Computational Methods for SPH-based Fluid Animation

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Stefan Holger Reinhardt

aus Tübingen

Hauptberichter: Prof. Dr. Daniel Weiskopf
Mitberichter: Prof. Dr. Bernhard Eberhardt
Prof. Dr. Jan Bender

Tag der mündlichen Prüfung: 11.12.2020

Visualisierungsinstitut der Universität Stuttgart

2021
Acknowledgments

First, I want to express my gratitude to Daniel Weiskopf and Bernd Eberhardt for giving me the opportunity to be their Ph.D. student and to pursue my research. Thank you for sharing your experience, your ongoing support, and especially your patience. The numerous discussions were essential to refine my work. In the end, your supervision and guidance made this thesis possible. Thank you for the great time I had at VISUS and HdM.

Moreover, I want to thank Jan Bender for taking the time to review my thesis and taking part in my Ph.D. defense. Additionally, I am glad that Markus Huber supervised my Bachelor and Master Thesis and that Bernd Eberhard aroused my interest in physical-based animations. This was the foundation for my Ph.D. research. Without this time, I don't think I would have conducted my Ph.D. at VISUS. In addition, I wanna thank the ‘Kooperatives Promotionskolleg Digital Media’ for funding years of my research.

I also want to thank all my co-authors and collaborators for their input and their contribution to my research: Otilia Dumitrescu, Markus Huber, Tim Krake, Michael Krone, Dieter Morgenroth, and Benjamin Wollet. You all made this work possible thank you for that.

I am especially thankful for my roommates at VISUS, namely: Moataz Abdelaal, Valentin Bruder, Moritz Heinemann, Cristina Morariu, Christoph Schulz, and Alexander Straub and – of course – for all other VIS/VISUS colleagues. You all made the (sometimes very extensive and not necessarily work-related) activities very fun. Furthermore, I am grateful for my colleagues and roommates at HdM: Jan Fröhlich, Tim Krake, Dieter Morgenroth, Hendrik Paukert, Ingmar Rieger, Benjamin Wollet, and Jonathan Ziegler. We always had fun at our office and also at our sometimes late-night extracurricular activities. Not to forget Jochen Bomm, Martin Fuchs, Johannes Schaugg, Andreas Schmid, Robin Schulte, and Uwe Schulz for giving shelter in their offices with the best coffees at HdM as well as the fun and interesting conversations during sometimes longer coffee breaks. And of course all my other colleagues. I will miss the great time we all had together.
Special thanks go to Tim Krake, Cristina Morariu, Yannic Schoof, and Benjamin Wollet for proofreading of this thesis and their useful comments. Thank you for your valuable feedback. Especially Tim Krake, had to read quite a bunch of pages, thank you for that. And not to forget, thank you Markus Huber for providing the \LaTeX\ template for this thesis and saving me hours or merely days of headaches.

I am deeply thankful for my family and friends for supporting me all the time. Especially my parents, Klaus and Ute, supported me by letting me pursue my (admittedly not always straightforward) way. Not to forget all the baby-sitter, who took care of Linus, so I had more time to work on my research (especially in these days of living with the pandemic): Alfons, Andreas, Friederike, Mercedes, Sula, and Ute special thanks to all of you. Finally, I am thanking Marie-Therese for her endless support and being (almost) always so understanding through these nice but sometimes also rough times.

This thesis is dedicated to my partner Marie-Therese and my son Linus.

— Stefan Reinhardt
Table of Contents

Acknowledgments iii
List of Figures vii
List of Tables xi
List of Abbreviations xiii
Abstract xv

Zusammenfassung xvii

1 Introduction 1
   1.1 Animated Fluids 3
   1.2 Research Questions 4
   1.3 Overview and Contributions 7
   1.4 Reused and Copyrighted Material 9

2 SPH-based Fluid Animation 11
   2.1 Governing Equations of Fluid Motion 12
   2.2 Fundamentals of SPH 17
   2.3 Spatial Discretization 21
   2.4 Temporal Discretization 24
   2.5 SPH-based Fluid Solver 26

3 Visual Debugging of SPH Simulations 35
   3.1 Related Work 37
   3.2 Visualization Requirements 38
   3.3 Design of the Visual Debugging Environment 39
   3.4 Results 44
   3.5 Discussion and Future Work 51
   3.6 Summary 52
### Table of Contents

#### 4 Fully Asynchronous SPH
- 4.1 Related Work ........................................ 54
- 4.2 Asynchronous Time Integration ..................... 56
- 4.3 Results .............................................. 62
- 4.4 Discussion and Future Work ....................... 74
- 4.5 Summary ............................................ 76

#### 5 Consistent Shepard Interpolation ..................... 77
- 5.1 Related Work ........................................ 79
- 5.2 Classical Shepard Correction ....................... 81
- 5.3 Corrected Correction ................................ 82
- 5.4 Implementation ..................................... 88
- 5.5 Evaluation .......................................... 90
- 5.6 Discussion and Future Work ....................... 97
- 5.7 Summary ............................................ 98

#### 6 Solving PDEs on Evolving Surfaces .................... 99
- 6.1 Related Work ........................................ 100
- 6.2 Model ............................................... 101
- 6.3 Method .............................................. 109
- 6.4 Results .............................................. 111
- 6.5 Discussion and Future Work ....................... 117
- 6.6 Summary ............................................ 117

#### 7 Conclusion ........................................... 119
- 7.1 Summary ............................................ 120
- 7.2 Discussion ........................................... 121
- 7.3 Future Research Directions ....................... 125

#### Co-authored references ............................... 129

#### Bibliography ........................................ 131

#### A Supplemental Images for Visual Comparison – ATS  A1
- A.1 Corner dam break .................................. A2
- A.2 Radial flow .......................................... A3
- A.3 Fountain ............................................ A5

#### B Supplemental Material – Consistent Shepard ........ A7
- B.1 Fluid Block .......................................... A8
- B.2 Fluid Pillar ......................................... A11
- B.3 Corner Dam Break .................................. A13
- B.4 Fountain ............................................ A18
- B.5 Convergence Proof ................................ A19
## List of Figures

1.1 The process of creating a physically based fluid animation. .................................. 2  
1.2 Fluid body as continuum. ......................................................................................... 3  

2.1 Side-by-side comparison between the Eulerian and Lagrangian technique. .......... 13  
2.2 Plots of the smoothing kernels and their partial derivatives. ........................... 20  
2.3 Fluid particle neighborhood. .................................................................................. 22  
2.4 Spatial grid for neighborhood search. ................................................................. 28  

3.1 A screenshot of the visual debugging environment for SPH simulations. .......... 36  
3.2 Split view with a scatter plot and a 3D view of a simulated a drop. ................. 40  
3.3 Visual analysis of a fluid pillar using scatter plots. ............................................. 41  
3.4 A split view of a PCP and a 3D view with brushing and linking. ................. 42  
3.5 A screenshot of the visual debugging environment. ........................................... 43  
3.6 3D view of a large-scale SPH simulation ....................................................... 45  
3.7 Visual debugging of boundary handling in a dam break simulation. .............. 46  
3.8 Comparison of pressure models used to simulate a fluid pillar. ..................... 47  
3.9 Comparing a dam break simulation simulated with IISPH and WCSPH .......... 49  
3.10 Visual debugging of two surface tension models in combination with WCSPH. 50  

4.1 Time step distribution for a 10M dam break scenario. ..................................... 54  
4.2 Illustration of the different time stepping models. ............................................. 57  
4.3 Neighborhood reconstruction in asynchronous time integration. ................. 58  
4.4 The parallelization concept for asynchronous time stepping. ....................... 61  
4.5 Rendering of a simulation with asynchronous time integration. ................. 63  
4.6 Strong and weak scaling study. ................................................................. 65  
4.7 Visual comparison of the time stepping methods for the radial flow. .......... 67  
4.8 Side-by-side comparison of integration methods applied to the fountain scenario. 68
4.9 Clipping boxes for isolated investigations of the fluid velocities around the collision objects. ........................................ 69
4.10 Velocity measurements of the radial flow scenario. ............ 70
4.11 Velocity measurements of the overall fountain scenario. ....... 71
4.12 Comparison of the used time integration scheme and the semi-implicit Euler in the radial flow scenario. ...................... 72
4.13 Comparison of the used time integration scheme and the semi-implicit Euler in the fountain scenario. ......................... 73
4.14 Comparison of different combinations of $\lambda_{v}$ and $\lambda_{a}$ using ATS. .................................................. 74
4.15 Density noise occurring with asynchronous integration. ......... 75

5.1 Kernel correction comparison in the fluid pillar scenario. ...... 78
5.2 Density error and noise in the fountain scenario. ................. 79
5.3 Conceptual overview of a particle neighborhood, its discretization, and employed kernel correction schemes. ................. 83
5.4 Boundary density correction. ...................................... 88
5.5 Surface stiffness constant scaling in the corner dam break scenario. ................................................... 89
5.6 Density analysis on a collapsing fluid block. ....................... 90
5.7 Density noise analysis on a corner dam break. .................... 91
5.8 Mean local density variance measured for the corner dam break scenario. .............................................. 92
5.9 Mean local density variance measured for the fluid pillar scenario. .................................................. 93
5.10 Volume preservation tests. .......................................... 94
5.11 Convergence analysis. ............................................... 95
5.12 Kernel gradient correction study. .................................. 96
5.13 Displacement and scaling experiment. ............................ 97

6.1 Illustration of the map $O$, relating two simulation states. ....... 103
6.2 Length-preserving evolution of a vector field. ....................... 106
6.3 Different stages of the method for a two-phase input simulation. 109
6.4 Method overview. ..................................................... 110
6.5 Fluid flow on a rotating sphere. .................................... 112
6.6 Thermal convection on a hemisphere. .............................. 113
6.7 Reaction-diffusion and fluid flow on a dam break scenario. .... 114
6.8 Conservation tests. .................................................... 115
6.9 Velocity drift on curved surfaces. .................................. 116

A.1 Color map for the examples in Appendix A. ......................... A1
A.2 Side-by-side comparison of the different time stepping methods for the corner dam break. ................................. A2
A.3 Side-by-side comparison of the different time stepping methods in the radial flow scenario. ................................. A3
A.4 Top view on the radial flow scenario. .............................. A4
A.5 Side-by-side comparison of the different methods in the fountain scenario. .................................................. A5
B.1 Linear color map used in the examples throughout Appendix B. ................................................................. A7
B.2 Pictograms to understand the orientation of the camera in the test scenes. ................................................. A7
B.3 Selected renderings from the fluid block scenario simulated with WCSPH. .................................................. A8
B.4 Side-by-side comparison of the different methods in the fluid block scenario simulated with DFSPH. ............ A9
B.5 Density variance measurements of the fluid block scenario. ...................................................................... A10
B.6 Selected snapshots from the fluid pillar scenario conducted with WCSPH and DFSPH. ................................. A11
B.7 Density variance measurements of the fluid block scenario. ...................................................................... A12
B.8 Selected renderings of the corner dam break scenario conducted with WCSPH. ........................................ A13
B.9 Selected renderings of the corner dam break scenario conducted with DFSPH. .......................................... A14
B.10 Selected renderings of the corner dam break scenario conducted with DFSPH. ........................................ A15
B.11 Selected renderings of the corner dam break scenario conducted with DFSPH. ........................................ A16
B.12 Density variance measurements of the corner dam break scenario to quantify the observations. .............. A17
B.13 Renderings of the fountain scenario conducted with DFSPH. ................................................................. A18
B.14 Measurements of the mean and max local density variance of the fountain scenario. .............................. A19
List of Tables

3.1 Performance tests of the visual debugging environment . . . . . . . . . . 44
4.1 Runtime comparison of FTS, GTS, ITS, and ATS . . . . . . . . . . . . . . 64
5.1 Volume preservation comparison. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATS</td>
<td>Asynchronous Time Stepping</td>
</tr>
<tr>
<td>AVI</td>
<td>Asynchronous Variational Time Integration</td>
</tr>
<tr>
<td>CDB</td>
<td>Corner Dam Break</td>
</tr>
<tr>
<td>CFL</td>
<td>Courant-Friedrichs-Lewy</td>
</tr>
<tr>
<td>CPM</td>
<td>Closest Point Method</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DFSPH</td>
<td>Divergence-Free Smoothed Particle Hydrodynamics</td>
</tr>
<tr>
<td>EOS</td>
<td>Equation of State</td>
</tr>
<tr>
<td>FTS</td>
<td>Fixed Time Stepping</td>
</tr>
<tr>
<td>fps</td>
<td>Frames per Second</td>
</tr>
<tr>
<td>GHz</td>
<td>Giga Hertz</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>GTS</td>
<td>Global Adaptive Time Stepping</td>
</tr>
<tr>
<td>IIF</td>
<td>Inter-Particle Interaction Force</td>
</tr>
<tr>
<td>IISPH</td>
<td>Implicit Incompressible Smoothed Particle Hydrodynamics</td>
</tr>
<tr>
<td>ITS</td>
<td>Individual Time Stepping</td>
</tr>
<tr>
<td>MLS RKPM</td>
<td>Moving Least Squares Reproducing Kernel Particle Method</td>
</tr>
<tr>
<td>PCISPH</td>
<td>Predictive Corrective Incompressible SPH</td>
</tr>
<tr>
<td>PCP</td>
<td>Parallel Coordinate Plot</td>
</tr>
<tr>
<td>PDE</td>
<td>Partial Differential Equation</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>PPE</td>
<td>Pressure Poisson Equation</td>
</tr>
<tr>
<td>RKPM</td>
<td>Reproducing Kernel Particle Method</td>
</tr>
<tr>
<td>ODE</td>
<td>Ordinary Differential Equation</td>
</tr>
<tr>
<td>SPH</td>
<td>Smoothed Particle Hydrodynamics</td>
</tr>
<tr>
<td>WCSPH</td>
<td>Weakly Compressible Smoothed Particle Hydrodynamics</td>
</tr>
</tbody>
</table>
Abstract

Smoothed Particle Hydrodynamics (SPH) has become one of the major approaches for physically based fluid animation in the field of computer graphics. In the last decades, many authors developed techniques to extend the original formulation, making SPH a versatile method that can model a wide range of phenomena. One focus of recent research was the development of computational methods that improve the efficiency and accuracy of the original formulation as well as the visual quality. Despite all the improvements, research on SPH is still a very active field.

This thesis presents computational methods to improve the physically based animation of fluids with SPH. The contributions address various challenges arising from the simulation of fluids using SPH. After a thorough discussion of the fundamentals of SPH-based fluid animation, an approach to visually debug SPH simulations is presented. It is designed to support the development of new computational methods. To this end, a requirements analysis is conducted, and the application is designed accordingly. In the remainder of the thesis, a particular focus is given to the discretization process. First, an asynchronous time stepping model is presented. By using individual discrete time steps per particle, the efficiency of the simulation process is improved. Secondly, a consistent kernel correction method is developed. This approach helps reduce errors arising in the spatial discretization process and, therefore, improves the accuracy of the simulation model. It is based on the well-known Shepard correction but resolves inconsistencies that arise when applying it to SPH. Lastly, an approach is presented to add fine-scale details on an animated surface, e.g., resulting from an SPH-based fluid animation. These secondary effects, modeled on the surface, are driven by the base simulation’s velocity field. Fine-scaled details can be simulated efficiently using the presented method, enriching the appearance of the fluid.

The presented methods take special requirements of computer graphics applications into account. Particular attention is paid to efficiency, accuracy, and visual quality. The efficiency of the simulation is specifically important in the field of computer graphics. Typically, the animation of fluids is iteratively refined, which results in many simulation runs of the same sequence.
Additionally, improving the efficiency of the simulation process allows for finer spatial discretization and, therefore, the level of realism increases. In physically based animation with SPH, the accuracy not only depends on the simulation resolution but also on the accuracy of the SPH approximation. Accuracy improvements help model the physical phenomena more precisely and may help enhance the level of realism. Visual quality is always of particular importance for computer graphics applications, as realistic high-quality visualizations are usually the desired result.
ZUSAMMENFASSUNG


können feine Details sehr effizient simuliert werden und das Erscheinungsbild der Fluids wird verbessert.

Physically based animation of fluids has a long history in the field of computer graphics. It is of great importance for artists to create a realistic appearance of fluids. Especially in commercials and feature films, high requirements must be met, as a fluid animation shall be indistinguishable from captured images as both often need to be composed. Realistic animations also become of more central concern for interactive applications, e.g., games or augmented and virtual reality in general. Besides the degree of realism, the efficiency of the simulation process is also essential. In addition, the animations should be artistically controllable to create the desired result, for instance by manipulating the underlying physical model.

To formally describe viscous fluid flow, the Navier-Stokes equations are typically employed. Many different techniques have been developed to solve them numerically. These techniques base on two different perspectives, namely the Eulerian or the Lagrangian viewpoint. Both of them have proven their strengths and weaknesses. This thesis focuses on a Lagrangian approach popular in the field of computer graphics. In 1983, Reeves [104] presented the first Lagrangian approach to create fluid animations suitable for films. He developed a particle system to enable the simulation of fire, smoke, water, and clouds. Since then, the demand for visual quality has been continuously increasing. A variety of models have been designed to meet these requirements. Besides visual quality, new models have been proposed to improve the simulation’s efficiency and to enable the modeling of new effects. For example, Peachy [98] presented a procedural approach to model breaking waves on a beach. Kaas and Miller [74] presented an efficient approach solving the shallow water equations.
Nowadays, complex scenarios with fine-scaled details are animated and coupled with other physically based simulations of, e.g., rigid and soft bodies or fur. Many techniques to simulate fluids have been proposed, meeting the requirements present in the field of computer graphics. One of these techniques is Smoothed Particle Hydrodynamics (SPH). It has become one of the major approaches for fluid simulation in various research disciplines. Especially in the field of computer graphics, it has gained particular importance. SPH was developed by Gingold and Monaghan [49] as well as by Lucy [85], originally focusing on the modeling of astrophysical phenomena. It is a Lagrangian approach, used to simulate the mechanics of a continuum media. With SPH, the fluid is divided into a set of evolving particles and the physical quantities are calculated on them as a weighted mean of relevant properties of surrounding particles (i.e., particles closer than a certain distance to the currently considered one). As mentioned, viscous fluid flow is simulated with SPH by solving the Navier-Stokes equations numerically. To this end, an initial value problem is formulated and solved at discrete locations in time and space.

In general, the process of creating such animations can be divided into three different steps. An outline is given in Figure 1.1. First, the physical phenomenon is typically described by a set of partial differential equations (PDEs) called governing equations (in this case, the Navier-Stokes equations). Next, they are approximated using a numerical simulation system (here SPH). Finally, the computational solution (the simulated fluid) is visualized, e.g., by rendering selected states, resulting in an animation of the fluid given by the sequence of rendered images. In this thesis, computational methods to improve SPH-based fluid animation are investigated. More precisely, the presented research focuses on how to improve the accuracy, efficiency, and visual quality of the numerical simulation part; therefore, it pertains to the second step in the overall process of creating a physically based fluid animation (see Figure 1.1).

In the remainder of this chapter, necessary terms are described (Section 1.1), research aspects of this thesis are discussed conceptually, and research questions are formulated (Section 1.2). Subsequently, the particular contributions and the outline of this thesis are presented (Section 1.3).
1.1 Animated Fluids

A fluid body $\mathcal{F}$ can be considered as a set of infinitesimal spherical material elements $\mathbf{f} \in \mathcal{F}$, called fluid particles. They are located at positions $\mathbf{x}_f \in \mathbb{R}^3$ and the shape of $\mathcal{F}$ is determined by their configuration $\mathcal{C} = \{\mathbf{x}_f | f \in \mathcal{F}\}$. When animating the fluid body $\mathcal{F}$, the individual particle positions change over time, i.e., $\mathbf{x}_f(t)$ is actually a function with respect to time. The configuration $\mathcal{C}(t) = \{\mathbf{x}_f(t) | f \in \mathcal{F}\}$ defines how the shape of the fluid body evolves. An illustration of a fluid body is given in Figure 1.2, where two different states of the animation are shown. The shape of the fluid body at times $t_1$ and $t_2$ is determined by the configurations $\mathcal{C}(t_1)$ and $\mathcal{C}(t_2)$.

The motion of a body based on physical principles, is typically defined by a PDE of the form

$$\frac{D\mathbf{v}}{Dt} = f(t, q, \mathbf{v}, \partial_1 q, \partial_2 q, ..., \partial_1 \mathbf{v}, \partial_2 \mathbf{v}, ...),$$  \hspace{1cm} (1.1)

where $\mathbf{v}$ denotes the fluid’s velocity and $q$ is an arbitrary attribute (e.g., density or pressure). Without loss of generality, only one attribute $q$ is shown here.

The term $\partial_i := \left\{ \frac{\partial^{\left|\alpha\right|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} ... \partial x_n^{\alpha_n}} \right\}_{\left|\alpha\right| = i}$ denotes the set of all partial derivatives of order $i$, with $\alpha$ being a multi-index. For an inviscid fluid, $f$ is defined by the Euler momentum equation and for a viscous fluid by the Navier-Stokes momentum equation. Nonetheless, more than one PDE is usually needed to describe the underlying physical phenomena completely.

A solution of Equation 1.1 is typically computed numerically. To this end, a discrete representation of the fluid body is needed. With Lagrangian approaches, this representation is a set of points. The number of evaluation points defines...
the fluid resolution, i.e., the level of detail on which the fluid characteristics can be simulated. The more evaluation points are used, the more fine-scale details can be modeled. However, the computational cost also increases at the same time.

Having approximated the right-hand side of Equation 1.1, the fluid body can be evolved. Numerical time integration schemes are used to compute new particle positions \( x_f(t + \Delta t) \) for a defined time step \( \Delta t \). This means that \( t \) is also evaluated at discrete points in time. The smaller \( \Delta t \), the more accurate the temporal discretization, but also the higher the computational cost. The algorithm that computes the overall solution (i.e., approximates Equation 1.1) and evolves the fluid body, is called fluid solver.

### 1.2 Research Questions

SPH-based animation of fluids has been a very active research field in computer graphics over the last decades. Many authors have developed methods to improve the base algorithm. In classical engineering applications, the accuracy of simulations is essential. For computer graphics applications, the efficiency, visual quality, and level of realism are of more central concern. Many methods have been developed addressing a variety of aspects, e.g., modeling additional physical effects enhancing the visual quality and degree of realism (for instance volume conservation, surface tension, or surface extraction), or improving the efficiency and stability (e.g., spatial or temporal adaptive model).

Despite all progress, research on SPH is still a field of current interest. This thesis comprises work concerning the development of computational methods for SPH-based fluid animation. More precisely, the goal of this thesis is to improve the numerical simulation aspect (the second step in Figure 1.1). Thereby, special attention is given to the physical models’ discretization process, improving the accuracy, efficiency, and visual quality of the simulations. Although accuracy might typically be of more central concern for classical engineering applications, it is also essential in the field of computer graphics. For instance, increasing the accuracy of an approximation may help improve the stability and, therefore, also the efficiency of the simulation method. Furthermore, an accurate representation of the underlying physical model is essential for the degree of realism and visual quality, which is typically a central aspect of computer graphics applications. In essence, the overall research goal of this thesis is:

**Overall Goal**

*Improving the numerical simulation aspect of an SPH-based fluid animation.*
This goal can be achieved in various ways. In this thesis, it is pursued by examining certain aspects of the development process. First, it is investigated how existing methods can be analyzed and the development process of new ones can be supported. To perform a numerical simulation, the set of PDEs (describing the physical model) is evaluated at discrete locations in space. The result of the evaluation is used to evolve the fluid body using time integration schemes, i.e., it is used to compute the particle positions at discrete points in time. Therefore, it is discussed how the efficiency and accuracy of the spatial and temporal discretization can be improved. Finally, it is investigated how to enrich an existing fluid animation by modeling secondary effects on its surface, improving the visual quality. In the following, these aspects and their challenges are discussed in detail. Additionally, research questions are formulated for each aspect.

**Visual Analysis of Computational Methods.** A first step to develop new methods for SPH is to understand the characteristics of existing ones. The development process of novel methods is challenging, and continuous evaluation is crucial for success. Such an evaluation is challenging due to many reasons: First, a method can typically not be investigated isolated but often interacts with others. Moreover, a variety of parameters have to be adjusted and lead to varying results or instabilities of the simulation itself. Debugging such development processes can be tedious. With classical debugging approaches, such as analyzing the textual output of numeric values, it is almost impossible to approach this task. With SPH, a large number of particles with many associated attributes have to be processed, which makes it unfeasible to obtain many useful insights. These issues are approached by investigating **Research Question 1:**

**RQ 1**

How to support the development process of computational methods for SPH-based fluid simulation?

**Efficient Spatio-Temporal Discretization.** In general, the smaller the time step $\Delta t$ used for the time integration scheme, the more accurate the approximation. However, the increase in computational cost is not neglectable. For computer graphics applications, efficiency and artistic controllability are often preferred over accuracy, as long as the animation of the fluid is stable and physically plausible (i.e., the degree of realism is not decreased). For explicit time integration schemes, $\Delta t$ is usually chosen to be as large as possible while maintaining a stable simulation. A stability criterion needs to be evaluated to determine the maximum possible $\Delta t$. The so-called Courant-Friedrichs-
Lewy (CFL) condition provides a necessary condition for the stability of the simulation. When applying a uniform time step, the smallest $\Delta t$ needed is determined using the CFL at all the fluids spatial evaluation positions. Therefore, the computational cost (with respect to the temporal discretization) is reduced almost to a minimum when only allowing for a uniform time step scheme. Nonetheless, this time step restriction may only be necessary for some evaluation positions $x_F$ within the fluid body. To further reduce the computational cost, individual time step sizes for each fluid particle $f$ are needed. This aspect is addressed by investigating Research Question 2:

**RQ 2**

*How to improve the efficiency of SPH-based fluid animation with adaptive time discretization?*

**Accurate Spatial Discretization.** For the animation of fluids with SPH, a discrete representation of the fluid body is mandatory. Due to the spatial discretization process, inevitable errors are introduced. Even though increasing accuracy is not necessarily the primary objective, it may help increase the stability of the simulation and, therefore, the efficiency. For example, erroneous forces occur due to an incorrect approximation of fluid properties. These may cause instabilities which could be avoided. In this thesis, correction schemes that reduce such errors are investigated in Research Question 3:

**RQ 3**

*How to improve the accuracy of the spatial discretization process?*

**Secondary Effects on the Surface of an Animated Fluid.** On the one hand, a high spatial resolution of the discrete representation of the fluid body allows one to model fine-scaled details. These are very important for the degree of realism. On the other hand, a high resolution also means high computational cost. In computer graphics applications, typically, only the surface of the fluid body is visualized. Therefore, fine-scaled effects on the surface are particularly important for visual quality. In Research Question 4, it is discussed how to model such details on an evolving surface:

**RQ 4**

*How to model fine-scaled effects on the surface of an SPH-based fluid animation?*
1.3 Overview and Contributions

This section provides an outline of the thesis and my contributions to the presented topics. Chapter 2 provides the necessary background and basics of physically based animation of fluids with SPH. It summarizes necessary previous work. Chapters 3, 4, 5, and 6 present the technical contributions of this thesis. I am the first author for all of the covered topics, except for Chapter 6. If not stated otherwise, I developed the respective software prototypes and evaluated the methods. My supervisors, Bernhard Eberhardt and Daniel Weiskopf, were involved in all the presented topics and are co-authors of each publication.

