tag{A.2}$$

with $\mathbf{x} = (\alpha_1, ..., \alpha_n)^{\mathrm{T}}$ being the vector of unknowns, i.e., the opacities. In the following, we consider the terms of the energy individually, for each with *bounded* variables $0 \leq \alpha_i \leq 1$.

Maximization of Opacity The energy term that maximizes the opacity is rewritten as follows:

$$\begin{aligned} E_p &= \frac{p}{2}\sum_{i=1}^{n}(\alpha_i - 1)^2 && \text{(A.3)} \\ &= \frac{p}{2}(\mathbf{x} - \mathbf{1})^{\mathrm{T}}(\mathbf{x} - \mathbf{1}) && \text{(A.4)} \\ &= \frac{p}{2}(\mathbf{x}^{\mathrm{T}}\mathbf{x} - 2\mathbf{x}^{\mathrm{T}}\mathbf{1} + \mathbf{1}^{\mathrm{T}}\mathbf{1}) && \text{(A.5)} \\ &= \frac{1}{2}\mathbf{x}^{\mathrm{T}}(p \cdot \mathbf{I})\mathbf{x} + (-p \cdot \mathbf{1})^{\mathrm{T}}\mathbf{x} + \text{const} && \text{(A.6)} \end{aligned}$$

Removal of Occlusion Occlusions are removed by penalizing unimportant objects in front of important ones. The corresponding energy is turned into matrix notation by:

$$
\begin{aligned}
E_q &= \frac{q}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i(1-g_i)^{\lambda}h_{ij}g_j)^2 && \text{(A.7)}\\
&= \frac{q}{2}\sum_{i=1}^{n}\alpha_i\cdot\alpha_i(1-g_i)^{2\lambda}\sum_{j=1}^{n}h_{ij}^2g_j^2 && \text{(A.8)}\\
&= \frac{q}{2}\mathbf{x}^{\mathrm{T}}\begin{bmatrix}\ddots & & \\ & \alpha_i\cdot(1-g_i)^{2\lambda}\sum_{j=1}^{n}h_{ij}^2g_j^2 & \\ & & \ddots\end{bmatrix} && \text{(A.9)}\\
&= \frac{q}{2}\mathbf{x}^{\mathrm{T}}\begin{bmatrix}\ddots & & \\ & (1-g_i)^{2\lambda}\sum_{j=1}^{n}h_{ij}^2g_j^2 & \\ & & \ddots\end{bmatrix}\mathbf{x} && \text{(A.10)}\\
&= \frac{1}{2}\mathbf{x}^{\mathrm{T}}(q\cdot\mathbf{W})\mathbf{x} && \text{(A.11)}
\end{aligned}
$$

with the diagonal matrix $\mathbf{W}_{ii}=(1-g_i)^{2\lambda}\sum_{j=1}^{n}h_{ij}^2g_j^2$.

Removal of Background Clutter Unimportant objects behind important ones are removed by minimizing:

$$
\begin{aligned}
E_r &= \frac{r}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i(1-g_i)^{\lambda}h_{ji}g_j)^2 && \text{(A.12)}\\
&= \frac{r}{2}\sum_{i=1}^{n}\alpha_i\cdot\alpha_i(1-g_i)^{2\lambda}\sum_{j=1}^{n}h_{ji}^2g_j^2 && \text{(A.13)}\\
&= \frac{r}{2}\mathbf{x}^{\mathrm{T}}\begin{bmatrix}\ddots & & \\ & \alpha_i\cdot(1-g_i)^{2\lambda}\sum_{j=1}^{n}h_{ji}^2g_j^2 & \\ & & \ddots\end{bmatrix} && \text{(A.14)}\\
&= \frac{r}{2}\mathbf{x}^{\mathrm{T}}\begin{bmatrix}\ddots & & \\ & (1-g_i)^{2\lambda}\sum_{j=1}^{n}h_{ji}^2g_j^2 & \\ & & \ddots\end{bmatrix}\mathbf{x} && \text{(A.15)}\\
&= \frac{1}{2}\mathbf{x}^{\mathrm{T}}(r\cdot\mathbf{V})\mathbf{x} && \text{(A.16)}
\end{aligned}
$$

with the diagonal matrix $\mathbf{V}_{ii}=(1-g_i)^{2\lambda}\sum_{j=1}^{n}h_{ji}^2g_j^2$.

Spatial Smoothness To ensure continuity, the opacities are spatially smoothed:

$$E_s = \frac{s}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{a_{ij}}{2}(\alpha_i-\alpha_j)^2 \tag{A.17}$$

$$= \frac{s}{2}(\mathbf{Dx})^{\mathrm{T}}(\mathbf{Dx}) \tag{A.18}$$

$$= \frac{1}{2}\mathbf{x}^{\mathrm{T}}(s\cdot\mathbf{D}^{\mathrm{T}}\mathbf{D})\mathbf{x} \tag{A.19}$$

Thereby, a_{ij} indicates adjacency between segments. The matrix $\mathbf{D}$ is the forward difference operator, thus $\mathbf{D}^{\mathrm{T}}\mathbf{D}$ becomes a Laplacian matrix. For line primitives, $\mathbf{D}^{\mathrm{T}}\mathbf{D}$ is tridiagonal. As an example, for a single line consisting of four segments, $\mathbf{D}^{\mathrm{T}}\mathbf{D}$ becomes:

$$\mathbf{D}^{\mathrm{T}}\mathbf{D} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \tag{A.20}$$

Temporal Smoothness The opacities can be smoothed over time, which is useful for animated geometry.

$$E_t = \frac{u}{2}\sum_{i=1}^{n}(\alpha_i-\alpha_i')^2 \tag{A.21}$$

$$= \frac{u}{2}(\mathbf{x}-\mathbf{x}')^{\mathrm{T}}(\mathbf{x}-\mathbf{x}') \tag{A.22}$$

$$= \frac{u}{2}(\mathbf{x}^{\mathrm{T}}\mathbf{x}-2\mathbf{x}^{\mathrm{T}}\mathbf{x}'+\mathbf{x}'^{\mathrm{T}}\mathbf{x}') \tag{A.23}$$

$$= \frac{1}{2}\mathbf{x}^{\mathrm{T}}(u\cdot\mathbf{I})\mathbf{x}+(-u\cdot\mathbf{x}')^{\mathrm{T}}\mathbf{x}+\mathrm{const} \tag{A.24}$$

with $\mathbf{x}'=(\alpha_1',...,\alpha_n')^{\mathrm{T}}$ being the corresponding opacities of the last iteration.

System Matrix Assembly Summing up the individual energy terms of the Eqs. (A.6), (A.11), (A.16) and (A.19), and expressing them in the standard QP form (A.1) yields for steady data:

$$\mathbf{Q} = p\cdot\mathbf{I}+q\cdot\mathbf{W}+r\cdot\mathbf{V}+s\cdot\mathbf{D}^{\mathrm{T}}\mathbf{D} \tag{A.25}$$

$$\mathbf{c} = (-p,\ldots,-p)^{\mathrm{T}} \tag{A.26}$$

For unsteady data (Section 4.3.2), we additionally apply the temporal smoothing from Eq. (A.24):

$$\mathbf{Q} = p\cdot\mathbf{I}+q\cdot\mathbf{W}+r\cdot\mathbf{V}+s\cdot\mathbf{D}^{\mathrm{T}}\mathbf{D}+u\cdot\mathbf{I} \tag{A.27}$$

$$\mathbf{c} = (-p-u\cdot\alpha_1',\ldots,-p-u\cdot\alpha_n')^{\mathrm{T}} \tag{A.28}$$

B Appendix B. Rotation Invariant Closed-Forms

In the following, we derive the closed-forms of the rotation invariant quantities that have been given in Eqs. (8.7)–(8.9) as:

$$\mathbf{u}_r = \mathbf{u} \tag{B.1}$$

$$\mathbf{u}_{rt} = \frac{\partial \mathbf{u}_r}{\partial t} = \mathbf{u}_t \tag{B.2}$$

$$\mathbf{J}_r = \mathbf{J} + \frac{1}{d}\,\mathbf{R}\,\mathbf{H}\,\mathbf{R}^{\mathrm{T}} \tag{B.3}$$

For $n = 2$, we introduced in Section 8.1 a non-isometric domain deformation $\mathbf{g}_p(\mathbf{x}, t)$ and its inverse $\mathbf{h}_p(\mathbf{x}, t)$

$$\mathbf{g}_p(\mathbf{x}, t) = \begin{pmatrix} \arctan \frac{y-y_0}{x-x_0} \\ \|\mathbf{x} - \mathbf{x}_0\| \end{pmatrix}\ ,\ \mathbf{h}_p(\mathbf{x}, t) = \mathbf{x}_0 + y \begin{pmatrix} \cos x \\ \sin x \end{pmatrix} \tag{B.4}$$

along with its domain transformed vector field $\mathbf{w}_p$:

$$\mathbf{w}_p(\mathbf{x}, t) = (\nabla \mathbf{h}_p(\mathbf{x}, t))^{-1} \cdot \mathbf{u}(\mathbf{h}_p(\mathbf{x}, t), t) \tag{B.5}$$

Since $\mathbf{h}_p$ is the inverse of $\mathbf{g}_p$ the following identity holds:

$$\mathbf{h}_p(\mathbf{g}_p(\mathbf{x}, t), t) = \mathbf{x} \tag{B.6}$$

In the following, we derive the closed-form expressions for the terms:

$$\mathbf{u}_r(\mathbf{x}, t) = (\nabla \mathbf{g}_p(\mathbf{x}, t))^{-1} \cdot \mathbf{w}_p(\mathbf{g}_p(\mathbf{x}, t), t) \tag{B.7}$$

$$\mathbf{u}_{rt}(\mathbf{x}, t) = (\nabla \mathbf{g}_p(\mathbf{x}, t))^{-1} \cdot \frac{\partial}{\partial t}\mathbf{w}_p(\mathbf{g}_p(\mathbf{x}, t), t) \tag{B.8}$$

$$\mathbf{J}_r(\mathbf{x}, t) = (\nabla \mathbf{g}_p(\mathbf{x}, t))^{-1} \cdot \nabla(\mathbf{w}_p(\mathbf{g}_p(\mathbf{x}, t), t)). \tag{B.9}$$

By applying the chain rule to the identity (B.6), we get

$$\nabla(\mathbf{h}_p(\mathbf{g}_p(\mathbf{x}, t), t)) = \nabla \mathbf{h}_p(\mathbf{g}_p(\mathbf{x}, t), t) \cdot \nabla \mathbf{g}_p(\mathbf{x}, t) = \nabla \mathbf{x} = \mathbf{I} \tag{B.10}$$

from which we get

$$(\nabla \mathbf{h}_p(\mathbf{g}_p(\mathbf{x},t),t))^{-1} = \nabla \mathbf{g}_p(\mathbf{x},t) \tag{B.11}$$

Evaluating the transformed vector field $\mathbf{w}_p(\mathbf{x},t)$ at a transformed location $\mathbf{g}_p(\mathbf{x},t)$ gives:

$$\mathbf{w}_p(\mathbf{g}_p(\mathbf{x},t),t) = (\nabla \mathbf{h}_p(\mathbf{g}_p(\mathbf{x},t),t))^{-1} \cdot \mathbf{u}(\mathbf{h}_p(\mathbf{g}_p(\mathbf{x},t),t),t)\ , \tag{B.12}$$

which simplifies by inserting (B.11) and the identity (B.6) to

$$\mathbf{w}_p(\mathbf{g}_p(\mathbf{x},t),t) = \nabla \mathbf{g}_p(\mathbf{x},t) \cdot \mathbf{u}(\mathbf{x},t). \tag{B.13}$$

Inserting (B.13) into (B.7) yields:

$$\begin{aligned}
\mathbf{u}_r(\mathbf{x},t) &= (\nabla \mathbf{g}_p(\mathbf{x},t))^{-1} \cdot \mathbf{w}_p(\mathbf{g}_p(\mathbf{x},t),t) && \text{(B.14)}\\
&= (\nabla \mathbf{g}_p(\mathbf{x},t))^{-1} \cdot \nabla \mathbf{g}_p(\mathbf{x},t) \cdot \mathbf{u}(\mathbf{x},t) && \text{(B.15)}\\
&= \mathbf{u}(\mathbf{x},t) && \text{(B.16)}
\end{aligned}$$

Similarly, inserting (B.13) into (B.8) yields with application of the product rule and $\frac{\partial}{\partial t}\mathbf{g}_p(\mathbf{x},t) = \mathbf{0}$:

$$\begin{aligned}
\mathbf{u}_{rt}(\mathbf{x},t) &= (\nabla \mathbf{g}_p(\mathbf{x},t))^{-1} \cdot \frac{\partial}{\partial t}\mathbf{w}_p(\mathbf{g}_p(\mathbf{x},t),t) && \text{(B.17)}\\
&= (\nabla \mathbf{g}_p(\mathbf{x},t))^{-1} \cdot \frac{\partial}{\partial t}\left[\nabla \mathbf{g}_p(\mathbf{x},t) \cdot \mathbf{u}(\mathbf{x},t)\right] && \text{(B.18)}\\
&= (\nabla \mathbf{g}_p(\mathbf{x},t))^{-1} \cdot \left(\mathbf{0}_{3,3} \cdot \mathbf{u}(\mathbf{x},t) + \nabla \mathbf{g}_p(\mathbf{x},t) \cdot \frac{\partial}{\partial t}\mathbf{u}(\mathbf{x},t)\right) && \text{(B.19)}\\
&= \mathbf{u}_t(\mathbf{x},t) && \text{(B.20)}
\end{aligned}$$

