TU WIEN Informatics

Interactive Reference-Frame Computation for Observer-Relative Flow Visualization

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Bernhard Woschizka
Matrikelnummer 11809612

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Dipl.-Ing. Dr.techn. Peter Rautek

Wien, 23. Dezember 2021

Bernhard Woschizka	Eduard Gröller

TU WIEN Informatics

Interactive Reference-Frame Computation for Observer-Relative Flow Visualization

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Bernhard Woschizka
Registration Number 11809612

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Dipl.-Ing. Dr.techn. Peter Rautek

Vienna, 23rd December, 2021

Bernhard Woschizka	Eduard Gröller

Erklärung zur Verfassung der Arbeit

Bernhard Woschizka

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 23. Dezember 2021

Bernhard Woschizka

Danksagung

Zu Beginn möchte ich meinem Betreuer Peter Rautek für seine Unterstützung bei der Programmierung und Verfassung danken. Ich möchte mich bei Eduard Gröller bedanken, sowohl für seine Betreuung als auch für die Bekanntmachung mit Peter und der Forschungsgruppe an der King Abdullah University of Science and Technology (KAUST). Hier möchte ich speziell allen Mitgliedern der "High-Performance Visualization Group" an der KAUST danken. Danke für eure interessanten Präsentationen bei den wöchentlichen Meetings und die Möglichkeit, ein Teil einer aktiven Forschungsgemeinschaft zu sein. Nicht nur für diese Arbeit, sondern auch für mein ganzes Bachelorstudium, möchte ich meinen Eltern und meinem Bruder für deren Liebe, Freundschaft und all die (netten) Gespräche in dieser Zeit danken. Ebenso möchte ich meinen Kolleginnen, Kollegen, Freundinnen und Freunden, die Teil meiner akademischen und persönlichen Reise waren oder noch immer sind, danken.

Acknowledgements

First of all I want to thank my supervisor Peter Rautek for his support in both coding and writing tasks. I want to thank Eduard Gröller for his supervision and connecting me with Peter and the research group at King Abdullah University of Science and Technology (KAUST). Which leads me directly to thank all the people in the "High-Performance Visualization Group" at KAUST. Thank you for your interesting presentations on a weekly basis and the possibility to be part of an active research community. Not specifically for this thesis but for my whole bachelor studies, I want to thank my parents and my brother for their love, friendship and all the (good) talks we had throughout the years. I also want to thank my colleagues and friends that were or are still part of my journey through my academic and personal life.

Kurzfassung

Forscherinnen, Forscher, Technikerinnen und Techniker aus verschiedenen Bereichen arbeiten mit Strömungssimulation und -visualisierung. Diese Werkzeuge helfen beim Verstehen von physikalischen Phänomenen wie Wirbeln, welche als Wetterphänomene sowohl in Meeresströmungen als auch in der Aerodynamik vorkommen. Die Visualisierung von Strömungsfeldern profitiert von fortgeschrittenen Konzepten wie der Betrachterrelativität (Observer Relativity). Bereits bestehende interaktive Programme für betrachterzentrierte Visualisierung benötigen Vorberechnungen der Strömungsfelder, um sinnvolle Bezugsrahmen (reference frames) zu erhalten. Diese Vorberechnung ist ressourcenaufwendig und potenziell zeitintensiv für große Datensätze. Diese Arbeit stellt eine Methode ohne Vorberechnung vor. Die benutzende Person wählt eine Startposition, welche Ausgangspunkt für einen iterativen Vorgang ist. Abwechselnd wird das Beobachterfeld (Observer-Field) in einem Teil des Datensatzes optimiert und die Beobachterbewegung integriert (Observer-Worldline). Der Endpunkt der Integration dient als Startpunkt für die nächste Iteration. Die Visualisierung der Strömungslinien, bezogen auf den Beobachter, ist mit der genannten Berechnung verschachtelt. Die Kombination von schrittweiser Erstellung und systemnahen Optimierungstechniken wie Caching, Abschätzung und Parallelisierung, resultiert in einem Algorithmus, welcher eine beobachterzentrierte Visualisierung ohne Vorberechnung bereitstellt. Da diese Technik ohne Vorberechnung auskommt, kann sie für In-Situ Szenarien und für die Erkundung des Parameterbereiches der Optimierungstechniken angewendet werden. Dies ist für die Berechnung von Beobachtern notwendig.

Abstract

Scientists and engineers from many domains work with fluid simulation and visualization. These tools help to understand physical phenomena like vortices that occur as weather phenomena, in ocean currents and in aerodynamics. The visualization of flow fields profits from advanced concepts like observer relativity. Existing interactive frameworks for observer-relative visualization need to pre-process the flow data to obtain meaningful reference frames. This pre-processing task is computationally expensive and potentially time consuming for large data sets. In this thesis a method without pre-processing is proposed. Starting at a user specified location we iterate the optimization of the observer field in a small subset of the data, and the integration of the observer worldline to find the next start location. The visualization of the observed pathlines is interleaved with the observer-calculation. The combination of progressive buildup, as well as low-level optimization techniques like caching, prediction, and parallelization, provides an algorithm that visualizes observer-relative flow interactively without the need of pre-processing. The avoidance of pre-processing makes this technique suitable for in-situ scenarios, and for the exploration of the parameter space of optimization techniques that are needed to compute reference-frame candidates.

Contents

CHAPTER 1

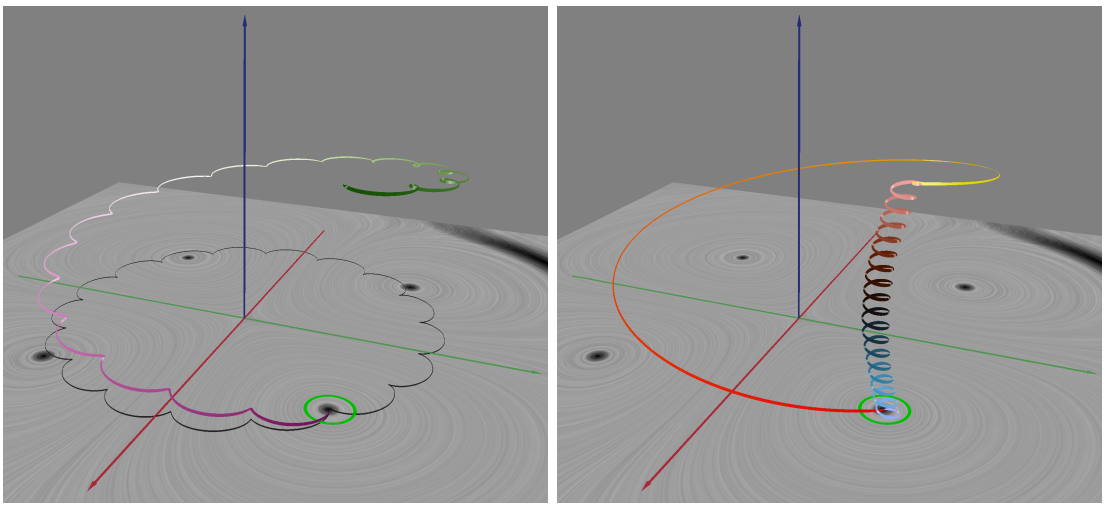
Introduction

The simulation and visualization of fluids are common tools in many fields of science and engineering. Analyzing flow phenomena such as vortex structures (e.g. hurricanes and ocean eddies) are especially important.
Some of the data characteristics are hidden and need additional computation and human intervention to be discovered and analyzed. This work focuses on an explorative human in-the-loop approach. The user searches for interesting phenomena in the data and is guided and supported by the visualization and computations that run in the background. This is combined with interaction to give the user the ability to freely explore the data and the parameter space of the employed visualization and optimization techniques.

1.1 Motivation

There is no widely accepted consensus about the definition of a vortex in flow data. Typically, a vortex is described as a swirling motion that is coherent over time. Since velocities are always given relative to some frame of reference, motions might appear different to different observers. In this sense vortex structures might sometimes be hidden inside the flow data. To better see them, current research [GGT17, HMTR18, RMB+20, ZHTR21] uses an observer-relative approach. Co-moving observers measure different relative velocities and hence vortex structures become more visible in certain reference frames than in others. Figure 1.1 shows an example of an unsteady 2D flow field visualized relative to two different observers. We visualize the instantaneous velocities of the first time step using Line Integral Convolution (LIC). A pathline was seeded close to a vortex in the data set. The vertical axis is time. The start point of the observer is at the center of the green ring. In Figure 1.1a the pathline (purple-white-green) is visualized relative to the default (lab-frame) observer. In Figure 1.1b it is visualized relative to a computed observer that follows the vortex over time. This observer is visualized as an orange-yellow line, the observer worldline. The observed pathline is shown with a color gradient from blue to black to light

red in Figure 1.1. The color encodes time. Figure 1.1 shows how two different observers lead to drastically different visualizations. The goal of observer-relative visualization is to make it easy and intuitive to switch between meaningful explicitly specified observers.



(a) (b)

Figure 1.1: This figure shows the same data set visualized relative to two different observers. The origin of all pathlines is the center of the green ring. (a) visualizes a pathline relative to the default observer. (b) visualizes it relative to a picked observer. The pathline (purple-white-green) is transformed by the observer worldline (yellow-orange) to form the observed pathline (blue-black-light red).