Chapter 3 – Visual Debugging of SPH Simulations. This chapter addresses Research Question 1. To this end, an approach to visually debug SPH-based simulations of fluids is presented. It differs from a conventional debugger, as properties of SPH simulations are systematically visualized rather than components of the simulation code. The goal of the visual debugging environment is to support developers and researchers in the process of developing and verifying simulation models. Besides the 3D visualization of the particles, parallel coordinate plots and scatterplots are used to show the multidimensional attributes of the simulation. Employing brushing and linking enables the interactive exploration of simulation results, and information about the simulations can be revealed. After describing the components of the visual debugging environment, four case studies for visual debugging of SPH simulation are discussed, to illustrate the versatility of our environment. The contributions of this chapter were published at the International Conference Information Visualisation [3]. The publication was co-authored with Otilia Dumitrescu, who provided most of the implementations as part of her Masters Thesis, Markus Huber, and Michael Krone, who supervised the implementation process and partly designed the visual debugging environment. I was responsible for the design of the visual debugging environment (together with Otilia Dumitrescu, Markus Huber, and Michael Krone), designing the case studies, and the systematic analysis of the simulations.

Chapter 4 – Fully Asynchronous SPH. An asynchronous time integration method that allows for dedicated time steps for each particle is presented in this chapter, investigating Research Question 2. By handling the time step restriction for each particle individually, computational resources are focused on important regions, and the overall computation time can be reduced considerably while ensuring stability. The presented method is tailored for SPH-based simulation of fluids but could also be used for any particle-based simulation model. In contrast to synchronous or regional time integration
models, each particle has its own timeline and is advanced individually. Hence, there is no global simulation time anymore. However, this means that there is also no consistent simulation state at any time. This is an issue as neighboring particles are needed to compute the particles’ attributes. It is resolved by synchronizing the neighboring attributes locally. To advance the simulation globally, the attributes are always computed for the particle that lags behind most. This is achieved by processing the particles within a priority queue. While SPH simulations with a uniform time step can easily be parallelized on shared memory architectures, parallelism with this asynchronous integration scheme is no longer straightforward. Therefore, a parallelization technique for asynchronous time integration is introduced. The contributions of this chapter were published at the Symposium on Computer Animation [4]. The paper was co-authored with Markus Huber. We both developed the idea of increasing the efficiency of SPH-based fluid animation using an asynchronous time integration scheme. I developed the actual time stepping model, including the algorithm for the parallelization. Moreover, I performed the experiments. Markus Huber helped refine the idea, supported the writing, developed the illustrations of the model, and created the particle renderings.

Chapter 5 – Consistent Shepard Interpolation for SPH-Based Fluid Animation

This chapter investigates Research Question 3. A technique to improve the accuracy of the discretization of a fluid body is presented. The approach is based on the Shepard correction, which reduces the interpolation errors from irregularly spaced data. The weighted sum of the smoothing kernel function values in the local neighborhood is used to normalize the kernel function. The kernel function values depend on the densities of neighboring particles. With classical Shepard correction, the densities are computed with the uncorrected kernel. This results in an inconsistent and error-prone formulation. The presented approach resolves the inconsistencies by using the corrected densities to compute the exact kernel correction factor and, thereby, increases the accuracy of the simulation. An efficient and easy-to-implement algorithm to solve the implicit problem by applying the power method is presented. Additionally, a kernel gradient correction scheme to achieve constant completeness is presented. It is demonstrated how the model can be applied to improve the density distribution on rigid bodies when using the rigid-fluid coupling approach of Akinci et al. [9]. The contributions of this chapter were presented at SIGGRAPH Asia and published in ACM Transactions on Graphics [5]. The publication was co-authored with Tim Krake. I came up with the idea of using corrected densities to normalize the smoothing kernel function, to end up with a consistent formulation. Tim Krake came up with the idea of using the power method to solve this problem iteratively. Moreover, he was responsible for the mathematical analysis of the problem and the convergence proof of
our algorithm. I developed the model for the gradient correction as well as the boundary treatment and the local density variance as a measuring tool. In addition, I was responsible for the implementation and evaluation of the proposed algorithm.

Chapter 6 – Solving PDEs on Evolving Surfaces  In this chapter, an efficient model to simulate fluid flow on an evolving surface is presented, addressing Research Question 4. Given any animated surface $M$ (such as a surface generated from an SPH-based fluid simulation), a secondary fluid simulation is performed on a narrow band around $M$. The idea is to describe the dynamics by numerically solving a PDE on an evolving manifold $M$. Such a PDE could, e.g., be the Navier-Stokes momentum equation or a reaction-diffusion equation. The modeled dynamics are influenced by two factors: the evolution of the surface itself and the secondary dynamics on the surface. In addition, the coupling between these two aspects has to be modeled. A unified method to model the dynamics on the surface by following physical principles is presented. The contributions of this chapter were presented at the Symposium on Computer Animation and published in Computer Graphics Forum [2]. It was co-authored with Dieter Morgenroth, who came up with the idea of performing a secondary simulation on top of an evolving surface. Moreover, he was responsible for the realization using the Closest Point Method, the actual implementation, and the evaluation of the method. I was responsible for developing the underlying mathematical model used in the simulation, as well as its analysis. In addition, I helped with generating the examples.

Prior to my doctoral research at the University of Stuttgart, I was involved in a publication presented at the Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS) in 2015 [1]. In this publication, a benchmark test to evaluate surface tension models that are developed for SPH-based fluid simulation is proposed.

1.4 Reused and Copyrighted Material

Material from an IEEE-copyrighted paper is partly reused with kind permission of IEEE (Thesis / Dissertation Reuse):

Moreover, material from the following papers is reused in this thesis. The copyright is owned by the authors according to the respective ACM Publishing License Forms:


In addition, material from a copyrighted paper is partly reused with the kind permission of John Wiley and Sons following the license agreement for Dissertation/Thesis use (License Number 4971560970239):

This chapter provides an introduction to SPH-based Fluid Animation. To this end, a derivation of the needed governing equations is given. Moreover, previous work is summarized that is used in the latter of the thesis and needed to discuss the presented research.

SPH is an approximation technique used in this thesis to simulate the motion of a fluid. It was initially introduced by Gingold and Monaghan [49] to solve astrophysical problems. Desbrun and Gascuel [39] introduced it to the computer graphics community to animate deformable solids. Several extensions were proposed to address drawbacks of the original method. An overview of state-of-the-art techniques can be found in the work of Koschier et al. [78]. In this chapter, mathematical and physical foundations to model incompressible viscous fluid flow with SPH are given. In Section 2.1, equations governing the motion of a fluid are derived based on the physical laws for the conservation of mass and momentum. Fundamentals of the SPH-method are derived in Section 2.2. Afterward, spatial and temporal discretization of the governing equations using SPH are given in Sections 2.3 and 2.4. By using these insights, a basic SPH algorithm can be formulated. This is discussed in Section 2.5, as well as additions needed to build a fully operational application for SPH-based fluid animation. It includes neighborhood search, collision handling, surface tension, and incompressibility constraints.
2.1 Governing Equations of Fluid Motion

The governing equations of fluid motion are based on three different fundamental physical laws:

1. **Conservation of Mass**
   In a control volume, mass is neither destroyed nor created, i.e., in a closed system, the overall mass of a continuum does not change over time.

2. **Conservation of Momentum**
   The sum of forces acting on the continuum equals its rate of change of total momentum.

3. **Conservation of Energy**
   In a closed system, the energy of a continuum remains constant. However, it may be converted from one form into another.

These conservation laws provide information about mass, momentum, and energy transport within a continuum. The energy conservation law deals with thermodynamic effects. As thermodynamic effects are not considered in this thesis, the energy conservation law is neglected. Therefore, the dynamics of a continuum is described by considering the two aspects: mass and momentum transport. The continuity equation defines the mass transport, and the Navier–Stokes momentum equation governs the momentum transport for a viscous fluid. Before going into detail about the considered governing equations, essential terms and perspectives are discussed that are needed to describe the motion of a fluid.

In this thesis, the considered physical phenomena are on a macroscopic length scale focusing on three-dimensional fluid flow. Therefore, the fluid is treated as a continuum. In the field of continuum mechanics, there exist different perspectives to describe the dynamics of a fluid, namely the Eulerian and the Lagrangian viewpoint. When using the Eulerian viewpoint, fluid properties are evaluated at fixed positions in space. In contrast, in the Lagrangian perspective, fluid properties are carried along with the fluid. A side-by-side comparison of both is given in Figure 2.1. As mentioned, the fluid body \( \mathcal{F} \) is considered to be a set of fluid particles \( f \in \mathcal{F} \). Each of the particles represents a certain amount of the fluid, i.e., they have constant mass. For compressible fluids, their volume may change over time. Fluid properties \( \Phi \) (such as velocity \( \mathbf{v} \) and density \( \rho \)) are either scalar or vector fields. They are defined as functions over space and time, i.e., \( \Phi : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^d \), where \( d \in \{1,3\} \) depending on the type of property (scalar or vectorial). As the fluid evolves, the quantity \( \Phi \) is altered. Its evaluation \( \Phi(\mathbf{x}_f,t) \) is moved along with the fluid particle \( f \). To compute
2.1. governing equations of fluid motion

Figure 2.1: Side-by-side comparison between the Eulerian and Lagrangian viewpoint. The velocities are visualized. In the Lagrangian perspective (left), the fluid quantities (such as velocity or density) are carried along with the moving particles. In the Eulerian perspective (right), they are evaluated at fixed locations in space.

the rate of change of a quantity at location \( x_f(t) \) the material derivative \( \frac{D}{Dt} \) is introduced. It reads

\[
\frac{D\Phi}{Dt}(x_f(t), t) = \lim_{\Delta t \to 0} \frac{\Phi(x_f(t) + \Delta t v_f(t), t + \Delta t) - \Phi(x_f(t), t)}{\Delta t},
\]

(2.1)

where \( v_f(t) = \frac{dx_f}{dt}(t) \). In the Lagrangian perspective, the quantity is already carried along with the fluid particles \( f \). This means that \( \frac{D\Phi}{Dt} \) simplifies to the partial derivative with respect to time: \( \frac{D\Phi}{Dt}(x_f(t), t) = \frac{\partial \Phi}{\partial t}(x_f(t), t) \). In contrast, with the Eulerian perspective fluid quantities are evaluated at fixed locations \( x \). Performing a Taylor series expansion around \((x, t)\) results in

\[
\Phi(x + \Delta t v(x,t), t + \Delta t) = \Phi(x,t) + \Delta t \frac{\partial \Phi}{\partial t}(x,t) + \Delta t v(x,t) \cdot \nabla \Phi(x,t) \]

\[
+ O(\Delta t^2 v(x,t), \Delta t^2).
\]

(2.2)

Note that \( \Phi \) is generally understood as a tensor field of arbitrary order. In this thesis, \( \Phi \) is either a scalar field (zeroth-order tensor) or a vector field (first-order tensor). That means, for a scalar field \( \Phi \), \( \nabla \Phi \) is the corresponding gradient field. If \( \Phi \) is a vector field, \( \nabla \Phi \) is the Jacobian and for Cartesian coordinates it holds that \( v \cdot \nabla \Phi = (\nabla \Phi) v \). With Equation 2.2, the material derivative can be defined as

\[
\frac{D\Phi}{Dt}(x,t) = \frac{\partial \Phi}{\partial t}(x,t) + v(x,t) \cdot \nabla \Phi(x,t).
\]

(2.3)

As mentioned in Chapter 1, this thesis focuses on computational methods for SPH-based fluid animation. SPH is a Lagrangian technique and, hence, the
Lagrangian frame of reference is used throughout this thesis. Nonetheless, to derive the governing equations for fluid motion, it is advantageous to take the Eulerian perspective, which is done for the rest of this section. Note that the resulting governing equations are independent of the viewpoint (with the Lagrangian perspective, the material derivative just simplifies to the partial derivative).

2.1.1 Conservation of Mass

The above mentioned physical law about the conservation of mass states that the total mass $m_F(t)$ of a fluid stays constant over time (if no sinks or sources are present), i.e.,

$$\frac{d}{dt} m_F(t) = \frac{d}{dt} \int_{C(t)} \rho(x,t) dV(x,t) = 0,$$

where $dV(x,t)$ denotes the material volume element at location $x$ and time $t$. Unless explicitly needed, $t$ and $x$ will be skipped for readability reasons. When applying Reynolds transport theorem and the divergence theorem, Equation 2.4 can be reformulated as

$$0 = \frac{d}{dt} \int_C \rho dV = \int_C \frac{\partial \rho}{\partial t} dV + \int_{C \setminus C^o} \rho \mathbf{v} \cdot \mathbf{n} dA$$

where $C \setminus C^o$ is the boundary (the closure $C$ minus the inner $C^o$) of $C$, and $\mathbf{n} dA$ the oriented surface element. As Equation 2.5 holds for arbitrary particle configurations $C$, it can be followed that the integrand has to be zero, i.e.,

$$\frac{D\rho}{Dt} + \rho (\nabla \cdot \mathbf{v}) = 0.$$

Equation 2.6 is called the mass continuity equation. It describes the mass transport within the fluid over time (or more precisely, the evolution of density).

Incompressible viscous fluids are considered in this work. For such a kind of fluid, the volume and, therefore, the density of a fluid particle does not change (i.e., $\frac{D\rho}{Dt} = 0$). Hence, Equation 2.6 simplifies to a volume conservation law stating that the divergence of the velocity field is zero:

$$\nabla \cdot \mathbf{v} = 0.$$
2.1.2 Conservation of Momentum

The incompressible Navier-Stokes momentum equation provides information about the dynamic behavior of an incompressible viscous fluid. It can be derived from the momentum conservation law. As mentioned, it states that the change of momentum equals the overall force $F$ acting on it, i.e.,

$$\frac{d}{dt} \int_C \rho v dV = F. \quad (2.8)$$

Applying Reynolds transport theorem, the divergence theorem, and Equation 2.6, the left-hand side of Equation 2.8 can be reformulated as

$$\frac{d}{dt} \int_C \rho v dV = \int_C \frac{D(\rho v)}{Dt} + \rho v \nabla \cdot v dV = \int_C \rho \frac{Dv}{Dt} dV. \quad (2.9)$$

The overall force $F$ on the right-hand side of Equation 2.8 is a sum of different forces that can be distinguished into two types: surface and body forces. Whereas the surface forces act as stress on the surface of $F$, the body forces are long-range forces acting on the entire fluid body $F$. According to Batchelor [16], the force acting on a surface element $dA$ with normal $n = (n_1, n_2, n_3)^T$ can be expressed as

$$\sigma n = \sum_{i=1}^{3} \begin{pmatrix} \sigma_{i1} n_1 \\ \sigma_{i2} n_2 \\ \sigma_{i3} n_3 \end{pmatrix}, \quad (2.10)$$

where $\sigma \in \mathbb{R}^{3 \times 3}$ denotes the Cauchy stress tensor. The overall force $F$ can therefore be expressed as

$$F = \int_C \rho a^b dV + \int_{C \setminus C_0} \sigma n dA, \quad (2.11)$$

where $a^b$ is the acceleration due to body forces, e.g., gravity. With these insights, equation 2.8 can be reformulated using the divergence theorem and Equation 2.9. It then reads

$$\int_C \rho \frac{Dv}{Dt} dV = \int_C (\rho a^b + \nabla \cdot \sigma) dV. \quad (2.12)$$

This is true for arbitrary configurations $C$ and it follows that

$$\frac{Dv}{Dt} = \frac{1}{\rho} \nabla \cdot \sigma + a^b. \quad (2.13)$$
This equation is known as the Cauchy momentum equation. It describes the momentum transport of any mass-conserving continuum. The Cauchy stress tensor $\sigma$ can be decomposed into deviatoric $\tau$ and volumetric (also known as hydro-static) $\pi$ stress via $\sigma = \tau + \pi$. If a fluid is at rest, the volumetric stress $\pi$ is defined by $\pi = -pI$, where $p$ is the hydro-static pressure. For a moving fluid, the volumetric stress is defined as the mean normal stress, i.e., $\pi = \frac{1}{3} \text{tr}(\sigma)I$ and pressure $p$ is accordingly given by $p = -\frac{1}{3} \text{tr}(\sigma)$. Therefore, $\sigma$ can be written as

$$
\sigma = \frac{1}{3} \text{tr}(\sigma)I + \sigma - \frac{1}{3} \text{tr}(\sigma)I = -pI + \tau.
$$

Inserting Equation 2.14 into Equation 2.13 leads to

$$
\frac{Dv}{Dt} = -\frac{1}{\rho} \nabla p + \frac{1}{\rho} \nabla \cdot \tau + a^b.
$$

Note that for an inviscid fluid, the deviatoric stress $\tau$ is zero, and Equation 2.15 simplifies to the Euler equations. As viscous flow is considered, a change in the particle rises shear stress that tends to oppose that change. For Newtonian fluids, the shear stress is proportional to the gradient of the velocity field, i.e., the Jacobian $\nabla v$ of the velocity $v$. As isotropic fluids are considered, $\nabla v$ can be projected onto its symmetric part $D = \frac{1}{2}(\nabla v + (\nabla v)^T)$. The second-order tensor $D$ is called rate-of-strain tensor, and the deviatoric stress $\tau$ is proportional to $D$. The dynamic viscosity coefficient $\mu$ describes the strength of the shear stress and, hence, $\tau$ can then be written as $\tau = \mu(\nabla v + (\nabla v)^T)$. In Equation 2.15, the divergence of $\tau$ is needed to compute the acceleration caused by shear stress:

$$
\nabla \cdot \tau = \mu \nabla \cdot (\nabla v + (\nabla v)^T) = \mu(\nabla (\nabla \cdot v) + \nabla^2 v),
$$

where $\nabla^2 v$ is the Laplacian of the velocity field. For $v = (v_0, v_1, v_2)^T$, it is defined as $\nabla^2 v := (\nabla^2 v_0, \nabla^2 v_1, \nabla^2 v_2)^T$. As mentioned above, the velocity field is divergence-free for incompressible fluids, i.e., $\nabla \cdot v = 0$. Therefore, the divergence of the deviatoric stress tensor reads $\nabla \cdot \tau = \mu \nabla^2 v$ and Equation 2.15 can be expressed as

$$
\frac{Dv}{Dt} = -\frac{1}{\rho} \nabla p + \frac{\mu}{\rho} \nabla^2 v + a^b.
$$

This equation is called the Navier-Stokes momentum equation for incompressible fluids.
2.2 Fundamentals of SPH

In this thesis, SPH is used to animate a fluid by employing the governing equations derived in the last section. Therefore, the properties $\Phi$ of the fluid need to be evaluated using SPH to describe the fluid’s motion. As discussed in the previous section, spatial derivatives of the fluid quantities need to be calculated as well. In particular, the density $\rho$, the pressure gradient $\nabla p$, and the Laplacian of the velocity field $\nabla^2 v$ need to be calculated to solve the governing equations. With SPH, the mentioned field quantities are approximated. Its basic concept will be described in this section, as well as how to approximate the fluid quantities. Subsequently, it will be discussed how the spatial derivatives of the fluid quantities can be approximated.

SPH is a technique that employs an integral representation of the evaluation $A(x,t)$ of any arbitrary field variable $A$ at location $x$ and time $t$. As it is a spatial approximation technique, $t$ is assumed to be constant for now and will be skipped for the rest of this section. The integral representation of $A(x)$ can be written as

$$A(x) = \int_{\mathbb{R}^3} A(x') \delta(\|x - x'\|) dV(x'),$$

where $\delta$ is the delta distribution.

The idea of SPH is to approximate the delta distribution by a smoothing kernel $W_h (h \in \mathbb{R}_>0)$ with compact support $\Omega(x,h) = \{ y \in \mathbb{R}^3; \|x - y\| \leq h \}$ and $\lim_{h \to 0} W_h = \delta$ pointwise. Note that $h$ denotes the support radius in this thesis, which can differ from the smoothing length. That leads to the so-called kernel approximation $\langle A \rangle$ of $A$ defined by

$$\langle A(x) \rangle = \int_{\Omega(x,h)} A(x') W_h(\|x - x'\|) dV(x').$$

In other words, the quantities $A(x)$ are smoothed over the compact area $\Omega(x,h)$ around position $x$. Gingold and Monaghan [49] proposed $W_h$ to be a Gaussian or a spline function. Nonetheless, a wide range of kernels has been used for different applications. In general, a smoothing kernel $W_h$ needs to fulfill specific properties:

1. Dirac limit
   The smoothing kernel $W_h$ converges to the delta distribution for $h \to 0$, i.e.,

   $$\lim_{h \to 0} W_h(\|x - x'\|) = \delta(\|x - x'\|).$$
2. **Normalization**
   The kernel function $W_h$ has to be normalized:
   \[
   \int_{\Omega(x,h)} W_h(\|x - x'\|) dV(x') = 1.
   \]

3. **Compact support**
   The smoothing kernel has the compact support $\Omega(x,h)$. As $W_h$ is assumed to be a continuous function, it follows that
   \[
   W_h(\|x - x'\|) = 0 \text{ if } \|x - x'\| \geq h.
   \]
   The compact set $\Omega(x,h)$ will be called the neighborhood of $x$.

4. **Positivity**
   The kernel function is positive on the inside of its compact support $\Omega(x,h)$, i.e.,
   \[
   W_h(\|x - x'\|) > 0 \text{ if } \|x - x'\| < h.
   \] (2.20)

Throughout this thesis, different kernel functions are used. Before describing them in detail, spatial derivatives of a fluid property approximated with SPH are discussed.

### 2.2.1 Spatial Derivatives with SPH

To animate a fluid using Equation 2.17, spatial derivatives of scalar and vector fields need to be computed. In Equation 2.19, $A$ is an abstract property and can be substituted with its gradient $\nabla_x A$:
\[
\langle \nabla_x A(x) \rangle = \int_{\Omega(x,h)} (\nabla_x A(x')) W_h(\|x - x'\|) dV(x').
\] (2.21)

While this formulation of the gradient approximation would work in general, it is not used as it requires the computation of $\nabla_x A(x')$ for all $x' \in \Omega(x,h)$. Therefore, an alternative formulation is used that is derived in the following. When applying the product rule, Equation 2.21 reads
\[
\langle \nabla_x A(x) \rangle = \int_{\Omega(x,h)} \nabla_x (A(x') W_h(\|x - x'\|)) dV(x')
\] (2.22)
\[
- \int_{\Omega(x,h)} A(x') \nabla_x W_h(\|x - x'\|) dV(x').
\]

The first term on the right-hand side can be transformed into a surface integral using a corollary of the divergence theorem: For an arbitrary control volume $CV$ and a quantity $A$, it holds
\[
\int_{CV} \nabla A dV = \int_{C(V\setminus CV)} A \cdot n dA.
\] (2.23)
To show that this equality holds the divergence theorem can be used. For an arbitrary but fixed vector \( \mathbf{c} \), it holds
\[
\mathbf{c} \cdot \nabla A = \nabla \cdot (\mathbf{c} A) - A \nabla \cdot \mathbf{c} = \nabla \cdot (\mathbf{c} A).
\]
Therefore, it can be stated that
\[
\mathbf{c} \cdot \int_{C_V} \nabla A \, dV = \int_{C_V} \nabla \cdot (\mathbf{c} A) \, dV = \mathbf{c} \cdot \int_{C_V \setminus C_V} A \cdot \mathbf{n} \, dA. \tag{2.24}
\]
This holds for any \( \mathbf{c} \) and, therefore, Equation 2.23 holds. With these insights, Equation 2.22 can be written as
\[
\langle \nabla_x A(x) \rangle = \int_{\Omega(x,h)} A(x') W_h(\|x - x'\|) \cdot \mathbf{n} \, dA(x')
\]
\[
- \int_{\Omega(x,h)} A(x') \nabla_x W_h(\|x - x'\|) \, dV(x'). \tag{2.25}
\]
As \( W_h(h) = 0 \), the first term of the right-hand side of Equation 2.25 vanishes. Defining \( r := \|x - x'\| \), the kernel gradient \( \nabla_x W_h(r) \) can be computed by
\[
\nabla_x W_h(r) = \nabla_x r \frac{dW_h(r)}{dr} = \frac{x - x'}{\|x - x'\|} \frac{dW_h(r)}{dr}. \tag{2.26}
\]
This implies that \( \nabla_x W_h(r) = -\nabla_x W_h(r) \) and, therefore, the equality
\[
\langle \nabla_x A(x) \rangle = \int_{\Omega(x,h)} A(x') \nabla_x W_h(\|x - x'\|) \, dV(x') \tag{2.27}
\]
holds. This formulation has the advantage that there is no need to evaluate the gradient of \( A(x') \) but only of \( W_h \) at location \( x \). The derivatives \( \langle \nabla_x^2 A \rangle \) and \( \langle \nabla_x \cdot A \rangle \) can be computed similarly.