The derivation of $\mathbf{J}_r(\mathbf{x},t)$ requires a few more steps. For $n = 2$, we intend to express Eq. (B.9) in terms of

$$d = \|\mathbf{x} - \mathbf{x}_0\| \quad , \quad \mathbf{r} = \frac{\mathbf{x} - \mathbf{x}_0}{d} = \begin{pmatrix} \frac{x-x_0}{d} \\ \frac{y-y_0}{d} \end{pmatrix} \quad , \quad \mathbf{r}_p = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{r} = \begin{pmatrix} \frac{y-y_0}{d} \\ \frac{x_0-x}{d} \end{pmatrix} \tag{B.21}$$

and the 2D matrices

$$\mathbf{R} = (\mathbf{r}_p, \mathbf{r}) = \begin{pmatrix} \frac{y-y_0}{d} & \frac{x-x_0}{d} \\ \frac{x_0-x}{d} & \frac{y-y_0}{d} \end{pmatrix} \quad , \quad \mathbf{H} = \begin{pmatrix} -\mathbf{u}^{\mathrm{T}}\mathbf{r} & -\mathbf{u}^{\mathrm{T}}\mathbf{r}_p \\ \mathbf{u}^{\mathrm{T}}\mathbf{r}_p & 0 \end{pmatrix}. \tag{B.22}$$

First, we reformulate the factors in (B.9). The spatial gradient of $\mathbf{g}_p(\mathbf{x},t)$ is

$$\nabla \mathbf{g}_p(\mathbf{x},t) = \begin{pmatrix} \frac{y_0-y}{d^2} & \frac{x-x_0}{d^2} \\ \frac{x-x_0}{d} & \frac{y-y_0}{d} \end{pmatrix} \tag{B.23}$$

and its inverse is

$$(\nabla \mathbf{g}_p(\mathbf{x},t))^{-1} = \begin{pmatrix} y_0 - y & \frac{x-x_0}{d} \\ x - x_0 & \frac{y-y_0}{d} \end{pmatrix} = \mathbf{R} \begin{pmatrix} -d & 0 \\ 0 & 1 \end{pmatrix} . \tag{B.24}$$

Inserting (B.23) into (B.13) gives

$$\mathbf{w}_p(\mathbf{g}_p(\mathbf{x},t),t) = \begin{pmatrix} -\mathbf{u}^{\mathrm{T}}(\frac{\mathbf{r}_p}{d}) \\ \mathbf{u}^{\mathrm{T}}\mathbf{r} \end{pmatrix} \tag{B.25}$$

The spatial gradient of Eq. (B.25) then becomes

$$\begin{aligned} \nabla(\mathbf{w}_p(\mathbf{g}_p(\mathbf{x},t),t)) &= \begin{pmatrix} -\mathbf{u}_x{}^{\mathrm{T}}(\frac{\mathbf{r}_p}{d}) - (\frac{\mathbf{r}_p}{d})_x{}^{\mathrm{T}}\mathbf{u} & -\mathbf{u}_y{}^{\mathrm{T}}(\frac{\mathbf{r}_p}{d}) - (\frac{\mathbf{r}_p}{d})_y{}^{\mathrm{T}}\mathbf{u} \\ \mathbf{u}_x{}^{\mathrm{T}}\mathbf{r} + \mathbf{r}_x{}^{\mathrm{T}}\mathbf{u} & \mathbf{u}_y{}^{\mathrm{T}}\mathbf{r} + \mathbf{r}_y{}^{\mathrm{T}}\mathbf{u} \end{pmatrix} \\ &= \begin{pmatrix} -\frac{1}{d} & 0 \\ 0 & 1 \end{pmatrix} \cdot \mathbf{R}^{\mathrm{T}} \cdot \mathbf{J} + \begin{pmatrix} -(\frac{\mathbf{r}_p}{d})_x{}^{\mathrm{T}}\mathbf{u} & -(\frac{\mathbf{r}_p}{d})_y{}^{\mathrm{T}}\mathbf{u} \\ \mathbf{r}_x{}^{\mathrm{T}}\mathbf{u} & \mathbf{r}_y{}^{\mathrm{T}}\mathbf{u} \end{pmatrix} \end{aligned} \tag{B.26}$$

Substituting (B.24) and (B.26) in (B.9) yields:

$$\mathbf{J}_r(\mathbf{x},t) = (\nabla \mathbf{g}_p(\mathbf{x},t))^{-1} \cdot \nabla(\mathbf{w}_p(\mathbf{g}_p(\mathbf{x},t),t)) \tag{B.27}$$

$$= \mathbf{J} + \mathbf{R} \begin{pmatrix} -d & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} -(\frac{\mathbf{r}_p}{d})_x{}^{\mathrm{T}}\mathbf{u} & -(\frac{\mathbf{r}_p}{d})_y{}^{\mathrm{T}}\mathbf{u} \\ \mathbf{r}_x{}^{\mathrm{T}}\mathbf{u} & \mathbf{r}_y{}^{\mathrm{T}}\mathbf{u} \end{pmatrix} \tag{B.28}$$

$$= \mathbf{J} + \frac{1}{d}\mathbf{R} \begin{pmatrix} d^2 \cdot (\frac{\mathbf{r}_p}{d})_x{}^{\mathrm{T}}\mathbf{u} & d^2 \cdot (\frac{\mathbf{r}_p}{d})_y{}^{\mathrm{T}}\mathbf{u} \\ d \cdot \mathbf{r}_x{}^{\mathrm{T}}\mathbf{u} & d \cdot \mathbf{r}_y{}^{\mathrm{T}}\mathbf{u} \end{pmatrix} \tag{B.29}$$

Inserting $\mathbf{r}$ and $\mathbf{r}_p$ from Eq.(B.21) in (B.29) gives:

$$\begin{aligned} \mathbf{J}_r(\mathbf{x},t) &= \mathbf{J} + \frac{1}{d}\mathbf{R} \begin{pmatrix} d^2 \cdot \mathbf{u}^{\mathrm{T}} \left[\frac{1}{d^2} \begin{pmatrix} y - y_0 \\ x_0 - x \end{pmatrix} \right]_x & d^2 \cdot \mathbf{u}^{\mathrm{T}} \left[\frac{1}{d^2} \begin{pmatrix} y - y_0 \\ x_0 - x \end{pmatrix} \right]_y \\ d \cdot \mathbf{u}^{\mathrm{T}} \left[\frac{1}{d} \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} \right]_x & d \cdot \mathbf{u}^{\mathrm{T}} \left[\frac{1}{d} \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} \right]_y \end{pmatrix} \\ &= \mathbf{J} + \frac{1}{d}\mathbf{R}\frac{1}{d} \begin{pmatrix} m_{00} & m_{01} \\ m_{10} & m_{11} \end{pmatrix} \end{aligned} \tag{B.30}$$

with the coefficients of the matrix resulting from differentiation and subsequent simplification:

$$\begin{aligned} m_{00} &= \frac{1}{d}\mathbf{u}^{\mathrm{T}} \begin{pmatrix} -2(x_0 - x)(y_0 - y) \\ -(y_0 - y - x_0 + x)(y_0 - y + x_0 - x) \end{pmatrix} \\ &= -\frac{y - y_0}{d}\,\mathbf{u}^{\mathrm{T}} \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} - \frac{x - x_0}{d}\,\mathbf{u}^{\mathrm{T}} \begin{pmatrix} y - y_0 \\ x_0 - x \end{pmatrix} \\ &= -\mathbf{u}^{\mathrm{T}}\mathbf{r}(y - y_0) - \mathbf{u}^{\mathrm{T}}\mathbf{r}_p(x - x_0) \end{aligned} \tag{B.31}$$

$$
\begin{aligned}
m_{01} &= \frac{1}{d}\mathbf{u}^{\mathrm{T}} \begin{pmatrix} -(y_0 - y - x_0 + x)(y_0 - y + x_0 - x) \\ 2(x_0 - x)(y_0 - y) \end{pmatrix} \\
&= -\frac{x_0 - x}{d}\,\mathbf{u}^{\mathrm{T}} \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} - \frac{y - y_0}{d}\,\mathbf{u}^{\mathrm{T}} \begin{pmatrix} y - y_0 \\ x_0 - x \end{pmatrix} \\
&= -\mathbf{u}^{\mathrm{T}}\mathbf{r}(x_0 - x) - \mathbf{u}^{\mathrm{T}}\mathbf{r}_p(y - y_0)
\end{aligned} \tag{B.32}
$$

$$
\begin{aligned}
m_{10} &= \frac{1}{d}\mathbf{u}^{\mathrm{T}} \begin{pmatrix} (y_0 - y)^2 \\ (x_0 - x)(y_0 - y) \end{pmatrix} \\
&= -\frac{y - y_0}{d}\,\mathbf{u}^{\mathrm{T}} \begin{pmatrix} y_0 - y \\ x_0 - x \end{pmatrix} \\
&= \mathbf{u}^{\mathrm{T}}\mathbf{r}_p(y - y_0)
\end{aligned} \tag{B.33}
$$

$$
\begin{aligned}
m_{11} &= \frac{1}{d}\mathbf{u}^{\mathrm{T}} \begin{pmatrix} (x_0 - x)(y_0 - y) \\ (x_0 - x)^2 \end{pmatrix} \\
&= -\frac{x_0 - x}{d}\,\mathbf{u}^{\mathrm{T}} \begin{pmatrix} y_0 - y \\ x_0 - x \end{pmatrix} \\
&= \mathbf{u}^{\mathrm{T}}\mathbf{r}_p(x_0 - x)
\end{aligned} \tag{B.34}
$$

Placing the simplified forms (B.31), (B.32), (B.33) and (B.34) in (B.30) gives:

$$
\begin{aligned}
\mathbf{J}_r(\mathbf{x}, t) &= \mathbf{J} + \frac{1}{d}\mathbf{R}\frac{1}{d} \begin{pmatrix} -\mathbf{u}^{\mathrm{T}}\mathbf{r}(y - y_0) - \mathbf{u}^{\mathrm{T}}\mathbf{r}_p(x - x_0) & -\mathbf{u}^{\mathrm{T}}\mathbf{r}(x_0 - x) - \mathbf{u}^{\mathrm{T}}\mathbf{r}_p(y - y_0) \\ \mathbf{u}^{\mathrm{T}}\mathbf{r}_p(y - y_0) & \mathbf{u}^{\mathrm{T}}\mathbf{r}_p(x_0 - x) \end{pmatrix} \\
&= \mathbf{J} + \frac{1}{d}\mathbf{R} \begin{pmatrix} -\mathbf{u}^{\mathrm{T}}\mathbf{r} & -\mathbf{u}^{\mathrm{T}}\mathbf{r}_p \\ \mathbf{u}^{\mathrm{T}}\mathbf{r}_p & 0 \end{pmatrix} \cdot \begin{pmatrix} \frac{y - y_0}{d} & \frac{x_0 - x}{d} \\ \frac{x - x_0}{d} & \frac{y - y_0}{d} \end{pmatrix} \\
&= \mathbf{J} + \frac{1}{d}\,\mathbf{R}\,\mathbf{H}\,\mathbf{R}^{\mathrm{T}}
\end{aligned} \tag{B.35}
$$

The derivation for $n = 3$ is similar.

C Appendix C.

Eigenvalues with Zero Real Parts in the Inertial Spatio-Velocity Domain

To show the condition for $Re(f_{j,2}) = 0$ in Eq. (12.7), we compute

$$\begin{aligned} Re(f_{j,2}) &= Re\left(\frac{-1+\sqrt{1+4\,r\,(Re(e_j)+i\,Im(e_j))}}{2r}\right) && \text{(C.1)} \\ &= Re\left(\frac{-1+\sqrt{1+4\,r\,(-r\,Im(e_j)^2+i\,Im(e_j))}}{2r}\right) && \text{(C.2)} \\ &= Re\left(\frac{-1+\sqrt{-(i-2\,r\,Im(e_j))^2}}{2r}\right) && \text{(C.3)} \end{aligned}$$

where the step from (C.1) to (C.2) inserts the right-hand side of Eq. (12.7) and the step from (C.2) to (C.3) factors the radicand. For brevity we substitute in Eq (C.3)

$$x = 2\,r\,Im(e_j) \tag{C.4}$$

and thereby obtain:

$$Re(f_{j,2}) = Re\left(\frac{-1+\sqrt{-(i-x)^2}}{2r}\right) \tag{C.5}$$

With $\sqrt{-(i-x)^2} = 1 + x\,i$ as shown later, Eq. (C.5) further reduces to:

$$Re(f_{j,2}) = Re\left(\frac{-1+1+x\,i}{2r}\right) \tag{C.6}$$

Substituting Eq. (C.4) back into Eq. (C.6) gives:

$$\begin{aligned} Re(f_{j,2}) &= Re\left(\frac{2\,r\,Im(e_j)\,i}{2r}\right) && \text{(C.7)} \\ &= Re\,(i\,Im(e_j)) = 0 && \text{(C.8)} \end{aligned}$$

which proves the condition of Eq. (12.7).