Figure 1.2 shows a comparison of the pre-computation (top) and on-demand computation (bottom) pipeline. Both pipelines transform input data (Fig. 1.2 a) by selecting an observer (Fig. 1.2 d) into an observer-relative visualization (Fig. 1.2 e). Current visualization research [ZHTR21] computes new reference frames that optimally show vortex structures in an expensive pre-computation step (Fig.1.2 b1). This approach is influenced by the research of vortex detection, which computes locations of vortices (without the help of humans). The pre-computed data offers the opportunity to interactively work with the data set and change the visualization in real-time. The pre-computation outputs a large number of observers where the user chooses one of them at a time (Fig. 1.2 c1). The visualization is then shown relative to this observer (Fig. 1.2 e). As pre-computation is a rather time-consuming process, this thesis takes an alternative approach of computing observers on-demand. The pipeline is changed, such that the point of interest is picked first (Fig. 1.2 b2) and then the reference frame optimization is computed only locally (Fig. 1.2 c2). This results in a big advantage, as the computation is now independent from the (spacial) size of the data set. This modification of the pipeline offers the opportunity to explore the large parameter space of the observer optimization methods, as well as to use the method in in-situ scenarios.

The computation time of the new approach still depends on the data set size in the time dimension. To provide a responsive user-experience we progressively update the visualization, meaning that a small section of the visualization is shown at the beginning and this section is extended over time. If the user remains at the picked position, the computation and visualization will continue to update till it reaches the full time range of the data set. The user has the possibility to interrupt the progressive computation by changing the position. This is useful if the current position does not seem to be promising to contain a flow structure of interest.

1.2 Goal of the Thesis

The goal of this thesis is to modify an existing explorative pipeline, which uses pre-computation of the optimal reference frames on the whole data set [ZHTR21]. This adaption and extension can be split up in the following parts:

1. Reduce the size of computation from the whole data set to a (local) neighborhood that is interactively chosen
2. Cache the calculated results for better performance
3. Progressively visualize the results as the computation progresses over the time dimension
4. Interactively visualize the observer relative flow data
5. Evaluate the overall performance of the tool

1.3 Structure of the Work

The thesis starts with the description of previous research, where vortex detection, reference frame optimization and physically-based observers are covered in detail in Chapter 2. Then the pipeline is described in depth in Chapter 3, followed by optimizations in Chapter 4. The results and an evaluation of the performance of the pipeline is shown in Chapter 5. The thesis concludes with a summary and improvements that can be made and possible future work in Chapter 6.

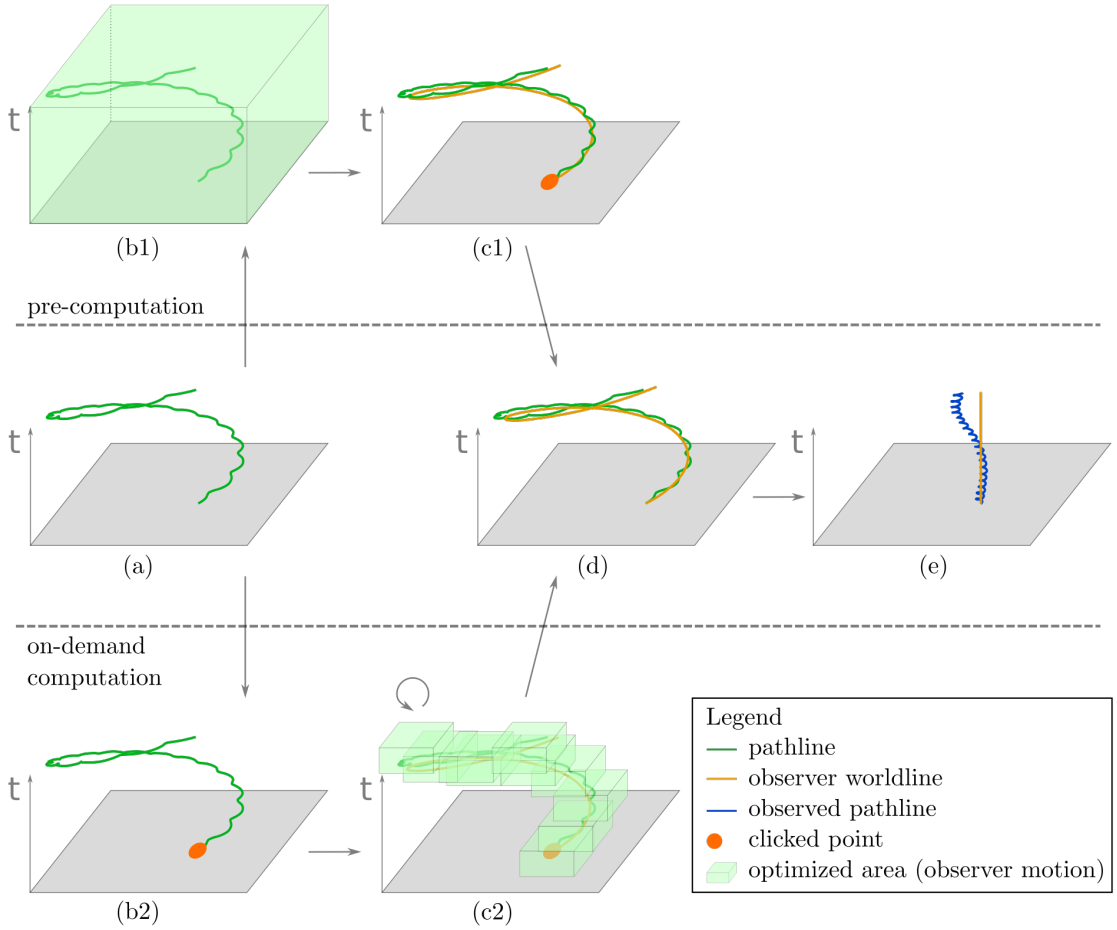


Figure 1.2: Pre-computation vs on-demand computation pipeline. The input data (a) is shown as a single pathline (green). The pre-computed pipeline calculates the *observer field* (light green box) on the whole data set (b1). Then a point of interest (orange dot) is picked and the *observer worldline* (orange) is integrated (c1). The new on-demand approach switches the steps (b) and (c). The point of interest is picked first (b2) and the observer field can be calculated only where it is needed. Then the observer worldline is integrated until it reaches the boundary of the computed region. This leads to the next needed region. This iterative process builds up the observer worldline, where intermediate results are visualized (c2). Both pipelines calculate the transformation from the observer worldline, which is the observer motion over time (d). The transformation visualizes the pathlines relative to the observer motion as *observed pathlines* (blue) in (e).

CHAPTER 2

Related work

Flow visualization is used in various different domains and helps people to understand the underlying phenomena more easily. Different techniques exist for instance for steady (time-independent) flow fields, e.g. *Line Integral Convolution* (LIC) [CL93] and for unsteady (time-dependent) flow fields, e.g. *Lagrangian-Eulerian Advection* (LEA) [JEH02]. Steady flow fields do not change over time. Unsteady ones are changing over time, as it is common in fluid and air flows. *Streamlines* [JL97] and *Streak Lines* [WT10] are techniques that are also often used. In the upcoming sections important terms and their related work will be discussed.

2.1 Vortex Definition and Detection

Vortices are structures of interest especially in weather simulation, climate simulation, aerodynamics simulation, etc. There have been many attempts to define a vortex, however, a precise definition does not exist [Hal05][GT18]. A vortex can be roughly described as "a swirling motion around a (curved) axis". Traditional measurements to extract vortex core lines (like λ_2, vorticity, and Sujudi-Haimes) are useful for steady fields, but for unsteady flow fields, detecting vortices with these methods is unreliable. Therefore the concept of *objectivity and reference frame optimization* has been introduced.

2.2 Reference Frame Invariance

Existing vortex detection criteria have been made objective by defining an optimal reference frame [GGT17]. Several measures have been defined to be insensitive to different kind of transformations:

- Galilean invariant vortex detection is invariant to translations [SWH05].

- Rotational invariant vortex detection is invariant to rotations around a fixed point [GST15].
- Objective vortex detection is invariant to time-dependent translations and rotations [Hal05].

2.3 Objectively Computed Optimal Reference Frames

Flow fields are velocity vector fields and must be given relative to a reference frame, or an observer. The flow field is captured (or simulated) relative to the default or lab-frame observer. An optimal reference frame can be found with objectively computed observer fields, so observer fields that are invariant to time-dependent translations and rotations. The optimal reference frame can be used to make other measures objective [GGT17, HMTR18, RMB+20, THR+21].

An objective global optimization method can also be used to calculate an *observer field* that minimizes the changes of the observed flow field over time. A local optimization efficiently finds an observer for every point in a given area. We use the method of Günther et al. [GGT17] to calculate the observer field in a neighborhood around a picked point.

2.4 Software Framework

This thesis is based on an existing framework for observer-relative flow visualization. The work is based on an interactive exploration tool [ZHTR21] that makes use of several optimization methods [GGT17, HMTR18]. As mentioned in Chapter 1, the method of Zhang et al. [ZHTR21] precomputes the optimization for the whole data set. Then a real-time interactive exploration is possible to examine and find structures of interest in the data. Pre-computation times depend on the data set size and the chosen optimization method. In contrast, we compute optimal observers in a small neighborhood on-demand reducing the computation times drastically. As the flow visualization can only be transformed relative to one observer at a time, we only calculate one (or a small set of observers). This enables us to reduce the computation time as the optimization is evaluated in a small area (neighborhood). Furthermore the performance of our local optimization is independent of the overall size of the data set.

CHAPTER 3

On-demand Observer-Relative Flow Visualization

The main novelty of the described pipeline is to quickly alternate between four algorithms to keep the observer-relative flow visualization interactive. In Section 3.1 the algorithm to compute the optimal observer field in a local neighborhood is described. In Section 3.2 the iterative method of computing the observer worldline is explained. Based on the observer worldline and the immediate neighborhood we compute the transformations that are applied to the pathlines. The computation of transformations is described in detail in Section 3.3. The last algorithm is the observer-relative visualization of the pathlines, described in Section 3.4. After each iteration the intermediate result is visualized. Figure 3.1 shows three intermediate results of our progressive visualization method.