Using the approximation
\[
\langle \nabla_x^2 A(x) \rangle = \int_{\Omega(x,h)} A(x') \nabla_x^2 W_h(\|x - x'\|) \, dV(x'), \tag{2.28}
\]
however, results in a poor estimation of the Laplacian [78]. To address this issue, Brookshaw [29] proposed an approximation of \( \nabla_x^2 A(x) \) via
\[
\langle \nabla_x^2 A(x) \rangle = 2 \int_{\Omega(x,h)} \frac{(A(x') - A(x))(x - x')}{\|x - x'\|^2} \nabla_x W_h(\|x - x'\|) \, dV(x'). \tag{2.29}
\]
This formulation has the advantage that it has no need to compute the second derivative of \( W_h \) and leads to a better estimation of the Laplacian [78].
Figure 2.2: Smoothing kernels and their derivatives are plotted for one particle pair with respect to their distance. The derivative \( \frac{d}{dr}W_{\text{poly}6} \) shows small values for close-by particles (i.e., small values for \( r \) of about 0.2 – 0.4), resulting in an underestimation of these when approximating the gradient of a fluid quantity. Therefore, the derivative \( \frac{d}{dr}W_{\text{spiky}} \) is typically used instead of \( \frac{d}{dr}W_{\text{poly}6} \). This effect is less strong when using the cubic spline kernel \( W_{\text{cbs}} \) and, hence, only one kernel is used.

2.2.2 Smoothing Kernel Functions

In general, a smoothing kernel function \( W_h \) should be similar to a Gaussian [92] but have a compact support and the other desired properties, as discussed above. Gingold and Monaghan [49] actually used a Gaussian kernel in their original work. Due to performance reasons, other kernels are typically employed for computer graphics applications. In this thesis, different sets of functions are used. In Chapters 3 and 4, the poly6 and spiky kernel [95] are applied. In Chapters 5 and 6, the cubic spline kernel [92] is employed. In Figure 2.2, all kernels and their derivatives are plotted.

The poly6 kernel is given by

\[
W_{h}^{\text{poly}6}(r) = \frac{315}{64\pi h^6} \begin{cases} 
(h - r^2)^3 & \text{if } r < h, \\
0 & \text{otherwise},
\end{cases}
\]  

(2.30)

and its derivative by

\[
\frac{d}{dr}W_{h}^{\text{poly}6}(r) = -\frac{945}{32\pi h^6} \begin{cases} 
(r(h - r^2)^2 & \text{if } r < h, \\
0 & \text{otherwise}.
\end{cases}
\]  

(2.31)
The gradient $\nabla_x W_{h}^{\text{poly}6}(r)$ can be computed with Equation 2.26. Nonetheless, using $W_{h}^{\text{poly}6}(r)$ to approximate the gradient of a field quantity $A$ has some drawbacks. As observable in Figure 2.2b, $\nabla_x W_{h}^{\text{poly}6}(r)$ has very small values if $r$ approaches zero. This means, close-by regions are underestimated in the approximation process. Therefore, Müller et al. [95] propose using the spiky kernel to approximate the gradient of a field quantity $A$ instead. It is defined by

$$W_{h}^{\text{spiky}}(r) = \frac{15}{\pi h^6} \left\{ \begin{array}{ll} (h - r)^3 & \text{if } r < h, \\ 0 & \text{otherwise,} \end{array} \right. \quad (2.32)$$

and its derivative reads

$$\frac{d}{dr} W_{h}^{\text{spiky}}(r) = -\frac{45}{\pi h^6} \left\{ \begin{array}{ll} (h - r)^2 & \text{if } r < h, \\ 0 & \text{otherwise.} \end{array} \right. \quad (2.33)$$

Monaghan [92] proposes using a unified kernel for both the approximation of the field quantity $A$ and its gradient $\nabla A$. A popular choice here is to use the cubic spline kernel. It is given by

$$W_{h}^{\text{cbs}}(r) = \frac{8}{\pi h^3} \left\{ \begin{array}{ll} 6 \left( \left( \frac{r}{h} \right)^3 - \left( \frac{r}{h} \right)^2 \right) + 1 & \text{if } r \leq \frac{h}{2}, \\ 2 \left( 1 - \frac{r}{h} \right)^3 & \text{if } \frac{h}{2} < r < h, \\ 0 & \text{otherwise.} \end{array} \right. \quad (2.34)$$

Its derivative reads

$$\frac{d}{dr} W_{h}^{\text{cbs}}(r) = -\frac{48}{\pi h^4} \left\{ \begin{array}{ll} 2 \left( \frac{r}{h} \right) - 3 \left( \frac{r}{h} \right)^2 & \text{if } r \leq \frac{h}{2}, \\ \left( 1 - \frac{r}{h} \right)^2 & \text{if } \frac{h}{2} < r < h, \\ 0 & \text{otherwise.} \end{array} \right. \quad (2.35)$$

### 2.3 Spatial Discretization

To animate a fluid the Navier-Stokes momentum equation (Equation 2.17) needs to be discretized in space and time. Before describing the temporal discretization in Section 2.4, the spatial discretization of the fluid domain is discussed in this section. To this end, the fluid domain is sampled with $n$ particles at discrete positions $x_i(t)$ with $i \in N = \{1, \ldots, n\}$ and $A(x_i(t))$ being evaluations at these locations $x_i(t)$. Again, $t$ is assumed to be constant and will be skipped for the rest of this section. To compute the fluid quantities with SPH, the kernel approximations $\langle A(x_i) \rangle$, described in the previous section, need to be discretized. Each particle $i \in N$ represents a certain amount of
Chapter 2. SPH-based Fluid Animation

Figure 2.3: Illustration of the spatial discretization of the fluid body in the neighborhood of particle \( i \) (highlighted as red dot). The blue radial gradient illustrates the values of \( W_h \) and the big gray circle describes the neighborhood \( \Omega(x, h) \). The dots inside the gray circle are the neighboring particles \( j \in N_i \). All particles \( k \) that do not contribute to the attribute computations (i.e., \( k \notin N_i \)) are grayed out.

The fluid and, therefore, a specific mass \( m(x_i) \) is linked to it. Together with the particles’ density \( \rho(x_i) \), the volume that particle \( i \) represents is given by \( V(x_i) = \frac{m(x_i)}{\rho(x_i)} \). Therefore, the discretization of the kernel approximation from Equation 2.19 reads

\[
\langle A(x_i) \rangle \approx \sum_{j \in N_i} \frac{m(x_j)}{\rho(x_j)} A(x_j) W_h(\|x_i - x_j\|),
\]

(2.36)

where \( N_i = \{j; j \in N, x_j \in \Omega(x_i, h)\} \) is the set of the discrete evaluation positions in the neighborhood of particle \( i \). For convenience, the following definitions will be used: \( A_i := \langle A(x_i) \rangle, \ m_i := m(x_i), \) and \( W_{ij} := W_h(\|x_i - x_j\|) \). An illustration of the discrete evaluation positions is given in Figure 2.3, where \( i \) is highlighted in red.

When computing the gradient \( \nabla A_i := \langle \nabla x A(x_i) \rangle \) of a fluid quantity, Equation 2.27 needs to be discretized as well:

\[
\nabla A_i \approx \sum_{j \in N_i} \frac{m_j}{\rho_j} A_j \nabla W_{ij}.
\]

(2.37)
The Laplacian of a fluid quantity $A$ (Equation 2.29) can be discretized in the same way. Brookshaw [29] calculates it as

$$\nabla^2 A_i \approx 2 \sum_{j \in N_i} \frac{m_j}{\rho_j} \frac{x_{ij}}{\|x_{ij}\|^2} \nabla W_{ij},$$

(2.38)

where $A_{ij} = A_i - A_j$ and $x_{ij} = x_i - x_j$.

### 2.3.1 Discretization of the Navier-Stokes Momentum Equation

The above equations can now be used for the spatial discretization of the right-hand side of Equation 2.17. As mentioned, with SPH, the fluid quantities are approximated at discrete locations $x_i$ in space. Therefore, the Navier-Stokes momentum equation is evaluated at these locations and can be written as

$$\frac{Dv_i}{Dt} = -\frac{1}{\rho_i} \nabla p_i + \frac{\mu}{\rho_i} \nabla^2 v_i + a_i^b. \quad (2.39)$$

In typical scenarios, the acceleration due to body forces contains only gravity, i.e., $a_i^b = g$. The acceleration $\frac{Dv_i}{Dt}$ of the fluid at location $x_i$ is computed using SPH. To this end, the two terms $a_i^p$ and $a_i^\mu$ are approximated. Details about this approximation will be discussed in the following.

In order to calculate $a_i^p$ and $a_i^\mu$, densities $\rho_i$ and pressure values $p_i$ are needed. The density $\rho_i$ is typically approximated using Equation 2.36 and reads

$$\rho_i = \sum_{j \in N_i} m_j W_{ij}. \quad (2.40)$$

There exist various ways to compute the pressure $p_i$. One intuitive way is to define pressure $p_i$ by density deviations from the fluid’s reference density $\rho_0$. This is typically achieved using an equation of state (EOS):

$$p_i = k \frac{\rho_0}{\gamma} \left( \left( \frac{\rho_i}{\rho_0} \right)^\gamma - 1 \right), \quad (2.41)$$

where $k$ is a constant controlling the stiffness and $\gamma$ defines the relation between density deviation and pressure. For example, if $\gamma = 1$ is chosen, there is a linear relation between density deviation and pressure as $p_i = k (\rho_i - \rho_0)$. For incompressible flows, the stiffness factor needs to be set to $k = c_s^2$, where $c_s$ is the speed of sound. In practice, $k$ is a user-defined stiffness constant that scales the pressure [69].
Approximation of the Pressure Term

The pressure term $a^p_i$ can be calculated using Equation 2.37 and reads

$$a^p_i = -\frac{1}{\rho_i} \nabla p_i = -\frac{1}{\rho_i} \sum_{j \in N_i} \frac{m_j}{\rho_j} p_j \nabla W_{ij}. \quad (2.42)$$

This formulation has some drawbacks when computing the pressure gradient $\nabla p_i$ to determine the acceleration due to pressure forces. It violates the momentum conservation principle. Therefore, modern SPH formulations make use of an antisymmetric formulation of $a^p_i$ to achieve momentum conservation [94]. To this end, Monaghan [91] proposes rewriting the pressure gradient via

$$\frac{1}{\rho_i} \nabla p_i = \nabla \left( \frac{p_i}{\rho_i} \right) + \frac{p_i}{\rho_i^2} \nabla \rho_i, \quad (2.43)$$

which includes the density into the pressure gradient approximation. The acceleration due to pressure can then be computed by

$$a^p_i = -\sum_{j \in N_i} m_j \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij}. \quad (2.44)$$

The result is an antisymmetric pressure gradient approximation that conserves linear and angular momentum [91].

Approximation of the Viscosity Term

Next, the calculation of the viscous term $a^\mu_i$ is described. To this end, the approximation of the Laplacian proposed by Brookshaw [29] is employed, which leads to

$$a^\mu_i = 2\mu \sum_{j \in N_i} m_j \frac{x_{ij}}{\rho_j \|x_{ij}\|^2 + 0.01h^2} \cdot \nabla W_{ij}. \quad (2.45)$$

The term $0.01h^2$ is added in the denominator to avoid instabilities. Otherwise, the denominator will approach zero if two particles are getting close [94].

2.4 Temporal Discretization

The evolution of the fluid also needs to be discussed to formulate a basic SPH solver. As mentioned, with SPH, the fluid body consist of a set of particles, and its quantities move along with these particles. Equation 2.39 defines how the particles’ velocities change over time, i.e., it defines an acceleration
\( \mathbf{a}_i(t) = \langle \frac{Dv}{Dt}(\mathbf{x}_i(t), t) \rangle \). Together with their initial positions \( \mathbf{x}_i(t_0) \) and velocity \( \mathbf{v}_i(t_0) \), an initial value problem is given by

\[
\frac{d}{dt} \mathbf{v}_i(t) = \mathbf{a}_i(t). \tag{2.46}
\]

This equation defines how the particle velocities change over time. A second differential equation is used to compute the resulting position changes, and a system of first-order ordinary differential equations (ODEs) is defined by

\[
\frac{d\mathbf{x}_i}{dt}(t) = \mathbf{v}_i(t),
\]
\[
\frac{d\mathbf{v}_i}{dt}(t) = \mathbf{a}_i(t). \tag{2.47}
\]

To solve the system of ODEs, they are discretized in time via numerical integration schemes. Given a fixed time step size \( \Delta t \), the new position \( \mathbf{x}_i(t + \Delta t) \) and velocity \( \mathbf{v}_i(t + \Delta t) \) of particle \( i \) can be computed.

There exist a variety of different time integration methods for this task. They can be divided into two classes: explicit and implicit integrators. While explicit integrators compute the new state (here \( \mathbf{x}_i(t + \Delta t) \)) based only on information of the previous one, implicit integrators make use of the new state as well. Implicit integrators are known to be stable for stiff problems and, therefore, large \( \Delta t \) are possible. However, they are complicated to implement and computationally expensive as they typically require solving a linear system of equations. Explicit integrators are very efficient and straight forward to implement but become unstable for larger \( \Delta t \) when solving stiff problems.

An explicit time integration scheme is given by the Euler method. Given the actual state \( \mathbf{x}_i(t) \) and \( \mathbf{v}_i(t) \) and a finite step size \( \Delta t \), the new state is computed via

\[
\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t),
\]
\[
\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \mathbf{a}_i(t). \tag{2.48}
\]

This method is very simple and efficient. Nonetheless, a variant of the explicit Euler method is used in most SPH solvers. It is called semi-implicit or symplectic Euler scheme \cite{78} and reads

\[
\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t),
\]
\[
\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \mathbf{a}_i(t). \tag{2.49}
\]

In this case, the velocity is first computed using an explicit scheme. Afterward, the new position is computed with the new velocities \( \mathbf{v}_i(t + \Delta t) \), i.e., the
position is updated using an implicit Euler scheme. This slightly increases the stability of the numerical integration scheme.

The Euler method can be seen as a first-order Taylor approximation of $x_i(t + \Delta t)$ and $v_i(t + \Delta t)$. Goswami and Batty [50] propose including the second-order term to compute the new particle positions. The full integration scheme then reads

$$x_i(t + \Delta t) = x_i(t) + \Delta t v_i(t + \Delta t) + \frac{\Delta t^2}{2} a_i(t),$$

(2.50)

$$v_i(t + \Delta t) = v_i(t) + \Delta t a_i(t).$$

This modification enhances the stability of the time integration step and, hence, allows for slightly larger time steps. However, this integration scheme is no longer symplectic and introduces numerical damping. In this thesis, the symplectic Euler scheme (Eqs. 2.49) is used, except for Chapter 4, where the modification to the symplectic Euler (Eqs. 2.50) is employed.

For all above mentioned time integration methods, the maximum possible time step size $\Delta t$ is restricted by the so-called Courant-Friedrichs-Lewy (CFL) condition that is given by

$$\Delta t \leq \lambda \frac{h_{\max}}{\max \{\|v_i\|\}},$$

(2.51)

with $\lambda \in [0,1]$ being the CFL-constant, and $\max \{\|v_i\|\}$ the maximum norm of the fluid’s velocity. The CFL condition is a necessary condition for the solver to stay stable. Nonetheless, sudden changes in acceleration $a_i$ may require to choose $\lambda$ very small. Therefore, such accelerations should also be considered when computing the time step [65], [92]. The possible time step can then be determined by

$$\Delta t = \min \left( \lambda_v \frac{h}{\max \{\|v_i\|\}}, \lambda_a \sqrt{\frac{h}{\max \{\|a_i\|\}}} \right),$$

(2.52)

where $\lambda_v$ is the CFL-constant restricting the velocity term and $\lambda_a$ the acceleration term.

### 2.5 SPH-based Fluid Solver

With all above discussed techniques, it is now possible to implement a basic SPH-based fluid solver. The first step is to sample the fluid body with particles. This can be achieved in several ways. The simplest way is to set the particles
in the fluid to form a structure similar to a Cartesian grid. In recent years, more sophisticated approaches has been presented. For example, Colagrossi et al. [34] presented an particle packing algorithm reducing velocity noise during the early simulation phase. Furthermore, a fixed mass \( m_i \) is set to each particle \( i \), typically \( m_i = \frac{m_F}{n} \).

After this initialization process, the fluid can be animated. An overview of the animation loop for an basic SPH solver is given in Algorithm 2.1. It consists of the following steps: First, the neighboring particles \( j \in N_i \) are calculated for each particle \( i \). Next, the particle densities and pressures are computed, e.g., by Eqs. 2.40 and 2.41. Afterward, the particles’ acceleration can be computed (Equation 2.39) and, finally, the new particle velocities and positions. This loop is repeated till the animation has reached the desired state.

While the concept of Algorithm 2.1 can be used to animate a fluid with SPH, it is not very efficient. Important building blocks are still missing to generate visually pleasing fluid animations with complex interactions. These will be discussed in the following. An efficient way to calculate the neighborhood \( N_i \) of particle \( i \) is crucial to be able to simulate a large number of particles. It will be discussed in Section 2.5.1. Moreover, to produce interesting effects, e.g. splashes, the fluid needs to interact with other objects. This will be discussed in Section 2.5.2. Subsequently, the modeling of surface tension is discussed in Section 2.5.3. Simulating incompressible flow with this basic solver (and using the EOS) would require a very large stiffness constant \( k \) and, therefore, very small time step sizes. Computational methods addressing this problem are discussed in Section 2.5.4.
Figure 2.4: Illustration of a axis-aligned grid. To accelerate the neighborhood search, particles are inserted into the grid using a spatial hash. Due to the spatial decomposition, only the cell containing the particle \( i \), and its neighboring cells have to be queried to determine \( N_i \). Usually, the grid cells are quadratic, with an edge length of \( h \).

2.5.1 Neighborhood Search

Computing the neighborhood \( N_i \) is a basic step in each SPH-based fluid solver. Typically, a particle \( i \) only interacts with a very small fraction of all particles sampled in the fluid body. The simplest, but not very efficient approach, is to consider every particle to be a possible neighboring particle and just neglecting them if \( \| x_i - x_j \| > h \). More efficient techniques employ spatial data structures allowing for local query of possible neighboring particles.

A straightforward neighborhood determination strategy is to use a regular axis-aligned grid with uniform cell sizes. First, the particles are inserted into grid cells. Typically, the cell size is chosen to be the support radius \( h \). To find the neighbors of particle \( i \), only the cell containing particle \( i \) and its neighboring cells have to be considered. An illustration of the search strategy is given in Figure 2.4. This technique has the advantage that it is generally very efficient in terms of computation time, as accessing a particle is possible in \( \mathcal{O}(1) \) and the construction of the grid is done in \( \mathcal{O}(n) \) (as mentioned, \( n \) is the number of particles). It has the disadvantage that the simulation domain needs to be specified by an axis-aligned bounding box and, more importantly, the memory consumption is high for large simulation domains, as many empty cells need to be stored. For this reason, more sophisticated data structures, such as adaptive grids or compact hashing, have been introduced. An analysis and comparison of neighborhood query strategies is given by Ihmsen et al. [64].
In this thesis, the regular grid is used, except for Chapter 5. The techniques presented in Chapter 5 are implemented in the open-source framework \textit{SPlisH-SPlasH} that employs a variant of compact hashing [64].

### 2.5.2 Collision Objects

The interaction with collision objects is essential to generate interesting fluid animations with SPH. The type of possible collision objects is very diverse. It ranges from thin deformable objects, such as cloth, to static, or dynamic rigid-bodies. In general, they can be distinguished by the representation and type of the boundary object. For example, a mesh-based representation has been used by Huber et al. [63] to model thin deformable objects, such as cloth. In this thesis, only rigid-body objects are considered. They are often represented using particles [9], [17], [19], [65]. Particle-based approaches have the advantage of a consistent representation of the fluid and boundary objects. However, they tend to produce inaccurate pressure forces and bumpy surfaces [24]. Koschier and Bender [77] proposed using an implicit boundary representation addressing these problems. A continuous extension of the density field is achieved with an implicit discretization based on signed distance fields. Recently, Bender et al. [24] presented an improvement to density maps called volume maps. They follow the concept of density maps but compute the volume contribution of the collision object instead. This approach is more consistent in the SPH sense as the boundary volume can be directly used to compute the derivatives of fluid quantities, such as the density gradient. Their improvement results in smoother pressure forces even with lower resolution, reducing the memory consumption.

Throughout this thesis, only rigid-bodies are considered as collision objects, and the particle-based coupling model of Akinci et al. [9] is used. It has the advantage that rigid boundary objects can be represented by a single layer of particles. Moreover, it allows for non-uniform particle sampling of the boundary object. To achieve this, an artificial mass $\Psi_{b_i}$ is assigned to each boundary particle $b_i$ individually. The idea is that the volume $V_{b_i}$ of a boundary particle is smaller for densely sampled areas and larger for sparsely sampled ones. It is defined by

$$V_{b_i} = \frac{1}{\sum_{k \in N_{b_i}^b} W_{b_i k}}, \quad (2.53)$$

where $N_{b_i}^b$ is the set of all neighboring boundary particles, i.e., $N_{b_i}$, but excluding fluid particles. The artificial mass $\Psi_{b_i}$ is then defined as $\Psi_{b_i} = \rho_0 V_{b_i}$, where $\rho_0$ is the fluids reference density. The computation of a fluid particles’ density
\( \rho_i \) is then given by
\[
\rho_i = \sum_{j \in N_i^f} m_j W_{ij} + \sum_{b_j \in N_i^b} \Psi_{b_j} W_{ib_f},
\]
(2.54)
with \( N_i^f \) being the set of all neighboring fluid particles. The acceleration \( a_{p_i-b_j} \)
due to pressure forces between a fluid particle \( i \) and its rigid neighbor \( b_j \) are computed via
\[
a_{p_i-b_j} = \Psi_{b_j} \frac{p_i}{\rho_i} \nabla W_{ib_f}.
\]
(2.55)
To this end, the pressure \( p_i \) of the fluid particle is mirrored onto the boundary particle, i.e., no pressure is computed for boundary particles. More details are given in the work of Akinci et al. [9].

### 2.5.3 Surface Tension

Another essential aspect to generate visually pleasing simulations is the modeling of surface tension. It has a strong influence on the visual appearance of the fluid. As only the surface is needed for rendering purposes, the fluid is often represented by its boundary. The surface of the fluid is highly influenced by surface tension and, therefore, should be included in the simulation [7], [18], [58]. There exist a variety of surface tension models for SPH-based fluid animation. Most of them are based on two different viewpoints:

- **Microscopic Model**
  These models are based on the physical principle of cohesive attraction, i.e., surface tension is seen as a particle-particle interaction force causing the particles to stick together.

- **Macroscopic Model**
  Surface tension tends to minimize the fluids surface area. To minimize the surface area, typically a force is modeled, counteracting surface curvature [7].

To include surface tension into the simulation, another source term \( a_{st_i} \) is added to the right-hand side of Equation 2.39.

In this thesis, a modification to the model of Becker and Teschner [18], as proposed by Huber et al. [1], is used to animate fluids with SPH. It is based on the microscopic viewpoint. Attractive forces are modeled as cohesion forces, i.e., cause the particles to accelerate toward each other:
\[
a_{st_i} = -\frac{\phi}{m_i} \sum_{j \in N_i} m_j (x_i - x_j) W_{ij}.
\]
(2.56)
One major drawback of this model is that particles tend to group together in clusters. Therefore, Huber et al. [1] propose cutting off the attraction at a certain particle distance $r^c$. The term $a_{si}^t$ then reads

$$a_{si}^t = -\frac{\varphi}{m_i} \sum_{j \in N_i} \begin{cases} m_j (x_i - x_j) W_{ij} & \text{if } \|x_i - x_j\| > r^c, \\ 0 & \text{else}. \end{cases}$$

(2.57)

A comparison between the model of Becker and Teschner and the proposed modification can be found in Section 3.4.4. More extensive details on surface tension are given in the work of Huber et al. [1].

### 2.5.4 Pressure Solver

Solving the pressure term of the incompressible Navier-Stokes momentum equation is paid special attention for several reasons. The two main reasons are that pressure accelerations are the dominating term (and, therefore, crucial for the stability of the solver) and that they are responsible for the volume preservation of the fluid. As mentioned in Section 2.1.1, the density of the fluid (and, therefore, of a fluid particle) is constant for incompressible flows. As a direct consequence, the overall fluid volume should stay constant over time. Inaccuracies in the computation of pressure term $a_p^t$ can cause oscillations of the fluid’s volume resulting in unrealistic behavior. Furthermore, such oscillations can cause instabilities in the simulation. Several authors examine this issue, either based on the EOS [18], [114] or pressure projection schemes [37], [41], [68].

When using the EOS (Equation 2.41) to compute the pressure, the stiffness factor has to be chosen as $k = c_s^2$ to obtain incompressible flow. For fresh water at about 20°C, this means $k = 1481^2 \frac{m^2}{s^2} \approx 2.2 \cdot 10^6 \frac{m^2}{s^2}$. The consequences are very large pressure forces and, accordingly, only small time steps are possible. To overcome this issue, Becker and Teschner [18] propose allowing for a small, user defined compression $\eta$ of the fluid. They restrict the maximum fluid velocity to a constant $\bar{v}_{\text{max}}$ and state that $c_s > \frac{\bar{v}_{\text{max}}}{\sqrt{\eta}}$. Therefore, $c_s$ can be replaced by the term $\frac{\bar{v}_{\text{max}}}{\sqrt{\eta}}$ and the EOS reads:

$$p_i = \frac{(\bar{v}_{\text{max}})^2 \rho_0}{\gamma} \left( \left( \frac{\rho_i}{\rho_0} \right)^\gamma - 1 \right).$$

(2.58)

For example, when allowing for a maximum fluid velocity of 50 m/s and a compression of 1% ($\eta = 0.01$), the stiffness constant $k$ is set to $k = 2.5 \cdot 10^5 \frac{m^2}{s^2}$. Allowing for smaller compression requires smaller time steps. Larger values of $\eta$ allow for larger time steps but result in less realistic simulations [78]. This
approach is commonly known as weakly compressible SPH (WCSPH) and is used in Chapter 3. Modern pressure solvers avoid this issue. They are all based on a splitting concept called projection method or Chorin’s method.