In the following, we show that $\sqrt{-(i-x)^2} = 1 + x\,i$, which was used in the proof above. The square root of a complex number z is commonly defined as $\sqrt{z} = \sqrt{|z|}\frac{z+|z|}{|z+|z||}$. With $\sqrt{z} = \sqrt{-(i-x)^2}$ we have:

$$z = -(i-x)^2 = (-x^2+1) + (2x)\,i \tag{C.9}$$

$$|z| = \sqrt{(-x^2+1)^2 + (2x)^2} = \sqrt{(x^2+1)^2} = x^2+1 \tag{C.10}$$

Then, the square root of our complex number becomes:

$$\sqrt{-(i-x)^2} = \sqrt{x^2+1}\,\frac{-x^2+1+x^2+1+(2x)\,i}{|-x^2+1+x^2+1+(2x)\,i|} \tag{C.11}$$

$$= \sqrt{x^2+1}\,\frac{2+(2x)\,i}{\sqrt{4+4x^2}} \tag{C.12}$$

$$= \sqrt{x^2+1}\,\frac{2(1+x\,i)}{2\sqrt{x^2+1}} \tag{C.13}$$

$$= 1 + x\,i \tag{C.14}$$

D Appendix D.
Vector Field in which Influence Curves Appear as Tangent Curves

Recalling from Chapter 13, the family of all influence curves can be characterized as pathlines of an n dimensional unsteady vector field $\mathbf{h}(\mathbf{x}, t)$, starting at $t = t_0$:

$$\frac{d\,\mathbf{c}}{d\,\tau} = \mathbf{h}(\mathbf{c}(\mathbf{x}, \tau),\, t_0 + \tau). \tag{D.1}$$

In the following, we derive the vector field, in which influence curves appear as tangent curves (Eq. (13.6)):

$$\mathbf{h}(\mathbf{x}, t) = -\,{\phi_{\mathbf{x}}}^{-1} \left[\phi_{\mathbf{v}} \left(\frac{\mathbf{u}(\mathbf{x}, t_0) - \mathbf{v}_0}{r} + \mathbf{g}\right) + \phi_t\right] - \mathbf{v}_0 \tag{D.2}$$

where $\phi_{\mathbf{x}} = \phi_{\mathbf{x}}(\mathbf{x}, \mathbf{v}_0, t_0, t - t_0)$, $\phi_{\mathbf{v}} = \phi_{\mathbf{v}}(\mathbf{x}, \mathbf{v}_0, t_0, t - t_0)$ and $\phi_t = \phi_t(\mathbf{x}, \mathbf{v}_0, t_0, t - t_0)$ are inertial flow map derivatives that can be computed by forward integration, cf. Eq. (13.3).

We search for the unknown vector field $\mathbf{h}$ that fulfills

$$\lim_{\Delta\tau \to 0} \frac{\hat{\phi}(\mathbf{x} + \Delta\tau\,\mathbf{h},\, \mathbf{v}_0,\, t_0,\, \tau + \Delta\tau) - \hat{\phi}(\mathbf{x},\, \mathbf{v}_0,\, t_0,\, \tau)}{\Delta\tau} = \begin{pmatrix} \mathbf{0} \\ \mathbf{v}_r \\ 0 \end{pmatrix}. \tag{D.3}$$

Eq. (D.3) means that moving the starting point of the inertial particle integration from $\mathbf{x}$ to $\mathbf{x} + \Delta\tau\,\mathbf{h}$ and increasing the integration time from τ to $\tau + \Delta\tau$ should not change the location where the integration ends, while the change of the observed particle velocity is $\mathbf{v}_r$. In the following, we construct $\mathbf{x} + \Delta\tau\mathbf{h}$ at time $\tau + \Delta\tau$, i.e., the "next" location on the influence curve. For $\Delta\tau \to 0$ the flow map at $\tau + \Delta t$ is constructed by taking another "step" after the flow map at τ:

$$\begin{aligned} \hat{\phi}(\mathbf{x} + \Delta\tau\,\mathbf{h},\, \mathbf{v}_0,\, t_0,\, \tau + \Delta\tau) = \hat{\phi}(\mathbf{x} + \Delta\tau\,\mathbf{h},\, \mathbf{v}_0,\, t_0,\, \tau) \\ + \Delta\tau\,\hat{\mathbf{p}}(\hat{\phi}(\mathbf{x},\, \mathbf{v}_0,\, t_0,\, \tau)) \end{aligned} \tag{D.4}$$

The first summand of the right hand side contains the flow map at an infinitesimal close location in direction $\mathbf{h}$, which can be constructed from the flow map at $\mathbf{x}$ plus the flow map derivative in direction $\mathbf{h}$:

$$\begin{aligned}\hat{\phi}(\mathbf{x} + \Delta\tau\,\mathbf{h},\, \mathbf{v}_0,\, t_0,\, \tau) = \hat{\phi}(\mathbf{x},\, \mathbf{v}_0,\, t_0,\, \tau) \\ + \Delta\tau\,(\nabla\hat{\phi}(\mathbf{x},\, \mathbf{v}_0,\, t_0,\, \tau)) \begin{pmatrix} \mathbf{h} \\ \mathbf{0} \\ 0 \end{pmatrix}.\end{aligned} \tag{D.5}$$

The second summand samples the phase space $\hat{\mathbf{p}}$ after flow map integration. This is the tangent of the curve at the reached location, which is also the flow map derivative in start direction $\hat{\mathbf{p}}$ at t_0:

$$\begin{aligned}\hat{\mathbf{p}}(\hat{\phi}(\mathbf{x},\, \mathbf{v}_0,\, t_0,\, \tau)) &= (\nabla\hat{\phi}(\mathbf{x},\, \mathbf{v}_0,\, t_0,\, \tau))\; \hat{\mathbf{p}}(\mathbf{x}, \mathbf{v}_0, t_0) \\ &= (\nabla\hat{\phi}(\mathbf{x},\, \mathbf{v}_0,\, t_0,\, \tau)) \begin{pmatrix} \mathbf{v}_0 \\ \frac{\mathbf{u}(\mathbf{x}, t_0) - \mathbf{v}_0}{r} + \mathbf{g} \\ 1 \end{pmatrix}\end{aligned} \tag{D.6}$$

Inserting Eqs. (D.5) and (D.6) into Eq. (D.4), and inserting Eq. (D.4) in turn into Eq. (D.3) gives as condition for $\mathbf{h}$:

$$\lim_{\Delta\tau\to 0} \frac{\Delta\tau\,(\nabla\hat{\phi}(\mathbf{x},\, \mathbf{v}_0,\, t_0,\, \tau)) \left(\begin{pmatrix} \mathbf{h} \\ \mathbf{0} \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{v}_0 \\ \frac{\mathbf{u}(\mathbf{x}, t_0) - \mathbf{v}_0}{r} + \mathbf{g} \\ 1 \end{pmatrix} \right)}{\Delta\tau} = \begin{pmatrix} \mathbf{0} \\ \mathbf{v}_r \\ 0 \end{pmatrix} \tag{D.7}$$

Expanding the flow map gradient $\nabla\hat{\phi}$ using Eq. (13.3) and considering only the first n components of Eq. (D.7) (i.e., the spatial subspace) gives

$$\phi_{\mathbf{x}}\,(\mathbf{h} + \mathbf{v}_0) + \phi_{\mathbf{v}}\,\left(\frac{\mathbf{u}(\mathbf{x},\, t_0) - \mathbf{v}_0}{r} + \mathbf{g} \right) + \phi_t = \mathbf{0}\,, \tag{D.8}$$

which can be rearranged for $\mathbf{h}$ by inverting $\phi_{\mathbf{x}}$ to obtain to Eq. (D.2).

Appendix E.

Galilean Invariance of Inertial Vortex Cores

We show that our local coreline extraction method from Section 14.1.3 is Galilean invariant by first showing the Galilean invariance of the 6D unsteady acceleration $\nabla\tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}} + \tilde{\mathbf{u}}_t$. We follow closely the argumentation of Jan Sahner's [Sah09] proof of Galilean invariance of the 3D unsteady acceleration and Ronald Peikert's proof of Galilean invariance of the cores of swirling particle motion. The latter proof was also given in full detail in the thesis of Jan Sahner [Sah09].

Given is a time-dependent 6D vector field $\tilde{\mathbf{u}}(\tilde{\mathbf{x}}, t)$, in which inertial particle trajectories arise as tangent curves. A Galilean transformation moves the 6D point $\tilde{\mathbf{x}}$ at constant speed in a constant direction $\tilde{\mathbf{c}}_0$, yielding the transformed location $\tilde{\mathbf{y}}$:

$$\tilde{\mathbf{y}} = \tilde{\mathbf{x}} + t\,\tilde{\mathbf{c}}_0 \tag{E.1}$$

The 6D vector field $\tilde{\mathbf{u}}$ is transformed accordingly to $\tilde{\mathbf{w}}$:

$$\tilde{\mathbf{w}}(\tilde{\mathbf{y}}, t) = \tilde{\mathbf{u}}(\tilde{\mathbf{x}}, t) + \tilde{\mathbf{c}}_0 = \tilde{\mathbf{u}}(\tilde{\mathbf{y}} - t\,\tilde{\mathbf{c}}_0, t) + \tilde{\mathbf{c}}_0 \tag{E.2}$$

Since $\tilde{\mathbf{c}}_0$ is constant it follows that

$$\nabla\tilde{\mathbf{w}} = \nabla\tilde{\mathbf{u}} \tag{E.3}$$

A vortex extraction method is called *Galilean invariant* if it detects structures that are subject to Galilean transformations.

Galilean Invariance of 6D Unsteady Acceleration

In the following, we show that the acceleration $\tilde{\mathbf{u}}_t + \nabla\tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}}$ is Galilean invariant. First, we rewrite the transformed acceleration in Leibniz notation:

$$\tilde{\mathbf{w}}_t + \nabla\tilde{\mathbf{w}} \cdot \tilde{\mathbf{w}} = \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{w}}|_{\tilde{\mathbf{y}},t} + \nabla\tilde{\mathbf{w}}|_{\tilde{\mathbf{y}},t} \cdot \tilde{\mathbf{w}}(\tilde{\mathbf{y}}, t) \tag{E.4}$$

By applying the transformation from (E.2), we express the transformed vector field $\tilde{\mathbf{w}}$ in terms of the vector field $\tilde{\mathbf{u}}$ in the original frame:

$$\begin{aligned}\tilde{\mathbf{w}}_t + \nabla\tilde{\mathbf{w}} \cdot \tilde{\mathbf{w}} &= \frac{\mathrm{d}}{\mathrm{d}t}(\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} + \tilde{\mathbf{c}}_0) && \text{(E.5)}\\ &\quad +\nabla(\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} + \tilde{\mathbf{c}}_0) \cdot (\tilde{\mathbf{u}}(\tilde{\mathbf{y}} - t\,\tilde{\mathbf{c}}_0, t) + \tilde{\mathbf{c}}_0) && \text{(E.6)}\end{aligned}$$

Using the chain rule, we differentiate

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}(\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} + \tilde{\mathbf{c}}_0) &= -\nabla\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} \cdot \tilde{\mathbf{c}}_0 + \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} + \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{c}}_0 && \text{(E.7)}\\ &= -\nabla\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} \cdot \tilde{\mathbf{c}}_0 + \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} && \text{(E.8)}\end{aligned}$$

with $\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{c}}_0 = \mathbf{0}$. By substituting (E.8) in (E.5), and using the identity $\nabla(\tilde{\mathbf{c}}_0) = \mathbf{0}$ in (E.6), we obtain:

$$\begin{aligned}\tilde{\mathbf{w}}_t + \nabla\tilde{\mathbf{w}} \cdot \tilde{\mathbf{w}} &= -\nabla\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} \cdot \tilde{\mathbf{c}}_0 + \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} && \text{(E.9)}\\ &\quad +\nabla\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} \cdot (\tilde{\mathbf{u}}(\tilde{\mathbf{y}} - t\,\tilde{\mathbf{c}}_0, t) + \tilde{\mathbf{c}}_0) && \text{(E.10)}\end{aligned}$$

Expanding (E.10) leads to:

$$\begin{aligned}\tilde{\mathbf{w}}_t + \nabla\tilde{\mathbf{w}} \cdot \tilde{\mathbf{w}} &= -\nabla\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} \cdot \tilde{\mathbf{c}}_0 + \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} && \text{(E.11)}\\ &\quad +\nabla\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} \cdot \tilde{\mathbf{u}}(\tilde{\mathbf{y}} - t\,\tilde{\mathbf{c}}_0, t) + \nabla\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} \cdot \tilde{\mathbf{c}}_0 && \text{(E.12)}\end{aligned}$$

Now, the spatial directional derivative in $\tilde{\mathbf{c}}_0$ cancel out, thus we further simplify to:

$$\tilde{\mathbf{w}}_t + \nabla\tilde{\mathbf{w}} \cdot \tilde{\mathbf{w}} = \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} + \nabla\tilde{\mathbf{u}}|_{(\tilde{\mathbf{y}}-t\,\tilde{\mathbf{c}}_0,t)} \cdot \tilde{\mathbf{u}}(\tilde{\mathbf{y}} - t\,\tilde{\mathbf{c}}_0, t) \qquad \text{(E.13)}$$

We already expressed $\tilde{\mathbf{w}}$ in terms of $\tilde{\mathbf{u}}$. Also expressing $\tilde{\mathbf{y}}$ in terms of $\tilde{\mathbf{x}}$ by using the identity in (E.1), yields:

$$\begin{aligned}\tilde{\mathbf{w}}_t + \nabla\tilde{\mathbf{w}} \cdot \tilde{\mathbf{w}} &= \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{u}}|_{\tilde{\mathbf{x}},t} + \nabla\tilde{\mathbf{u}}|_{\tilde{\mathbf{x}},t} \cdot \tilde{\mathbf{u}}(\tilde{\mathbf{x}}, t) && \text{(E.14)}\\ &= \tilde{\mathbf{u}}_t + \nabla\tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}} && \text{(E.15)}\end{aligned}$$

The unsteady 6D acceleration is therefore Galilean invariant.

Galilean Invariance of Inertial Vortex Cores

Eqs. (14.15) named the seven eigenvectors of the 7D Jacobian $\hat{\mathbf{J}}$. One of the eigenvectors contains the feature flow field, which has always eigenvalue zero:

$$\hat{\mathbf{J}} \cdot \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \\ 1 \end{pmatrix} = \mathbf{0}$$

Splitting this matrix product into its spatial-velocity and time component gives:

$$\nabla\tilde{\mathbf{u}} \cdot \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix} + \tilde{\mathbf{u}}_t = \mathbf{0} \tag{E.16}$$

We will make use of this equality later.

In the following, we show that $\tilde{\mathbf{u}} - \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix}$ and the acceleration $\nabla\tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}} + \tilde{\mathbf{u}}_t$ are colinear.