3.1 Interactive Observer Worldline Computation

The input data are time-dependent 2D flow fields (one flow field in consecutive time steps). Each flow field is given as velocity vectors on a regular grid. This field is denoted as $\boldsymbol{v}$. In $\boldsymbol{v}$ a point is picked, denoted as start point s (Fig. 3.2 a). s has a physical position in space and time. As the upcoming optimization is done on a grid, s is mapped to (the closest lower integer) indices (x, y, t). Around s a neighborhood is constructed with a minimum integer size of 4x4x2 voxels $(x-1, x+3, y-1, y+3, t, t+1)$. The local optimization calculates the observer motion on these 4x4x2=32 grid points (Fig. 3.2 b). The calculated vectors are then stored in an *observer field*. The observer field $\boldsymbol{u}$ has the same size as the input field $\boldsymbol{v}$. For better performance caching is introduced to the pipeline. $\boldsymbol{u}$ is initialized with a special value (representing empty) and progressively filled with values from the optimization. The optimization is calculated according [GGT17]. Before every optimization step the cache is queried, if the values are already computed

for the needed neighborhood. If not, a new optimization is started for that neighborhood. Furthermore, inside the optimization the cache is again consulted for existing values on a per-voxel level.

3.2 Integration

The integration of the observer worldline is done using physical positions/values, which are denoted with *phys*. (The physical x- and y-coordinate are denoted as x_{phys}, y_{phys}, respectively. The x- and y-velocity of the observer $u_y(x), u_y(x)$ are physical values by default.)
Starting from s we integrate the observer worldline, which represents the observer motion in space and time (Fig. 3.2 c). Conceptually, we follow the observer motion $u(s)$ for one step (of size $stepSize$). At this new position $(x_{phys} + u_x(s) * stepSize, y_{phys} + u_y(s) * stepSize, t_{phys} + stepSize)$ we need to interpolate the velocity of the observer. This procedure is repeated saving all points in memory. The integration continues until the boundary of the region is reached (Fig. 3.2 d). This point is the new start point for a local optimization (Fig. 3.2 e) and ongoing integration (Fig. 3.2 f), where (Fig. 3.2 g) shows again the point where the integration leaves the region. This iterative approach continues till the new start point is outside the borders of the data set (Fig. 3.2 h). At this point the process is finished. The described procedure works for integrating forward in time. If a start point is picked with $t > minimumTime$, the integration will be computed forward and backward. For integrating backward the stepSize is multiplied with negative one.

3.3 Transformation

To calculate the transformation the worldline is traversed in time (this is simple as the points are already ordered by time, constantly increasing by $stepSize$). At each point of the worldline a translation and rotation are calculated from the observer field $\boldsymbol{u}$. The transformation is a time-dependent isometry $t \mapsto \Phi_t$. Explicitly described as

$$\Phi_t(x) = w(t) + \boldsymbol{Q}(t)(x - w(t_0)), \tag{3.1}$$

where $w(t)$ describes the translation and $\boldsymbol{Q}(t)$ the rotation. We work with Cartesian coordinates, so

$$\boldsymbol{Q}(t) = \begin{bmatrix} cos\Theta(t) & -sin\Theta(t) \\ sin\Theta(t) & cos\Theta(t) \end{bmatrix}. \tag{3.2}$$

The observer worldline $w(t)$ and $\Theta(t)$ are calculated with

$$\frac{d}{dt}w(t) = \boldsymbol{u}(w(t), t), \qquad \frac{d}{dt}\Theta(t) = c(t), \tag{3.3}$$

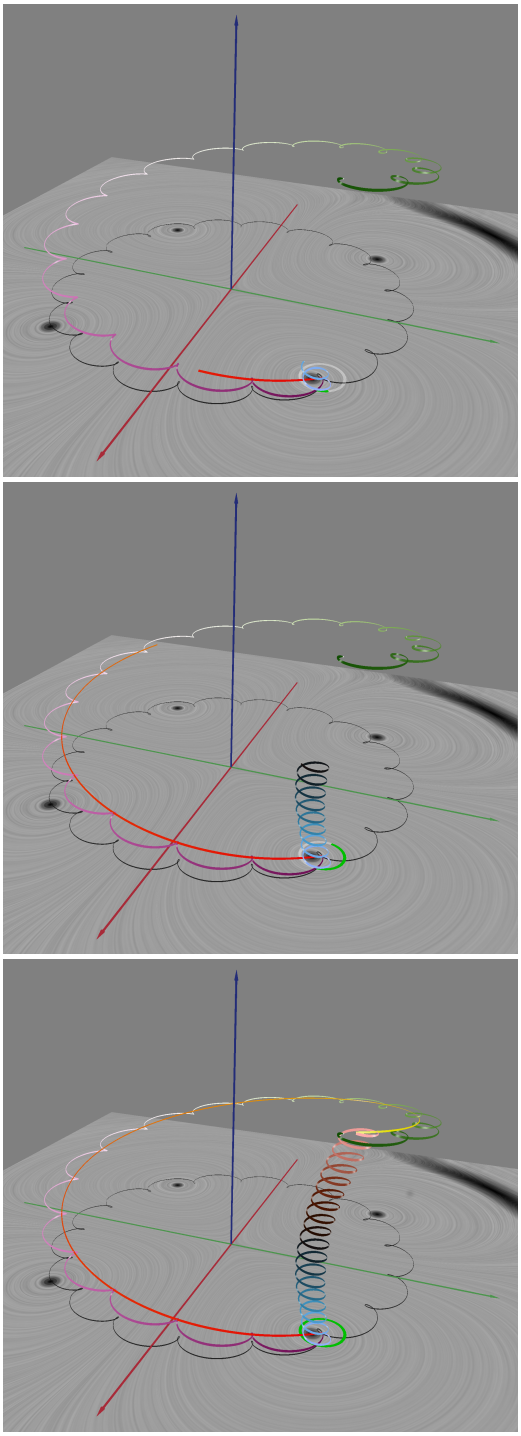
where $c(t)$ describes the curvature. These ordinary differential equations can be solved with

$$w(t) = w(t_0) + \int_{t_0}^{t} \boldsymbol{u}(w(\tau), \tau)\, d\tau, \qquad \Theta(t) = \int_{t_0}^{t} c(\tau)\, d\tau. \tag{3.4}$$

To compute $\boldsymbol{u}$ and c at any point p we calculate $\boldsymbol{u}$ on a discrete grid at a 4x4 area around p. With x and y, the nearest x- and y-indices to point p, the area is defined as $(x-1, x+2, y-1, y+2)$. Then the values for position p are interpolated. Therefore the minimal spacial size of the neighborhood for the optimizations is 4x4. Combining the transformations at all time steps $\Phi_{t1,\ldots,tn}$ forms the overall transformation Φ_t (shown in Fig. 3.3).

3.4 Observer-Relative Pathlines

We compute a set of pathlines for the input field. The transformation Φ_t is applied to each pathline (at the top of Fig. 3.3). For better visibility the transformation is shown for one pathline only. The transformation is calculated from the observer worldline (orange) and is applied to the pathline (green). This results in the observed pathline (blue). A detailed view of this transformation is shown at the bottom of Fig. 3.3, where each point of a pathline (black outline) is transformed individually by its time-dependent $\Phi_{t1,\ldots,tn}$ to its observed relative (white outline).



(a)

(b)

(c)

Figure 3.1: The algorithm builds up the visualization over time (only one pathline is visualized). At the beginning only the pathline (purple-white-green) and its shadow (black) is shown. (a) shows the result a little after the start. In (b) half of the result is visualized. In (c) the algorithm is finished and the full result, containing the observer worldline (orange-yellow) and the observed pathline (blue-black-light red) is shown.