**Chorin’s Method**

Chorin [31] presented an operator splitting concept that allows for an individual computation of the pressure term \( a^P_i \). It is based on the Helmholtz-Hodge decomposition, which states that the vector field given by Equation 2.17 can be decomposed into a curl-free (thus a gradient field), and a divergence-free vector field. In the case of the incompressible Navier-Stokes momentum equation, the conservative vector field is exactly characterized by the pressure term \( a^P_i \). Chorin’s method consists of two steps:

1. Compute intermediate velocities \( v^*_i \) ignoring the pressure term \( a^P_i \):
   \[
   v^*_i = v_i(t) + \Delta t (a^U_i(t) + a^b_i(t)).
   \]  
   (2.59)

2. Determine the pressure using the intermediate solution and project the velocity field onto its divergence-free part by
   \[
   v_i(t + \Delta t) = v^*_i + \Delta t a^P_i.
   \]  
   (2.60)

This concept can be used to improve WCSPH [18]. Employing Chorin’s method, an intermediate velocity \( v^*_i \) is computed first, using non-pressure forces only. Afterward, the intermediate density \( \rho^*_i \) is computed based on a discretization of the continuity equation (Equation 2.6):

\[
\rho^*_i = \rho_i(t) - \Delta t \rho_i(t) \nabla \cdot v^*_i = \rho_i(t) + \Delta t \sum_{j \in N_i} m_j (v^*_i - v^*_j) \cdot \nabla W_{ij}.
\]  
(2.61)

This intermediate density is then used to compute the pressure. Finally, the new velocities are determined using the intermediate velocities \( v^*_i \) and the pressure term \( a^P_i \). An overview of this method is given in Algorithm 2.2. It is commonly known as non-iterative EOS with splitting. In Algorithm 2.1 accelerations due to non-pressure forces compete with accelerations resulting from pressure forces. Chorin’s method solves this issue by using the non-pressure related accelerations to compute \( a^P_i \) and may make the solver more stable [69]. This variant is used in Chapter 4, employing a version of the EOS as presented by Desbrun and Gascuel [39].

To enforce incompressibility, modern pressure solvers typically use a pressure Poisson equation (PPE) to calculate the pressure. To formulate a PPE with SPH two main concepts can be followed, either using the divergence-free condition \( \nabla \cdot v_i = 0 \) or the constant density condition \( \frac{D \rho_i}{D T} = 0 \). While they are conceptually equivalent conditions, they lead to different formulations of the PPE. In the following, they are described in more detail.
Algorithm 2.2 EOS-based Fluid Solver with Splitting

while animating do
    for all particle $i$ do
        find neighbors $j$
    for all particle $i$ do
        compute density $\rho_i$
        $v_i^* = v_i(t) + \Delta t(a_i^L(t) + a_i^b(t))$
    for all particle $i$ do
        compute $\rho_i^* = \rho_i + \Delta t \sum_{j \in N_i} m_j (v_i^* - v_j^*) \cdot \nabla W_{ij}$
        compute $p_i$ using $\rho_i^*$
    for all particle $i$ do
        compute $a_i^p = - \frac{1}{\rho_i^*} \nabla p_i$
    for all particle $i$ do
        compute new velocity $v_i(t + \Delta t) = v_i^* + \Delta t a_i^p$
        compute new position $x_i(t + \Delta t)$

Constant Density Condition  This variant assumes that the density does not change over time, i.e., $\frac{D\rho}{Dt} = 0$. To derive the PPE, $\rho_i(t + \Delta t)$ is set to be the reference density, i.e., $\rho_i(t + \Delta t) = \rho_0$, as $\rho_0$ is the desired fluid density for incompressible flow. Using implicit time integration and Equation 2.60, the continuity equation (Equation 2.6) can be written as

$$\frac{D\rho_i}{Dt}(t + \Delta t) = -\rho_i(t + \Delta t) \nabla \cdot v_i(t + \Delta t)$$

$$= -\rho_0 \nabla \cdot (v_i^* - \frac{\Delta t}{\rho_0} \nabla p_i(t + \Delta t)) \tag{2.62}$$

$$= -\rho_0 \nabla \cdot v_i^* + \Delta t \nabla^2 p_i(t + \Delta t).$$

This can be further simplified with Equation 2.61, but using the implicit Euler scheme ($\rho_i^* = \rho_i(t) - \Delta t p_i(t + \Delta t) \nabla \cdot v_i^*$):

$$\frac{D\rho_i}{Dt}(t + \Delta t) = -\frac{\rho_i(t) - \rho_i^*}{\Delta t} + \Delta t \nabla^2 p_i(t + \Delta t). \tag{2.63}$$

The term $\frac{D\rho}{Dt}(t + \Delta t)$ is reformulated using the implicit Euler step, i.e.,

$$\frac{D\rho_i}{Dt}(t + \Delta t) = \frac{\rho_i(t + \Delta t) - \rho_i(t)}{\Delta t}. \tag{2.64}$$

Inserting this into Equation 2.63 results in the following PPE:

$$\nabla^2 p_i(t + \Delta t) = \frac{\rho_0 - \rho_i^*}{\Delta t^2}. \tag{2.65}$$
Ihmsen et al. [68] use this formulation of the PPE to derive a pressure solver. Typically it is solved iteratively, e.g., using a relaxed Jacobi scheme. This variant is commonly known as implicit incompressible SPH (IISPH) [68]. It is used in this thesis in Chapter 3.

**Divergence-Free Condition** In this variant, the derivation of the PPE aims that the new velocity is divergence-free, i.e., \( \nabla \cdot \mathbf{v}_i(t + \Delta t) = 0 \). This assumption and Equation 2.60 result in

\[
0 = \nabla \cdot \mathbf{v}_i(t + \Delta t) = \nabla \cdot \mathbf{v}_i^* + \nabla \cdot \Delta \mathbf{a}_i^p.
\]  

(2.66)

Inserting the pressure term \( \Delta \mathbf{a}_i^p \) results in the following formulation of the PPE:

\[
\nabla^2 p_i = \frac{\rho_i(t)}{\Delta t} \nabla \cdot \mathbf{v}_i^*.
\]  

(2.67)

This formulation is not widely used as it suffers from severe volume drift over time [99]. Nonetheless, employing solely Equation 2.65 will, in practice, not result in a divergence-free velocity field as Bender and Koschier [21] show. Therefore, they propose combining both formulations. After computing the advection velocity and density, the density error is corrected using a pressure solver based on Equation 2.65. Next, the particle positions are updated by the velocities that result from the pressure term and, subsequently, the new densities are computed. Finally, the velocities are adjusted to fulfill the divergence-free condition (Equation 2.67). This approach is commonly known as divergence-free SPH (DFSPH) [21] and is used in Chapters 5 and 6.
In the process of developing computational models for SPH-based fluid animation, several challenges arise. Implementations and simulation models have to be evaluated, to ensure the correctness and accuracy of calculations, and the suitability to the given problem. A developer typically needs to overcome several problems: the combination of different interchangeable models, as well as the variety of their user-definable parameters, lead to different results, coding errors cause the simulation to crash or to behave incorrectly, and application-specific requirements have to be met.

There are several reasons why conventional debugging approaches fail for SPH-based fluid animation: Due to a large number of particles with multiple attributes, there are too many values that have to be displayed or processed in a meaningful way to obtain useful insights. Additionally, many of the particle attributes depend on the surrounding neighbors and may change over time. Furthermore, it is difficult to extract important characteristics of the attributes that obey spatial or temporal locality. Aside from that, there is usually no interactivity in the process, which means the output cannot be rearranged or altered in any way.

These challenges are addressed in this chapter by investigating Research Question 1. To this end, an approach to visual debugging of SPH-based fluid simulations is presented. The visual debugging approach assists the development process of computational methods for SPH-based fluid animation. The idea is to systematically visualize properties of the SPH simulations to detect errors and identify critical aspects. The presented technique differs from conventional debugging of software since no components of the software
Figure 3.1: A screenshot of the visual debugging environment for SPH simulations (left) that combines methods from information visualization and scientific visualization to provide a comprehensive overview of the simulation attributes. Non-spatial views like the scatter plot and the parallel coordinates plot on the left are combined with 3D views to analyze internal attributes of the simulation interactively via brushing and linking. Properties of surface tension models can be evaluated to improve the simulation, which directly impacts the appearance of surfaces shown in a photorealistic offline rendering to the right.

are visualized but rather properties of SPH simulations. The visual debugging environment combines a 3D visualization of the particles with multiple 2D views showing the multidimensional attributes of the simulation, to reveal correlations and important patterns. Interactive explorations of simulation sequences of arbitrary length and consisting of up to several millions of particles are supported. A screenshot with a typical configuration of the environment is shown in Figure 3.1. By linking individual views, information about the simulations can be revealed that goes beyond the possibilities of conventional debugging and simple particle visualizations.

The environment supports spatio-temporal particle data with multiple associated attributes and enables visual debugging. The components can be summarized in the following way:

- A 3D particle view using ray casting is provided. It allows the visualization of scenarios containing up to millions of particles at interactive frame rates.
- Scatter plots and parallel coordinates plots (PCPs) are used, to simultaneously display multidimensional SPH particle attributes.
- Brushing and linking is supported to connect individual views and to allow interactive exploration of the dataset.
- As typical SPH simulations consist of a large number of particles, stochastic sampling and alpha blending is used in the non-spatial data views to reduce visual clutter.
3.1 Related Work

Crossno et al. [35] highlighted the importance of visualization as a tool for debugging simulations. They used simple representations, such as boxes, spheres, and lines, to reveal underlying, raw data and tested their approach by visual debugging finite element implementations [36].

The scientific visualization of raw particle data as spheres (whose radius and color can be used to provide additional information) has a long tradition. The survey paper by Lipşa et al. [83] provides an overview of visualization methods used in the physical sciences. They identified GPU-based ray casting [56], as the fastest method to render large numbers of spheres. Reina et al. [106] showed that GPU-based ray casting is also applicable to visualize large numbers of complex, composed glyphs. Grottel et al. [55] optimized this approach for dynamic data by improving rendering speed and uploading strategies. Using their approach, it is possible to ray cast data sets of multiple millions of particles interactively on a single GPU, even with dynamic data updates. Their rendering methods are implemented in the open-source framework MegaMol [54], which is used as a basis for the visual debugging environment.

Price [101] presented SPLASH, an interactive visualization tool specifically designed for visual analysis of SPH simulation data. SPLASH uses volume rendering to show particle densities. Additionally, vector glyphs can be overlayed to show additional properties like the velocity, or the magnetic field. Fraedrich et al. [44] used voxelized SPH particles on a perspective grid, and visualize the resulting grid, using raymarching. This method enables the extraction of direct volume renderings as well as isosurfaces from the data. They extended their approach using a level-of-detail representation of particle trajectories that show the motion in the data [43]. Both SPLASH and the work of Fraedrich et al. only use spatial visualization, which limits the number of additional attributes that can be conveyed to the user without introducing too much clutter.

Linsen et al. [82] presented an isosurface extraction method intended for the visual analysis of SPH data. Instead of resampling the particles to a grid, their extraction method uses the radial basis functions representing the particles. To enhance the analysis, they extracted clusters and showed their features in linked views using parallel and radial coordinates. Their method is included in SmoothViz [89], which is an interactive framework for the analysis of astrophysical SPH simulations. Nonetheless, the focus of SmoothViz is on the visual exploration of the simulation results, not on the debugging of the underlying simulation. Reichl et al. [105] also use voxelization techniques to render a high-quality surface of particle-based fluid simulations. Their method reaches interactive frame rates for up to $500M$ particles. However, it requires a preprocessing step, which makes the interactive exploration unfeasible. Krone et al. [79] presented a visual debugging tool for SPH simulations of a phase
inversion process, where the area of the isosurface separating the two phases has to adhere to a certain power law. Their application uses linked views to show the interactively extracted isosurface as well as a plot of the surface area over time. The presented approach follows the concept of this approach but offers more flexibility and, therefore, applies to a much wider range of visual debugging tasks. Moreover, the mentioned techniques mainly focus on surfaces of the fluid body simulated with SPH, ignoring properties at individual data points inside the fluid, which is crucial for the debugging process.

Similar to SmoothViz [82], [89], additional plots are used to show the multidimensional properties of the SPH particles. Multiple coordinated views are a widely used concept for the visual analysis of simulations or other scientific data. Most commonly, 3D visualizations of the data are shown along with 2D scatter plots or PCPs. Selections made in one view are typically synchronized to the others, following the principle of brushing and linking. To fully support the user during data exploration, the presented debugging environment also makes use of this established interaction method. Piringer et al. [100] proposed using multiple linked scatter plots in 2D and 3D. Rübel et al. [107] presented a visualization application that supports particle and volume rendering and uses PCPs to show high-dimensional attributes of the simulation data. The presented visual debugging environment follows similar design principles. However, the focus is on the investigation of the underlying simulation (which created the data) rather than interactive exploration. Since it is intended to verify that the simulation results are valid, the application scenario of the visual debugging environment is located between the simulation that produces the data and the classical data analysis tools discussed above.

### 3.2 Visualization Requirements

The investigated data consists of 3D particle locations with corresponding influence radii. Additionally, an arbitrary number of attributes from the simulation is associated with each particle. For particle-based simulation of fluids, typical attributes are density, pressure, velocity, different types of forces, and the number of neighboring particles. The attributes, as well as physical models, influence each other and incorporate information of nearby particles. In a simulation, typically, a combination of models for pressure, viscosity, surface tension, spray and foam generation, and boundary handling is used. Hence, there does not exist one single SPH-fluid solver, but rather a large collection of interchangeable models.

Debugging such simulation systems introduces several challenges. During development, errors can appear in different parts of the program and may cause the simulation to behave incorrectly or crash at some point. At the same
time, not every error leads to instabilities or can be identified by inspecting calculated attributes. Finding such errors can be tedious, and using only a spatial analysis of the simulation runs is often not sufficient. Due to the large number of particles, printing out numeric values will often not suffice. For example, incorrect pressure or surface tension force values at specific locations may be caused by wrong pressure estimation, which itself can be caused by erroneous density computation. Even assuming an error-free implementation, the debugging process usually does not stop. Further analysis is needed to understand the behavior of the simulation, configure initial setups, and select parameters. Especially for the interplay of the different interchangeable models, it is also desirable to compare different implementations during the development of simulation codes.

To address these challenges, several visualization requirements have to be met. Like common debugging techniques, it should be easy to include visual debugging into the development process of simulations. Furthermore, it has to be tailored to the properties of the data and the arising challenges. To provide visual assistance in error detection and problem identification, the most important requirements are:

- **R1** Quantitative representation of the raw multi-dimensional simulation data that is difficult to grasp by analyzing numerical values or visualizing with simple color-coding.
- **R2** Simultaneous representations of spatial data and its multi-dimensional attributes.
- **R3** Representation that allows for the comparison of simulations models.
- **R4** Attribute representations that indicate incorrect implementation or inaccuracies in calculations.
- **R5** Good scalability of the representations for large amounts of particles.
- **R6** Simple and smooth usage during debugging, in particular, by providing interactive rendering and feedback.

### 3.3 Design of the Visual Debugging Environment

To satisfy the above visualization requirements (R1–R6), a visualization environment is presented that allows the representation of large multidimensional attributes using linked views. 3D particle views with adaptable non-spatial 2D views are combined. Simulation sequences are stored in the main memory or, depending on the size of the data set, can also be streamed for examining animations. To allow the user to analyze desired properties visually (R2), an
arbitrary number of spatial and non-spatial views are supported. Moreover, it is possible to visualize multiple data sets at the same time for a side-by-side comparison (R3). In the following, the main components of the views and interaction mechanisms are highlighted, and implementation details are given.

### 3.3.1 Spatial Data View

To obtain a view on the spatial representation of a simulation, 3D rendering of the particles is achieved using GPU-based ray casting of spheres. Spatial navigation through the data is possible by common interaction methods such as translation, rotation, and zooming of the camera. The properties of particles are visualized by color mapping. A linear transfer function can be used to map the value of a data attribute to a color and a freely adjustable clip plane allows the user to explore occluded regions of the fluid particles. As demonstrated in the results, 3D rendering and basic interaction meet the demand of interactivity for visual debugging (R6). In Figure 3.2 right, a 3D particle view with a clip plane is shown for an SPH simulation containing $27k$ particles.

### 3.3.2 Non-Spatial Data Views

The environment complements the spatial view with non-spatial views to enable a different perspective on the data and support the simultaneous display of multiple attributes (R2). To this end, scatter plots and PCPs are employed. Both views support panning and zooming.
3.3. design of the visual debugging environment

Figure 3.3: Visual analysis of a fluid pillar (left) simulated using implicit incompressible SPH (II SPH) and approximately 150$k$ particles. In both of the scatter plots, the pressure is mapped to density. Using all samples (left scatter plot), the distribution splits in two principal directions. With alpha blending based on the density of points (right scatter plot), it is possible to identify that the main portion of samples is included in the left part.

Scatter Plots  As pointed out in Section 3.2, several properties in SPH simulations depend on each other. Thus, investigating pairwise relationships between attributes via scatter plots can be beneficial to easily determine correlations, detect outliers, and identify clusters in attributes. The user can specify the attribute that is mapped to each axis and change the mapping interactively. Setting a user-defined range on the axes facilitates inspecting narrow regions of attributes. As SPH simulations typically consist of a large number of particles, alpha blending is used to improve the identification of high-density areas in the scatter plot (see Figure 3.3). Figure 3.2 shows an example of a combined spatial view together with the corresponding scatter plot of the relation neighbor count versus surface tension force.

Parallel Coordinates Plots  PCPs have been proven to be very useful for the analysis of multidimensional data [59], [71]. This visualization technique is included to identify patterns and trends between multiple interchangeable attributes. Therefore, important relations of quantities influenced by simulation models can be investigated. Figure 3.4 shows a PCP linked to a 3D view. Similar to the scatter plot, the user can also apply alpha blending to reduce clutter by highlighting dense regions.

In both of the provided non-spatial data views, overplotting becomes an issue when dealing with a high number of particles. Structures and trends will no longer be visible, which can lead to misinterpretations during analysis. This issue is only partly addressed by the aforementioned blending. To overcome
Figure 3.4: A split view showing a PCP and the corresponding 3D view of a simulated drop (27k particles). Particles selected in the 3D view are also highlighted in the linked PCP (green). In the PCP, four particle attributes are mapped to the axes and exhibit a linear correlation between the magnitudes of surface tension forces and advection forces. To reduce visual clutter, the lines are alpha-blended and stochastic sampling is used to reduce the number of data points (i.e., lines) to 25%.

this issue, stochastic sampling [40] is additionally employed, which displays only a percentage of the data (R5). The sampling can be enabled optionally, as it potentially leads to information loss (i.e., details or outliers can get lost).

Brushing and linking is used across all views. In the scatter plot, multiple selections with rectangular brushes as conjunctions are possible for localization of features that are then also highlighted in the other views (see Figure 3.2). One-dimensional brushing in the PCP is possible by selecting samples on a single axis.

3.3.3 Implementation

The visual debugging environment is implemented as a plug-in for the open-source framework MegaMol [54]. Due to its modular design and the strong focus on interactive rendering of very large numbers of particles, MegaMol provides a good foundation for further extensions. Moreover, it already offers optimized data structures for fast particle rendering and built-in mechanisms for data loading, including streaming of large data from disk. The spatial sphere visualization was extended to colorize the SPH particles based on different attributes and to support brushing and linking. Figure 3.5 shows a typical setup of our visual debugging environment. The 3D view mapping color based on density is shown in the lower left. In the lower right, a PCP with alpha blending and stochastic sampling to display 1% of the data is depicted. The PCP is linked to the particle selection performed in the 3D view. In the upper right, two scatter plots with different attribute relations are shown, where stochastic sampling is set to 75%.
3.3. Design of the Visual Debugging Environment

Figure 3.5: A screenshot of the visual debugging environment for a typical fluid simulation: The 3D particle view is linked to two scatter plots and a PCP. In the spatial data view, a linear color map is used to illustrate density distribution inside the fluid. Using the parameter interface (upper left), the user can set colors, mappings of attributes to coordinate axes in the non-spatial data views, and control stochastic sampling (here: 1% in the PCP). The selected particles in the 3D view are highlighted in the non-spatial data views.

Since interactivity is crucial for usability (R6), the performance for SPH simulations of different sizes was measured (see Table 3.1). The spatial view easily maintains interactive frame rates for all tested data sets. Even for a data set of 50M particles (as shown in Figure 3.6) an interactive feedback rate of 3.0 fps was observed in the 3D view. The scatter plot also remains interactive even on the test data set with 5M particles. Only the PCP exhibits significantly lower frame rates. However, the usability of the PCP for very large data is also limited due to overplotting. The stochastic sampling mentioned in Section 3.3.2 not only reduces clutter but also provides interactive frame rates for larger data sets. Normally multiple views are used for simulation debugging, as shown in Figure 3.5. As expected, the frame rate is bound by the slowest view in this case. That is, the requirement of interactivity (R6) of individual and combined views is satisfied for up to several millions of data points, which is the typical data size used for debugging.
3.4 Results

In this section, case studies are conducted to show the versatility of the presented visual debugging environment. To demonstrate its benefits, four applications of visual debugging are presented to gain insights into problems typically observed in SPH-based simulations. It is demonstrated how implementation errors can be detected and how a detailed examination of multidimensional particle attributes helps the model development process. Scenarios with different models for pressure, surface tension, and boundaries are considered. If not stated otherwise, the following parameter combinations are used for all of the presented simulations: The reference density is set to $\rho_0 = 1000 \text{ kg/m}^3$ and the smoothing length to $h = 0.1 \text{ m}$. Each particle $i$ has a mass $m_i = 0.125 \text{ kg}$. The viscosity parameter to $\mu = 0.01$ using the viscosity model as described by Equation 2.45. A fixed time step size of $\Delta t = 0.002 \text{ s}$ was used in all simulation runs.

3.4.1 Debugging Implementation Errors

In this case study, it is shown how to detect an error in the collision handling with rigid bodies using a non-spatial view. As mentioned in Section 2.5.2, the mass of a boundary particle is initialized with the local number density. These boundary particles do not have a dedicated pressure. Though when modeling friction as adhesive forces and other attractive forces, e.g., caused by surface tension, the boundary particles need to be included in the calculations. In this case, the density of a boundary particle is needed. Usually, they are initialized with the reference density $\rho_0$, and the density is computed in a standard SPH manner including boundary particles only and using artificial mass. Not performing the density computation could cause errors near boundaries. This

Table 3.1: Performance tests of the visual debugging environment for different data set sizes using a single spatial view (3D), a scatter plot (SP), a PCP, and a PCP showing only 5% of the data set attributes (stochastically sampled). All measurements were taken on an Intel Xeon E5 CPU ($6 \times 3.5 \text{ GHz}$) with 64 GB RAM, and an NVidia GeForce GTX 960. The least retained frame rate in frames per second (fps) is presented whilst interacting with the application. Note that the maximum possible frame rate is bound by the display refresh rate (which was 60 fps).

<table>
<thead>
<tr>
<th># particles</th>
<th>3D</th>
<th>SP</th>
<th>PCP</th>
<th>PCP 5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>27$k$</td>
<td>59.86</td>
<td>59.70</td>
<td>45.33</td>
<td>59.95</td>
</tr>
<tr>
<td>100$k$</td>
<td>59.91</td>
<td>59.31</td>
<td>8.54</td>
<td>59.86</td>
</tr>
<tr>
<td>500$k$</td>
<td>52.32</td>
<td>57.98</td>
<td>3.81</td>
<td>44.33</td>
</tr>
<tr>
<td>5$M$</td>
<td>41.28</td>
<td>8.98</td>
<td>0.31</td>
<td>4.91</td>
</tr>
</tbody>
</table>
3.4. results

Figure 3.6: Spatial view of a large-scale SPH simulation (50M particles). The 3D rendering at a frame rate of approximately 3fps still allows basic user interaction. The applied rendering uses normal correction to achieve a surface-like appearance at larger distances [55]. The inset shows a zoomed-in view that uses silhouettes and screen-space ambient occlusion to improve depth perception. All rendering methods were already provided by MegaMol [54].

error seems to have no big visual influence and may not be detected by visual inspection (using only a spatial view). However, it may lead to calculation errors and cause instabilities.

The upper row of Figure 3.7 shows a PCP with particle type, density, pressure, and surface tension forces, linked to the spatial view. It can be seen that all boundary particles have a fixed density $\rho_b = 1000 \, \text{kg/m}^3$, which is a direct indicator for this particular error in the implementation. Using only the spatial visualization of the particles shown in the right column of Figure 3.7 is not sufficient in this case. It may also be possible to find the error with particle coloring based on the density attribute, but this would include time-consuming calibration since the density variations on the boundary are in a very small range. Using the PCP and the spatial view combined with brushing and linking, it is possible to directly identify the issue, thus implementing requirement R4.

3.4.2 Identification of Surface Particles

Boundary handling at free surfaces and solid obstacles is a challenging topic in fluid simulation using SPH. One important aspect is the reconstruction of surfaces to achieve visually pleasing renderings. To this end, surface particles need to be identified [128], [130]. Using the presented environment, it is possible to identify the particles at free surfaces and fluid-solid interfaces based on attribute combinations. Figure 3.1 provides an example of identifying the surface of the fluid body using selections in the scatter plot. For this, a combination of surface tension and pressure forces are used. The identification
Figure 3.7: Visual debugging of boundary handling in a dam break simulation of 280\,k particles. The upper row shows a 3D view of the particles and a PCP of their attributes for a simulation with an implementation error. Using only the 3D view, the influence of the error is not visible in the corrected simulation (lower row). The PCP reveals that the density is fixed and not computed correctly at the selected boundary particles (p_type = 0) highlighted in green.

is performed in two steps: First, all particles with large surface tension forces are selected and, additionally, all particles with small pressure forces. With the first selection, all particles at the surface of the main fluid body are detected. The second selection covers isolated particles and splashes.

This example reveals the usefulness of brushing and linking using the scatter plot in combination with the spatial view to support a visual model exploration (R2). The combination helps the user verify an assumed model behavior and assists in developing new models. In this particular case, the visualization can be used to confirm the assumption that surface tension forces are only high near boundaries. It is possible to extract fluid particles at the interface with the multiple selections. By isolating them, other properties of surface and isolated particles can then further be investigated. The gained insights may also be used to improve surface extraction.