Let $\tilde{\mathbf{u}} - \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix}$ be parallel to an eigenvector with real eigenvalue. Then, it is also parallel to the acceleration:

$$\tilde{\mathbf{u}} - \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix} \parallel \nabla\tilde{\mathbf{u}} \cdot \left(\tilde{\mathbf{u}} - \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix} \right) \tag{E.17}$$

$$= \nabla\tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}} - \nabla\tilde{\mathbf{u}} \cdot \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix} \tag{E.18}$$

$$= \nabla\tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}} + \tilde{\mathbf{u}}_t \tag{E.19}$$

The step from Eq. (E.18) to Eq. (E.19) used Eq. (E.16).

Next, we show that the converse is also true. Let the acceleration be parallel to an eigenvector with real eigenvalue. Then, it is mapped to the same one-dimensional subspace as $\tilde{\mathbf{u}} - \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix}$:

$$\nabla\tilde{\mathbf{u}} \cdot (\nabla\tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}} + \tilde{\mathbf{u}}_t) \parallel \nabla\tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}} + \tilde{\mathbf{u}}_t \tag{E.20}$$

$$= \nabla\tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}} - \nabla\tilde{\mathbf{u}} \cdot \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix} \tag{E.21}$$

$$= \nabla\tilde{\mathbf{u}} \cdot \left(\tilde{\mathbf{u}} - \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix} \right) \tag{E.22}$$

Hence, $\nabla\tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}} + \tilde{\mathbf{u}}_t$ and $\tilde{\mathbf{u}} - \begin{pmatrix} \mathbf{f} \\ \mathbf{0}_3 \end{pmatrix}$ are colinear. Thus, we inherit the Galilean invariance of the 6D unsteady acceleration.

Bibliography

[ABG*03] AKÇELIK V., BIROS G., GHATTAS O., LONG K. R., VAN BLOEMEN WAANDERS B.: A variational finite element method for source inversion for convective-diffusive transport. *Finite Elements in Analysis and Design 39*, 8 (2003), 683–705.

[AD16] AMENT M., DACHSBACHER C.: Anisotropic ambient volume shading. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 1015–1024.

[AMHH08] AKENINE-MÖLLER T., HAINES E., HOFFMAN N.: *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008.

[Ang10] ANGILELLA J.-R.: Dust trapping in vortex pairs. *Physica D: Nonlinear Phenomena 239* (Sept. 2010), 1789–1797.

[AOGJ15] AGRANOVSKY A., OBERMAIER H., GARTH C., JOY K. I.: A multi-resolution interpolation scheme for pathline based Lagrangian flow representations. In *Proc. SPIE, Visual Data Analysis Conference* (2015).

[APH*03] AGRAWALA M., PHAN D., HEISER J., HAYMAKER J., KLINGNER J., HANRAHAN P., TVERSKY B.: Designing effective step-by-step assembly instructions. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (2003), 828–837.

[AS84] ABRAHAM R. H., SHAW C. D.: *Dynamics – The Geometry of Behaviour*. The Visual Mathematics Library. Aerial Press, Incorporated, 1984.

[ATR*08] ANNEN T., THEISEL H., RÖSSL C., ZIEGLER G., SEIDEL H.-P.: Vector field contours. In *Proc. Graphics Interface* (2008), pp. 97–105.

[BBC*09] BENZI R., BIFERALE L., CALZAVARINI E., LOHSE D., TOSCHI F.: Velocity-gradient statistics along particle trajectories in turbulent flows: The refined similarity hypothesis in the Lagrangian frame. *Phys. Rev. E 80* (Dec 2009), 066318.

[BBC*11] BEC J., BIFERALE L., CENCINI M., LANOTTE A. S., TOSCHI F.: Spatial and velocity statistics of inertial particles in turbulent flows. *Journal of Physics: Conference Series 333*, 1 (2011), 012003.

[BBK*08] BORDÁS R., BENDICKS C., KUHN R., WUNDERLICH B., THÉVENIN D., MICHAELIS B.: Coloured tracer particles employed for 3D-PTV in gas flows. In *Proc. International Symposium on Flow Visualization (ISFV)* (July 2008).

[BCP*12] BRAMBILLA A., CARNECKY R., PEIKERT R., VIOLA I., HAUSER H.: Illustrative flow visualization: State of the art, trends and challenges. In *EuroGraphics 2012 State of the Art Reports (STARs)* (2012), pp. 75–94.

[BDH92] BOYCE W. E., DIPRIMA R. C., HAINES C. W.: *Elementary differential equations and boundary value problems*, vol. 9. Wiley New York, 1992.

[BE10] BALACHANDAR S., EATON J. K.: Turbulent dispersed multiphase flow. *Annual Review of Fluid Mechanics 42*, 1 (2010), 111–133.

[Bet37] BETZ A.: The ground effect on lifting propellers. *Zeitschrift für angewandte Mathematik und Mechanik 17*, 2 (1937).

[BFMW12] BÜRGER K., FRAEDRICH R., MERHOF D., WESTERMANN R.: Instant visitation maps for interactive visualization of uncertain particle trajectories. In *Proc. SPIE 8294, Visualization and Data Analysis* (2012), SPIE, p. 82940P.

[BFTW09] BÜRGER K., FERSTL F., THEISEL H., WESTERMANN R.: Interactive streak surface visualization on the GPU. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2009) 15*, 6 (November-December 2009), 1259–1266.

[BG07] BRUCKNER S., GRÖLLER E.: Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1344–1351.

[BGT12] BARAKAT S. S., GARTH C., TRICOCHE X.: Interactive computation and rendering of finite-time Lyapunov exponent fields. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (2012), 1368–1380.

[BH02] BROWN J. K. M., HOVMØLLER M. S.: Aerial dispersal of pathogens on the global and continental scales and its impact on plant disease. *Science 297*, 5581 (2002), 537–541.

[BH15] BRAMBILLA A., HAUSER H.: Expressive seeding of multiple stream surfaces for interactive flow exploration. *Computers & Graphics 47* (2015), 123–134.

[BHD*15] BISWAS A., HE W., DENG Q., CHEN C.-M., SHEN H.-W., MACHIRAJU R., RANGARAJAN A.: An uncertainty-driven approach to vortex analy-

sis using oracle consensus and spatial proximity. In *Proc. IEEE Pacific Visualization Symposium* (Hangzhou, China, 2015).

[BHDH05] BADIA A. E., HA-DUONG T., HAMDI A.: Identification of a point source in a linear advection-dispersion-reaction equation: Application to a pollution source problem. *Inverse Problems 21*, 3 (2005), 1121.

[BHP14] BEYER J., HADWIGER M., PFISTER H.: A survey of GPU-based large-scale volume visualization. In *Proc. EuroVis State of the Art Reports* (2014).

[BJB*12] BHATIA H., JADHAV S., BREMER P., CHEN G., LEVINE J. A., NONATO L. G., PASCUCCI V.: Flow visualization with quantified spatial and temporal errors using edge maps. *IEEE Transactions on Visualization and Computer Graphics 18*, 9 (2012), 1383–1396.

[BKC*13] BORGO R., KEHRER J., CHUNG D. H. S., MAGUIRE E., LARAMEE R. S., HAUSER H., WARD M., CHEN M.: Glyph-based visualization: Foundations, design guidelines, techniques and applications. In *EuroGraphics 2013 State of the Art Reports (STARs)* (2013), pp. 39–63.

[BKC15] BARTOŇ M., KOSINKA J., CALO V. M.: Stretch-minimising stream surfaces. *Graphical Models 79* (2015), 12 – 22.

[BLRV01] BOFFETTA G., LACORATA G., REDAELLI G., VULPIANI A.: Detecting barriers to transport: a review of different techniques. *Physica D: Nonlinear Phenomena 159* (2001), 58–70.

[BNPB13] BHATIA H., NORGARD G., PASCUCCI V., BREMER P.-T.: The Helmholtz-Hodge decomposition–A survey. *IEEE Transactions on Visualization and Computer Graphics 19*, 8 (2013), 1386–1404.

[Bor11] BORDÁS R.: *Optical measurements in disperse two-phase flows: Application to rain formation in cumulus clouds.* PhD thesis, University of Magdeburg, 2011.

[BP02] BAUER D., PEIKERT R.: Vortex tracking in scale-space. In *Proc. Symposium on Data Visualisation* (2002), pp. 233–240.

[BPKB14] BHATIA H., PASCUCCI V., KIRBY R. M., BREMER P.-T.: Extracting features from time-dependent vector fields using internal reference frames. *Computer Graphics Forum (Proc. EuroVis) 33*, 3 (2014), 21–30.

[BR10] BRUNTON S. L., ROWLEY C. W.: Fast computation of FTLE fields for unsteady flows: A comparison of methods. *Chaos 20* (2010), 017503.

[BRGIG*14] BALSA RODRÍGUEZ M., GOBBETTI E., IGLESIAS GUITIÁN J., MAKHINYA

M., MARTON F., PAJAROLA R., SUTER S.: State-of-the-art in compressed GPU-based direct volume rendering. *CGF 33*, 6 (2014), 77–100.

[BRR05] BOANO F., REVELLI R., RIDOLFI L.: Source identification in river pollution problems: A geostatistical approach. *Water Resources Research 41*, 7 (2005), WS07023.

[BS95] BANKS D. C., SINGER B. A.: A predictor-corrector technique for visualizing unsteady flow. *IEEE Transactions on Visualization and Computer Graphics 1* (1995), 151–163.

[BS13] BISWAS A., SHEN H.-W.: Evaluation of stream surfaces using error quantification metrics. In *IS&T/SPIE Electronic Imaging* (2013), International Society for Optics and Photonics, pp. 90170Z–90170Z.

[BSDW12] BACHTHALER S., SADLO F., DACHSBACHER C., WEISKOPF D.: Space-time visualization of dynamics in Lagrangian coherent structures of time-dependent 2D vector fields. In *Proc. International Conference on Information Visualization Theory and Applications (IVAPP)* (2012), pp. 573–583.

[BT13] BARAKAT S. S., TRICOCHE X.: Adaptive refinement of the flow map using sparse samples. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE SciVis) 19*, 12 (2013), 2753–2762.

[BWF*10] BORN S., WIEBEL A., FRIEDRICH J., SCHEUERMANN G., BARTZ D.: Illustrative stream surfaces. *IEEE TVCG 16*, 6 (2010), 1329–1338.

[BZ89] BÜCHNER J., ZELENYI L. M.: Regular and chaotic charged particle motion in magnetotaillike field reversals: 1. Basic theory of trapped motion. *Journal of Geophysical Research: Space Physics (1978–2012) 94*, A9 (1989), 11821–11842.

[CB11] CANDELARESI S., BRANDENBURG A.: Decay of helical and nonhelical magnetic knots. *Phys. Rev. E 84* (2011), 016406.

[CBP*14] CHILDS H., BIERSDORFF S., POLIAKOFF D., CAMP D., MALONY A. D.: Particle advection performance over varied architectures and workloads. In *IEEE International Conference on High Performance Computing, Goa, India* (2014).

[CCK07] CHEN Y., COHEN J., KROLIK J.: Similarity-guided streamline placement with error evaluation. *IEEE Transactions on Visualization and Computer Graphics 13* (2007), 1448–1455.

[CCT72] CARTER L. L., CASHWELL E. D., TAYLOR W. M.: Monte Carlo sampling

with continuously varying cross sections along flight paths. *Nuclear Science and Engineering 48*, 4 (1972), 403–411.

[CFK*10] CARTWRIGHT J. H. E., FEUDEL U., KÁROLYI G., MOURA A., PIRO O., TÉL T.: Dynamics of finite-size particles in chaotic fluid flows. In *Nonlinear Dynamics and Chaos: Advances and Perspectives*, Thiel M., Kurths J., Romano M. C., Károlyi G., Moura A., (Eds.), Understanding Complex Systems. Springer Berlin Heidelberg, 2010, pp. 51–87.

[CFM*13] CARNECKY R., FUCHS R., MEHL S., JANG Y., PEIKERT R.: Smart transparency for illustrative visualization of complex flow surfaces. *IEEE TVCG 19*, 5 (2013), 838–851.

[CGP*10] CASCIOLA C. M., GUALTIERI P., PICANO F., SARDINA G., TROIANI G.: Dynamics of inertial particles in free jets. *Physica Scripta 2010*, T142 (2010), 014001.

[CH97] CAI W., HENG P.-A.: Principal stream surfaces. In *IEEE Visualization* (Los Alamitos, CA, USA, 1997), IEEE Computer Society Press, pp. 75–ff.

[CKC08] CHOW F. K., KOSOVIC B., CHAN S.: Source inversion for contaminant plume dispersion in urban environments using building-resolving simulations. *Journal of Applied Meteorology and Climatology 47*, 6 (2008), 1553–1572.

[CL93] CABRAL B., LEEDOM L.: Imaging vector fields using line integral convolution. *Computer Graphics (Proc. SIGGRAPH) 27* (1993), 263–272.

[CL96] COLEMAN T. F., LI Y.: A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM J. on Optimization 6*, 4 (1996), 1040–1058.

[CM11] CORREA C. D., MA K.-L.: Visibility histograms and visibility-driven transfer functions. *IEEE TVCG 17*, 2 (2011), 192–204.

[CML*07] CHEN G., MISCHAIKOW K., LARAMEE R. S., PILARCZYK P., ZHANG E.: Vector field editing and periodic orbit extraction using morse decomposition. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 769–785.

[COJ15] CHANDLER J., OBERMAIER H., JOY K. I.: Interpolation-based pathline tracing in particle-based flow visualization. *IEEE Transactions on Visualization and Computer Graphics 21*, 1 (2015), 68–80.

[Col68] COLEMAN W. A.: Mathematical verification of a certain Monte Carlo sampling technique and applications of the technique to radiation transport

problems. *Nuclear Science and Engineering 32*, 1 (1968), 76–81.

[CPC90] CHONG M. S., PERRY A. E., CANTWELL B. J.: A general classification of three-dimensional flow fields. *Physics of Fluids A: Fluid Dynamics 2*, 5 (1990), 765–777.