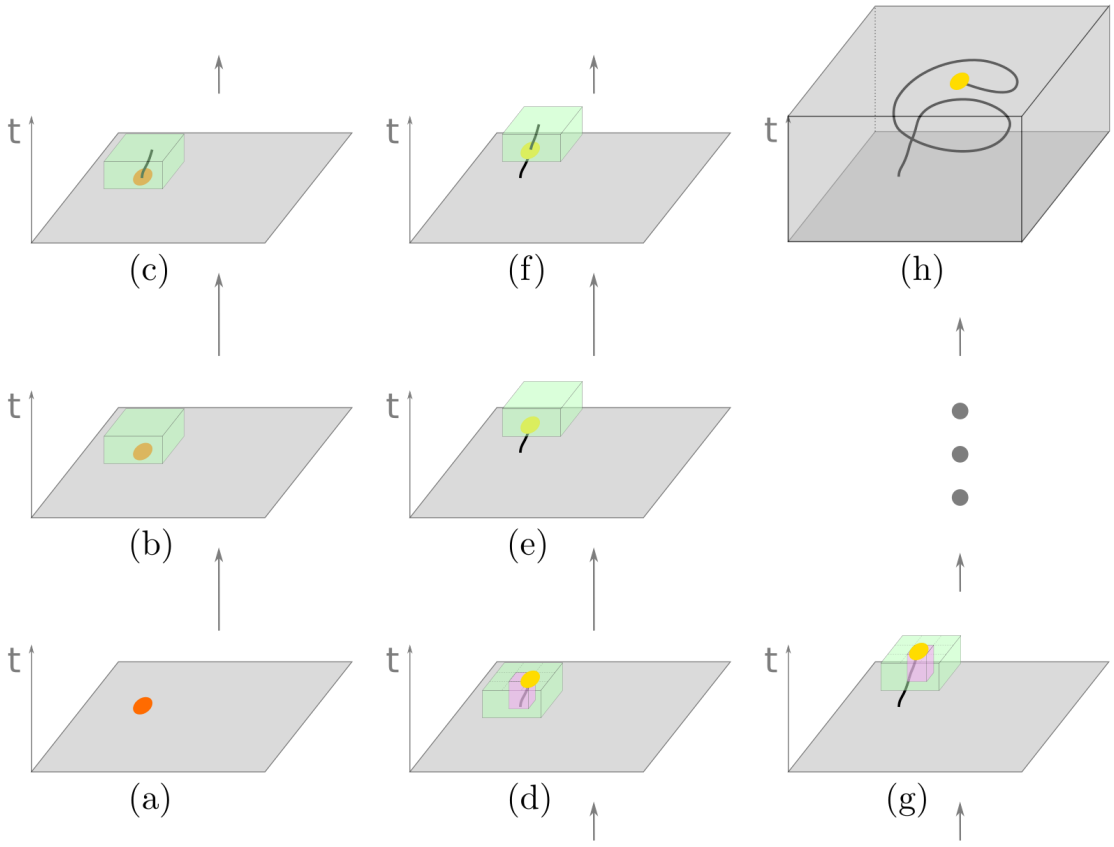


Figure 3.2: The computation of the observer worldline is started when the user picks a point of interest (a). Around that point the observer field is calculated using an optimization method in a local neighborhood (green) (b). The integration of the *observer worldline* (c) is started at the point of interest and continues till it leaves the region where the observer transformation is computable (pink) (d). This point is the new start point for a local optimization (e) and ongoing integration (f), where (g) shows again the point where the integration leaves the region. This iterative approach continues till the new start point is outside the boarders of the data set (h). At this point the process is finished.

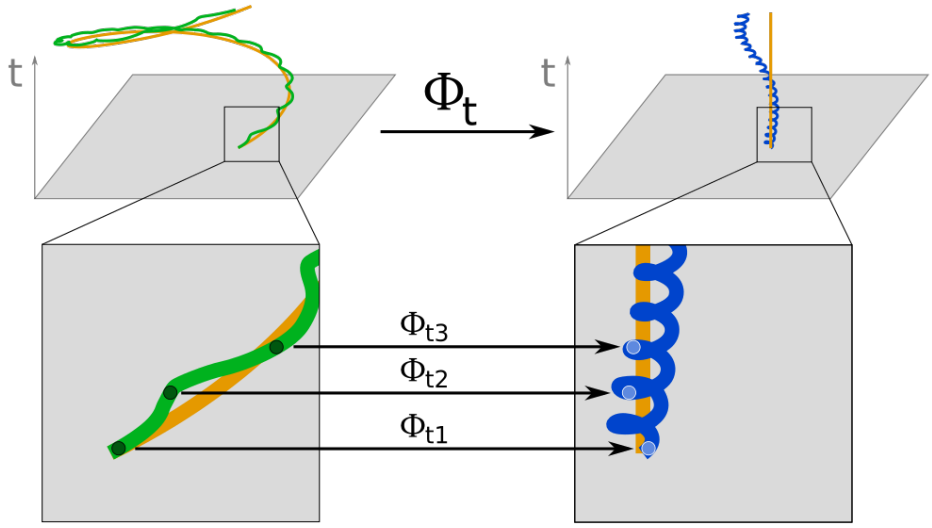


Figure 3.3: For better visibility the transformation is shown for one pathline only. Each pathline (green) is transformed with Φ_t, which was calculated from the *observer worldline* (orange). This results in the *observed pathline* (blue). In the detailed view it can be seen, that each point of the pathline (black outline) is transformed by the time-dependent transformation $\Phi_{t1,\ldots,tn}$ to its observed relative point (white outline).

CHAPTER 4

Performance Optimizations

In this chapter three low-level performance optimizations of the novel on-demand pipeline are discussed.

4.1 Parallelization

In each iteration-step of the pipeline we are concerned with three different local areas that are defined for local optimization, integration, and transformation (shown in Fig. 4.1). The integration is restricted by the *region*, the smallest area (pink). The calculation of the transformation needs the *neighborhood* (green), which is the region enlarged by one in the spacial dimensions. The *optimization neighborhood* (blue) provides the necessary data for the local optimization and it is the neighborhood enlarged by an adjustable value (10 was used in our experiments) in the spacial dimensions. The extent in the time domain is the same for all three areas (minimal two, but six was used in our experiments).

The local optimization can be split up in the calculation of the derivatives, the construction of the matrices containing this data, and the solving of the system. These three parts are parallelized with synchronization points in between. The calculation of the derivatives and the construction of the matrices are parallelized over each voxel of the optimization neighborhood. The system is solved in parallel for every voxel of the neighborhood.

4.2 Neighborhood Prediction

The performance of the algorithm is influenced by the number of iterations and the neighborhood size for the local optimization. Based on the characteristics of the data set, the neighborhood size can be changed to influence the number of iterations. Data sets with little movement benefit of larger neighborhood sizes in time. Data sets with much movement benefit of larger neighborhood sizes in space. Reasonably increasing the size

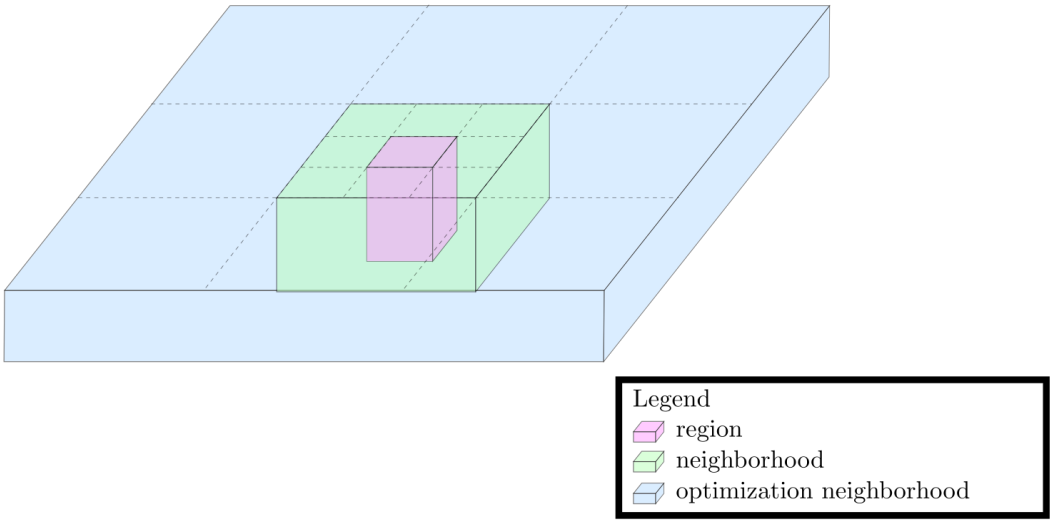


Figure 4.1: In each iteration-step of the pipeline three different areas are needed for the calculations. The pink area (*region*) restricts the integration of the observer worldline. To calculate the transformation the region is enlarged by one in each spacial dimension. This forms the *neighborhood* (green) where the optimization method [GGT17] is computed. In order to calculate the observer field in the neighborhood, we calculate partial derivatives in the space of the input data in the *optimization neighborhood* (blue). All three areas have the same extent in the time-domain.

of the neighborhood lowers the number of iterations while increasing the runtime of the local optimization. (For our experiments we increased the size from 4x4x2 to 5x5x6.) This increases the overall performance of the algorithm, because the local optimization can reuse values (e.g. derivatives) and use other optimizations (summed area tables, parallelization). A generally bigger, not reasonably picked, neighborhood size reduces the responsiveness of the algorithm due to the longer runtime of the local optimization (a neighborhood size of e.g. 15x15x20 is not recommended). The best performance is achieved if all computed values of the local optimization are used. Data sets with varying movement of vortex structures may benefit from a continuous adaption of the neighborhood size and shape. One way to continuously adapt to the (local) characteristics of the data set is to enlarge the neighborhood in the direction where the integration left the region. This procedure is repeated until the region is left in another direction and the neighborhood size is reset. This is shown in Figure 4.2, where the turquoise boxes indicate the fixed sized neighborhoods and the white boxes represent the enlargement of the neighborhood.
Too big neighborhood sizes reduce responsiveness and may result in visualizations that are perceived to be slow, which degrades the user experience.