3.4.3 Investigation of the Density-Pressure Relationship

As mentioned, pressure computation is a crucial part when animating fluids with SPH. It has a major influence on the stability of the simulation. The pressure acting on a particle is mainly influenced by its density. Different pressure models are investigated by visualizing the relation between density
3.4. results

Figure 3.8: Comparison of pressure models for the simulation of a fluid pillar using approximately 150k particles in an equilibrium state. For each model, a scatter plot showing density and pressure is linked to the 3D view of the particles, where density is mapped to particle colors. From left to right: WCSPH using $\gamma = 1$, WCSPH using $\gamma = 7$, and IISPH. The particle colors illustrate the density distribution inside the fluid, and with the scatter plots, the correlation between density and pressure can be investigated for each pressure model. All three models were loaded in the visual debugging environment to examine the different models simultaneously.

and pressure in scatter plots for the simulation of a fluid pillar. A container of 8 m height, and 1.5 m width is filled with a liquid, and the simulation runs until an equilibrium state is reached. The pressure computation using IISPH is compared with WCSPH using $\gamma = 1$, and WCSPH using $\gamma = 7$. In case of IISPH, a maximum average density deviation of 0.1%, and a maximum compression of 3% is allowed. Using EOS-based models, the stiffness constant is set to $k = 10^4\, \text{m}^2\,\text{s}^{-2}$, which corresponds to a compressibility of 1% at $100\,\text{m}\,\text{s}^{-1}$. A side-by-side comparison of linked pairs of particle views and scatter plots are depicted in Figure 3.8.

The relationships shown in the scatter plots reveal the differences between these models. When calculating pressure with WCSPH using $\gamma = 1$, an exact linear correlation between density and pressure can be observed. When simulating the same scenario using $\gamma = 7$, the correlation becomes non-linear (Figure 3.8, center). With IISPH, this relation is not easily predictable anymore. Without these visualization techniques, the user may not be able to observe the properties of this model. In contrast to the EOS-based model, there is no exact linear or non-linear correlation. More likely, a strong linear correlation evolves if density gets larger than the reference density $\rho_0$. The reason for this fact is that IISPH employs a PPE to compute the pressure from density deviations, i.e., a linear system of equations is solved.

The visualization reveals another interesting fact: At a positive density deviation of approximately 1%, the density pressure relation splits up, following two main directions. Using the proposed alpha blending technique, it is possible to confirm that the lower direction is formed by a few outliers, as shown
in the right plot of Figure 3.3. Nonetheless, this behavior is not obvious and may raise some questions: for example, which particles follow which direction and why? The first question can now be investigated using brushing and linking of the scatter plot and the spatial view: Particles that follow the lower direction are all located at the fluid-solid interface. Hence, it can be assumed that this fact is related to the boundary model or, more likely, the combination of pressure and boundary model.

The visual debugging environment opens up new perspectives for the investigation of this particular research question. It enables a visual impression of the fluid properties and a deeper knowledge of specific models and their behavior. This task would be tedious to achieve with conventional debugging methods (R1). In this case, it is crucial to not only use the scatter plot itself, but also additional included features. For example, the scatter plot of the IISPH density pressure relation could provoke the false assumption that the upper line is formed only by a few particles. Hence, the density plot depicted in Figure 3.3 is also needed to fully understand this relation.

### 3.4.4 Model Comparison

The comparison of different SPH models is challenging or even impossible if only spatial views are used. In this case study, the capabilities of the visual debugging environment are investigated for the comparison of models in two scenarios. First, two models for solving the pressure term of the Navier-Stokes equations are compared and, secondly, properties of surface tension models.

The first model comparison is conducted on a breaking dam scenario with 200k particles, using WCSPH (with \( \gamma = 1 \), and IISPH. IISPH and WCSPH are based on distinct source terms. Hence, a comparison based only on spatial visualization is difficult. Figure 3.9 depicts the simulation at time \( t = 0.5 \) s, where IISPH is shown in the upper and WCSPH in the lower row. The right column shows a 3D view of the simulation and the middle and left column scatter plots of different attribute combinations. Regarding the mapping of density to pressure values in the scatter plots, it is possible to reproduce the insights gained in the third case study (Section 3.4.3). But a counterintuitive fact is additionally revealed: Although the pressure is much higher with IISPH, most of the resulting pressure forces are smaller by more than a factor of ten compared to WCSPH, even though both models use Equation 2.44 to derive the resulting forces. This may be one explanation for the fact that larger time steps are possible using IISPH compared to WCSPH. By the visualization of the data using a scatter plot, this can directly be observed, and the time-consuming numeric investigation is not necessary.

Lastly, a comparison of two microscopic surface tension models is conducted. Namely, the model of Becker and Teschner [18] and the modification proposed
3.4. results

Figure 3.9: Comparing a frame of a dam break simulation consisting of 200k particles with IISPH (top) and WCSPH using $\gamma = 1$ (bottom). In the left column, differences between the relation of density and pressure are shown. As expected, exact linear correlation can be identified for WCSPH. For IISPH, the correlation differs, and resulting pressure forces are smaller. The visual differences can be seen in the particle renderings on the right.

The model of Becker and Teschner is known as inter-particle interaction force (IIF) model. As described in Section 2.5.3, the modification of the IIF model neglects attractive forces for close particles by clamping cohesion forces to zero if particles are closer than a certain distance. As initial state, 27k particles are arranged in a cube. Without applying gravity, the surface tension forces should lead to the formation of a spherical drop. In this scenario, a smoothing length of $h = 0.125$ m is used and the time step size is set to $\Delta t = 0.001$ s. Figure 3.10 depicts the equilibrium state at the end of the simulation. The upper row shows the original IIF model and the lower row the modification. As shown in the upper spatial view, deformations occur at the surface using the original IIF model. Huber et al. [1] assumed that the reason for these deformations is the competition between the surface tension and pressure forces. The scatter plot depicting this relation (middle column of Figure 3.10) points out the differences between the two models. Regarding the modification of the IIF model in the lower row, two clusters can be identified and interpreted: The surface tension force is acting on isolated particles at the surface where the pressure force is quite small and vice versa.

by Huber et al. [1].
Additional internal friction is introduced when using a microscopic model. For this reason, the surface tension forces are considered to be a part of the advection forces. In the scatter plot shown in Figure 3.10 illustrating the relation between surface tension and advection forces (left column), it is possible to discover an interesting fact: With the modification, a strong positive correlation between surface tension and advection force is achieved. With the original formulation, this correlation is less pronounced. This indicates that the particle velocities are higher and less uniform and, hence, cause larger viscosity forces. It is possible to further investigate this fact with the features of our spatial view by using a clipping plane and particle coloring by velocity.

The case studies serve as an example of how the applied tools help identify similarities and differences of different models. By adding scatter plots to the visualization, deeper insights into the simulation process can be gained. With the possibility of comparative evaluation (R3) by combining the different views using brushing and linking, it is possible to identify spatio-temporal and local effects during the simulation.
3.5 Discussion and Future Work

The presented visual debugging environment helps identify implementation errors in the simulation and supports the development process of new simulation models. When deriving physical models through the design of forces, there is not just one force but many superposing to the overall acting force. Hence, reflections upon these forces are crucial. Several questions arise during the development process. For example, how do forces influence each other, counteract, lead to instabilities, or small step sizes? With the number of particles usually considered in an SPH simulation, traditional debugging may not lead to success. Using the presented visual debugging environment, it is possible to investigate such questions. It allows for a broad investigation of the data set, combining one or more internal attributes of the simulated dataset. By using non-spatial views, another perspective on the data is revealed. Combined with brushing and linking, alpha blending, and stochastic sampling, it is possible to identify many issues in a fast and convenient way. The various combinations of attributes and views lead to new insights into the given particle data set. Moreover, errors in the implementation can be isolated.

However, there are still some limitations in the presented approach of visual debugging SPH simulations: For very large particle counts (> 50M), low refresh rates in the spatial view are observed. Furthermore, overplotting in non-spatial views becomes an issue already at a lower particle count. To improve the performance of the spatial visualization for large data sets, it might be necessary to move to a cluster computer, e.g., using the CPU-based ray tracing proposed by Wald et al. [123]. Nonetheless, for large data sets, it is more important to reduce visual clutter due to overplotting. Using alpha blending in combination with stochastic sampling (reduced to 1%), it is possible to identify trends the PCP for data set of 200k particles (see Figure 3.5). However, the risk of losing important information is quite high.

In future work, this issue could be addressed by adding special techniques like the outlier-preserving focus+context-method for PCP by Novotný and Hauser [96]. With the scatter plot, it is still possible to analyze data sets with increased particle counts. Using alpha blending based on point density, patterns are still identifiable and the refresh rate maintains interactive even with higher particle counts (up to 5M). However, state-of-the-art simulation systems can produce simulations with much higher particle counts where the visual limits of conventional scatter plots are exceeded. Nonetheless, when debugging simulations or developing new models, usually lower particle counts (similar to the ones used in the case studies) are used, since the simulation itself takes some time and higher particle numbers would not be suitable for efficient debugging.
3.6 Summary

In this chapter, an approach for visual debugging of SPH simulations with large particle counts and multi-dimensional attributes was presented. This provides one possible answer to Research Question 1. After discussing previous work, the visualization requirements (R1–R6, Section 3.2) were identified. They are tailored to the characteristics arising for SPH-based fluid animations. Next, the application design was discussed to meet these requirements (Section 3.3). To show the versatility and benefits of the presented approach, visual debugging was performed in four real-world case studies (Section 3.4).

The presented approach supports the development process of novel methods for SPH-based fluid animation. It was used to develop the methods presented in the following chapters. In particular, it is used for visualization of the simulation results supporting interactive exploration.
In this chapter, Research Question 2 is addressed by presenting a method for asynchronous time stepping. It applies to any SPH-based fluid solver that solves the pressure term non-iteratively. In particular, it is tested with non-iterative EOS with splitting as described in Algorithm 2.2.

Most SPH solvers use a uniform time step for every particle that can be adapted in each simulation step according to the CFL condition. For stability reasons, this restriction may only be necessary for a few particles. As shown in the histogram in Figure 4.1, the possible time steps for a common scenario may vary substantially. As they cover up to three magnitudes, much unnecessary computational effort is spent when using a fixed or global adaptive time stepping method.

In comparison to previous approaches that address locally adaptive time integration, the presented method avoids synchronizing particle states at time barriers [13], [38], [39] at any time during the simulation as well as the division of the fluid domain into regions [50]. The technical contributions of the method can be summarized as:

- A fully asynchronous time integration method is presented, avoiding global synchronization barriers. Throughout the simulation, each particle has its dedicated time step depending on the state of the simulation, and particles are processed using a priority queue.

- While SPH simulations with a uniform time step for every particle can easily be parallelized on shared memory architectures, parallelism with asynchronous integration is not straightforward. To overcome this issue a parallelization technique for asynchronous time integration using multiple queues is introduced.
4.1 Related Work

To increase the computational efficiency of simulations, especially for large particle numbers, adaptive models provide a suitable solution. A general overview of adaptive models in computer graphics can be found in the survey of Manteaux et al. [86].

To employ spatial adaptivity in SPH, different concepts have been proposed. Desbrun and Cani [38] use a basic splitting and merging concept to increase the simulation resolution only in areas of interest. Adams et al. [6] introduce a condition based on geometric local feature sizes for splitting and merging, and vary particle sampling by weighting according to visual importance. With the approach of Ando et al. [10], particles inside the fluid are collapsed and a high-resolution simulation is performed on the surface only. To preserve thin features, they base the resampling criterion on the anisotropy of the neighborhood of the particle. Orthmann and Kolb [97] use diffusive flux as a resampling criterion. Another approach to spatial adaptivity employs coupled simulations at different scales as proposed by Solenthaler and Gross [113]. They perform a fine-scale simulation on predefined regions that can dynamically change. As this approach does not guarantee mass conservation, Horvath and Solenthaler [61] propose an emission scheme that tracks the total mass and iteratively compensates volume changes.

The presented method addresses temporal adaptivity. A straightforward approach is to determine a globally adaptive time step, using the CFL condition as proposed by Desbrun and Gascuel [39]. This is also used for predictive-corrective incompressible SPH (PCISPH) with additional shock handling presented by Ihmsen et al. [65]. Goswami and Batty [50] introduce a regional time integration model. Regions with individual time steps are defined and particles in this region are advanced accordingly. They define a base time step $\Delta t_b$. 

Figure 4.1: In SPH-based fluid simulation, the maximum possible time step size varies across particles as shown in the histogram. While previous methods use the global minimum to advance the particles, a dedicated time step for each particle is allowed through asynchronous time integration. In the left figure, the used time step is color-coded for a 10M particle simulation. For this large-scale scenario, speedups of a factor of 6.3 are achieved compared to a fixed time step.
and restrict the possible time steps to a multiple of $\Delta t_b$, hence, introducing equidistant buckets. Again, Desbrun and Gascuel [39] propose using an individual time step per particle. Based on a reformulation of the CFL condition, the simulation time step is subdivided by powers of two. Forces are evaluated only if required by the stability criterion, and positions are integrated with the smallest time step. Ban et al. [13] follow this idea and compute the time step for each particle, hence, employing individual stepping. If the individual time step of a particle is smaller than a global time step, it is set active and evaluated in the SPH loop. Otherwise, the particle is marked as inactive and is linearly interpolated to the next global time step. As in globally adaptive stepping, the global time step size is determined by the minimal individual time step size. Particles are synchronized every export sample as presented by Desbrun and Gascuel [39]. In contrast to these methods, the presented time integration scheme has no need for global synchronization. In Section 4.3, the method is compared to globally adaptive and individual time stepping.

For elastodynamics, asynchronous variational time integration (AVI) allows substantial speedups without reducing accuracy [81]. In computer graphics, AVIs have been successfully applied to cloth simulations, including asynchronous collision handling [57], [120]. Schroeder et al. [110] propose a hybrid implicit and semi-implicit asynchronous time integration method to simulate deformable objects. Asynchronous integration is also used with fully implicit time integration [129]. They all employ AVI on finite element simulation models with predetermined neighborhood information. The presented asynchronous integration method, in contrast, addresses Lagrangian flow with varying particle neighborhoods.

Speeding up SPH simulations can also be achieved by exploiting parallelism. On shared-memory systems, SPH simulation models using global synchronizations can be parallelized using loop parallelization. Efficient data structures for parallel neighborhood query and processing are discussed in the survey of Ihmsen et al. [64]. Thaler et al. [119] present an architecture to compute implicit incompressible SPH simulations in parallel on distributed memory systems. Kale and Lew [73] propose a scheme for the parallel execution of AVI for elastodynamic objects. The discussed parallelization scheme builds upon their idea of multiple parallel queues to parallelize asynchronous time integration for SPH. For load balancing, they decompose the domain by solving a graph partitioning problem using weighted edges for the predetermined neighbors. As no predetermined neighborhood exists in Lagrangian flow, an over-decomposition of the problem size is proposed.
4.2 Asynchronous Time Integration

In this section, the approach for the asynchronous simulation of SPH-based fluids is presented. A time integration model, where each particle $i$ can be integrated with a dedicated step $\Delta t_i$ determined by Equation 2.52 is proposed. Using the common approach for particle processing, as shown in Algorithm 2.2, the particles’ attribute computations need to be synchronized several times during one simulation step. That introduces a large number of synchronization barriers, as illustrated by the combined particle updates in Figure 4.2 (a) and (b). Each particle has to be processed five times per update to compute its new state. With the presented approach, it is possible to compute the new state of a particle at once, as outlined in Algorithm 4.1. Furthermore, there is no need for global synchronization of the simulation as no global simulation state exists. Only a virtual barrier is added to export specific states of the simulation for visualization. The presented approach for asynchronous particle handling follows the concept presented by Lew et al. [81] and enforces a strict particle processing order using a priority queue. Figure 4.2 (c) illustrates the time integration concept and the particle processing order.

4.2.1 Asynchronous Particle Handling

SPH-based fluids consist of individual particles that are locally influenced by their surrounding neighbors. This means that the problem is of compact support, at least when using a non-iterative pressure solver. The particles can be handled individually, as long as the needed neighborhood information remains consistent. As each particle has its own timestamp $t_i$, it usually differs from the timestamp $t_j$ of its neighbors, which is illustrated in Figure 4.3. To compute the fluid properties at particle $i$, the attributes of the neighboring particles need to be computed at time $t_i$ to obtain a consistent state. This is done by reconstructing the attribute values $A_j(t_i)$ of the neighboring particles $j$ as illustrated in Figure 4.3. The attributes of neighboring particles are traced back in time using their derivatives:

$$ A_i(t_i) = A_j(t_j) + (t_i - t_j) \frac{dA_j(t_j)}{dt}. $$

(4.1)

The following attributes from neighboring particles are needed to compute the fluid properties at particle $i$: mass $m_j(t_i)$, density $\rho_j(t_i)$, pressure $p_j(t_i)$, position $x_j(t_i)$, and velocity $v_j(t_i)$. $x_j(t_i)$ and $v_j(t_i)$ are reconstructed using the already computed accelerations. The employed scheme for this task is described in Equation 2.50. The derivative $\frac{d\rho_j}{dt}$ is needed to determine the advection density and to reconstruct $\rho_j(t_i)$. Since non-iterative EOS with splitting
Figure 4.2: Illustration of different time stepping methods: (a) fixed time stepping (FTS), (b) global adaptive time stepping (GTS), and (c) asynchronous time stepping (ATS). With FTS and GTS, particles are updated synchronously based on the minimum time step. With ATS, varying time steps are used for the evolution of different particles, which leads to a strict particle processing order using a priority queue ((c) bottom). No global synchronization is needed and virtual barriers are only employed to export simulation states. By enlarging the time step size for each particle, the overall amount of needed operations is reduced.
Figure 4.3: Neighborhood reconstruction in asynchronous time integration for a particle \( p_3 \): on the right, neighbors contributing to the attribute computation are shown, and on the left, the individual time steps. Attributes \( A_j(t_j) \) are traced back in time to \( t_i \) to compute \( A_i(t_i) \).

is used, it is already computed, and \( \rho_j(t_i) \) can be reconstructed without any extra cost. The pressure \( p_j(t_i) \) is computed on the fly based on the EOS. As the mass \( m_j \) is constant, there is no need to compute \( m_j(t_i) \).

After synchronizing the neighborhood, it is possible to compute the attributes of particle \( i \) including time integration, while processing it only once. An outline of the asynchronous step is given in Algorithm 4.1.

To handle all attributes at once and to ensure that no important information (such as shocks) is missed in the simulation, one important constraint is added to the time integration method: a strict particle processing order \( t_i \leq t_j \) for all neighbors \( j \) is enforced. That also ensures that neighbors are only traced back in time, which further enhances the stability of the simulation system.

To identify the particle that has to be processed next, a priority queue is employed, where particle indices are stored and sorted by their current simulation time. Figure 4.2 (c) illustrates how the particles evolve. At the bottom, the particle queue is depicted. The front-most particle is dequeued and its attributes are computed as described in Algorithm 4.1. Then, the particle is evolved and pushed back into the queue. The top row of Figure 4.2 shows the timeline of selected particles. It is observable that the presented model does not introduce any global barriers into the simulation. Only a virtual, non-invasive, barrier is added to export the current simulation state at \( t_e \): once the last particle has reached \( t_e \), it is possible to store the current simulation state. For export, the particle attributes are traced back to \( t_e \) using the same reconstruction scheme as used for the neighborhood synchronization (Equation 4.1).
Algorithm 4.1 The asynchronous simulation step of one particle

function \textsc{do\_step}(i)
\begin{itemize}
  \item determine possible time step $\Delta t_i = \min_j(\Delta t_j)$
  \item reconstruct neighbor attributes $A_j(t_i)$
  \item compute $\rho_i(t_i)$
  \item compute $a^\mu_i(t_i)$
  \item compute $v^*_i$ using non-pressure accelerations (e.g., $a^\mu_i$ and $a^b_i$)
  \item compute $\rho^*_i(t_i)$ using $v^*_i(t_i)$ and $v^*_j(t_i)$
  \item compute $p_i(t_i) = k(\rho^*_i(t_i) - \rho_0)$ and $a^p_i(t_i)$
  \item integrate particle $i$ over time using $\Delta t_i$
\end{itemize}

Algorithm 4.2 presents the queue concept. The queue is processed until all particles reach the next $t_e$, and the simulation state is exported. After exporting, this step is repeated until the end of the simulation. Algorithm 4.2 also includes the parallelization concept discussed in the following section. The additional operations needed for parallel processing are highlighted in blue.