[CPP*05] CEREZO E., PÉREZ F., PUEYO X., SERON F. J., SILLION F. X.: A survey on participating media rendering techniques. *The Visual Computer 21*, 5 (2005), 303–328.

[CQB99] CUCITORE R., QUADRIO M., BARON A.: On the effectiveness and limitations of local criteria for the identification of a vortex. *European Journal of Mechanics - B/Fluids 18*, 2 (1999), 261–282.

[CSBI05] CAMARRI S., SALVETTI M.-V., BUFFONI M., IOLLO A.: Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate Reynolds numbers. In *XVII Congresso di Meccanica Teorica ed Applicata* (2005).

[CSC07] CORREA C., SILVER D., CHEN M.: Illustrative deformation for data exploration. *IEEE TVCG 13*, 6 (2007), 1320–1327.

[CSFP12] CARNECKY R., SCHINDLER B., FUCHS R., PEIKERT R.: Multi-layer illustrative dense flow visualization. *Computer Graphics Forum 31*, 3pt1 (2012), 895–904.

[CST98] CROWE C., SOMMERFIELD M., TSUJI Y.: *Multiphase Flows with Droplets and Particles*. CRC Press, 1998.

[CWM*09] CHAN M.-Y., WU Y., MAK W.-H., CHEN W., QU H.: Perception-based transparency optimization for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1283–1290.

[CWW13] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 5 (2013), 152:1–152:11.

[CYY*11] CHEN C.-K., YAN S., YU H., MAX N., MA K.-L.: An illustrative visualization framework for 3D vector fields. *Computer Graphics Forum 30*, 7 (2011), 1941–1951.

[DC04] DANCA M.-F., CHEN G.: Bifurcation and chaos in a complex model of dissipative medium. *International Journal of Bifurcation and Chaos 14*, 10 (2004), 3409–3447.

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Sugges-

tive contours for conveying shape. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (2003), 848–855.

[dLvL99] DE LEEUW W., VAN LIERE R.: Collapsing flow topology using area metrics. In *Proceedings of the Conference on Visualization* (1999), VIS '99, pp. 349–354.

[DUS*11] DEE D. P., UPPALA S. M., SIMMONS A. J., BERRISFORD P., POLI P., KOBAYASHI S., ANDRAE U., BALMASEDA M. A., BALSAMO G., BAUER P., BECHTOLD P., BELJAARS A. C. M., VAN DE BERG L., BIDLOT J., BORMANN N., DELSOL C., DRAGANI R., FUENTES M., GEER A. J., HAIMBERGER L., HEALY S. B., HERSBACH H., HÓLM E. V., ISAKSEN L., K0.5ALLBERG P., KÖHLER M., MATRICARDI M., MCNALLY A. P., MONGE-SANZ B. M., MORCRETTE J.-J., PARK B.-K., PEUBEY C., DE ROSNAY P., TAVOLATO C., THÉPAUT J.-N., VITART F.: The ERA-Interim reanalysis: configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society 137*, 656 (2011), 553–597.

[DWE02] DIEPSTRATEN J., WEISKOPF D., ERTL T.: Transparency in interactive technical illustrations. *Computer Graphics Forum 21*, 3 (2002), 317–325.

[EBRI09] EVERTS M. H., BEKKER H., ROERDINK J. B. T. M., ISENBERG T.: Depth-dependent halos: Illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics 15* (2009), 1299–1306.

[EBRI11] EVERTS M. H., BEKKER H., ROERDINK J. B. T. M., ISENBERG T.: Illustrative line styles for flow visualization. In *Pacific Graphics – Short Papers* (2011), The Eurographics Association.

[EBRI15] EVERTS M. H., BEKKER H., ROERDINK J. B. T. M., ISENBERG T.: Interactive illustrative line styles and line style transfer functions for flow visualization. *ArXiv e-prints* (2015). 1503.05787.

[EHS13] EICHELBAUM S., HLAWITSCHKA M., SCHEUERMANN G.: LineAO – improved three-dimensional line rendering. *IEEE Transactions on Visualization and Computer Graphics 19*, 3 (2013), 433–445.

[ELC*12a] EDMUNDS M., LARAMEE R. S., CHEN G., MAX N., ZHANG E., WARE C.: Surface-based flow visualization. *Computers & Graphics 36*, 8 (2012), 974–990.

[ELC*12b] EDMUNDS M., LARAMEE R. S., CHEN G., ZHANG E., MAX N.: Advanced, automatic stream surface seeding and filtering. In *Theory and Practice of Computer Graphics* (2012), pp. 53–60.

[ELM*12] EDMUNDS M., LARAMEE R., MALKI R., MASTERS I., CROFT T., CHEN G., ZHANG E.: Automatic stream surface seeding: A feature centered approach. *Computer Graphics Forum (Proc. EuroVis) 31*, 3 (2012), 1095–1104.

[FH15] FARAZMAND M., HALLER G.: The Maxey–Riley equation: Existence, uniqueness and regularity of solutions. *Nonlinear Analysis: Real World Applications 22* (2015), 98–106.

[FI08] FURUYA S., ITOH T.: A streamline selection technique for integrated scalar and vector visualization. In *IEEE Visualization Poster Session* (2008).

[FPH*08] FUCHS R., PEIKERT R., HAUSER H., SADLO F., MUIGG P.: Parallel Vectors Criteria for Unsteady Flow Vortices. *IEEE Transactions on Visualization and Computer Graphics 14*, 3 (2008), 615–626.

[FPS*08] FUCHS R., PEIKERT R., SADLO F., ALSALLAKH B., GRÖLLER M. E.: Delocalized unsteady vortex region detectors. In *Proc. Vision, Modeling and Visualization* (2008), pp. 81–90.

[FSME14] FREY S., SADLO F., MA K.-L., ERTL T.: Interactive progressive visualization with space-time error control. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2397–2406.

[FW12] FRAEDRICH R., WESTERMANN R.: Motion visualization of large particle simulations. In *Proceedings of IS&T/SPIE Electronic Imaging 2012, Conference on Visualization and Data Analysis* (2012), SPIE, pp. 82940Q–1 – 12.

[FWT08] FREDERICH O., WASSEN E., THIELE F.: Prediction of the flow around a short wall-mounted cylinder using LES and DES. *Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM) 3*, 3-4 (2008), 231–247.

[GA10] GREEN-ARMYTAGE P.: A colour alphabet and the limits of colour coding. *JAIC-Journal of the International Colour Association 5* (2010).

[GBC*13] GALTIER M., BLANCO S., CALIOT C., COUSTET C., DAUCHET J., HAFI M. E., EYMET V., FOURNIER R., GAUTRAIS J., KHUONG A., PIAUD B., TERRÉE G.: Integral formulation of null-collision Monte Carlo algorithms. *Journal of Quantitative Spectroscopy and Radiative Transfer 125* (2013), 57–68.

[GBWT11] GÜNTHER T., BÜRGER K., WESTERMANN R., THEISEL H.: A view-

dependent and inter-frame coherent visualization of integral lines using screen contribution. *Proc. Vision, Modeling, and Visualization (VMV)* (2011), 215–222.

[GDN98] GRIEBEL M., DORNSEIFER T., NEUNHOEFFER T.: *Numerical Simulation in Fluid Dynamics, a Practical Introduction*. SIAM, 1998.

[GGTH07] GARTH C., GERHARDT F., TRICOCHE X., HAGEN H.: Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization) 13*, 6 (2007), 1464–1471.

[GJ10] GARTH C., JOY K. I.: Fast, memory-efficient cell location in unstructured grids for visualization. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization) 16*, 6 (2010), 1541–1550.

[GKKT13] GÜNTHER T., KUHN A., KUTZ B., THEISEL H.: Mass-dependent integral curves in unsteady vector fields. *Computer Graphics Forum (Proc. EuroVis) 32*, 3 (2013), 211–220.

[GKT*08] GARTH C., KRISHNAN H., TRICOCHE X., TRICOCHE T., JOY K. I.: Generation of accurate integral surfaces in time-dependent vector fields. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization) 14*, 6 (2008), 1404–1411.

[GKT16] GÜNTHER T., KUHN A., THEISEL H.: MCFTLE: Monte Carlo rendering of finite-time Lyapunov exponent fields. *Computer Graphics Forum (Proc. EuroVis) 35*, 3 (2016), to appear.

[GLL91] GLOBUS A., LEVIT C., LASINSKI T.: A tool for visualizing the topology of three-dimensional vector fields. In *Proc. IEEE Visualization* (1991), pp. 33–40.

[GLT*07] GARTH C., LARAMEE R. S., TRICOCHE X., SCHNEIDER J., HAGEN H.: Extraction and visualization of swirl and tumble motion from engine simulation data. In *Topology-based Methods in Visualization*, Visualization and Mathematics. Springer Berlin Heidelberg, 2007, pp. 121–135.

[GMG01] GRAFTIEAUX L., MICHARD M., GROSJEAN N.: Combining PIV, POD and vortex identification algorithms for the study of unsteady turbulent swirling flows. *Measurement Science and Technology 12*, 9 (2001), 1422.

[GPPMn15] GARABOA-PAZ D., PÉREZ-MUÑUZURI V.: A method to calculate finite-time Lyapunov exponents for inertial particles in incompressible flows. *Nonlinear Processes in Geophysics 22*, 5 (2015), 571–577.

[GRT13] GÜNTHER T., RÖSSL C., THEISEL H.: Opacity optimization for 3D line fields. *ACM Transaction on Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 120:1–120:8.

[GRT14] GÜNTHER T., RÖSSL C., THEISEL H.: Hierarchical opacity optimization for sets of 3D line fields. *Computer Graphics Forum (Proc. Eurographics) 33*, 2 (2014), 507–516.

[GSFT16] GÜNTHER T., SCHULZE M., FRIEDERICI A., THEISEL H.: Visualizing volcanic clouds in the atmosphere and their impact on air traffic. *IEEE Computer Graphics and Applications 36*, 3 (2016), 36–47.

[GSM*14] GÜNTHER T., SCHULZE M., MARTINEZ ESTURO J., RÖSSL C., THEISEL H.: Opacity optimization for surfaces. *Computer Graphics Forum (Proc. EuroVis) 33*, 3 (2014), 11–20.

[GST16] GÜNTHER T., SCHULZE M., THEISEL H.: Rotation invariant vortices for flow visualization. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization 2015) 22*, 1 (2016), 817–826.

[GT14] GÜNTHER T., THEISEL H.: Vortex cores of inertial particles. *IEEE Trans. on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization) 20*, 12 (2014), 2535–2544.

[GT15] GÜNTHER T., THEISEL H.: Finite-time mass separation for comparative visualizations of inertial particles. *Computer Graphics Forum (Proc. EuroVis) 34*, 3 (2015), 471–480.

[GT16a] GÜNTHER T., THEISEL H.: Inertial steady 2D vector field topology. *Computer Graphics Forum (Proc. Eurographics) 35*, 2 (2016), 455–466.

[GT16b] GÜNTHER T., THEISEL H.: Source inversion by forward integration in inertial flows. *Computer Graphics Forum (Proc. EuroVis) 35*, 3 (2016), to appear.

[GTS*04] GARTH C., TRICOCHE X., SALZBRUNN T., BOBACH T., SCHEUERMANN G.: Surface techniques for vortex visualization. In *Proc. Joint Eurographics - IEEE TCVG Conference on Visualization* (Aire-la-Ville, Switzerland, Switzerland, 2004), VISSYM'04, Eurographics Association, pp. 155–164.

[Hal60] HALTON J.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik 2*, 1 (1960), 84–90.

[Hal01] HALLER G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D: Nonlinear Phenomena 149*, 4

(Mar. 2001), 248–277.

[Hal05] HALLER G.: An objective definition of a vortex. *Journal of Fluid Mechanics 525* (2005), 1–26.

[Hal15] HALLER G.: Lagrangian coherent structures. *Annual Review of Fluid Mechanics 47* (2015), 137–162.

[HGH*10] HUMMEL M., GARTH C., HAMANN B., HAGEN H., JOY K. I.: IRIS: Illustrative rendering for integral surfaces. *IEEE TVCG 16*, 6 (2010), 1319–1328.

[HH89] HELMAN J. L., HESSELINK L.: Representation and display of vector field topology in fluid flow data sets. *Computer 22*, 8 (1989), 27–36.

[HH91] HELMAN J. L., HESSELINK L.: Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications 11* (May 1991), 36–46.

[HL93] HOSCHEK J., LASSER D.: *Fundamentals of Computer Aided Geometric Design*. AK Peters, 1993.

[HLNW11] HLAWATSCH M., LEUBE P. C., NOWAK W., WEISKOPF D.: Flow radar glyphs – Static visualization of unsteady flow with uncertainty. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization) 17*, 12 (2011), 1949–1958.

[HLSR08] HADWIGER M., LJUNG P., SALAMA C. R., ROPINSKI T.: Advanced illumination techniques for GPU volume raycasting. In *ACM SIGGRAPH ASIA Courses* (2008).

[HNW93] HAIRER E., NØRSETT S. P., WANNER G.: *Solving Ordinary Differential Equations I (2Nd Revised. Ed.): Nonstiff Problems*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. *ACM Transaction on Graphics (Proc. SIGGRAPH Asia) 27*, 5 (2008), 130:1–130:8.

[HS08] HALLER G., SAPSIS T.: Where do inertial particles go in fluid flows? *Physica D: Nonlinear Phenomena 237* (May 2008), 573–583.

[HS11] HALLER G., SAPSIS T.: Lagrangian coherent structures and the smallest finite-time Lyapunov exponent. *Chaos 21*, 2 (2011), 023115.

[HSJW14] HLAWATSCH M., SADLO F., JANG H., WEISKOPF D.: Pathline glyphs. *Computer Graphics Forum (Proc. Eurographics) 33*, 2 (2014), 497–506.