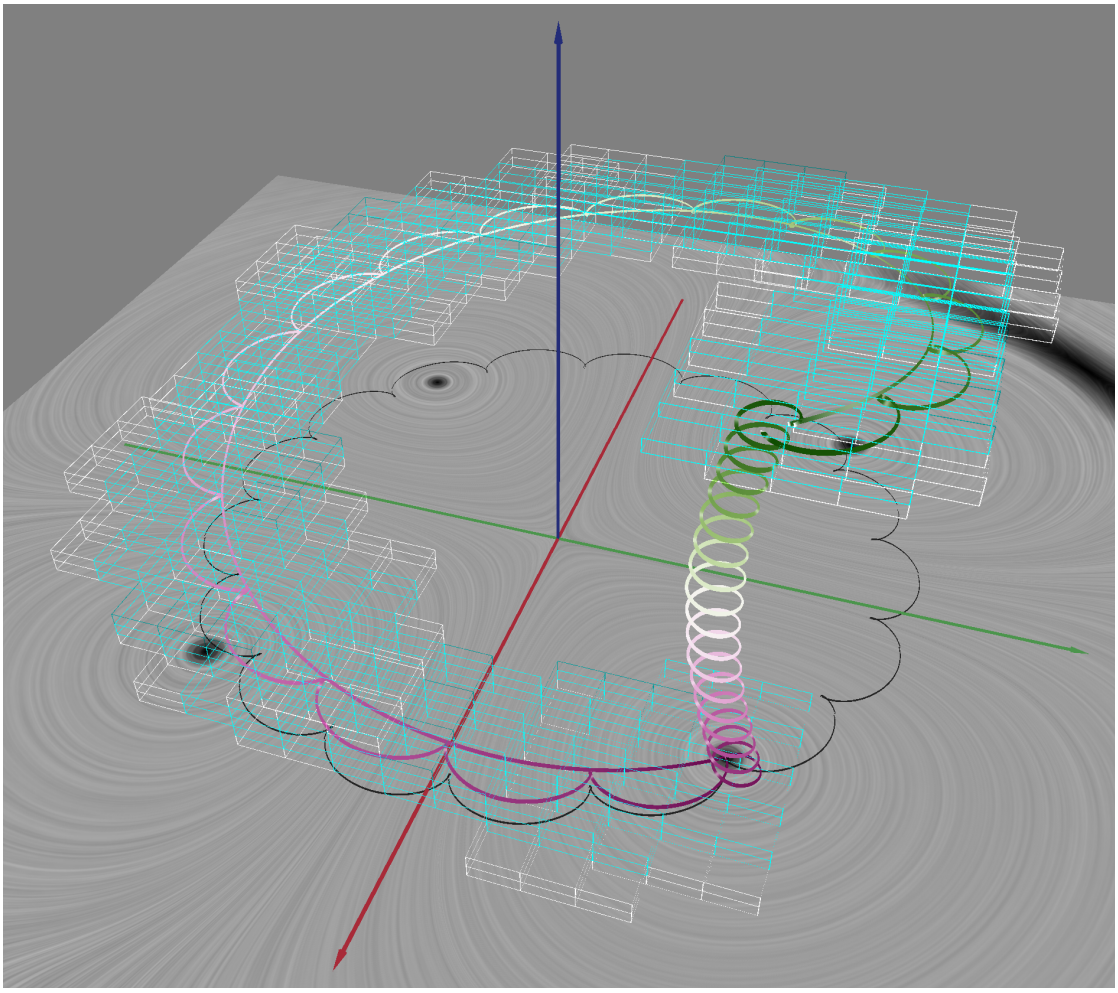
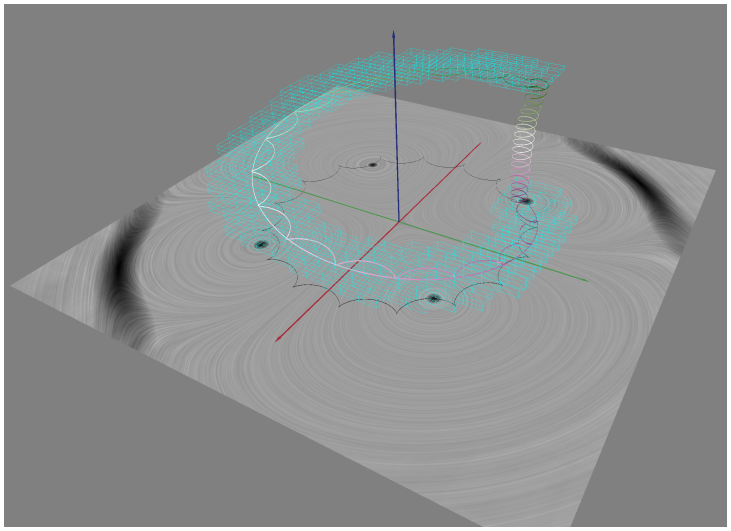


Figure 4.2: This figure shows the comparison of the fixed and predicted neighborhood size. The white boxes mark the enlargement done by the prediction algorithm. The turquoise boxes mark the fixed sized neighborhoods.

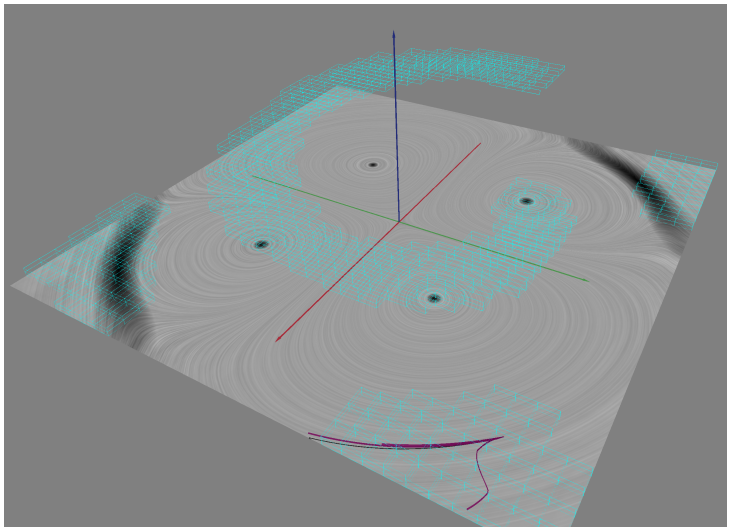
4.3 Caching

Our algorithm caches the observer field $\boldsymbol{u}$ at voxel level. The cache is initialized with a flag value (marking empty). Before the local optimization is started, the cache is queried. If all values of the neighborhood exist, the local optimization is skipped (all values can be fetched from the cache). Inside the local optimization the cache is consulted on a per-voxel level. Depending where the integration leaves the region 25% to 75% of the neighborhood values can be gathered from the cache in the next iteration. In Figure 4.3 the data saved in the cache is visualized by connecting two existing, neighboring values with a turquoise edge. This results in boxes as the neighborhood has always a cuboid shape. In Figure 4.3a the user clicked once, a 4x4x2 neighborhood was chosen and the

algorithm stopped at maximum time. In Figure 4.3b the user continued the exploration at the corners of the data set. From the flat optimized areas it can be concluded, that these integrations reached the spacial boundaries.



(a)



(b)

Figure 4.3: This figure shows which areas of the cache are filled with values. This is visualized by connecting two existing, neighboring values with a turquoise edge. (a) shows the cache after a single user input with a neighborhood size of 4x4x2. After exploration of the edges of the data set, the cache is filled with values in these regions (b).

CHAPTER 5

Visual Results and Performance Analysis

In this chapter the results of the novel on-demand pipeline are shown with three data sets: "Four Rotating Centers" [GGT17], "Heated Cylinder with Boussinesq Approximation" [GGT17, Pop04] and "Cylinder Flow Around Corners" [Pop04, RG19]. The visual results are presented first which are followed by a performance analysis.

5.1 Visual Results

In the visualizations the following color maps are used for the different lines: Original pathlines (purple-white-green), observer worldline (orange-yellow), observed pathlines (blue-black-light red).

5.1.1 Four Rotating Centers

The "Four Rotating Centers" data set is used with dimensions 32x32x32 (X x Y x T) instead of 128x128x512 to test the pipeline on a small data set. In Figure 5.1 four vortices are visualized in the "Four Rotating Centers" data set [GGT17]. For better visibility Figure 5.1b shows only the observer worldline and the observed pathlines. The form of the worldline reveals the uniform rotation of the four centers.

5.1.2 Heated Cylinder with Boussinesq Approximation

The "Heated Cylinder with Boussinesq Approximation" data set [GGT17, Pop04] has a grid resolution of 150x450x2001. In Figure 5.2 three different vortices are identified. The search was done from timeslice 500 to 1200. For each vortex a different set of original pathlines was chosen and different start points for the observers were picked.

A data set is loaded with a selected timespan. If an observer start time bigger than the minimum is selected the pipeline starts a forward and a backward integration. This simultaneous operation is shown in Figure 5.3. The start point is picked in the middle of the timespan (Fig. 5.3a). Figure 5.3b shows the visualization in an intermediate state. Figure 5.3c shows the final state of the visualization. Two LICs are shown per state, one at the minimum of the timespan and one at the selected time. For this figure the timespan from timeslice 1000 to 1500 is used.

The progress of the visualization is shown with an indicator in the form of an annulus (ring). The indicator is positioned at the selected start point. When the start point is selected, the indicator is gray. It will turn green clockwise according to the percentage of the progress.

$$progress = \begin{cases} 1, & \text{if backward and forward integration finished} \\ \frac{finishedTimeSlices}{totalNumberOfTimeSlices}, & \text{otherwise} \end{cases}$$

Figure 5.3 shows the progress indicator with different progress values.

The capability to select observers at different times provides an intuitive search as the start point can be directly selected on the LIC of the selected timeslice. This is shown in Figure 5.4 where two vortices are identified by picking the observer on the LIC of the selected timeslice. This can be seen by the vertical position of the progress indicator (green ring). For a better orientation the LIC of timeslice 500 (at the bottom of selected timespan) is shown additionally to the LIC of the selected time slice. It can be seen that these vortices are also visualized in Figure 5.2, but the spacial position is shifted due the different starting times.

Producing meaningful visualizations requires that the observer worldline is placed in a way, that the original pathline swirls around the observer worldline. This is shown in Figure 5.5. For better visibility only one pathline is visualized. Two LIC images give orientation how the whole field changes over time. For this figure the data set is used from timeslice 1000 to 1500.

5.1.3 Cylinder Flow Around Corners

The "Cylinder Flow Around Corners" data set [Pop04, RG19] has a grid resolution of 450x150x1501. In Figure 5.6 one vortex is identified. The data set was loaded from timeslice 500 to 1200.