4.2.2 Parallelization

As modern hardware supports multi processing, it is important that the method can be executed in parallel. In the following, it is discussed how the method can be adapted for parallel execution. The parallelization of a typical SPH step (Algorithm 2.2) can be achieved by parallel execution of the loops over all particles. Using the asynchronous time stepping, this is not possible as there exists no predetermined loop count. To overcome this issue, the concept of Kale and Lew [73] is adapted to parallelize the simulation by using multiple queues $q_a$ (with $1 \leq a \leq b$ and $b$ the number of used threads). Spawning $b$ queues is sufficient for particle counts up to around $1M$. For large scale scenarios, such as the $10M$ dam break depicted in Figure 4.1, the solver suffers from waiting threads and load balancing needs to be considered. Kale and Lew [73] propose calculating the computational cost per element and use neighbor relations to perform a domain decomposition to distribute the elements in such a way that the computational effort is equal for all threads. Due to the lack of predetermined neighborhoods, this approach is not feasible in this case. Instead, the task is solved using over-decomposition of the problem size, i.e., spawning multiple queues per thread. In most cases, spawning three queues per thread is sufficient. The individual queues are handled in parallel. The concept is depicted in Figure 4.4. As the different priority queues do not communicate, the enforced particle processing order ($t_i \leq t_j$) may be violated. If a neighbor $j$ of the current particle $i$ is held in a different queue, the correct processing order cannot be guaranteed. To resolve this issue, a waiting list $wl_{q_a}$
Algorithm 4.2 Particle handling using a priority queue (operations for parallel execution are highlighted in blue)

```plaintext
function run_queue(queue, t_e)
    Initialize waiting list wl
    Initialize step counter ctr
    while t_queue.top() < t_e do
        i = queue.pop()
        find neighbors j
        if t_i > t_j then
            ▶ t_i ≤ t_j is not guaranteed for multiple queues
            wl.push(i)
        else
            DO_STEP(i)
            t_i += ∆t_i
            compute ∆t_i new (Equation 2.52)
        queue.push(i)
        ctr = ctr + 1
        if ctr % ⌊0.025 · queue.size()⌋ == 0 then
            insert all particles i of waiting list wl into queue
    per queue is introduced. The particle i is removed from the queue and added to the waiting list if any of its neighbors has not yet reached the necessary simulation time t_i. Figure 4.4 shows this process. The list is cleared, and the particles are reinserted back into the queue in regular intervals and, thus, it is possible to enforce the processing order again. The parallel asynchronous process of one queue between two export steps is shown in Algorithm 4.2.

4.2.3 Implementation Details

At first glance, organizing the particles in a queue seems to be expensive, but storing only the particle indices, and using a heap to organize the queue grants access to the next processed particle in constant time. The attributes of the particles are stored in contiguous arrays.

For the neighborhood search, a uniform grid approach is used to store the cells in an array using Z-index ordering [64]. Since a global update of the neighbor cells after processing each particle would be expensive, a global update is only done at each virtual export barrier. To avoid missing a neighbor contributing to the attribute computation, it is checked if the particle has changed the cell after evolving it. If this is true, the particle is added into the new cell without removing it from its previous cell. Combined with the strict particle processing
4.2. asynchronous time integration

![Diagram](image)

Figure 4.4: The parallelization concept for asynchronous time stepping: when using multiple queues, the strict particle processing order may be violated. It is enforced by introducing a waiting list per queue. If the condition $t_i \leq t_j$ is violated by any neighboring particle, the current particle $i$ is pushed into the waiting list.

order, it can be ensured that no neighboring particles are missed. The global update after each export also helps to increase performance.

The introduced waiting list adds a slight amount of extra work, especially if many of the neighboring particles are stored in other queues. As Z-index ordering of the particle arrays is employed, it is possible to simply split the queues by the particle index and enhance the separation of particles stored in different queues. Since particles move during the simulation, this separation will be reduced. Therefore, reordering of the particles at specific timestamps is performed. Afterward, the queues are reinitialized and, hence, the particle separation across the different queues increases again. This reinitialization is performed after several virtual export barriers. In the experiments, a reinitialization of the queues $q_a$ was done after $t_e$ has reached a multiple of 0.5s, which provides a good trade-off between the overhead of reinitializing the queues and postponing particles that violate the strict particle processing order. The global reconstruction is performed without particle synchronization. It enhances the performance of the time integration method.

Defining the time of reinserting the postponed particles into the queue is crucial for performance optimization. Reinserting them too early could lead to postponing these particles multiple times. In contrast, reinserting them too late could lead to unnecessary postpone actions in other queues. Unfortunately, there is no predefined value for this issue and needs some testing, as the optimal time to clear the waiting list $wl_{q_a}$ depends on the scenario. In the experiments, setting the size to 2.5 % of the queue size has proven to work well in many cases.
One global step is performed to initialize the simulation. Afterward, the individual $\Delta t_i$ are determined. Similar as Monhagan [92], Ihmsen et al. [65] propose using $\lambda_v = 0.4$ and $\lambda_a = 0.25$ to compute the time step sizes. Since the EOS is used, stiff forces arise that lead to the need of smaller CFL constants. Ban et al. [13] use $\lambda_v = 0.1$ and $\lambda_a = 0.025$. Employing the splitting concept, it is possible to enlarge the constants to $\lambda_v = 0.25$ and $\lambda_a = 0.05$. Similar to Goswami and Batty [50], the step sizes are clipped to be a multiple of 0.5ms, which further enhances the stability of the simulation.

### 4.3 Results

In this section, results obtained with the presented asynchronous time stepping (ATS) model are discussed and compared to previous methods: fixed time stepping (FTS), globally adaptive time stepping (GTS) [39], and individual stepping (ITS) [13]. First, the performance is investigated, followed by a visual comparison. In the visual comparison, increased velocity damping can be observed when the fluid interacts with rigid collision objects. To further investigate these differences, a quantitative analysis is performed.

All results are performed on a workstation with an Intel Xeon E5-1650 v3 hexacore CPU at 3.5 GHz and 64 GB DDR4 memory. For parallelization of the simulations, OpenMP is used to employ loop parallelization. All tested approaches are implemented in the same simulation system, i.e., use the same code base for all simulations. For visual comparison, the particles are rendered as spheres to avoid losing details in surface reconstruction. The sphere rendering is performed with the custom Plugin for MegaMol [54], as described in Chapter 3. For surface rendering, the level-set technique by Bhattacharya et al. [25] is used to extract the fluid surface.

If not stated otherwise, the smoothing kernel presented by Müller et al. [95] are used for the rest of this chapter. The viscosity parameter is set to $\mu = 0.01$, and to model surface tension Equation 2.57 is used with $\varphi = 0.02$.

#### 4.3.1 Performance Comparison

The performance study is conducted separately for serial and parallel execution. Since ATS processes particles differently than previous methods, runtime using serial code is investigated first. Then, performance and scalability using the parallelization presented in Section 4.2.2 is analyzed.

Four different scenarios are used, with different characteristics and particle counts, varying from $27k$ up to $10M$. The first example is a corner dam break (CDB) with $27k$ particles arranged in a cube of $30 \times 30 \times 30$ particles. This scenario is then scaled up to $125k$ ($50 \times 50 \times 50$) particles. Visualizations
4.3. results

Figure 4.5: A simulation with asynchronous time integration using 300 k particles with a radial force field and a bunny as a collision object to disturb the flow. A speedup of a factor of 7.5 compared to fixed time stepping is achieved.

of this scenario can be found in Appendix A. For parallel performance comparison, three larger scenarios are additionally included. The first test case is shown in Figure 4.5. It consists of 300k fluid and 20k boundary particles. To generate an almost homogeneous flow disturbed by a collision object, a radial force field is added as external forces in this test case. The second scenario consists of 1.3M fluid and 16k boundary particles as shown in Figure 4.8. The collision object is a slightly deformed cone. A linear force field in positive y-direction is added beneath the cone to produce a fountain. Due to the deformed shape of the cone, the splash on the top starts to move around and a non-symmetric flow is formed. In this scenario, most of the particles are barely moving, and only a few are involved in the interaction. A large-scale scenario with 10M particles (see Figure 4.1) is included in the performance comparison to test the scalability of the presented method. As the focus is on performance, the largest possible time step size is used that still maintains a stable simulation in all scenarios.

In Table 4.1, overall runtimes and speedups of ATS compared to FTS, GTS, and ITS are given for all test scenes. Using serial execution, ATS is approximately 7.5 times faster than FTS, around three times faster than GTS, and up to twice as fast as ITS. Parallelization introduces computational overhead, and the speedup of all tested methods is slightly reduced. With ATS, additional work is added to the solver when introducing the waiting lists $wl_{qr}$. Even though the speedup decreases, the difference to GTS and ITS is only minor, and ATS outperforms all other tested methods.
Table 4.1: Runtime comparison with performance tests conducted on five different scenarios: a 27k corner dam break, the radial flow (Figure 4.7), the fountain scenario (Figure 4.8), and a large scale dam break (Figure 4.1). Runtime measurements are given in seconds and elapsed computation time is shown for 1 s simulation using fixed time stepping (FTS), global adaptive time stepping (GTS), individual time stepping (ITS), and fully asynchronous time stepping (ATS). For FTS, the time step size is set $\Delta t = 0.25$ ms for all scenarios. The radial flow (Figure 4.7), the fountain scenario (Figure 4.8), and a large scale dam break (Figure 4.1) have different measurement times due to runtime constraints.

<table>
<thead>
<tr>
<th>Scene</th>
<th>#particles</th>
<th>FTS runtime</th>
<th>GTS runtime</th>
<th>ITS runtime</th>
<th>ATS runtime</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corner Dam Break</td>
<td>27k</td>
<td>396.9</td>
<td>130.4</td>
<td>3.0</td>
<td>89.7</td>
<td>4.4</td>
</tr>
<tr>
<td>Corner Dam Break</td>
<td>125k</td>
<td>1896.7</td>
<td>768.1</td>
<td>2.5</td>
<td>532.1</td>
<td>3.6</td>
</tr>
<tr>
<td>Corner Dam Break</td>
<td>6 threads</td>
<td>87.17</td>
<td>30.8</td>
<td>2.8</td>
<td>21.12</td>
<td>4.1</td>
</tr>
<tr>
<td>Corner Dam Break</td>
<td>27k</td>
<td>375.9</td>
<td>170.8</td>
<td>2.2</td>
<td>117.6</td>
<td>3.2</td>
</tr>
<tr>
<td>Corner Dam Break</td>
<td>300k</td>
<td>926.7</td>
<td>723.9</td>
<td>1.3</td>
<td>304.2</td>
<td>3.0</td>
</tr>
<tr>
<td>Corner Dam Break</td>
<td>1.3 M</td>
<td>3245.6</td>
<td>1954.0</td>
<td>1.7</td>
<td>1427.2</td>
<td>2.3</td>
</tr>
<tr>
<td>Corner Dam Break</td>
<td>10 M</td>
<td>25973.0</td>
<td>14560.0</td>
<td>1.8</td>
<td>25973.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

6 threads

1 thread

ATS

ITS

GTS

FTS

#particles

Scene

Runtime comparison with performance tests conducted on five different scenarios: a 27k corner dam break, the radial flow (Figure 4.7), the fountain scenario (Figure 4.8), and a large scale dam break (Figure 4.1).
Figure 4.6: Scaling tests: on the left, strong scaling is shown and on the right, weak scaling. In strong scaling, ATS scales slightly worse compared to other methods using only a few cores, but as the number of cores increases, it outperforms ITS. On weak scaling, all other tested methods are outperformed.

When the scenes become bigger, the additional workload becomes smaller compared to the overall computational cost and the speedup increases. In the radial flow scenario (Figure 4.7), the performance gain of ATS compared to GTS and ITS is apparent. For the fountain scene (Figure 4.8), ATS is still faster than the compared methods. However, the speedup is less in this situation. While still being almost twice as fast as ITS, ATS maintains only an overall speedup of a factor of 4.4. It is observable that ATS shows the best performance if there is much interaction present. As the fountain scenario has a large amount of barely moving particles, the speedup gained with ATS slightly decreases.

ATS is also tested on a 10M dam break with collision objects as shown in Figure 4.1. Almost linear scaling of the computation time can be observed when increasing the number of particles. Table 4.1 shows that a speedup of a factor of 6.3 is achieved compared to FTS. GTS is outperformed by a factor of 3.5. In absolute numbers, this means that it is possible to reduce simulation time from 7.22 hours using fixed stepping to 1.14 hours employing ATS.

Further, a scaling study is conducted, measuring strong and weak scaling of all methods. The strong scaling study is performed on the 125k CDB. The overall computation time for the simulation of 3s is measured, and the average over three simulation runs is computed. The number of cores is increased from one up to six, and the relative speedup is computed. The left of Figure 4.6 shows the comparison of the different time integration methods. As ATS introduces extra work to the parallelization, it scales slightly worse than the
other methods. However, with an increasing number of cores, the difference gets smaller. Using six cores, the speed up gets higher compared to ITS. FTS, GTS, and ITS first scale well with few cores, but cannot maintain the relative speedup if the number of cores increases. These methods are based on simple loop parallelization. They introduce a number of barriers to the simulation, as they have to synchronize multiple times per step. Therefore, scaling decreases with an increasing number of cores. ATS only enforces a virtual barrier at the export time \( t_e \), and the threads can run more independently and longer.

As load balancing is employed only for large-scale scenarios, weak scaling needs to be investigated as well. To this end, a fixed number of 10\( k \) particles per core is used in a test scene similar to the 125\( k \) corner dam break. Again, the relation between serial and parallel execution time is computed. In Figure 4.6 (right), the results of the weak scaling study are shown. Here, ATS outperforms all other investigated methods. In this test case, one queue per thread is used for ATS. Some areas in the simulation require more computation time, as there is more interaction, and smaller time steps are needed. Hence, ATS introduces waiting threads, and the scaling slightly decreases. This issue is addressed by spawning at least three queues per thread when simulating larger scenarios, such as the 10\( M \) dam break.

In the conducted experiments, the overhead introduced by the queue handling was almost negligible compared to neighborhood search and the particle attribute computation itself. Although some extra work is added, ATS reduces the overall number of operations for the particles, compensating this overhead.

4.3.2 Visual Comparison

A visual comparison to explore differences of the considered time integration methods is performed as well. To this end, the resulting animations of the fountain and the radial flow scene are compared.

One noticeable difference between ATS and the other time integration schemes can be observed: ATS dampens the fluid velocities to a small extent, in particular, at interaction with collision objects. In the radial flow scenario, only small differences in the velocity field are observed between the tested methods. To identify them, the particles are colored by their velocity magnitude, as shown in Figure 4.7. Overall, the velocity magnitudes look similar. At the flow behind the collision object, smaller velocities occur in our model compared to FTS. Using ITS or GTS, the velocities are also damped, but not as much as with ATS.

The variations become larger in the fountain scenario, as shown in Figure 4.8. Here, the overall shape behaves differently. Due to the damped velocity magnitudes with ATS, the fluid particles move more slowly than for FTS and GTS when pouring down from the top of the cone. The fluid on the top also moves
4.3. Results

Figure 4.7: Visual comparison of the methods for the radial flow as presented in Figure 4.5. Magnitudes of the velocities are color-coded. Only minor differences in the velocity field are observed. However, in the flow behind the collision object, ATS leads to slightly damped velocities compared to GTS and ITS.

Figure 4.7: Visual comparison of the methods for the radial flow as presented in Figure 4.5. Magnitudes of the velocities are color-coded. Only minor differences in the velocity field are observed. However, in the flow behind the collision object, ATS leads to slightly damped velocities compared to GTS and ITS.

less compared to ITS. As in ITS, inactive particles are linearly interpolated, and the particles on top allow for large time steps. That introduces slight instabilities, and the shape on top starts to move even more compared to FTS. In both scenes, GTS and FTS look quite similar. However, GTS produces many splashes of single particles. No explicit shock handling is employed on GTS and, therefore, these shocks produce droplets when the fluid hits the top of the cone, and when the waterfall hits the fluid surface.

Apart from the above-mentioned differences, ATS can equally be used to create visually appealing results. However, there is one important aspect to mention: besides stability, the focus is on performance. Therefore, the aim is to use the largest possible constants $\lambda_v$ and $\lambda_a$ for the CFL condition. With ITS and GTS,
increased instabilities can be observed at some points in the simulation. In the radial flow scenario, this occurs when the fluid hits the collision object, and in the fountain scenario, it results in large movements of the fluid particles on top of the fountain.

### 4.3.3 Study on Velocity Damping

As mentioned, increased damping of velocity magnitudes at interaction with collision objects with ATS. This fact is further investigated with a quantitative comparison of the fluid velocities. To this end, the mean of particle velocity magnitudes is measured over time for the radial flow and the fountain scenario. The study consists of three parts: First, velocity differences between the different time stepping models (Section 4.3.3.1) are inspected. Then, measurements to evaluate velocity damping induced by the modification of the semi-implicit Euler (Section 4.3.3.2) are provided. As the time step size of an individual particle $i$ is governed by the CFL condition, velocities using different coefficients for $\lambda_v$ (Section 4.3.3.3) are investigated. Since forces acting on particle $i$ are also considered in the CFL [65], [92], the influence of factor $\lambda_a$ is included as well. This study is conducted by comparing the mean as well as the 95th percentile velocities on the fountain scenario.

---

**Figure 4.8:** Side-by-side comparison of integration methods applied to the fountain scenario. Magnitudes of the velocities are color-coded using the same color map as in Figure 4.7. Due to the velocity damping at the collision object, the fluid falls steeper in ATS compared to FTS, GTS, and ITS.
4.3. results

Figure 4.9: Clipping boxes for isolated investigations of the fluid velocities around the collision objects. On the left, a top view of the radial flow scenario is shown. The considered region is outlined by a yellow rectangle. On the right, a side view of the fountain scenario including the considered region is shown.

As mentioned, differences in the fluid flow can in particular be observed at interaction with collision objects. In both investigated scenarios, only slight differences are noticeable when considering the mean of the velocity magnitudes (upper left plots of Figures 4.10 and 4.11). In the radial flow scenario, mean velocities are mainly governed by the radial force field. In the fountain scenario, they are quite small as most of the fluid is at rest. To reflect the observable variations (see the visual comparison in Section 4.3.2) in the quantitative study, the fluid velocities around the collision objects are separately investigated. Figure 4.9 shows the regions considered in this study.

4.3.3.1 Model Comparison

In the radial flow scenario, differences in the velocity field can be observed by visual inspection, in particular, behind the bunny (see e.g. Figures 4.7). As expected, the mean velocities are smaller using ATS compared to FTS, and are slightly larger when using GTS or ITS when considering only the selected region (Figure 4.10 lower left). Regarding the 95th percentile (Figure 4.10, lower right), the same characteristics are present.

Regarding the fountain scenario, differences in the fluid behavior become more apparent. As mentioned in Section 4.3.2, noticeable variances in the overall fluid flow can be recognized using the considered time stepping models. They result in distinct overall shapes of the fluid fountain. As expected, this fact can also be observed in a quantitative study of the particle velocities. Figure 4.11 illustrates measurements of the velocity magnitudes. In the upper row, the mean and the 95th percentile are depicted for all particles. The mean velocity is slightly larger when using ATS compared to FTS, GTS, and ITS.
Another important fact is revealed: using ITS, the mean velocities show large oscillations. This can also be observed by visual inspection as the resting fluid beneath the fountain starts to vibrate at the beginning of the simulation. At $t_e = 4\,\text{s}$, the fluid hits the top of the collision object and bounces off the structure. In the 95th percentile plot considering all particles (Figure 4.11, upper right), this is reflected in a rapid change of the velocity magnitudes at this point when using FTS, GTS, or ITS. With ATS, particle velocities are damped and almost no change is observable. Neglecting the resting fluid beneath the fountain (Figure 4.11, lower row), this effect becomes more prominent when using ATS. However, it is less noticeable compared to the other investigated time stepping models. This complies with the visual inspection: due to damped velocities, the fluid particles have a steeper drop down from the collision object using ATS.
4.3. results

![Graphs showing velocity measurements with Mean and 95th percentile for All particles and Selected region.

Figure 4.11: Velocity measurements of the fountain scenario. Mean and 95th percentile of all particles are shown on the upper row and for the selected region (Figure 4.9) on the lower row. Regarding the mean and the 95th percentile of the selected region, we observe smaller velocities using our time stepping model (ATS) compared to FTS, GTS, and ITS, which show almost similar behavior.

4.3.3.2 Comparison of the Semi-Implicit Euler and the Used Modification

As mentioned, a modification of the semi-implicit Euler step is used as time integration scheme. It may introduce additional numerical damping and, hence, its influence on the velocity field is investigated by comparing it to a simulation conducted using the unmodified semi-implicit Euler step. Similar to the study conducted in Section 4.3.3.1, the mean velocities are measured. The investigations in this study are restricted to GTS and ATS, and measurements are given for the radial flow as well as for the fountain scenario. In Figure 4.12, results of this study are plotted. No major differences can be observed between the semi-implicit Euler and the modification. Regarding GTS, both simulations lead to almost identical plots. Using ATS, there are only minor differences in the velocity field. Overall, these variations are negligible.
Figure 4.12: Comparison of the used time integration scheme and the semi-implicit Euler. Velocity measurements are given for the radial flow scenario using GTS and ATS. Mean and 95th percentile of all particles are shown on the upper row and for the selected region (Figure 4.9) on the lower row. There are no major differences in the velocity plots using the different time integration methods. Using GTS, there are no important differences regarding the mean and the 95th percentile. With ATS, there are only slight differences, which are negligible.

Inspecting the measurements of the fountain scenario, it can be seen that the simulations produce the same results when using GTS (see Figure 4.13). With ATS, higher (and slightly oscillating) velocities occur when using the modified semi-implicit Euler compared to the unmodified if considering all particles. If neglecting the fluid beneath the fountain, these variations decrease. In particular, there is no velocity damping caused by the integration scheme. Nonetheless, it is possible to use slightly larger values for $\lambda_v$ and $\lambda_a$ using this integration scheme and, as the focus is on performance, the modification is preferred over the semi-implicit Euler.
4.3. Results

Figure 4.13: Comparison of the used integrator and the semi-implicit Euler. Velocity measurements are given for the fountain scenario using GTS and ATS. Mean and 95th percentile of all particles are shown in the upper row and for the selected region (Figure 4.9) in the lower row. Using GTS, no important differences could be observed. With ATS, different mean and 95th velocities occur if considering all particles.

4.3.3.3 Influence of the Step Size

In the previous studies, it was recognized that ATS leads to more damping of velocities compared to FTS, GTS, and ITS, especially when interacting with collision objects. In this section, the influence of the time step size on this effect is investigated by varying the values of the coefficients $\lambda_v$ and $\lambda_F$ for the CFL condition. As these differences become most apparent in the fountain scenario, measurements for this scenario are presented in Figure 4.14.

Some effects can be recognized: When neglecting the resting fluid, no differences can be observed in the mean velocities nor in the 95th percentile. Including all particles in the measurements, larger fluid velocities occur when increasing the factor $\lambda_a$. This is caused by larger movements of the almost resting particles beneath the fountain. It can be concluded that there is no velocity damping due to the time step size and that the time step size of a particle is mostly restricted by the force term of the CFL condition.
4.4 Discussion and Future Work

Compared to GTS and ITS, the presented time stepping model ATS leads to higher velocity damping of the fluid when interacting with collision objects. Investigating the combination of ATS with different boundary models would be an interesting topic for future work. Another research direction would be the exploration of how ATS behaves in a massively parallel environment, such as a cluster computer or using GPUs. On the one hand, the simulation domain would need to be split into many more and smaller tasks, and it would be necessary to spawn many more queues. Hence, the possibility that neighboring particles are stored in other queues would increase. On the other hand, the threads could run more independently, and the particle accessing overhead would be reduced to a minimum. Therefore, it can be assumed that ATS would be viable for such environments.
Even though outperforming previous models, there are still possible improvements to ATS. For example, only small values for $\lambda_v$ and $\lambda_a$ are possible, as only basic shock handling is employed by clamping $\Delta t$ to the minimum of its neighborhood, thus restricting the possible time step size. A more sophisticated shock handling, as presented by Ihmsen et al. [65], could improve stability and open up the possibility to use larger values for $\lambda_v$ and $\lambda_a$. Since local constraints based on the compact neighborhood are allowed with ATS, an equivalent formulation depending only on neighborhood information would be needed. For the same reason, it is currently not possible to incorporate iterative SPH models to handle incompressibility constraints. Iterative models such as implicit incompressible SPH [68] or divergence-free SPH [21] solve a system of linear equations (see Section 2.5.4) to compute the pressure globally. That is not possible with the presented method. An alternative locally bound solution would be needed.

If most of the fluid is barely moving, the speedup of ATS decreases. While still outperforming GTS and ITS, only smaller time steps are possible for resting regions compared to moving ones. As the work of Goswami and Pajarola [51] shows promising results for barely active regions, a natural extension would be to follow their concept and handle barely active particles as static ones. One reason for small time steps is a noisy density field that especially occurs at resting regions. A small experiment shows this fact. It is illustrated in Figure 4.15. A cube with $125k$ particles was simulated till the fluid comes to rest. The particles are colored according to their density. On the left of Figure 4.15, a relatively large colored noise in the density field occurs. This results in a noisy pressure field and, therefore, large pressure forces. To reduce this noise, a kernel correction scheme, called Shepard correction [111] was employed.
(Figure 4.15). This kernel correction reduces the noise in the density field but, still, some a coarse-scale noise is visible. That is because classical Shepard correction is used inconsistently when applied to SPH. A solution to this issue is presented in Chapter 5. Unfortunately, it computes the correction factor iteratively using global constraints and, therefore, does not apply to ATS in its current form.

4.5 Summary