[HSW11] HLAWATSCH M., SADLO F., WEISKOPF D.: Hierarchical line integration. *IEEE Transactions on Visualization and Computer Graphics 17*, 8 (2011),

1148–1163.

[Hul92] HULTQUIST J. P. M.: Constructing stream surfaces in steady 3D vector fields. In *Proc. Visualization* (1992), pp. 171–178.

[Hun87] HUNT J. C. R.: Vorticity and vortex dynamics in complex turbulent flows. *Transactions on Canadian Society for Mechanical Engineering (Proc. CANCAM) 11*, 1 (1987), 21–35.

[HY00] HALLER G., YUAN G.: Lagrangian coherent structures and mixing in two-dimensional turbulence. *Physica D: Nonlinear Phenomena 147*, 3–4 (2000), 352–370.

[IFP96] INTERRANTE V., FUCHS H., PIZER S.: Illustrating transparent surfaces with curvature-directed strokes. In *Proc. Conference on Visualization* (1996), pp. 211–ff.

[Jar08] JAROSZ W.: *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego, La Jolla, CA, USA, 2008.

[JB10] JANSEN J., BAVOIL L.: Fourier opacity mapping. In *Proc. Symposium on Interactive 3D Graphics and Games* (2010), pp. 165–172.

[JCWD14] JU T., CHENG M., WANG X., DUAN Y.: A robust parity test for extracting parallel vectors in 3D. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE SciVis) 20*, 12 (2014), 2526–2534.

[JDA07] JUDD T., DURAND F., ADELSON E.: Apparent ridges for line drawing. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007).

[JH95] JEONG J., HUSSAIN F.: On the identification of a vortex. *Journal of Fluid Mechanics 285* (1995), 69–94.

[JL97] JOBARD B., LEFER W.: Creating evenly-spaced streamlines of arbitrary density. *Proc. Eurographics Workshop on Visualization in Scientific Computing 7* (1997), 45–55.

[JL00] JOBARD B., LEFER W.: Unsteady flow visualization by animating evenly-spaced streamlines. *Computer Graphics Forum (Proc. Eurographics) 19*, 3 (2000), 31–39.

[JL01] JOBARD B., LEFER W.: Multiresolution flow visualization. *WSCG 2001 Conference Proceedings* (2001), 33–37.

[JMT02a] JIANG M., MACHIRAJU R., THOMPSON D.: Geometric verification of swirling features in flow fields. In *Proc. IEEE Visualization* (2002), pp. 307–314.

[JMT02b] JIANG M., MACHIRAJU R., THOMPSON D.: A novel approach to vortex core region detection. In *Proc. Symposium on Data Visualisation (VISSYM)* (2002), pp. 217–226.

[JNT*11] JAROSZ W., NOWROUZEZAHRAI D., THOMAS R., SLOAN P.-P., ZWICKER M.: Progressive photon beams. *ACM Transaction on Graphics (SIGGRAPH Asia) 30*, 6 (2011), 181:1–181:12.

[KER*14] KUHN A., ENGELKE W., RÖSSL C., HADWIGER M., THEISEL H.: Time line cell tracking for the approximation of Lagrangian coherent structures with subgrid accuracy. *Computer Graphics Forum 33*, 1 (2014), 222–234.

[KF12] KULLA C., FAJARDO M.: Importance sampling techniques for path tracing in participating media. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering) 31*, 4 (2012), 1519–1528.

[KGG*10] KONOPKA P., GROOSS J.-U., GÜNTHER G., PLOEGER F., POMMRICH R., MÜLLER R., LIVESEY N.: Annual cycle of ozone at and above the tropical tropopause: observations versus simulations with the Chemical Lagrangian Model of the Stratosphere (CLaMS). *Atmospheric Chemistry and Physics 10*, 1 (2010), 121–132.

[KGH*14] KŘIVÁNEK J., GEORGIEV I., HACHISUKA T., VÉVODA P., ŠIK M., NOWROUZEZAHRAI D., JAROSZ W.: Unifying points, beams, and paths in volumetric light transport simulation. *ACM Transaction on Graphics (Proc. SIGGRAPH) 33*, 4 (2014), 103:1–103:13.

[KGJ09] KRISHNAN H., GARTH C., JOY K.: Time and streak surfaces for flow visualization in large time-varying data sets. *Visualization and Computer Graphics, IEEE Transactions on 15*, 6 (2009), 1267–1274.

[KGM*07] KONOPKA P., GÜNTHER G., MÜLLER R., DOS SANTOS F. H. S., SCHILLER C., RAVEGNANI F., ULANOVSKY A., SCHLAGER H., VOLK C. M., VICIANI S., PAN L. L., MCKENNA D.-S., RIESE M.: Contribution of mixing to upward transport across the tropical tropopause layer (TTL). *Atmospheric Chemistry and Physics 7*, 12 (2007), 3285–3308.

[KGP*13] KÖHLER B., GASTEIGER R., PREIM U., THEISEL H., GUTBERLET M., PREIM B.: Semi-automatic vortex extraction in 4D PC-MRI cardiac blood flow data using line predicates. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization) 19*, 12 (2013), 2773–2782.

[KGRK14] KUTZ B. M., GÜNTHER T., RUMPF A., KUHN A.: Numerical examination of a model rotor in brownout conditions. In *Proceedings of the American*

Helicopter Society (May 2014), no. AHS2014-000343 in AHS 70th Annual Forum.

[KHNH11] KASTEN J., HOTZ I., NOACK B. R., HEGE H.-C.: On the extraction of long-living features in unsteady fluid flows. In *Topological Methods in Data Analysis and Visualization*, Mathematics and Visualization. Springer Berlin Heidelberg, 2011, pp. 115 – 126.

[KHNH12] KASTEN J., HOTZ I., NOACK B. R., HEGE H.-C.: Vortex merge graphs in two-dimensional unsteady flow fields. In *EuroVis - Short Papers* (2012), pp. 1–5.

[KKKK12] KUTZ B. M., KOWARSCH U., KESSLER M., KRÄMER E.: Numerical investigation of helicopter rotors in ground effect. In *30th AIAA Applied Aerodynamics Conference* (2012).

[KKKW05] KRÜGER J., KIPFER P., KONDRATIEVA P., WESTERMANN R.: A particle system for interactive visualization of 3D flows. *IEEE Transactions on Visualization and Computer Graphics 11* (2005), 744–756.

[KPB12] KROES T., POST F. H., BOTHA C. P.: Exposure render: An interactive photo-realistic volume rendering framework. *PLoS ONE 7*, 7 (07 2012), e38586.

[KPH*09] KASTEN J., PETZ C., HOTZ I., NOACK B., HEGE. H.-C.: Localized finite-time Lyapunov exponent for unsteady flow analysis. In *Proceedings of Vision, Modeling and Visualization* (2009), pp. 265–274.

[KRHH11] KASTEN J., REININGHAUS J., HOTZ I., HEGE H.-C.: Two-dimensional time-dependent vortex regions based on the acceleration magnitude. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE SciVis) 17*, 12 (2011), 2080–2087.

[KRWT12] KUHN A., RÖSSL C., WEINKAUF T., THEISEL H.: A benchmark for evaluating FTLE computations. In *Proceedings of 5th IEEE Pacific Visualization Symposium (PacificVis 2012)* (Songdo, Korea, March 2012), pp. 121–128.

[KSW04] KIPFER P., SEGAL M., WESTERMANN R.: UberFlow: a GPU-based particle engine. In *Proc. ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware* (New York, NY, USA, 2004), ACM, pp. 115–122.

[KSW*12] KARCH G. K., SADLO F., WEISKOPF D., MUNZ C.-D., ERTL T.: Visualization of advection-diffusion in unsteady fluid flow. *Computer Graphics Forum 31* (2012), 1105–1114.

[KSWE15] KARCH G. K., SADLO F., WEISKOPF D., ERTL T.: Visualization of 2D

unsteady flow using streamline-based concepts in space-time. *Journal of Visualization 19*, 1 (2015), 115–128.

[LCD06] LUFT T., COLDITZ C., DEUSSEN O.: Image enhancement by unsharp masking the depth buffer. *ACM Trans. Graph. (Proc. SIGGRAPH) 25*, 3 (2006), 1206–1213.

[LDG98] LÖFFELMANN H., DOLEISCH H., GRÖLLER E.: Visualizing dynamical systems near critical points. In *In Spring Conference on Computer Graphics and its Applications* (1998), pp. 175–184.

[LEG*08] LARAMEE R. S., ERLEBACHER G., GARTH C., THEISEL H., TRICOCHE X., WEINKAUF T., WEISKOPF D.: Applications of texture-based flow visualization. *Engineering Applications of Computational Fluid Mechanics (EACFM) 2*, 3 (2008), 264–274.

[LGP14] LAWONN K., GÜNTHER T., PREIM B.: Coherent view-dependent streamlines for understanding blood flow. In *EuroVis - Short Papers* (2014), pp. 19–23.

[LHD*04] LARAMEE R. S., HAUSER H., DOLEISCH H., VROLIJK B., POST F. H., WEISKOPF D.: The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum 23*, 2 (2004), 203–221.

[LHS08] LI L., HSIEH H.-H., SHEN H.-W.: Illustrative streamline placement and visualization. *IEEE Pacific Visualization Symposium 2008* (2008), 79–86.

[LHZP07] LARAMEE R., HAUSER H., ZHAO L., POST F.: Topology-based flow visualization, the state of the art. In *Topology-based Methods in Visualization*, Hauser H., Hagen H., Theisel H., (Eds.), Mathematics and Visualization. Springer Berlin Heidelberg, 2007, pp. 1–19.

[Llo82] LLOYD S. P.: Least square quantization in PCM. *IEEE Information Theory 28*, 2 (1982), 129–137.

[LLPH15] LAWONN K., LUZ M., PREIM B., HANSEN C.: Illustrative visualization of vascular models for static 2D representations. In *Proc. Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2015), Springer International Publishing, pp. 399–406.

[LMC92] LANDAHL M. T., MOLLO-CHRISTENSEN E.: *Turbulence and random processes in fluid mechanics.* Cambridge University Press, 1992.

[LMG06] LIU Z., MOORHEAD R., GRONER J.: An advanced evenly-spaced streamline placement algorithm. *IEEE Transactions on Visualization and Com-*

puter Graphics 12 (2006), 965–972.

[LMSC11] LEE T.-Y., MISHCHENKO O., SHEN H.-W., CRAWFIS R.: View point evaluation and streamline filtering for flow visualization. In *Proc. IEEE Pacific Visualization* (2011), pp. 83–90.

[LRR00] LODHA S., RENTERIA J., ROSKIN K.: Topology preserving compression of 2D vector fields. In *In Proc. IEEE Visualization* (2000), pp. 343–350.

[LS07] LI L., SHEN H.-W.: Image-based streamline generation and rendering. *IEEE Transactions on Visualization and Computer Graphics 13* (2007), 630–640.

[LSP14] LU K., SHEN H.-W., PETERKA T.: Scalable computation of stream surfaces on large scale vector fields. In *International Conference for High Performance Computing, Networking, Storage and Analysis* (2014), pp. 1008–1019.

[LTH08] LI G.-S., TRICOCHE X., HANSEN C.: Physically-based dye advection for flow visualization. *Computer Graphics Forum 27*, 3 (2008), 727–734.

[LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bi-directional path tracing. *Proc. Compugraphics* (1993), 145–153.

[MAD05] MEBARKI A., ALLIEZ P., DEVILLERS O.: Farthest point seeding for efficient placement of streamlines. In *IEEE Visualization* (2005), pp. 479–486.

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 99–108.

[MBS*04] MAHROUS K., BENNETT J., SCHEUERMANN G., HAMANN B., JOY K. I.: Topological segmentation in three-dimensional vector fields. *IEEE Transactions on Visualization and Computer Graphics 10*, 2 (2004), 198–205.

[MBZ06] MOGRABI E., BAR-ZIV E.: On the asymptotic solution of the Maxey-Riley equation. *Physics of Fluids 18*, 5 (2006).

[MCHM10] MARCHESIN S., CHEN C.-K., HO C., MA K.-L.: View-dependent streamlines for 3D vector fields. *IEEE Transactions on Visualization and Computer Graphics 16* (2010), 1578–1586.

[MCTB11] MAULE M., COMBA J. L., TORCHELSEN R. P., BASTOS R.: A survey of raster-based transparency techniques. *Computers & Graphics 35*, 6 (2011), 1023–1034.

[MET*15] MCLOUGHLIN T., EDMUNDS M., TONG C., LARAMEE R. S., MASTERS I., CHEN G., MAX N., YEH H., ZHANG E.: Visualization of input parameters

for stream and pathline seeding. *International Journal of Advanced Computer Science and Applications 6*, 4 (2015), 124–135.

[MHB*00] MEISSNER M., HUANG J., BARTZ D., MUELLER K., CRAWFIS R.: A practical evaluation of popular volume rendering algorithms. In *Proc. IEEE Symposium on Volume Visualization* (2000), pp. 81–90.

[MJL*13] MCLOUGHLIN T., JONES M. W., LARAMEE R. S., MALKI R., MASTERS I., HANSEN C. D.: Similarity measures for enhancing interactive streamline seeding. *IEEE Transactions on Visualization and Computer Graphics 19*, 8 (2013), 1342–1353.

[MKG*02] MCKENNA D. S., KONOPKA P., GROOSS J.-U., GÜNTHER G., MÜLLER R., SPANG R., OFFERMANN D., ORSOLINI Y.: A new Chemical Lagrangian Model of the Stratosphere (CLaMS) 1. formulation of advection and mixing. *Journal of Geophysical Research: Atmospheres 107*, D16 (2002), DOI: 10.1029/2000JD000114.

[MKW08] MAPES P., KENT R., WOOD R.: *DoD Helicopter Mishaps FY85-05: Findings and Recommendations*. Tech. rep., U.S. Air Force, 2008.