5.2 Performance Analysis

The performance of the pipeline was measured on three different data sets with varying dimensions (X x Y x T):

- "Four Rotating Centers" [GGT17], 32 x 32 x 32

- "Cylinder Flow Around Corners" [Pop04, RG19], 450 x 150 x 1501
- "Heated Cylinder with Boussinesq Approximation" [GGT17, Pop04], 150 x 450 x 2001

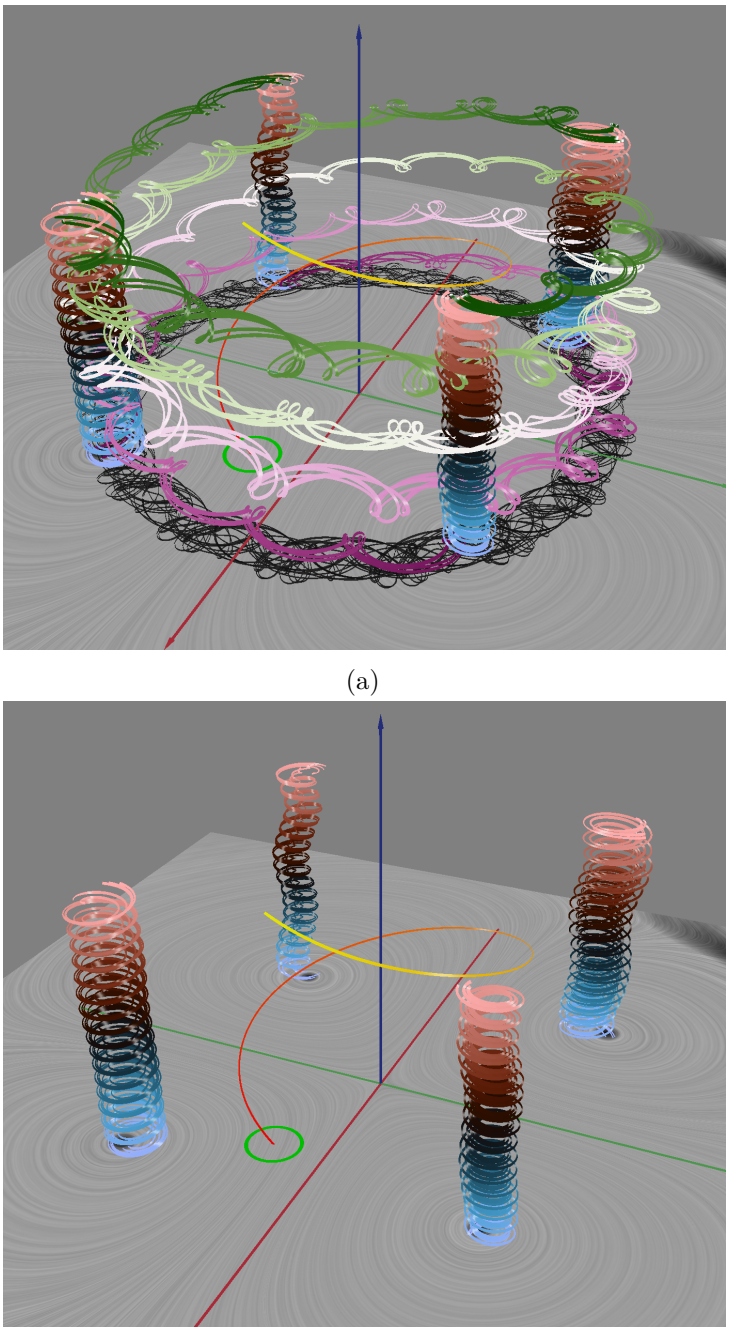
The following performance indicators were identified and measured:

- FPS idle: Default average frames per second of the framework
- FPS operation: Average frames per second during operation
- OptimizedPoints/s: Number of grid points computed by the optimization per second
- IterationTime (ms): Average duration of one iteration (optimization, integration, transformation, and visualization) in milliseconds
- TotalDuration (ms): Duration from setting of the start point to reaching the borders of the data set in the time domain
- DurationGlobal (ms): Duration of the global optimization used by Zhang et al. [ZHTR21] in milliseconds

Table 5.1 shows the performance on a dual Intel Xeon 2687W v4 (24 cores) and an NVIDIA GeForce RTX 3090 (Windows 10 64-bit, 256GB RAM). For the test data sets or data sets of comparable size, the pre-computed pipeline takes on the order of tens of seconds to compute a global solution. This might seem appropriate since the cost of pre-computation is feasible when only computed once. However, an interactive manipulation of the optimization parameters is not feasible with the pre-computed pipeline, since it requires a complete re-computation of the global solution every time a parameter is changed. Furthermore, for large data sets and more compute intensive optimization methods, the decoupling of the size of the data set from the response time of the framework should proof a significant improvement in usability in the future.

Indicator\Data Set	Four Centers	Cylinder Corners	Cylinder Boussinesq
FPS idle	24.3	19.5	15.2
FPS operation	8.1	5.2	3.8
OptimizedPoints/s	646.32	1.03	0.55
IterationTime (ms)	64.10	101.47	159.10
TotalDuration (ms)	4341.74	31407.20	95178.98
DurationGlobal (ms)	82.73	60955.73	77344.02

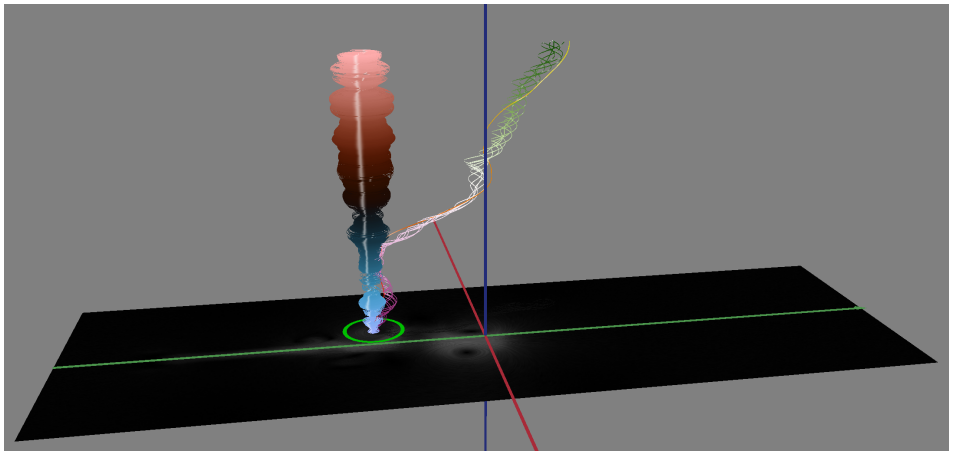
Table 5.1: The performance was tested with three data sets: "Four Rotating Centers" [GGT17], "Cylinder Flow Around Corners" [Pop04, RG19] and "Heated Cylinder with Boussinesq Approximation" [GGT17, Pop04]. The table is split into three sections. The first section compares the frames per second when the framework is idle or in operation. The second section shows measurements of the novel pipeline: Number of optimized points ($\boldsymbol{u}$-field positions) per second and time needed for one iteration (optimization, integration, transformation and visualization). The third section compares the runtime of one full run of the novel pipeline (from setting the start point to the global time boundaries of the data set) and the runtime of a global optimization done by Zhang et al. [ZHTR21].



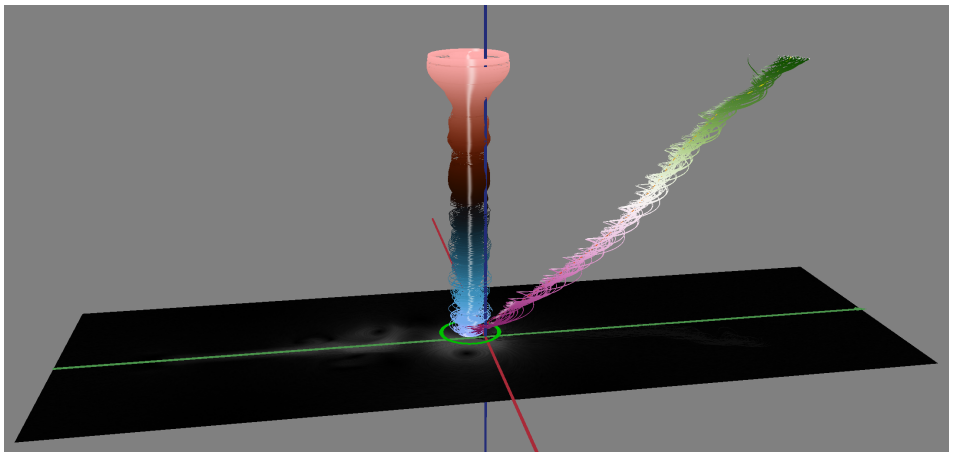
(a)

(b)

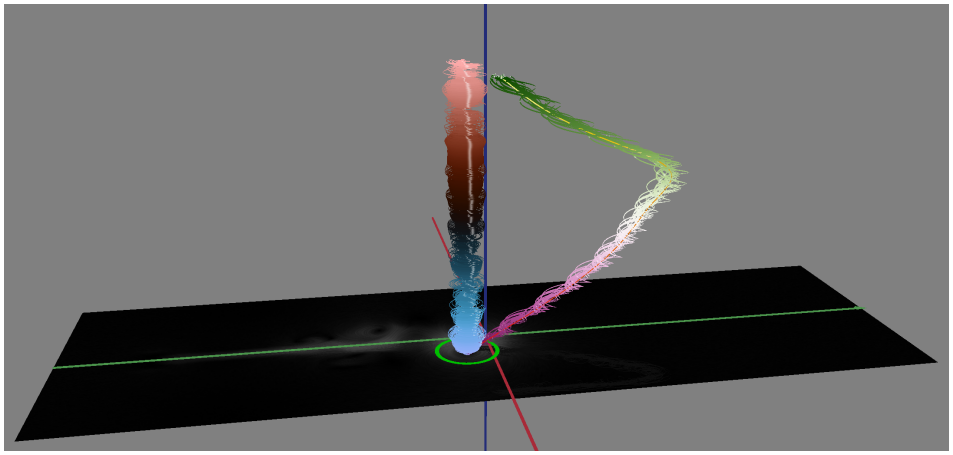
Figure 5.1: Four vortices are visualized in the "Four Rotating Centers" data set [GGT17]. For better visibility (b) shows only the worldline and the observed pathlines. The form of the worldline reveals the uniform rotation of the four centers.