In this chapter, a possible answer to Research Question 2 was given. To this end, an asynchronous time integration method for SPH was presented. It allows for dedicated time step sizes for each particle and, thus, increases efficiency of simulations. No barriers are inserted in the simulation, and each particle runs on its own timeline. With ATS, it is possible to achieve speedups up to a factor of 7.5. Additionally, an efficient concept for parallelization of asynchronous integration for SPH has been presented and tested in terms of absolute runtime and scaling. Performance comparisons to FTS, GTS, and ITS were conducted on five scenarios with different characteristics. Moreover, a visual comparison of the simulation results was performed to show the specific properties of the presented model. As increased velocity damping was observed with ATS in combination with the used boundary model, a detailed quantitative study was performed. It includes the analysis of influence of the time step size, the time integration schemes, and the modification to the semi-implicit Euler.
In this chapter, Research Question 3 is investigated. To this end, a smoothing kernel correction scheme is presented that reduces errors in the spatial discretization of the fluid body. As mentioned, to animate the fluid body using SPH, a spatially discrete representation of the fluid body is needed. Due to this spatial discretization, inevitable errors are introduced. For example, the smoothing kernel does not necessarily fulfill the normalization property anymore when used with the discrete version of the fluid body. That introduces errors when computing the fluid quantities as the approximation becomes inaccurate. The Shepard correction [111] (also referred to as Shepard kernel) is often used to resolve the approximation errors. The idea is to normalize the smoothing kernel again by computing correction factors using the discrete evaluation positions. Nonetheless, this approach has one drawback as the correction factors depend on the volume that is represented by a particle (which is an approximated quantity itself). The volume that a particle represents changes over time and space as the particles’ mass is constant and the density changes. More importantly, it depends on the density itself. Shepard correction employs the uncorrected particle volumes to scale the kernel and, therefore, results in an inconsistent and error-prone correction scheme. As indicated in Figure 4.15, it reduces the noise in the density field, but still results in a coarse noise due to the inconsistency.

To resolve this issue, a consistent method to correct the smoothing kernel is presented. An iterative method is proposed that includes the corrected densities to compute the Shepard kernel. As Figure 5.1 demonstrates, a smoother density distribution is achieved throughout the fluid body. The main technical contributions of this chapter are:
Figure 5.1: Snapshot of the fluid pillar scenario consisting of 60$k$ particles simulated with WCSPH. From left to right: simulation without kernel correction, with classical Shepard correction, and with consistent Shepard. Particles are colored with respect to density. The front left quarter of the tower is sliced out, to show the inside of the fluid. With consistent Shepard, a significantly smoother density field is obtained, compared to simulations conducted without kernel correction or with classical Shepard.

- A consistent and stable method to compute the kernel correction.
- A linear formulation of the problem is solved using the power method.
- A kernel gradient correction scheme that accounts for discretization errors in the kernel gradient approximation.
- The rigid-fluid coupling model of Akinci et al. [9] is adapted, and the local number densities are adjusted with corrected kernels.

Employing the power method is very efficient, e.g., for the fountain scenario (Figure 5.2), on average, only 2.1 iterations are needed for the algorithm to converge. Even in highly dynamic scenes with very large time steps, the solver needs, on average, less than five iterations to converge. Therefore, the introduced overhead can almost be neglected. The model is evaluated on weakly compressible SPH (WCSPH) [18] and divergence-free SPH (DFSPH) [21] (see also Chapter 2.5.4). With a suitable norm, it is shown that the density distribution is smoother with the presented method compared to Shepard correction and simulation with an uncorrected kernel.
Figure 5.2: In SPH-based fluid animation, the continuous fluid domain is sampled with particles. This discretization procedure induces errors resulting in a noisy density field (left part of the particle rendering shown on the left). A correction method to reduce this noise is proposed (right part of the particle rendering). The observations are quantified using a suitable norm (right).

5.1 Related Work

The discretization errors due to non-normalized kernels can occur due to sampling deficiency, e.g., near solid and air boundaries. To address these issues near solid and air boundaries, Schechter and Bridson [109] add a narrow layer of ghost particles. Nonetheless, these errors can occur all over the fluid body. Shepard [111] presents an operational solution to reduce discretization errors and produce a continuous surface from the interpolation of irregularly spaced data using weighted averages. This kernel normalization, often referred to as Shepard kernel, is applied by many authors to preserve the discontinuity at interfaces accurately (e.g., by Johnson and Beissel [72] or Colagrossi [33]), such as liquid–liquid or liquid–air interfaces. Grenier et al. [53] additionally use a gradient normalization based on the Shepard kernel. The Shepard kernel is also referred to as zeroth-order correction (or constant completeness) [20], i.e., constant functions are exactly represented by the approximation. In this chapter, a method based on Shepard kernel is proposed, obtaining constant completeness but resolving inconsistencies from the original formulation.

When simulating a fluid, methods that satisfy higher-order completeness conditions are often applied (a completeness condition to the order \( k \) is satisfied if any polynomial function to the degree \( k \) is represented exactly) [20]. An overview of completeness methods for SPH can be found in the works of Belytschko et al. [20] and Vaughan et al. [122]. They examine different approximation techniques that restore various levels of completeness. Krongauz and Belytschko [80] achieve linear (and higher-level) completeness by using a moving least square approximation. This approach is not a kernel correction in a classical sense but computes the approximation directly, achieving a specified

\[
\| \sigma(\bar{\rho}) \|_{\sigma,1} = 6000
\]
level of completeness. Liu et al. [84] use the moving least square interpolation scheme to correct the kernel function, the so-called moving least-square reproducing kernel (MLS RKPM). Similar to Krongauz and Belytschko, a polynomial of arbitrary degree is exactly represented. MLS RKPM applies to any meshless approximation for Galerkin procedures, in particular for SPH.

Johnson and Beissel [72] present an algorithm that uses normalized smoothing kernel functions for SPH. They adjust the smoothing kernel so that constant normal strain rates are represented exactly. Even though their approach does not ensure linear completeness, their results show improved accuracy, especially at free boundaries. Bonet and Lok [27] and Bonet et al. [26] present a variational formulation of SPH that ensures linear completeness and works for both fluids and solids. Their approach reduces density noise remarkably, as discussed by Vaughan et al. [122]. Similar to the presented model, higher-order kernel corrections reduce the noise in the density field. Nonetheless, all of the mentioned methods neglected that the volume represented by a particle changes over space and time. Since the mass is usually set to a constant value and the density changes, the volume has to change. Therefore, it needs to be included in the kernel correction. The presented method resolves this issue. It accounts for the fact that the volume of a particle is also a function over time.

Monaghan [90] proposes a skew-symmetric form of the kernel gradient. This symmetrization results in constant completeness of the gradient [20]. Morris [93] proposes the same formulation to compute the velocity changes due to pressure forces. This formulation is stable and independent of background pressure. Note that it does not conserve momentum exactly [92]. Therefore, alternative variants are often used (see, e.g., Morris et al. [94] or Monaghan [92]). Randles and Libersky [102] propose a kernel gradient correction scheme to model generalized boundary conditions. It can be understood as a generalization of the method by Johnson and Beissel [72] and ensures that the gradient of any linear tensor field is exactly reproduced. Bonet et al. [26] also use modified kernel gradients and achieve linear completeness. Nonetheless, these modifications lead to the problem that the gradient is no longer anti-symmetric and, hence, linear and angular momentum are not necessarily preserved.

All the mentioned methods alter the gradient of the smoothing kernel itself to obtain constant completeness. In this chapter, a method for constant completeness is proposed that does not alter the kernel gradient but implicitly adds a ghost particle to compensate for the discretization errors. A similar idea was used by Ganzenmüller [48]. Instead of adding an integration point (particle) to correct the kernel gradient, he computes a correction force to suppress zero-energy modes (the deformation of an element without change of its potential energy). This approach is similar to the presented concept: a particle is placed virtually, and the resulting force that would occur is used. However, the presented approach has a different goal: While Ganzenmüller suppresses
zero-energy modes, the presented method ensures the kernel gradient sums up to zero. Note that it does not replace the mentioned kernel and gradient correction schemes but can be combined with them.

5.2 Classical Shepard Correction

Before describing the method in detail, it is necessary to recapitulate classical Shepard correction. As mentioned in Section 2.2, using SPH, the fluid quantities are approximated via

$$A(x) = \int_{\Omega(x,h)} A(x') W_h(\|x - x'\|) dV(x').$$ (5.1)

The smoothing kernel $W_h$ must be normalized (see Chapter 2.2), i.e.,

$$\int_{\Omega(x,h)} W_h(\|x - x'\|) dV(x') = 1.$$ (5.2)

This ensures that the constant function $A(x) = 1$ is exactly reproduced. The gradient $\nabla A$ of the fluid quantity $A$ is evaluated via

$$\nabla A(x) = \int_{\Omega(x,h)} A(x') \nabla W_h(\|x - x'\|) dV(x').$$ (5.3)

Equation 5.3 enables the specification of another desired property of the smoothing kernel $W$: To exactly reproduce the derivative of a constant function, such as $A(x) = 1$, the integral of the kernel $W$ over the smoothing domain $\Omega$ must be zero:

$$0 = \nabla 1 = \int_{\Omega(x,h)} \nabla W_h(\|x - x'\|) dV(x').$$ (5.4)

As mentioned, the momentum equation (Equation 2.17) needs to be discretized over space and time to animate the fluid. Therefore, the formulation in Equation 5.1 is discretized (approximating the integral) to evaluate the fluid quantities with SPH. In this step, the normalization property typically gets lost.

The Shepard [111] correction of the smoothing kernel $W_h$ addresses errors introduced by the SPH discretization process due to irregularly distributed particles inside the fluid domain. Especially near open boundaries (such as fluid-air interfaces), the computation of the fluid quantity $A_i$ is error-prone due to the lack of neighboring particles. The following correction factor $c_{sh}^i$ is computed to account for this fact:

$$c_{sh}^i = \frac{1}{\sum_{k \in N_i} \frac{m_k}{p_k} W_{ik}}.$$ (5.5)
The factor $c_i^{sh}$ can then be used to correct the smoothing kernel $W_{ij}$ via

$$\tilde{W}_{ij} = c_i^{sh} W_{ij} = \frac{W_{ij}}{\sum_{k \in N_i} \frac{m_k}{\rho_k} W_{jk}}. \quad (5.6)$$

This correction shall ensure that the property described in Equation 5.2 still holds for the discrete case, i.e., the kernel is normalized (note that this only holds if uncorrected densities $\rho_k$ are considered). The fluid attributes are then computed with the corrected kernel

$$\tilde{A}_i = \sum_{j \in N_i} \frac{m_j}{\rho_j} A_j \tilde{W}_{ij}. \quad (5.7)$$

The classical Shepard correction is illustrated in Figure 5.3 (c).

## 5.3 Corrected Correction

In this section, the approach to compute consistent correction factors $c_i$ by considering the corrected densities is presented. As pointed out earlier, the densities are fluid quantities themselves and need to undergo the correction step as well. They are computed using Equation 2.36, i.e., $\rho_i = \sum_{j \in N_i} m_j W_{ij}$. In essence, this means that with the classical Shepard correction, a different smoothing kernel function (the kernel $W$) is used to compute the correction factors than for the computation of the fluid quantities (the kernel $\tilde{W}$). This use of kernel functions is inconsistent and introduces new errors. Instead of smoothing the density field, the Shepard correction may distort it even more compared to simulating without a kernel correction. The presented method resolves this inconsistency and uses the Shepard kernel $\tilde{W}$ to compute the correction factors $c_i$.

### 5.3.1 Kernel Correction

It is worth to consider another view on the problem: On account of the fluid discretization, the particle $i$ represents a certain amount $V_i = \frac{m_i}{\rho_i}$ of the total fluid volume $V$, as illustrated in Figure 5.3 (b). Using the classical Shepard correction, the density $\rho_i$ and, therefore, the volume element $V_i$ is corrected (Figure 5.3 (c)). This correction process neglects that the volume elements $V_j$ of the neighboring particles $j$ change as well. The presented approach accounts for this fact and includes the corrected neighboring volume elements $V_j^c = \frac{m_j}{c_j \rho_j}$ (Figure 5.3 (d)). Equation 5.5 then becomes

$$c_i = \frac{1}{\sum_{j \in N_i} \frac{m_j}{\rho_j} c_j} \frac{1}{\sum_{k \in N_j} \frac{m_k}{\rho_k} W_{jk}} W_{ij} = \frac{1}{\sum_{j \in N_i} \frac{m_j}{c_j \rho_j}} W_{ij}. \quad (5.8)$$
Figure 5.3: Illustration of the fluid body in the neighborhood of particle \( i \). In the continuous case, the kernel normalization property is satisfied (a), whereas this property gets lost in the discretization process (b). Classical Shepard correction (c) addresses this problem by adjusting the volume of particle \( i \). To compute the correction factor, the volumes of the neighboring particles are needed. However, classical Shepard correction neglects that they undergo a correction (indicated by the green circles in (c)) and uses the uncorrected volumes (black circles) to compute the correction factors. Consequently, the correction factor and, therefore, the volume is calculated inconsistently. A consistent formulation is proposed by computing the correction factors simultaneously (d). It results in a smoother density distribution and, hence, more uniform volumes.
The factor \( c_i \) cannot be computed explicitly anymore as the correction factors depend on each other. This problem can, however, be solved iteratively, e.g., using fixed-point iteration. By \( c = (c_1,...,c_n) \), the vector of the correction factors \( c_i \) is denoted, where \( n \) is the total number of particles. Let \( F(c) = (f_1(c),...,f_n(c)) \) be a vector of functions \( f_i \) defined as

\[
f_i(c) = \frac{1}{\sum_{j \in N_i, \tilde{c}_j \neq 0} \tilde{c}_j m_j W_{ij}}. \tag{5.9}
\]

To solve Equation 5.8, a \( c \) is searched that satisfies \( F(c) = c \). To this end, the following steps are repeated:

1. For all particles \( i \), set \( c^0 = (1,...,1) \).
2. Repeat until converged: \( c^{r+1} = F(c^r) \).

Note that if only one iteration is performed in step (2), it results in the classic Shepard correction. Therefore, the presented method could be interpreted as a generalization of the Shepard correction.

Employing a fixed-point iteration to solve Equation 5.8 has some drawbacks, though. First, the convergence of the method is not ensured, and, secondly, temporal coherence of the solution cannot be guaranteed. This may induce discontinuities in the density field and, therefore, cause instabilities in the simulation. To reduce the likelihood of incoherent solutions, \( c \) can be initialized with the previous solution, i.e., \( c^0(t + \Delta t) = c(t) \). Still instabilities may occur if a repulsive fixed point is obtained. Besides the instability issues, the speed of convergence is typically very slow for fixed-point iterations.

The presented approach accounts for these issues, and an unconditionally stable and fast converging algorithm is developed employing the power method. To this end, an equivalent reformulation of this problem is used. We substitute \( c_i \) with \( \tilde{c}_i = \frac{1}{c_i} \):

\[
\tilde{c}_i = \frac{1}{c_i} = \sum_{j \in N_i} \tilde{c}_j \frac{m_j}{p_j} W_{ij} a_{ij}. \tag{5.10}
\]

If Equation 5.10 is used instead of Equation 5.9, a fixed-point problem \( A\tilde{c} = \tilde{c} \) with the linear function \( A \) is obtained. Solving this is equivalent to finding an eigenvector \( \tilde{c} \) to the eigenvalue \( \lambda = 1 \) of the matrix \( A = (a_{ij})_{(i,j) \in N \times N} \). For this purpose, the matrix \( A \) is examined in detail. First, it needs to be ensured that \( \lambda = 1 \) exists. Fortunately, the column sum of \( A \) equals one, i.e., \( \sum_{i \in N} m_i W_{ij} = 1 \), and the entries \( a_{ij} \) of \( A \) satisfy the condition \( 0 \leq a_{ij} \leq 1 \). This implies that \( A \) is a column (or left) stochastic matrix, and it can be concluded that the
Algorithm 5.1 Compute $\tilde{c}$ using the power method

1: for particle $i$ do
2:   Neighborhood search
3:   If first step then $\tilde{c}_i^0 = 1$
4:   $\rho_i = \sum_{j \in N_i} m_j W_{ij}$
5:   do
6:     $\tilde{c}_i^{r+1} = A \tilde{c}_i^r \|A \tilde{c}_i^r\|$
7:   while $\frac{1}{|N|} \sum_{i \in N} |\tilde{c}_i^{r+1} - \tilde{c}_i^r| > \varepsilon_1$ or $\max_{i \in N} \left\{|\tilde{c}_i^{r+1} - \tilde{c}_i^r|\right\} > \varepsilon_2$

Eigenvalue $\lambda = 1$ exists. In addition, for all other eigenvalues $\lambda$, it holds true that $|\lambda| < 1$ (see Lemma B.2 in Appendix B). With these insights into matrix $A$, it is possible to use the power method [88] to find a solution vector $\tilde{c}$ by considering the recurrence relation

$$\tilde{c}_i^{r+1} = A \tilde{c}_i^r \|A \tilde{c}_i^r\|. \quad (5.11)$$

It is ensured that the algorithm converges and returns a solution for $A \tilde{c} = \tilde{c}$, when applying this method. Furthermore, a unique solution is obtained if the fluid body is connected. By the term connected fluid body, it is understood that every particle is a direct or indirect neighbor of any other particle of the fluid (see Definition B.1). More details and a mathematical derivation are given in Appendix B.5. If splashes occur, there is not one connected fluid body. For a non-fully connected fluid body, the matrix $A$ is not irreducible and, hence, $\lambda = 1$ is not necessarily a simple eigenvalue. However, in the conducted experiments, temporal coherence was obtained when initializing the algorithm with $\tilde{c}_i(t + \Delta t) = \tilde{c}(t)$.

Algorithm 5.1 describes the outline of the method. At least one iteration is always performed. The algorithm stops if the average change rate of $\tilde{c}_i$ is smaller or equal than $\varepsilon_1$, and the maximum rate is smaller than $\varepsilon_2$. Since only direct neighbors are needed to compute $\tilde{c}_i^{r+1}$, the problem can be solved efficiently in a matrix-free way. After computing the kernel correction factors, the kernel is corrected via

$$\tilde{W}_{ij} = \frac{W_{ij}}{\tilde{c}_i}, \quad (5.12)$$

and then the SPH simulation step is performed.

5.3.2 Kernel Gradient Correction

As mentioned, the kernel gradient $\nabla W_{ij}$ is needed as well to determine the fluid quantities. Since the kernel corrections (Section 5.3.1) alter the kernel
function, the computation of the gradient $\nabla W_{ij}$ has to be adjusted. To obtain a consistent formulation for the kernel gradient, Equation 5.12 is taken into account, and the gradient $\nabla \tilde{W}_{ij}$ is computed via

$$\nabla \tilde{W}_{ij} = \nabla \left( \frac{W_{ij}}{\tilde{c}_i} \right) = \frac{\nabla W_{ij} - W_{ij} \nabla \tilde{c}_i}{\tilde{c}_i},$$  \hspace{1cm} (5.13)$$

where $\nabla \tilde{c}_i$ is obtained from Equation 2.37 but considering the corrected densities:

$$\nabla \tilde{c}_i = \sum_{j \in N_i} \frac{m_j}{c_i \rho_j} \nabla W_{ij}.$$  \hspace{1cm} (5.14)$$

Even though Equation 5.13 provides a consistent formulation of the gradient and, hence, increases the accuracy, it does not necessarily satisfy the reproducing condition stated in Equation 5.4. In other words, to exactly reproduce the gradient of constant functions $\sum_{j \in N_i} \frac{m_j}{c_i \rho_j} \nabla W_{ij} = 0$ needs to be satisfied. In general, this condition is violated. The error of the kernel gradient sum is denoted by

$$\xi_{s_i} = \sum_{j \in N_i} \frac{m_j}{c_i \rho_j} \nabla \tilde{W}_{ij}.$$  \hspace{1cm} (5.15)$$

When subtracting $\xi_{s_i}$ from the right-hand side of Equation 5.15, it sums up to zero: $0 = \sum_{j \in N_i} \frac{m_j}{c_i \rho_j} \nabla \tilde{W}_{ij} - \xi_{s_i}$. One element in the sum corresponds to exactly one neighboring particle. Therefore, the term $\xi_{s_i}$ can be seen as an additional addend to the sum, considering the value $\xi_{s_i}$ to correspond to a ghost particle $s_i$ at an unknown location $x_{s_i}$. In other words, a ghost particle $s_i$ is added into the neighborhood of particle $i$ to correct the error. Therefore, $\xi_{s_i}$ can be expressible in the form

$$\xi_{s_i} = \sum_{j \in N_i} \frac{m_j}{c_i \rho_j} \nabla \tilde{W}_{ij},$$  \hspace{1cm} (5.16)$$

To place the ghost particle $s_i$, its location $x_{s_i}$ is needed. To this end, the function $\kappa_i(x) = \frac{m(x)}{c(x) \rho(x)} \nabla \tilde{W}(\|x_i - x\|, h)$ is defined. The location $x_{s_i}$ is then given by

$$x_{s_i} = \kappa_i^{-1}( -\xi_{s_i}).$$  \hspace{1cm} (5.17)$$

Knowing the position $x_{s_i}$ of the ghost particle, the fluid quantity $A_{s_i}$ can be computed using standard SPH interpolation as stated in Equation 2.36 and correct the kernel gradient sum.

Although this would provide an exact solution, computing $\kappa_i^{-1}$ is computationally expensive. Furthermore, it depends on the choice of the kernel function,
is not always well defined, or may even not exist\(^1\). Fortunately, \(A_{s_i}\) can be computed without explicit knowledge of \(x_{s_i}\). As mentioned, the goal is to reproduce constant functions; therefore, it is valid to assume that \(A_{s_i} = A_i\). The corrected gradient \(\nabla A_i\) of the fluid quantity \(A\) is then computed as:

\[
\nabla A_i = \sum_{j \in N_i} \frac{m_j}{c_j \rho_j} A_j \nabla \tilde{W}_{ij} - A_i \xi_{s_i},
\]

(5.18)

When animating the fluid using WCSPH, an antisymmetric kernel is used as stated in Equation 2.44. In this case, \(\xi_{s_i}\) results in

\[
\xi_{s_i} = c_i \rho_i \sum_{j \in N_i} m_j \left( \frac{1}{(c_j \rho_j)^2} + \frac{1}{(c_i \rho_i)^2} \right) \nabla \tilde{W}_{ij},
\]

(5.19)

and the gradient of the fluid quantity \(A\) is then adjusted accordingly.

### 5.3.3 Rigid Boundary Correction

As mentioned in Chapter 2, the approach of Akinci et al. [9] is used to model rigid boundary objects throughout this thesis. The boundary particles contribute to the density computation of the fluid particles, and the density \(\rho_i\) of particle \(i\) reads:

\[
\rho_i = \sum_{j \in N_i^f} m_j \tilde{W}_{ij} + \sum_{k \in N_i^b} \Psi_{b_k}(\rho_0) W_{ik},
\]

(5.20)

where \(\Psi_{b_k}(\rho_0)\) is the so-called local number density, \(N_i^f\) is the set of fluid neighbors of particle \(i\), and \(N_i^b\) the set of boundary neighbors. In essence, this is a ‘pseudo mass’ that is computed based on the configuration of the neighborhood particles:

\[
\Psi_{b_k}(\rho_0) = \frac{\rho_0}{\sum_{j \in N_k^b} \tilde{W}_{kj}}.
\]

(5.21)

The boundary particles are included into the computation of the kernel correction factors to maintain a consistent approach, i.e., correction factors are determined for rigid bodies as well, and the kernel \(W_{ik}\) in Equations 5.20 and 5.21 is replaced by \(\tilde{W}_{ik}\).

The boundary model of Akinci et al. [9] has the advantage that only one layer of boundary particles is needed, and the local number densities are computed

\(^1\)For example, when using \(W_{cbs}^h\), \(\nabla W_{cbs}^h\) is only injective on the interval \([0, h/3]\). In addition, it cannot be ensured that \(\xi_{s_i}\) is in the value range of function \(\kappa_i\).
Figure 5.4: Calculating the density on rigid boundaries results in a noisy density field (left). In contrast, a smooth density distribution is achieved (right) when applying consistent Shepard to the rigid boundary objects as a preprocessing step to the simulation correcting the volumes accordingly.

only with the boundary neighbors, as stated in Equation 5.21. However, the discretization of the smoothing kernel function tends to be error-prone, especially at free surfaces where only a few neighboring particles are available. Therefore, it is proposed to adjust the computation of $\Psi_{b_i}(\rho_0)$ when calculating correction factors using the presented approach, i.e., Equation 5.21 is altered accordingly using corrected kernels. This is done as a preprocessing step to the simulation and, hence, produces overhead only in the initialization but not during the simulation. For the presented approach, the density of a boundary particle $b_i$ needs to be computed via $ho_{b_i} = \sum_{j \in N_i^{b}} \Psi_{b_i}(\rho_0) \tilde{W}_{b_ij}$. Without the kernel correction, a noisy density field is obtained, as the example in Figure 5.4 (left) indicates. When adjusting the boundary volumes and computing the densities by employing the corrected kernel, a smooth density distribution over the complete rigid body (Figure 5.4, right) is achieved with barely any deviations from the reference density $\rho_0$.

5.4 Implementation

The approach is implemented in the open-source framework SPDisHSPlasH, which is also used by several other authors, e.g., Bender and Koschier [21], Bender et al. [22] or Weiler et al. [125]. For all simulations, the cubic spline kernel function [91] is employed. In most examples, surface tension is excluded, but if included, the modification of the IIF model, as presented by Huber et al. (Equation 2.57), is used. For WCSPH, the pressure is computed via Equation 2.41, where the stiffness constant is set as $k = 5.250$ and $\gamma = 7$, if not
5.4. Implementation

\[ t = 0.5 \text{s} \]

\[ t = 2.5 \text{s} \]

\[ t = 4.5 \text{s} \]

With Scaling

Without Scaling

**Figure 5.5:** Selected renderings of the corner dam break scenario. The fluid was simulated using DFSPH and consistent Shepard correction. In the left column, renderings of the simulation with scaling of the stiffness constant \( \kappa \) are shown, in the right one, without scaling. A decreased amount of splashes at \( t = 2.5 \text{s} \) and a slightly smoother surface at \( t = 4.5 \text{s} \) can be observed, when scaling the stiffness constant.

stated otherwise. Fixed time stepping is used for WCSPH with \( \Delta t = 1 \text{ms} \), and adaptive time stepping for DFSPH as presented by Monaghan [92], where the maximum time step is restricted to \( \Delta t \leq 5 \text{ms} \), except for measuring volume preservation. In this case, a fixed time step of \( \Delta t = 2 \text{ms} \) is used for DFSPH. The gradient correction scheme is always incorporated when using the presented method as well as with the classic Shepard kernel.

When applying the consistent Shepard kernel to DFSPH, a little roughness at free surfaces is observed. This is attributed to the interplay of the presented approach and the constant-density solver. With consistent Shepard, densities are corrected, especially at free surfaces, and a slight overestimation of the pressure forces can be observed in these areas. To overcome this issue, the stiffness constant is scaled with \( c_i \) at free surfaces, which leads to a smooth appearance of the fluid surfaces. A side-by-side comparison is given in Figure 5.5.
Figure 5.6: The collapsing fluid block simulated with DFSPH (upper row) and WCSPH (lower row). From left to right: no kernel correction, classical Shepard correction, and consistent Shepard. For both uncorrected and classical Shepard correction, a fine-grained noise in the density field can be observed. Using consistent Shepard, a completely smooth density field is achieved for WCSPH (lower right). Considering DFSPH, the method still significantly improves the smoothness of the density field.

5.5 Evaluation

To show the versatility of the consistent Shepard kernel correction, scenarios of different characteristics are considered, and the approach is compared to simulations conducted with uncorrected as well as with classical Shepard correction. One class of examples is an almost steady fluid, such as the fluid pillar example depicted in Figure 5.1. To complement the evaluation, the behavior of the presented method is explored in highly dynamic scenes like the corner dam break shown in Figure 5.7. Additionally, the density distribution of a collapsing fluid block standing on the ground is examined to reduce the influence of the boundary model (see Figure 5.6).

The method is evaluated considering two important aspects for the animation of a fluid: the smoothness of the density field and the volume the fluid occupies. A smooth density distribution is crucial for the stability because most fluid quantities depend on the density, especially the pressure, which usually produces the dominating forces in the system and, hence, is crucial for the stability of the simulation. With consistent Shepard correction, a slightly increased volume can be observed, which indicates that a more accurate simulation is obtained. Besides the density and volume discussion, convergence, and stability characteristics of the presented method are investigated.

5.5.1 Density Field Evaluation

The density distribution is inspected visually, and the observations are substantiated by measuring the smoothness of the density field locally and globally.
5.5. evaluation

Figure 5.7: A snapshot of the corner dam break scenario simulated with DFSPH (upper row) and WCSPH (lower row). The particles are color-coded with respect to the local density variance. On the left, the simulations without kernel corrections are shown. High local density variances can be observed at the surface but also beneath. With classical Shepard correction (center), there are only slight improvements recognizable. Using consistent Shepard (right), there are only a few regions with a high local density variance (where we would expect them), and the majority of the fluid body has a variance close to zero.