[MLP*10] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum 29*, 6 (September 2010), 1807–1829.

[MLZ10] MCLOUGHLIN T., LARAMEE R. S., ZHANG E.: Constructing streak surfaces for 3D unsteady vector fields. In *Proc. Spring Conference on Computer Graphics* (2010), pp. 17–26.

[MR83] MAXEY M. R., RILEY J. J.: Equation of motion for a small rigid sphere in a nonuniform flow. *Physics of Fluids 26*, 4 (1983), 883–889.

[MSE14] MACHADO G. M., SADLO F., ERTL T.: Image-based streamsurfaces. In *Proc. SIBGRAPI Conference on Graphics, Patterns and Images* (Washington, DC, USA, 2014), IEEE Computer Society, pp. 343–350.

[MSRT13a] MARTINEZ ESTURO J., SCHULZE M., RÖSSL C., THEISEL H.: Global selection of stream surfaces. *Computer Graphics Forum (Proc. Eurographics) 32*, 2 (2013), 113–122.

[MSRT13b] MARTINEZ ESTURO J., SCHULZE M., RÖSSL C., THEISEL H.: Poisson-based tools for flow visualization. In *IEEE PacificVis* (2013), pp. 241–248.

[MTHG03] MATTAUSCH O., THEUSSL T., HAUSER H., GRÖLLER E.: Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines. In *Proc. Spring Conference on Computer Graphics (SSCG)* (2003),

ACM, pp. 213–222.

[MTP*02] MACPHEE A. G., TATE M. W., POWELL C. F., YUE Y., RENZI M. J., ERCAN A., NARAYANAN S., FONTES E., WALTHER J., SCHALLER J., GRUNER S. M., WANG J.: X-ray imaging of shock waves generated by high-pressure fuel sprays. *Science 295*, 5558 (2002), 1261–1263.

[MWS13] MA J., WANG C., SHENE C.-K.: Coherent view-dependent streamline selection for importance-driven flow visualization. *Proc. SPIE 8654, Visualization and Data Analysis* (2013).

[ND04] NIENHAUS M., DÖLLNER J.: Blueprints: Illustrating architecture and technical parts using hardware-accelerated non-photorealistic rendering. In *Proc. Graphics Interface* (2004), pp. 49–56.

[NLL*12] NOUANESENGSY B., LEE T.-Y., LU K., SHEN H.-W., PETERKA T.: Parallel particle advection and FTLE computation for time-varying flow fields. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Los Alamitos, CA, USA, 2012), pp. 61:1–61:11.

[NSJ14] NOVÁK J., SELLE A., JAROSZ W.: Residual ratio tracking for estimating attenuation in participating media. *ACM Transaction on Graphics (SIGGRAPH Asia) 33*, 6 (2014), 179:1–179:11.

[OHH15] ONU K., HUHN F., HALLER G.: LCS tool: a computational platform for Lagrangian coherent structures. *Journal of Computational Science 7* (2015), 26–36.

[Oku70] OKUBO A.: Horizontal dispersion of floatable particles in the vicinity of velocity singularities such as convergences. *Deep Sea Research and Oceanographic Abstracts 17*, 3 (1970), 445–454.

[OOG08] OUELLETTE N. T., O'MALLEY P. J. J., GOLLUB J. P.: Transport of finite-sized particles in chaotic flow. *Phys. Rev. Lett. 101* (2008), 174504.

[OT12] OTTO M., THEISEL H.: Vortex analysis in uncertain vector fields. *Computer Graphics Forum (Proc. EuroVis) 31*, 3 (2012), 1035–1044.

[PD09] PENG J., DABIRI J. O.: Transport of inertial particles by Lagrangian coherent structures: Application to predator–prey interaction in jellyfish feeding. *Journal of Fluid Mechanics 623* (3 2009), 75–84.

[PHR99] PAGENDARM H.-G., HENNE B., RUTTEN M.: Detecting vortical phenomena in vector data by medium-scale correlation. In *Proc. Visualization* (1999), pp. 409–552.

[PKM*12] PLOEGER F., KONOPKA P., MÜLLER R., FUEGLISTALER S., SCHMIDT T., MANNERS J. C., GROOSS J.-U., GÜNTHER G., FORSTER P. M., RIESE M.: Horizontal transport affecting trace gas seasonality in the Tropical Tropopause Layer (TTL). *Journal of Geophysical Research: Atmospheres 117*, D09303 (2012), DOI: 10.1029/2011JD017267.

[PKPH09] PETZ C., KASTEN J., PROHASKA S., HEGE H.-C.: Hierarchical vortex regions in swirling flow. *Computer Graphics Forum (Proc. EuroVis) 28*, 3 (2009), 863–870.

[Pop00] POPE S. B.: *Turbulent Flows*. Cambridge University Press & Beijing World Publishing Corporation, 2000.

[Pop04] POPINET S.: Free computational fluid dynamics. *Cluster World 2*, 6 (2004).

[POS*11] PAGOT C., OSMARI D., SADLO F., WEISKOPF D., ERTL T., COMBA J.: Efficient parallel vectors feature extraction from higher-order data. *Computer Graphics Forum (Proc. EuroVis) 30*, 3 (2011), 751–760.

[PPF*11] POBITZER A., PEIKERT R., FUCHS R., SCHINDLER B., KUHN A., THEISEL H., MATKOVIC K., HAUSER H.: The state of the art in topology-based visualization of unsteady flow. *Computer Graphics Forum 30*, 6 (2011), 1789–1811.

[PR99] PEIKERT R., ROTH M.: The "parallel vectors" operator – a vector field visualization primitive. In *Proc. IEEE Visualization* (1999), pp. 263–270.

[PRN*11] PETERKA T., ROSS R., NOUANESENGSY B., LEE T.-Y., SHEN H.-W., KENDALL W., HUANG J.: A study of parallel particle tracing for steady-state and time-varying flow fields. In *Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium* (Washington, DC, USA, 2011), IPDPS '11, IEEE Computer Society, pp. 580–591.

[PSGC11a] PICANO F., SARDINA G., GUALTIERI P., CASCIOLA C.: DNS of a free turbulent jet laden with small inertial particles. In *Direct and Large-Eddy Simulation VIII*, Kuerten H., Geurts B., Armenio V., Fröhlich J., (Eds.), vol. 15 of *ERCOFTAC Series*. Springer Netherlands, 2011, pp. 189–194.

[PSGC11b] PICANO F., SARDINA G., GUALTIERI P., CASCIOLA C. M.: Particle-laden jets: Particle distribution and back-reaction on the flow. *Journal of Physics: Conference Series 318*, 5 (2011), 052018.

[PVH*03] POST F. H., VROLIJK B., HAUSER H., LARAMEE R. S., DOLEISCH H.: The state of the art in flow visualisation: Feature extraction and tracking.

Computer Graphics Forum 22, 4 (2003), 775–792.

[RP98] ROTH M., PEIKERT R.: A higher-order method for finding vortex core lines. In *Proc. IEEE Visualization* (1998), pp. 143–150.

[RPP*09] ROSANWO O., PETZ C., PROHASKA S., HEGE H.-C., HOTZ I.: Dual streamline seeding. In *IEEE Pacific Visualization Symposium* (April 2009), pp. 9–16.

[RRV14] RABEN S. G., ROSS S. D., VLACHOS P. P.: Experimental determination of three-dimensional finite-time Lyapunov exponents in multi-component flows. *Experiments in Fluids 55*, 10 (2014), 1–6.

[RSBE01] ROETTGER S., SCHULZ M., BARTELHEIMER W., ERTL T.: Automotive soiling simulation based on massive particle tracing. In *Data Visualization 2001*, Eurographics. Springer Vienna, 2001, pp. 309–317.

[RSK08] RAAB M., SEIBERT D., KELLER A.: Unbiased global illumination with participating media. In *Monte Carlo and Quasi-MC Methods 2006*. Springer Berlin Heidelberg, 2008, pp. 591–605.

[Sad15] SADLO F.: Lyapunov time for 2D Lagrangian visualization. In *Topological and Statistical Methods for Complex Data*. Springer, 2015, pp. 167–181.

[Sah09] SAHNER J.: *Extraction of Vortex Structures in 3D Flow Fields*. PhD thesis, University of Magdeburg, Germany, April 2009.

[SBH*01] SCHEUERMANN G., BOBACH T., HAGEN H., MAHROUS K., HAMANN B., JOY K. I., KOLLMANN W.: A tetrahedra-based stream surface algorithm. In *Proc. Visualization* (2001), pp. 151–158.

[SBR16] SUDHARSAN M., BRUNTON S. L., RILEY J. J.: Lagrangian coherent structures and inertial particle dynamics. *Phys. Rev. E 93* (Mar 2016), 033108.

[SGG15] STAIB J., GROTTEL S., GUMHOLD S.: Visualization of particle-based data with transparency and ambient occlusion. *Computer Graphics Forum (Proc. EuroVis) 34*, 3 (2015), 151–160.

[SGL10] SYAL M., GOVINDARAJAN B., LEISHMAN J. G.: Mesoscale sediment tracking methodology to analyze brownout cloud developments. In *Proceedings of the American Helicopter Society, 66th Annual Forum* (2010).

[SGRT12] SCHULZE M., GERMER T., RÖSSL C., THEISEL H.: Stream surface parametrization by flow-orthogonal front lines. *Computer Graphics Forum (Proc. Symposium on Geometry Processing) 31*, 5 (2012), 1725–1734.

[SGS05] STOLL C., GUMHOLD S., SEIDEL H.-P.: Visualization with stylized line

primitives. In *IEEE Visualization Conference* (Oct 2005), pp. 695–702.

[SH95] SUJUDI D., HAIMES R.: *Identification of Swirling Flow in 3D Vector Fields*. Tech. rep., Departement of Aeronautics and Astronautics, MIT, 1995. AIAA Paper 95-1715.

[SH08] SAPSIS T., HALLER G.: Instabilities in the dynamics of neutrally buoyant particles. *Phys. Fluids 20* (2008), 017102.

[SH09] SAPSIS T. P., HALLER G.: Inertial particle dynamics in a hurricane. *Journal of the Atmospheric Sciences* (2009).

[Sha05] SHADDEN S. C.: Lagrangian coherent structures. http://mmae.iit.edu/shadden/LCS-tutorial/, 2005.

[SHH*07] SCHLEMMER M., HOTZ I., HAMANN B., MORR F., HAGEN H.: Priority streamlines: A context-based visualization of flow fields. In *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization* (Aire-la-Ville, Switzerland, Switzerland, 2007), EuroVis'07, Eurographics Association, pp. 227–234.

[SKK*15] STANGER C., KUTZ B., KOWARSCH U., BUSCH E. R., KESSLER M., KRÄMER E.: *High Performance Computing in Science and Engineering '14: Transactions of the High Performance Computing Center, Stuttgart (HLRS) 2014*. Springer International Publishing, Cham, 2015, ch. Enhancement and Applications of a Structural URANS Solver, pp. 433–446.

[SKMR98] SCHEUERMANN G., KRUGER H., MENZEL M., ROCKWOOD A.: Visualizing nonlinear vector field topology. *IEEE Transactions on Visualization and Computer Graphics 4*, 2 (1998), 109–116.

[SKTM11] SZIRMAY-KALOS L., TÓTH B., MAGDICS M.: Free path sampling in high resolution inhomogeneous participating media. *Computer Graphics Forum 30*, 1 (2011), 85–97.

[SL99] SHAO Y., LI A.: Numerical modelling of saltation in the atmospheric surface layer. *Boundary-Layer Meteorology 91* (1999), 199–225.

[SLCZ09] SPENCER B., LARAMEE R. S., CHEN G., ZHANG E.: Evenly spaced streamlines for surfaces: An image-based approach. *Computer Graphics Forum 28*, 6 (2009), 1618–1631.

[SLM05] SHADDEN S. C., LEKIEN F., MARSDEN J. E.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena 212*, 3-4 (2005), 271–304.

[SMG*14] SCHULZE M., MARTINEZ ESTURO J., GÜNTHER T., RÖSSL C., SEIDEL H.-P., WEINKAUF T., THEISEL H.: Sets of globally optimal stream surfaces for flow visualization. *Computer Graphics Forum (Proc. EuroVis) 33*, 3 (2014), 1–10.

[SOK*13] SHAFII S., OBERMAIER H., KOLÁŘ V., HLAWITSCHKA M., GARTH C., HAMANN B., JOY K. I.: Illustrative rendering of vortex cores. In *EuroVis - Short Papers* (2013), pp. 61–65.

[SP99] SADARJOEN I. A., POST F. H.: Geometric methods for vortex extraction. In *Data Visualization '99*, Eurographics. Springer Vienna, 1999, pp. 53–62.

[SP07] SADLO F., PEIKERT R.: Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction. *IEEE Transactions on Visualization and Computer Graphics (IEEE Visualization) 13*, 6 (2007), 1456–1463.

[SPH11] SAPSIS T., PENG J., HALLER G.: Instabilities on prey dynamics in jellyfish feeding. *Bull Math Biol. 73*, 8 (2011), 1841–1856.

[SRWS10] SCHNEIDER D., REICH W., WIEBEL A., SCHEUERMANN G.: Topology aware stream surfaces. *Computer Graphics Forum (Proc. EuroVis) 29*, 3 (2010), 1153–1161.

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3D shapes. *SIGGRAPH Comput. Graph. 24*, 4 (1990), 197–206.

[Sta98] STALLING D.: *Fast Texture-Based Algorithms for Vector Field Visualization*. PhD thesis, Freie Universität Berlin, 1998.

[Ste13] STEVENS B.: Introduction to UCLA-LES, 2013.

[STH*09] SHI K., THEISEL H., HAUSER H., WEINKAUF T., MATKOVIC K., HEGE H.-C., SEIDEL H.-P.: Path line attributes - an information visualization approach to analyzing the dynamic behavior of 3D time-dependent flow fields. In *Topology-Based Methods in Visualization II*, Mathematics and Visualization. Springer, 2009, pp. 75–88.