(a)

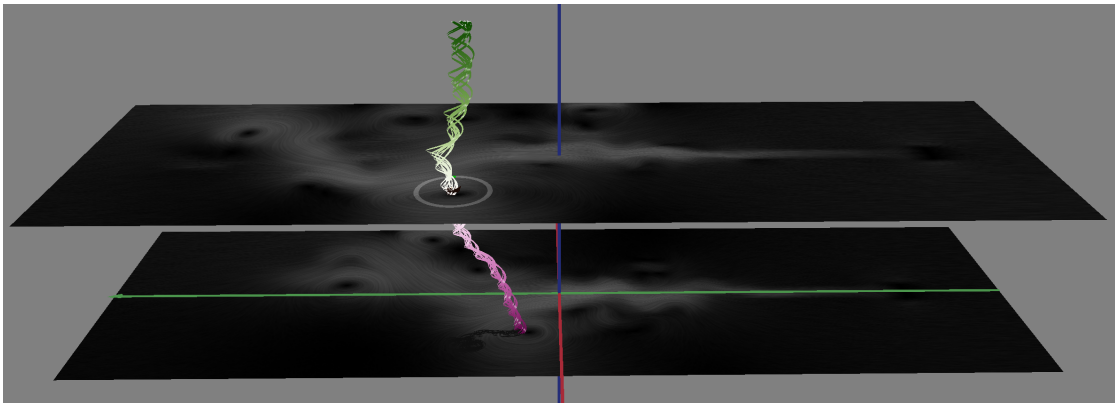


(b)

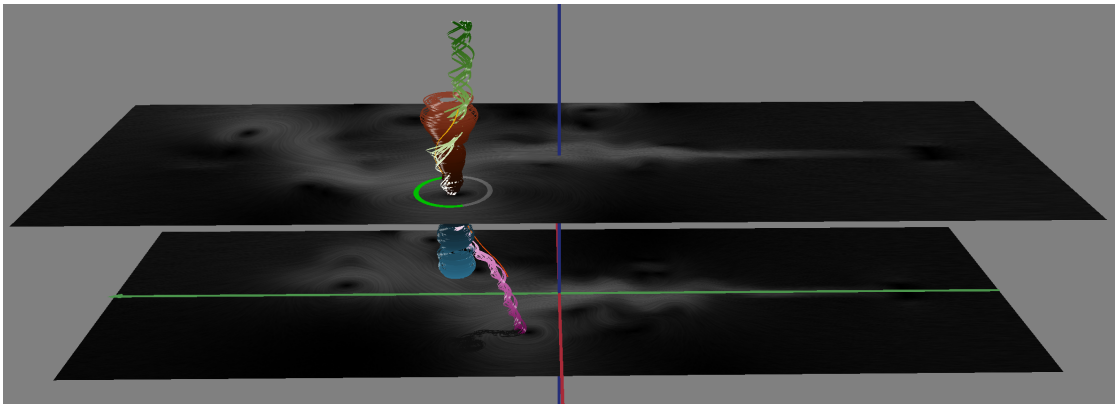


(c)

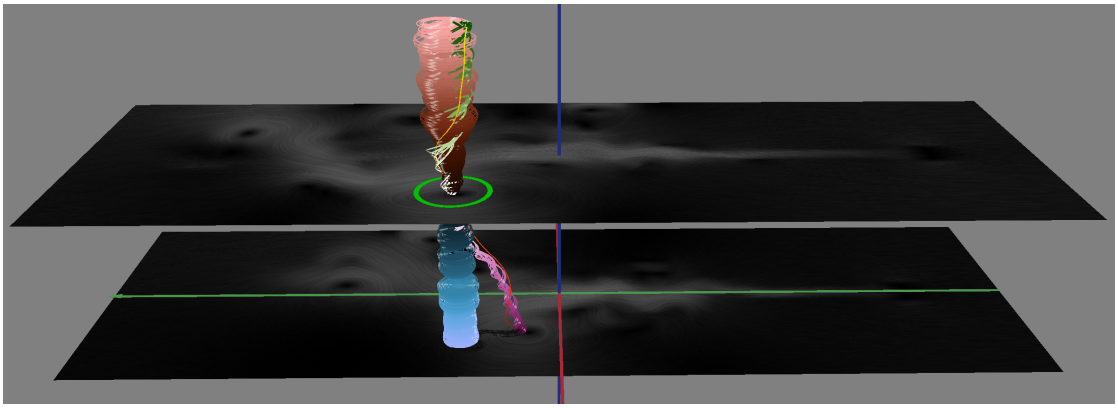
Figure 5.2: Three different vortices in the "Heated Cylinder with Boussinesq Approximation" data set [GGT17, Pop04] from timeslice 500 to 1200. All three vortices use a different set of original pathlines and a different observer.



(a)

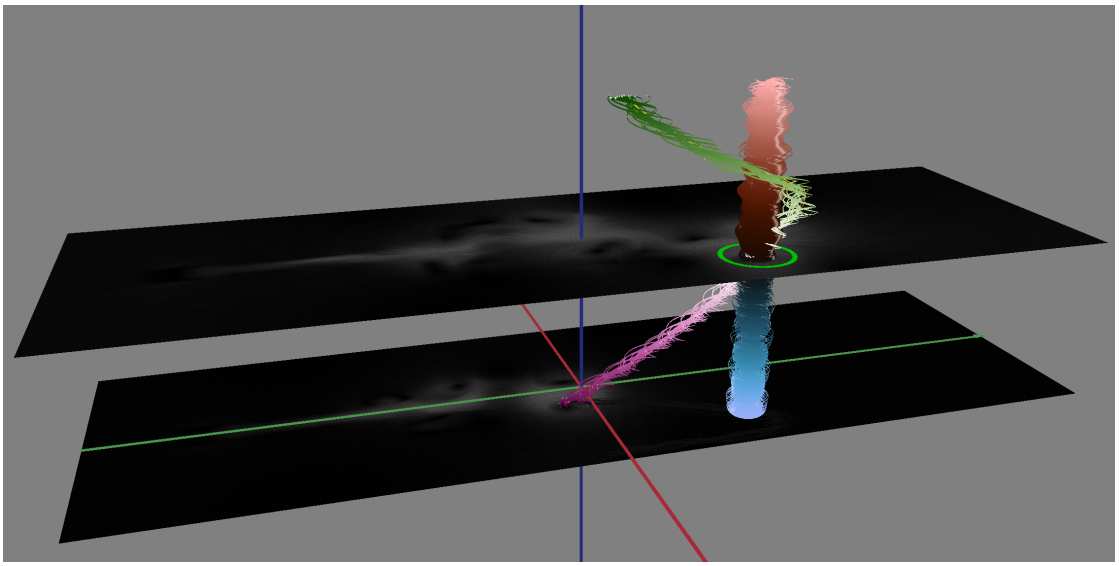


(b)

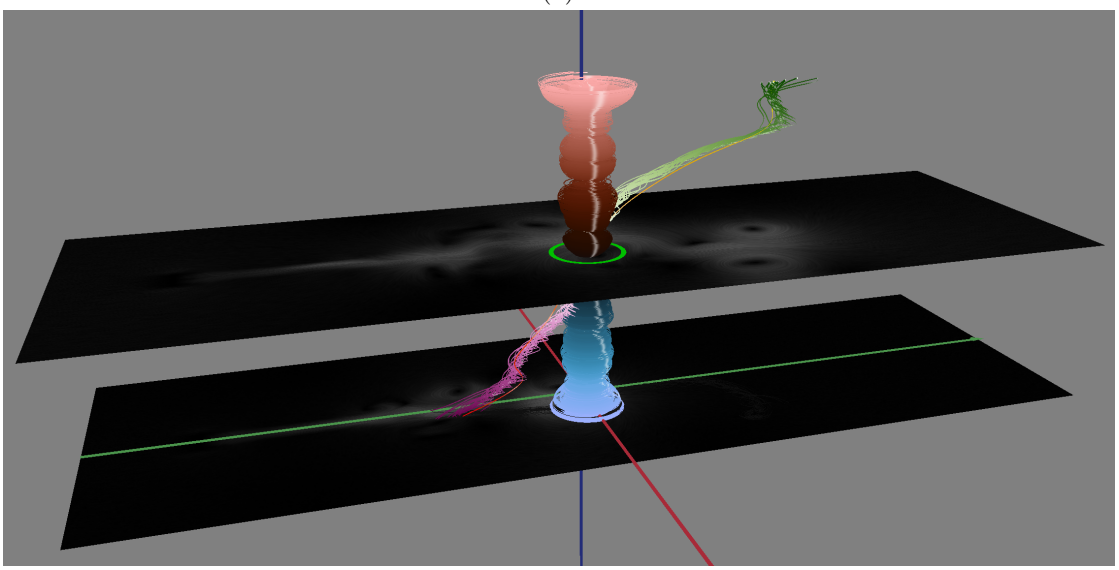


(c)

Figure 5.3: This figure shows the simultaneous forward and backward operation of the algorithm. The start point is picked in the middle of the timespan (a). (b) shows the visualization in an intermediate state. (c) shows the final state of the visualization. Two LICs are shown per state, one at the beginning of the timespan and one at the selected time. For this figure the "Heated Cylinder with Boussinesq Approximation" data set [GGT17, Pop04] is used from timeslice 1000 to 1500.



(a)



(b)

Figure 5.4: Two vortices in the "Heated Cylinder with Boussinesq Approximation" data set [GGT17, Pop04] from timeslice 500 to 1200. The observer is picked in the middle of the selected timespan. This can be seen by the vertical position of the progress indicator (green ring). For better orientation LICs are shown at timeslice 500 (at the bottom of the selected timespan) and the selected time slice. It can be seen that these vortices are also visualized in Figure 5.2, but the spacial position is shifted due the different starting times.