[STWE07] SCHAFHITZEL T., TEJADA E., WEISKOPF D., ERTL T.: Point-based stream surfaces and path surfaces. In *Proc. Graphics Interface* (2007), pp. 289–296.

[SVG*08] SCHAFHITZEL T., VOLLRATH J., GOIS J., WEISKOPF D., CASTELO A., ERTL T.: Topology-preserving lambda2-based vortex core line detection for flow visualization. *Computer Graphics Forum (Proc. EuroVis) 27*, 3 (2008), 1023–1030.

[SWH05a] SAHNER J., WEINKAUF T., HEGE H.-C.: Galilean invariant extraction and iconic representation of vortex core lines. In *Proc. Eurographics / IEEE VGTC Symposium on Visualization (EuroVis)* (2005), pp. 151–160.

[SWH05b] STALLING D., WESTERHOFF M., HEGE H.-C.: Amira: A highly interactive system for visual data analysis. In *The Visualization Handbook*. Elsevier, 2005, pp. 749–767.

[SWS09] SCHNEIDER D., WIEBEL A., SCHEUERMANN G.: Smooth stream surfaces of fourth order precision. *Computer Graphics Forum (Proc. EuroVis) 28*, 3 (2009), 871–878.

[SWTH07] SAHNER J., WEINKAUF T., TEUBER N., HEGE H.-C.: Vortex and strain skeletons in Eulerian and Lagrangian frames. *IEEE Transactions on Visualization and Computer Graphics 13*, 5 (2007), 980–990.

[TAB15] THOMAS S., AMIRAUX M., BAEDER J. D.: Modeling the two-phase flow field beneath a hovering rotor on graphics processing units using a hybrid vortex/CFD methodology. *Journal of the American Helicopter Society 60*, 2 (2015), 1–17.

[TB96] TURK G., BANKS D.: Image-guided streamline placement. In *Proc. SIGGRAPH* (1996), pp. 453–460.

[The02] THEISEL H.: Designing 2D vector fields of arbitrary topology. *Computer Graphics Forum (Proc. Eurographics) 21*, 3 (2002), 595–604.

[TKB12] THOMAS S., KALRA T., BAEDER J.: A hybrid CFD methodology to model the two phase flowfield beneath a hovering laboratory scale rotor. *Proceedings of the 42nd AIAA Fluid Dynamics Converence* (June 2012).

[TLHD03] TONG Y., LOMBEYDA S., HIRANI A. N., DESBRUN M.: Discrete multiscale vector field decomposition. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (2003), 445–452.

[TLKB11] THOMAS S., LAKSHMINARAYAN V. K., KALRA T. S., BAEDER J. D.: Eulerian-lagrangian analysis of cloud evolution using cfd coupled with a sediment tracking algorithm. In *Proceedings of the American Helicopter Society* (Virginia Beach, VA, May 2011), 67th Annual Forum.

[TMWS13] TAO J., MA J., WANG C., SHENE C.: A unified approach to streamline selection and viewpoint selection for 3D flow visualization. *IEEE Transactions on Visualization and Computer Graphics 19* (2013), 393–406.

[TRS03] THEISEL H., RÖSSL C., SEIDEL H.-P.: Compression of 2D vector fields under guaranteed topology preservation. *Computer Graphics Forum*

(Proc. Eurographics) 22, 3 (2003), 333–342.

[TS03] THEISEL H., SEIDEL H.-P.: Feature flow fields. In *Proc. Symposium on Data Visualisation* (2003), pp. 141–148.

[TSW*05] THEISEL H., SAHNER J., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Extraction of parallel vector surfaces in 3D time-dependent fields and application to vortex core line tracking. In *Proc. IEEE Visualization* (2005), pp. 631–638.

[TWHS03] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Saddle connectors - an approach to visualizing the topological skeleton of complex 3D vector fields. In *Proc. IEEE Visualization* (2003), pp. 225–232.

[ÜSE13] ÜFFINGER M., SADLO F., ERTL T.: A time-dependent vector field topology based on streak surfaces. *IEEE Transactions on Visualization and Computer Graphics 19*, 3 (2013), 379–392.

[ÜSK*12] ÜFFINGER M., SADLO F., KIRBY M., HANSEN C., ERTL T.: FTLE computation beyond first-order approximation. In *Eurographics (Short Papers)* (2012), pp. 61–64.

[USM96] UENG S.-K., SIKORSKI C., MA K.-L.: Efficient streamline, streamribbon, and streamtube constructions on unstructured grids. *IEEE Transactions on Visualization and Computer Graphics 2*, 2 (1996), 100–110.

[Vea98] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, Stanford, CA, USA, 1998.

[vFWTS08] VON FUNCK W., WEINKAUF T., THEISEL H., SEIDEL H.-P.: Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE TVCG 14*, 6 (November 2008), 1396–1403.

[VG05] VIOLA I., GRÖLLER E.: Smart visibility in visualization. In *Proc. Computational Aesthetics* (2005), pp. 209–216.

[VG12] VAN GELDER A.: Vortex core detection: Back to basics. In *IS&T/SPIE Electronic Imaging* (2012), International Society for Optics and Photonics, pp. 829413–829413.

[VGP09] VAN GELDER A., PANG A.: Using PVsolve to analyze and locate positions of parallel vectors. *IEEE Transactions on Visualization and Computer Graphics 15*, 4 (2009), 682–695.

[VKG04] VIOLA I., KANITSAR A., GRÖLLER E.: Importance-driven volume rendering. In *Proc. Visualization* (2004), pp. 139–146.

[VKP00] VERMA V., KAO D., PANG A.: A flow-guided streamline seeding strategy. In *IEEE Visualization* (2000), pp. 163–170.

[VPB*09] VERGNE R., PACANOWSKI R., BARLA P., GRANIER X., SCHLICK C.: Light warping for enhanced surface depiction. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3 (2009), 25:1–25:8.

[vW93] VAN WIJK J. J.: Implicit stream surfaces. In *IEEE Visualization* (Washington, DC, USA, 1993), IEEE Computer Society, pp. 245–252.

[War08] WARD M. O.: Multivariate data glyphs: Principles and practice. In *Handbook of Data Visualization*, Springer Handbooks Comp. Statistics. Springer Berlin Heidelberg, 2008, pp. 179–198.

[WCW*09] WIEBEL A., CHAN R., WOLF C., ROBITZKI A., STEVENS A., SCHEUERMANN G.: Topological flow structures in a mathematical model for rotation-mediated cell aggregation. *Topological Data Analysis and Visualization: Theory, Algorithms and Applications* (2009), 1–12.

[Wei91] WEISS J.: The dynamics of enstrophy transfer in two-dimensional hydrodynamics. *Physica D: Nonlinear Phenomena 48*, 2-3 (1991), 273–294.

[Wei08] WEINKAUF T.: *Extraction of Topological Structures in 2D and 3D Vector Fields*. PhD thesis, University Magdeburg, 2008.

[WGM*08] WANG L., GIESEN J., MCDONNELL K. T., ZOLLIKER P., MUELLER K.: Color design for illustrative visualization. *IEEE TVCG 14*, 6 (2008), 1739–1754.

[WHT12] WEINKAUF T., HEGE H.-C., THEISEL H.: Advected tangent curves: A general scheme for characteristic curves of flow fields. *Computer Graphics Forum (Proc. Eurographics) 31*, 2 (May 2012), 825–834.

[Wie04] WIEBEL A.: *Feature Detection in Vector Fields Using the Helmholtz-Hodge Decomposition*. Diploma thesis, Univ. Kaiserslautern, 2004.

[WJE01] WESTERMANN R., JOHNSON C., ERTL T.: Topology-preserving smoothing of vector fields. *IEEE Transactions on Visualization and Computer Graphics 7*, 3 (2001), 222–229.

[WMHL65] WOODCOCK E., MURPHY T., HEMMINGS P., LONGWORTH S.: Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Proc. Conf. Applications of Computing Methods to Reactor Problems* (1965), vol. 557, Argonne National Laboratory.

[WS01] WISCHGOLL T., SCHEUERMANN G.: Detection and visualization of closed

streamlines in planar flows. *IEEE Transactions on Visualization and Computer Graphics 7*, 2 (2001), 165–172.

[WS05] WIEBEL A., SCHEUERMANN G.: Eyelet particle tracing - steady visualization of unsteady flow. In *Proc. IEEE Visualization* (2005), pp. 607–614.

[WS11] WANG C., SHEN H.-W.: Information theory in scientific visualization. *Entropy 13*, 1 (2011), 254–273.

[WSTH07] WEINKAUF T., SAHNER J., THEISEL H., HEGE H.-C.: Cores of swirling particle motion in unsteady flows. *IEEE Transactions on Visualization and Computer Graphics (Proc. Visualization) 13*, 6 (2007), 1759–1766.

[WT10a] WEINKAUF T., THEISEL H.: Streak lines as tangent curves of a derived vector field. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2010) 16*, 6 (November - December 2010), 1225–1234.

[WT10b] WONG O. D., TANNER P. E.: Photogrammetric measurements of an eh-60l brownout cloud. In *Proceedings of the American Helicopter Society* (Phoenix, AZ, May 2010), 66th Annual Forum.

[WTGP11] WEINKAUF T., THEISEL H., GELDER A. V., PANG A.: Stable feature flow fields. *IEEE Transactions on Visualization and Computer Graphics 17*, 6 (2011), 770–780.

[WTHS04a] WEINKAUF T., THEISEL H., HEGE H.-C., SEIDEL H.-P.: Boundary switch connectors for topological visualization of complex 3D vector fields. In *VisSym* (2004), pp. 183–192.

[WTHS04b] WEINKAUF T., THEISEL H., HEGE H.-C., SEIDEL H.-P.: Topological construction and visualization of higher order 3D vector fields. *Computer Graphics Forum (Proc. Eurographics) 23*, 3 (2004), 469–478.

[WTHS06] WEINKAUF T., THEISEL H., HEGE H.-C., SEIDEL H.-P.: Topological structures in two-parameter-dependent 2D vector fields. *Computer Graphics Forum (Proc. Eurographics) 25*, 3 (2006), 607–616.

[WTS*07] WIEBEL A., TRICOCHE X., SCHNEIDER D., JANICKE H., SCHEUERMANN G.: Generalized streak lines: Analysis and visualization of boundary induced vortices. *Visualization and Computer Graphics, IEEE Transactions on 13*, 6 (2007), 1735–1742.

[WTS12] WEINKAUF T., THEISEL H., SORKINE O.: Cusps of characteristic curves and intersection-aware visualization of path and streak lines. In *Topological Methods in Data Analysis and Visualization II*, Mathematics and

Visualization. Springer Berlin Heidelberg, 2012, pp. 161–175.

[WWSS10] WIEBEL A., WANG Q., SCHNEIDER D., SCHEUERMANN G.: Accelerated streak line computation using adaptive refinement. *Journal of WSCG 18* (2010), 17–23.

[XLS10] XU L., LEE T.-Y., SHEN H.-W.: An information-theoretic framework for flow visualization. In *IEEE Transactions on Visualization and Computer Graphics* (2010), pp. 1216–1224.

[YHGT10] YANG J. C., HENSLEY J., GRÜN H., THIBIEROZ N.: Real-time concurrent linked list construction on the GPU. *Computer Graphics Forum 29*, 4 (2010), 1297–1304.

[YIC*10] YUE Y., IWASAKI K., CHEN B.-Y., DOBASHI Y., NISHITA T.: Unbiased, adaptive stochastic sampling for rendering inhomogeneous participating media. *ACM Transaction on Graphics (Proc. SIGGRAPH Asia) 29*, 6 (2010), 177:1–177:8.

[YKP05] YE X., KAO D., PANG A.: Strategy for seeding 3D streamlines. *IEEE Visualization Conference* (2005), 471–478.

[YTvdW*02] YU Y., TUNG C., VAN DER WALL B., PAUSDER H.-J., BURLEY C., BROOKS T., BEAUMIER P., MERCKER Y. D. E., PENGEL K.: The HART-II test: Rotor wakes and aeroacoustics with higher-harmonic pitch control (HHC) inputs – the joint German/French/Dutch/US project. *American Helicopter Society 58th Annual Forum* (2002).

[YWSC12] YU H., WANG C., SHENE C.-K., CHEN J.: Hierarchical streamline bundles. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (2012), 1353–1367.

[ZHXC09] ZHANG L., HE Y., XIE X., CHEN W.: Laplacian lines for real-time shape illustration. In *Proc. Symposium on Interactive 3D Graphics and Games* (2009), pp. 129–136.

[ZJL*15] ZWICKER M., JAROSZ W., LEHTINEN J., MOON B., RAMAMOORTHI R., ROUSSELLE F., SEN P., SOLER C., YOON S.-E.: Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer Graphics Forum (Proc. Eurographics) 34*, 2 (2015), 667–681.

[ZSH96] ZÖCKLER M., STALLING D., HEGE H.-C.: Interactive visualization of 3D vector fields using illuminated stream lines. In *IEEE Visualization* (1996), pp. 107–113.

[ZZW02] ZHU T., ZHANG D.-L., WENG F.: Impact of the advanced microwave

sounding unit data on hurricane prediction. *Monthly Weather Review 130* (2002), 2416–2432.

[ZZW04] ZHU T., ZHANG D.-L., WENG F.: Numerical simulation of Hurricane Bonnie (1998). part I: Eyewall evolution and intensity changes. *Monthly Weather Review 132* (2004), 225–241.

Declaration

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; verwendete fremde und eigene Quellen sind als solche kenntlich gemacht. Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Ich habe insbesondere nicht wissentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich missbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert,
- fremde Forschungsergebnisse verzerrt wiedergegeben.

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadensersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Magdeburg, den 30. Juni 2016

Tobias Günther