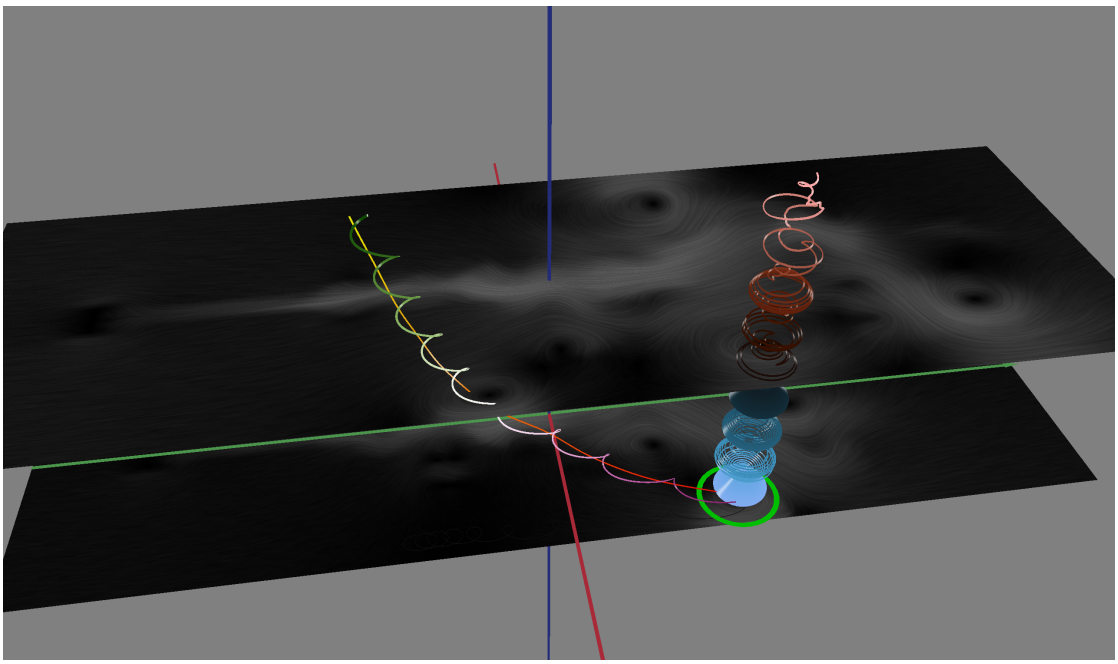


Figure 5.5: The observer worldline is in the middle of the original pathline (the pathline swirls around the worldline). This results in a meaningful visualization of the observed pathlines. For better visibility only one pathline is visualized. Two LIC images give orientation how the whole field changes over time. The data is from the "Heated Cylinder with Boussinesq Approximation" data set [GGT17, Pop04] from timeslice 1000 to 1500.

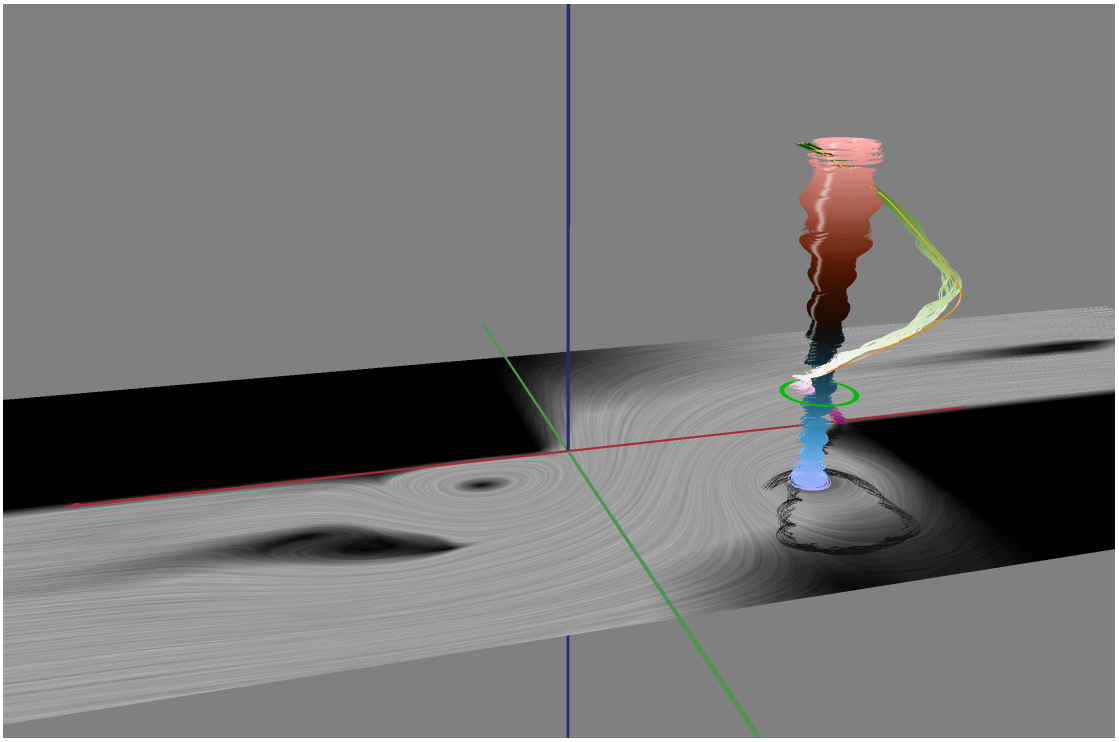


Figure 5.6: One vortex in the "Cylinder Flow Around Corners" data set [Pop04, RG19] from timeslice 500 to 1200.

CHAPTER 6

Conclusion

This thesis introduces an algorithm for interactive observer-relative flow visualization without pre-processing. This was achieved by restructuring and optimizing an existing algorithm that used pre-processing. We took advantage of reducing the area of optimization of the observer movement from the full size to a small subset of the data. The algorithm loops trough observer movement local optimization, observer worldline integration, observer pathline transformation, and visualization. After each iteration the result is directly delivered to the user. This progressive buildup provides a reactive and interactive user experience. This allows for rapid exploration of the data set as well as the parameter space of the available optimization algorithms.

6.1 Limitations

The current implementation of the pipeline is subject to different limitations. Firstly, it is only implemented for planar geometry. Enabling the algorithm to operate on other geometric objects would extend the applicability. The second limitation is the natural connection of running time and size of the data set in the time domain. A large timespan of the data set results in a long total runtime. Lastly, the proposed pipeline only addresses the responsiveness of the system, but does not enable novel exploration techniques that would make it easier to find good reference frames.

6.2 Future Work

The search for vortices is currently a tedious process that could be supported by automatic parallel search and evaluation. Around the start point additional seeds could be introduced and at all seed points the optimization and integration would be computed in parallel. After each (or several) iteration(s) the most promising observer worldline could be evaluated with a quality metric such as the time derivative. As the optimal observer may

change throughout the process, animation between the transformations could be used to provide a more pleasant interface to the user.

List of Figures

Bibliography

[CL93] Brian Cabral and Leith Casey Leedom. Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer Graphics and Interactive Techniques*, pages 263–270, 1993.

[GGT17] Tobias Günther, Markus Gross, and Holger Theisel. Generic objective vortices for flow visualization. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017.

[GST15] Tobias Günther, Maik Schulze, and Holger Theisel. Rotation invariant vortices for flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):817–826, 2015.

[GT18] Tobias Günther and Holger Theisel. The state of the art in vortex extraction. In *Proceedings of Computer Graphics Forum*, volume 37, pages 149–173. Wiley Online Library, 2018.

[Hal05] George Haller. An objective definition of a vortex. *Journal of Fluid Mechanics*, 525:1–26, 2005.

[HMTR18] Markus Hadwiger, Matej Mlejnek, Thomas Theußl, and Peter Rautek. Time-dependent flow seen through approximate observer killing fields. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1257–1266, 2018.

[JEH02] Bruno Jobard, Gordon Erlebacher, and M Yousuff Hussaini. Lagrangian-Eulerian advection of noise and dye textures for unsteady flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):211–222, 2002.

[JL97] Bruno Jobard and Wilfrid Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Visualization in Scientific Computing'97*, pages 43–55. Springer, 1997.

[Pop04] Stéphane Popinet. Free computational fluid dynamics. *ClusterWorld*, 2(6), 2004.

[RG19] Irene Baeza Rojo and Tobias Günther. Vector field topology of time-dependent flows in a steady reference frame. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):280–290, 2019.

[RMB+20] Peter Rautek, Matej Mlejnek, Johanna Beyer, Jakob Troidl, Hanspeter Pfister, Thomas Theußl, and Markus Hadwiger. Objective observer-relative flow visualization in curved spaces for unsteady 2d geophysical flows. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):283–293, 2020.

[SWH05] Jan Sahner, Tino Weinkauf, and Hans-Christian Hege. Galilean invariant extraction and iconic representation of vortex core lines. In *Proceedings of EuroVis*, volume 5, pages 151–160, 2005.

[THR+21] Holger Theisel, Markus Hadwiger, Peter Rautek, Thomas Theußl, and Tobias Günther. Vortex criteria can be objectivized by unsteadiness minimization. *Physics of Fluids*, 33(10):107–115, 2021.

[WT10] Tino Weinkauf and Holger Theisel. Streak lines as tangent curves of a derived vector field. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1225–1234, 2010.

[ZHTR21] Xingdi Zhang, Markus Hadwiger, Thomas Theußl, and Peter Rautek. Interactive exploration of physically-observable objective vortices in unsteady 2d flow. *IEEE Transactions on Visualization and Computer Graphics*, 2021.