Figure 5.1 depicts a fluid pillar consisting of 60k particles simulated with WCSPH. Without any kernel correction (left), a quite noisy density distribution can be observed throughout the fluid. This issue is slightly improved by applying classical Shepard correction (center), but still, large fluctuations in the density field can be recognized. As WCSPH employs a state equation to compute the pressure, this will result in oscillations of the pressure field and large pressure forces in diverse directions, which cause instabilities in the simulation process. The presented method (right) achieves a smoother density field. Almost no density fluctuations can be observed in the horizontal direction. When simulating the fluid pillar using DFSPH, the density field is already comparatively smooth without any correction, and the effects of consistent Shepard are less significant. Nonetheless, slight density fluctuations can be observed throughout the fluid body, which become even worse when employing classical Shepard correction. In contrast, when using consistent Shepard correction, they are less significant, and a smoother overall density distribution can be achieved (see Appendix B.3).

In all tested scenarios, a smooth density field could be obtained, including the collapsing fluid block (Figure 5.6), where a smooth density distribution over the whole fluid domain can be observed for WCSPH (lower row). With DFSPH (upper row), a smoother density distribution is obtained as well. Although the effect is less prominent, it is still a significant improvement.
Figure 5.8: The mean local density variance $\|\sigma\|_{\sigma,1}$ over time for the corner dam break scenario (Figure 5.7) simulated with WCSPH (left) and DFSPH (right). Consistent Shepard results in a much lower density variance, which is consistent with the visualization of the local density variance in Figure 5.7.

The observations are confirmed by quantifying the noise in the density field. To that end, a seminorm $\sigma_i(\bar{\rho})$ is defined to measure local density variance:

$$\sigma_i(\bar{\rho}) = \left( \sum_{j \in N_i} \left( \frac{\rho_{ij}}{\|x_{ij}\|} \right)^2 \right)^{\frac{1}{2}},$$

(5.22)

where $\bar{\rho} = (\rho_1, ..., \rho_n)$ is the vector of the particles’ densities. That could be interpreted as a local $L_2$-norm of finite differences. Moreover, two other norms $\|\sigma(\bar{\rho})\|_{\sigma,1}$ and $\|\sigma(\bar{\rho})\|_{\sigma,\infty}$ are defined by

$$\|\sigma(\bar{\rho})\|_{\sigma,1} = \frac{1}{n} \sum_{i \in N} |\sigma_i(\bar{\rho})| \text{ and } \|\sigma(\bar{\rho})\|_{\sigma,\infty} = \max_{i \in N} |\sigma_i(\bar{\rho})|,$$

(5.23)

where $\sigma(\bar{\rho}) = (\sigma_1(\bar{\rho}), ..., \sigma_n(\bar{\rho}))$. While $\sigma_i(\bar{\rho})$ characterizes local smoothness, the norms $\|\sigma(\bar{\rho})\|_{\sigma,1}$ and $\|\sigma(\bar{\rho})\|_{\sigma,\infty}$ provide a global characterization.

A snapshot of the dam break scenario is depicted in Figure 5.7, where the particles are color-coded with respect to $\sigma_i(\bar{\rho})$. It is observable that consistent Shepard correction reduces the local density variance, both at the surface and inside the fluid.

Figure 5.8 shows the norm $\|\sigma(\bar{\rho})\|_{\sigma,1}$ over time. This plot confirms that a smoother density field could be achieved for both pressure models, i.e., the local density variance is reduced by orders of magnitudes. In all examples,
5.5. Evaluation

Figure 5.9: Measurements of the mean local density variance for the fluid pillar scenario (Figure 5.1) simulated with WCSPH (left) and DFSPH (right) comparing the density smoothness with the norm $\|\sigma(\bar{\rho})\|_{\sigma,1}$. Classical Shepard correction reduces the local density variance compared to the simulation conducted without correction. Consistent Shepard outperforms both and achieve almost zero local density variance.

A reduced mean and max density variance is obtained. In particular, in the fluid pillar scenario, the mean density variance is almost zero at all times in the simulation, as Figure 5.9 shows. While a noisy density field occurs for the uncorrected and the Shepard correction (at least until the fluid comes to rest), a smooth density gradient is obtained using the consistent Shepard correction. The same behavior can be observed for $\|\sigma(\bar{\rho})\|_{\sigma,\infty}$. Measurements for other scenarios and plots of $\|\sigma(\bar{\rho})\|_{\sigma,\infty}$ can be found in Appendix B.

5.5.2 Volume Evaluation

A slightly increased volume of the fluid can be observed when applying consistent Shepard correction. To further investigate this fact, the volume occupied by the fluid is measured for two scenarios. The first one is the fluid pillar depicted in Figure 5.10. Additionally, a fluid cube with an edge length of 2m is simulated. In this scenario, hard-coded reflective boundaries (i.e., particle positions are clipped at the domain boundaries and velocities are reflected) are used to exclude distortion of the measurement due to the boundary model. Both scenarios are simulated until the fluid comes to rest, then the enclosed volume is computed. To this end, an iso-surface of the fluid with an offset of 0.025m is generated, and its volume is measured. Table 5.1 provides an overview of the measurements. In both scenarios, all methods slightly underestimate the fluid volume. With consistent Shepard, this underestimation is
reduced by a factor of about 2.9 for DFSPH in the fluid block scenario. In the fluid pillar, there is a volume loss of 0.6% with consistent Shepard compared to 4.1% without kernel correction. For WCSPH, the volume loss is reduced by a factor of 4.33.

5.5.3 Convergence and Performance

The presented algorithm is analyzed regarding the convergence rate and stability. Additionally, the performance of the simulation is investigated. The study is conducted on the corner dam break scenario consisting of 125\(k\) particles with an initial distance of 0.05 m, and a smoothing kernel size of \(h = 0.1\) m. When simulating with WCSPH, the stiffness constant is set to \(k = 75000\). For

<table>
<thead>
<tr>
<th></th>
<th>Ref.</th>
<th>WCSPH</th>
<th>DFSPH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fluid Block</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Corr.</td>
<td></td>
<td>7.49 (6.3%)</td>
<td>7.6 (4.9%)</td>
</tr>
<tr>
<td>Class. Shep</td>
<td>8</td>
<td>7.53 (5.9%)</td>
<td>7.73 (3.3%)</td>
</tr>
<tr>
<td>Cons. Shep</td>
<td></td>
<td>7.69 (3.8%)</td>
<td>7.85 (1.7%)</td>
</tr>
<tr>
<td>Fluid Pillar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Corr.</td>
<td></td>
<td>7.01 (6.5%)</td>
<td>7.19 (4.1%)</td>
</tr>
<tr>
<td>Class. Shep</td>
<td>7.5</td>
<td>7.09 (5.5%)</td>
<td>7.29 (2.9%)</td>
</tr>
<tr>
<td>Cons. Shep</td>
<td></td>
<td>7.39 (1.5%)</td>
<td>7.45 (0.6%)</td>
</tr>
</tbody>
</table>
DFSPH, the maximum allowed density error is set to 0.01%, and the maximum allowed divergence error to 0.1%. The time step is increased from $\Delta t = 1\text{ ms}$ up to $\Delta t = 7.5\text{ ms}$ by steps of 0.5 ms, measuring the needed iterations of the power method to converge. The convergence criteria are set $\varepsilon_1 = 0.0001$ and $\varepsilon_2 = 0.0005$. On the left of Figure 5.11, the measurements considering the convergence rate of our method are given. The power method shows a very fast average convergence rate. For DFSPH and $\Delta t = 4\text{ ms}$, it is possible to maintain an average convergence rate of 1.45 iterations, and even with very large time steps such as $\Delta t = 7.5\text{ ms}$, it only needs on average 4.7 iterations to converge. Note that the convergence of the method is independent of the particle count. In Figure 5.13 (right), the average convergence rate is plotted for the collapsing fluid block, while increasing the particle count. The time step is set to $\Delta t = 1\text{ ms}$. It is observable that an almost constant convergence rate can be maintained, and that the presented algorithm scales well with increasing particle count.

Since DFSPH itself employs two iterative solvers, their convergence rate is investigated when combining them with consistent Shepard correction. To this end, the needed iterations of the divergence-free and constant-density solver are measured when including consistent Shepard correction as well as without kernel corrections (Figure 5.11, right). It can be observed that the constant-density solver needs more iterations to converge when using consistent Shepard, e.g., the constant-density solver of DFSPH requires about ten iterations without kernel corrections and 16 with our method. Nonetheless,
it is possible to maintain a stable simulation up to $\Delta t = 7.5\,\text{ms}$ while the largest possible time step for the simulation with the uncorrected kernel is $\Delta t = 5\,\text{ms}$.

The presented method introduces a little computational overhead. In the tested scenarios, the computation time per step is increased on average by about 20% when employing consistent Shepard. Most of the overhead can be attributed to the fact that the boundary particles need to be included in the kernel correction. Therefore, an additional loop over them is needed. It is observable that the overhead is reduced in scenes that incorporate only a few boundary particles.

### 5.5.4 Kernel Gradient Correction

The error of the kernel gradient is examined by computing $\|\xi_s_i\|$, and the mean distance between the active particle and the ghost particle. The measurements are given for the corner dam break scenario. In Figure 5.12, the kernel gradient error is color-coded according to $\|\xi_s_i\|$. It is observable that $\|\xi_s_i\|$ is considerably reduced when using classical Shepard correction (center) compared to a simulation without correction (left). Consistent Shepard further reduces the error, especially in sparsely sampled regions such as within the splashes. At first glance, the error appears to be rather small, but in fact, it is not negligible. To gain a deeper understanding of the error, the mean distance between particle $i$ and the virtually placed particle $s_i$ is computed. In Figure 5.13 (left), the mean distance in relation to $h$ is plotted over time. When simulating without any kernel correction, particle $s_i$ would be placed in a distance between $0.08h$ and $0.12h$ on average. This distance is rather small with – both classical and consistent – Shepard correction.
5.6. Discussion and Future Work

Considering DFSPH, varying run-time overhead could be recognized. When incorporating the presented method, the constant-density solver of DFSPH sometimes needed fewer iterations and, in others, more. Unfortunately, no pattern could be identified when these cases occur. It is assumed that, in some cases, the pressure solver is badly preconditioned, when including consistent Shepard into DFSPH. Choosing a suitable preconditioner could help solve this issue. A more elaborative study on this fact might be an interesting future research direction.

Since the boundary particles contribute to the density computation, they are included in the presented algorithm to compute correction factors. That is a performance issue if the boundary objects dominate the scenario. However, this issue could be addressed by incorporating other state-of-the-art boundary models, such as pressure-based [14], implicit frictional [23], or moving least squares boundaries [15].

When applying the presented gradient correction method, ghost forces are introduced into the simulation. When additionally using consistent Shepard correction, these forces become subtle. These ghost forces alter the simulation in a way that the discretization improves in accuracy, and that the gradient kernel sum equals zero (at least for any constant field). Combining the gradient correction method with other kernel and gradient correction schemes would be interesting to investigate.
As documented in Table 5.1, a relatively high volume loss was observed for DFSPH. At first sight, this seems to be a contradiction to the employed constant-density condition. However, when considering the fluid volume as a sum of individual particle volumes, i.e., \( V_{\text{tot}} = \sum_{i=1}^{N} \frac{m_i}{\rho_i} \), there is no observable volume loss. That means that the volume loss is not reflected in density errors. Furthermore, it indicates that there might be a systematic error hidden in the SPH model itself, in the discretization process, or in other approximations made during the simulation. A more detailed investigation of this fact is a goal for future research.

Consistent Shepard correction was compared with classical Shepard correction. As mentioned in Section 5.1, it also works with higher-order kernel correction schemes, and it would be interesting to combine the presented techniques with such schemes.

Compared to classical Shepard correction, the smoothness of the density field was improved by reducing the noise. Therefore, the stability of the simulation was improved. That also improves the configuration of the pressure field and the pressure force field. The presented technique improves the volume preservation of the fluid and, hence, increases the accuracy of the simulation. Overall, consistent Shepard correction complies with the SPH concept, is efficient to compute, and can be used independently of the pressure model.

5.7 Summary

In this chapter, a novel kernel correction technique was presented that accounts for the error introduced by the discretization of the fluid. It is one possible answer to Research Question 3. The method is unconditionally stable and efficient, as it converges very fast, i.e., the presented algorithm needs, on average, fewer than five iterations to converge, even in dynamic scenes with very large time steps up to \( \Delta t = 7 \text{ms} \). After discussing the details of the presented kernel and gradient correction schemes in Section 5.3, the method was evaluated. In the evaluation, the smoothness of the density field was investigated. To this end, a suitable norm to measure both local and global smoothness was presented. Additionally, the volume changes induced by the presented technique were discussed, and a convergence and performance analysis of the presented algorithm was conducted.
Chapter 6

Solving PDEs on Evolving Surfaces

Having animated a fluid with SPH, a typical next step is the generation of high-quality renderings. To accomplish this task, the surface from an animated fluid needs to be extracted. Secondary effects like foam, bubbles, or wetting effects (of interacting objects) are often added to improve the degree of realism. They can be added either as a secondary simulation in 3D-space or on the surface directly. In this chapter, a method is discussed that describes how to add secondary effects constraint to an animated surface, investigating Research Question 4. It can be used to model secondary effects, such as oil on water, or the reaction and diffusion of two reactive liquids.

A dynamical system is defined by a set of PDEs to describe the secondary effects. The idea is to solve this set of PDEs on an animated surface $M$. To this end, a generic model is designed. It enables the simulation of arbitrary physical phenomena on a moving surface. Fluid flow, buoyancy-induced flow, and reaction-diffusion are used to define a dynamical system and to test the model. As the effects are simulated only on the surface, they allow for efficient modeling of secondary effects. In particular, the presented contributions can be summarized as:

- Design of a generic model, following the physical laws of mass and momentum conservation. The presented model enables the simulation of any physical phenomena that can be defined by a set of PDEs.
- The animated surface defines an outer velocity field. This velocity field is used to model (one-way) coupling of the outer process with the dynamics on the surface. Additionally, mass transfer is modeled using sinks and sources.
6.1 Related Work

The modeling of secondary effects can be achieved in two ways, either directly integrated into the main simulation and two-way coupled, or as a post-process, only one-way coupled. The first one is used when secondary effects influence the overall fluid behavior and appearance. For example, when modeling bubbles in boiling water [121], trapped air in the fluid due to pouring water [67], or foam and bubbles arising in the fluid body [32]. In these cases, the secondary effects are simulated directly with the base fluid simulation. That has the advantage that both simulations can influence each other using two-way coupling. However, if there is a need to modify the secondary ones, both need to be simulated again.

One-way coupling is employed to allow for post-processing of existing simulations, where the secondary simulation does not affect the base simulation. For example, Takahashi et al. [118] proposed a method to model foam and splashes without influencing the base simulation. Kim et al. [75] presented a method to model bubbles with physically plausible movements but controlling their target shape. Ihmsen et al. [66] developed an approach to add foam and spray on top of a fluid simulation. These methods are added as a post-process on the finished base simulation. Nonetheless, they work in 3D, and, therefore, a simulation with high spatial resolution is computationally expensive. The method presented in this chapter simulates secondary effects only on the animated surface and, therefore, high-resolution simulations can be achieved very efficiently. As foam is only present on the surface of a fluid, Akinci et al. [8] presented a method for screen space rendering of foam on top of the original simulation. Gagnon et al. [47] presented a more general approach. They focused on the texturing of fluids and accounted for deformations and topological changes. The presented method is similar as it models secondary effects solely on the surface, but a generalized system is proposed that enables the modeling of any physical phenomena, or animated textures. To this end, a set of PDEs is solved on the surface.

Solving PDEs on static as well as on evolving surfaces has been proposed before. Stam [115] described how fluid flow could be modeled on a static surface. A similar work was presented by Shi et al. [112], but accounting for the characteristics of polygonal meshes. Auer et al. [12] presented a method to simulate fluid flow on static surfaces in real-time using the Closest Point Method (CPM) [108]. Auer and Westermann [11] also presented a method to simulate fluid flow on evolving surfaces. Mercier et al. [87] added turbulence onto the surface of the base simulation by solving a PDE on a narrow-band grid. Recently, material flow on a moving soap bubble was discussed by Ishida et al. [70] and Huang et al. [62]. Xu and Zhao [127] give a good overview of how PDEs can generally be solved on evolving interfaces. In contrast to
these methods, the presented one respects mass conservation on the surface. Intensive quantities are corrected using the velocity divergence of the outer process. The total momentum cannot be conserved, as the velocities are restricted to move along the surface. Nonetheless, the ‘absolute momentum’ can be conserved. An approach to achieve ‘absolute momentum conservation’ is presented. Additionally, one-way coupling is modeled by mass and momentum

6.2 Model

In this section, the fundamentals of the presented method are described and how the physics are modeled on the surface. Additionally, the physical phenomena are described, which are modeled as secondary effects on the surface.

6.2.1 Overview

Consider an evolving surface $M$. Such a surface can result from a finite-difference or SPH simulation, a keyframe animation, or any kind of time-dependent process. On $M$, another process (e.g., the simulation of a substance) is defined by a set of PDEs and coupled to the base process $M$ results from. Therefore, the overall system can be divided into three aspects:

- **Evolution of the surface $M$**
  Due to the surface evolution, the space is altered on which the dynamic process is modeled. In this step, it is accounted for how quantities evolve alongside the surface.

- **Coupling**
  This aspect determines how the base process affects the one on the surface.

- **Modeling the dynamics on the surface**
  This step accounts for the secondary dynamic process and how it is modeled on the surface, i.e., how a set of PDEs can be solved on the surface.

An example of such a process could be a drop of ink on a plastic sheet moving in space. The first aspect (evolution of the surface) describes how the ink is transported in space when the plastic sheet is moving in normal direction. The second aspect (coupling) accounts for the movement of the plastic sheet in tangential direction, i.e., the acceleration of the ink due to friction forces, and the third aspect (the dynamic process on the surface) is concerned with the fluid behavior of the ink itself.
6.2.2 Evolution of the Surface

To model such a dynamic process on an evolving surface $M(t) \subset \mathbb{R}^3$, scalar as well as vector-valued quantities have to be defined on $M(t)$. To this end, scalar $a(x,t) \in \mathbb{R}$ and vectorial quantities $v(x,t) \in T_xM$ are attached to every point $x \in M$, where $T_xM$ denotes the tangential space at location $x$ defined by the surface normal $n(x,t)$ and $t$ denotes a certain point in time. Typical quantities are, e.g., density or velocity when modeling fluid flow on the surface. The velocity field $u(x,t)$ defines how the surface evolves, i.e., how $M(t_0 + \Delta t)$ results from $M(t_0)$, where $\Delta t \in \mathbb{R}_{>0}$. For convenience, $M$ is written instead of $M(t_0)$, $M'$ instead of $M(t_0 + \Delta t)$, and $x$ and $t$ will be skipped if possible. In the following, the surface evolution characterized by $u$ will be referred to as the outer process, and the one on the surface will be called inner dynamics.

A map $O$ is constructed that describes the evolution of quantities due to the outer process. To this end, the space $\mathcal{M}$ is defined as

$$\mathcal{M} = \bigcup_{x \in M} \{x\} \times \mathbb{R} \times T_xM.$$  \hspace{1cm} (6.1)

The quantities needed for the inner dynamics can now be described as elements of $\mathcal{M}$. Without loss of generality, only one scalar and one vectorial quantity are attached to point $x$. It is possible to map an arbitrary number of quantities to $x$ in the same way. The map $O$ relates elements from $\mathcal{M}$ to elements in space $\mathcal{M}' = \bigcup_{x' \in M'} \{x'\} \times \mathbb{R} \times T_{x'}M'$ by

$$O : \mathcal{M} \to \mathcal{M}' : \begin{pmatrix} x \\ a \\ v \end{pmatrix} \mapsto \begin{pmatrix} x' \\ a' \\ v' \end{pmatrix}.$$ \hspace{1cm} (6.2)

An illustration of $O$ can be found in Figure 6.1.

The influence of the outer process defined by $u$ is decomposed in normal $u_n$ and tangential $u_t$ components. The tangential component is used to model friction between the outer and inner dynamics and is discussed in Section 6.2.3. For now, it is assumed that the inner dynamics are coupled to the outer process without friction, and therefore $u_t$ does not affect the inner dynamics, i.e., $m \in \mathcal{M}$ is only altered by $u_n$.

The rate of change $\frac{D}{Dt}x$ that a point $x$ experiences from the action of $O$ is then defined by $\tilde{u}_n := k u_n$, where $\frac{D}{Dt}$ denotes the material derivative and $k \in \mathbb{R}$ has to be chosen in such a way that $x' \in M'$ holds. The motion of point $x$ is directly governed by $\frac{D}{Dt}x$, in particular its velocity is $\tilde{u}_n$ and, hence, $\frac{D}{Dt}x = \tilde{u}_n$. As mentioned, scalar and vectorial quantities are advected alongside the point $x$, respecting certain conservation laws. Details to the evolution of the quantities are given in the following. First, the evolution of scalar quantities is discussed and, next, the evolution of vectorial ones.
6.2. Model

Figure 6.1: Illustration of $O$. The function $O$ maps $x \in M$ to $x' \in M'$. All to $x$ associated scalar and vector-valued attributes are transformed too. Hence, the tangent space $T_xM$ is mapped to $T_{x'}M'$ by $O$.

Evolution of Scalar Quantities Scalar quantities can be distinguished between two types: intensive and extensive quantities [103]. Depending on the type they are evolved differently. An extensive property is a global property (e.g., mass or volume). Such a property is only advected alongside $x$ and not changed, i.e., $\frac{DO}{Dt}a = 0$. In contrast, if $a$ describes an intensive physical property, such as density, it is demanded that

$$0 = \frac{DO}{Dt} \int_{M(t)} a(x,t) dM(x,t)$$

holds, where $dM(x,t)$ are infinitesimal surface elements at position $x$ on the surface $M$ at time $t$. In the following, $t$ and $x$ will be skipped for convenience. Equation 6.3 ensures that the total amount of such a quantity does not change if no phenomena like mass transfer or chemical reactions are present. In this case, the rate of change of an intensive quantity $a$ is given by

$$\frac{DO}{Dt}a = -a(\nabla \cdot (\tilde{u} n))_{T_xM},$$

where $\tilde{u} = (I - nn^T)u$ is the velocity projected onto the tangential space $T_xM$ at point $x$. This means that, if the surface diverges, the concentration of $a$ decreases, and vice versa.

In the following, it is shown that evolving a scalar quantity by employing Equation 6.4 respects the conservation law, stated in Equation 6.3. Changing the differentiation and integration order in Equation 6.3 results in

$$0 = \int_M \frac{DO}{Dt} (a dM) = \int_M \left( \frac{DOa}{Dr} dM + a \frac{DOM}{Dt} \right).$$

As the surface element $dM$ evolves, the term $\frac{DOM}{Dt}$ needs to be evaluated.
According to Stone [116], the material derivative of a surface element can be
described by
\[ \frac{D}{Dt} dM = (\nabla_{T_xM} \cdot \tilde{u}_n) dM, \]  \tag{6.6} 

where \( \nabla_{T_xM} = (I - nn^T) \nabla \) is the gradient operator projected onto the tangen-
tial space \( T_xM \) at point \( x \), and \( \tilde{u}_n \) is the velocity evolving the surface. Without
loss of generality, it is assumed for the latter of the paragraph that the outer
process is described by \( u \) instead of \( \tilde{u}_n \) and \( \frac{dM}{Dt} \) is used instead of \( \frac{DOdM}{Dt} \).
To derive Equation 6.6, it is first shown that the identity
\[ \frac{D}{Dt} dM = (\nabla \cdot u) dM - (\nabla u)^T dM \]  \tag{6.7} 

holds for an arbitrary oriented surface element \( dM = ndM \) [16]. To this end, it
is investigated how volume elements \( dV \) and line elements \( dl \) are changed due
to the outer process. The elements are considered to be that small, that they
are subject only to pure straining motion and rigid rotations [16].
The rate of change of the volume of the volume element \( dV \) is described by
\[ \frac{D}{Dt} dV = \nabla \cdot u dV. \]  \tag{6.8} 

It is assumed that the line element \( dl \) is linear (and, therefore, can be described
by a vector) and stays approximately straight. Under these assumptions, its
rate of change is simply the difference of the velocities at the two ends of the
element and can be described by
\[ \frac{D}{Dt} dl = \nabla u dl. \]  \tag{6.9} 

The volume \( dV \) of the volume element can also be described by \( dV = dM \cdot dl \).
Inserting \( dV \) into Equation 6.8 results in:
\[ (\nabla \cdot u) dM \cdot dl = \frac{D(dM \cdot dl)}{Dt} \]
\[ = dM \cdot \frac{D}{Dt} dl + \frac{D}{Dt} dM \cdot dl \]
\[ = dM \cdot (\nabla u dl) + \frac{D}{Dt} dM \cdot dl \]
\[ = ((\nabla u)^T dM) \cdot dl + \frac{D}{Dt} dM \cdot dl. \]  \tag{6.10}
6.2. model

As Equation 6.10 has to hold for arbitrary line elements $dl$, Equation 6.7 is obtained. To get the formulation in Equation 6.6 the inner-product of $n$ with Equation 6.7 is taken and the identity $n^T(\nabla u)^T n = (nn^T \nabla) \cdot u$ is used:

$$
\frac{D}{Dt} n^T dM = \frac{D}{Dt} n^T n dM = n^T \frac{D}{Dt} dM
$$

$$
= n^T (\nabla \cdot u) dM - n^T (\nabla u)^T dM
$$

$$
= n^T n(\nabla \cdot u) dM - n^T (\nabla u)^T n dM
$$

$$
= (\nabla \cdot u) dM - (nn^T \nabla) \cdot u dM
$$

$$
= ((I - nn^T) \nabla) \cdot u dM = (\nabla_{T_xM} \cdot u) dM.
$$

(6.11)

Inserting Equation 6.6 into Equation 6.5 results in

$$
0 = \int_{M(t)} \left( \frac{D_o a}{Dt} + a(\nabla_{T_xM} \cdot \hat{u}_n) \right) dM
$$

(6.12)

As Equation 6.12 holds for an arbitrary surface $M$, it follows that

$$
0 = \frac{D_o a}{Dt} + a(\nabla_{T_xM} \cdot \hat{u}_n) = \frac{D_o a}{Dt} + a(\nabla \cdot (\hat{u}_n)_{T_xM}).
$$

(6.13)

Therefore, the rate of change of a intensive quantity $a$ is defined by

$$
\frac{D_o a}{Dt} = -a(\nabla \cdot (\hat{u}_n)_{T_xM}).
$$

(6.14)

**Evolution of Vectorial Quantities**  When advecting a vectorial quantity $v$ alongside $x$, it is demanded that $v' \in T_xM'$ holds after applying $O$. That can be ensured by considering $v$ to be a small linear material line element subjected to the velocity field $\hat{u}_n$. Its rate of change is the difference of the velocities at the two ends of the element and can be described by $\frac{D_o}{Dt} v = \nabla \hat{u}_n v$. In the presented model, the vector field $v$ describes the velocity of the inner dynamics. As the directions of the velocities are constrained by the surface evolution, the total momentum will be altered and cannot be conserved. Nonetheless, it is possible to ensure that the sum of the magnitudes does not change, i.e.,

$$
0 = \frac{D_o}{Dt} \int_M \|av\| dM.
$$

(6.15)


**Figure 6.2:** Due to surface evolution the vector $v$ is transformed into vector $v'$. To this end, the map $O$ is employed. Methodically, the mapping of $v$ to $v'$ is divided into three steps. First, $v$ is advected to $x'$. Secondly, it is projected back onto the surface so that $v' \in T_{x'}M'$ holds. Thirdly, the length is scaled that $\|v\| = \|v'\|$. Note that this complies with the vector field evolution defined in Equation 6.20.

Changing the differentiation and integration order and employing Equation 6.4 as well as Equation 6.6, Equation 6.15 reads:

$$0 = \frac{DO}{Dt} \int_M \|av\|dM = \frac{DO}{Dt} \int_M a\|v\|dM$$

$$= \int_M \left( \frac{DOa}{Dt} \|v\|dM + a \frac{DO\|v\|}{Dt} dM + a\|v\| \frac{DOdM}{Dt} \right)$$

$$= \int_M a \frac{DO\|v\|}{Dt} dM.$$

As this is true for an arbitrary surface $M$, it leads to:

$$\frac{DO\|v\|}{Dt} = 0. \quad (6.17)$$

This means that $\frac{Dv}{Dt}$ needs to be adjusted so that the length of $v$ is conserved. Metaphorically, this means that $v$ is advected and rotated back onto the surface. An illustration of the advection of a vectorial quantity is given in Figure 6.2.

To calculate the rate of change of $v$ that compensates for length changes, the derivative is determined by the limit of the difference quotients. To this end, first-order Taylor expansion of $v$ with respect to $t$ is used to define $\hat{v} = v + \Delta t \nabla \hat{u}_n v$. Compensating for length changes, $v' := v(t + \Delta t)$ can be approximated by:

$$v' = \hat{v} - (\|\hat{v}\| - \|v\|) \frac{\hat{v}}{\|\hat{v}\|}$$

$$(6.18)$$
and the rate of change of $v$ is given by

\[
\frac{D_O v}{D t} = \lim_{\Delta t \to 0} \frac{v' - v}{\Delta t} = \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left( \Delta t \nabla \tilde{u}_n v - (\|\hat{\psi}\| - \|v\|) \frac{\hat{\psi}}{\|\hat{\psi}\|} \right)
\]

\[
= \lim_{\Delta t \to 0} \left( \nabla \tilde{u}_n v - \frac{1}{\Delta t} \left( 1 - \|v\| \right) \hat{\psi} \right)
\]

\[
= \lim_{\Delta t \to 0} \left( \nabla \tilde{u}_n v - \frac{\|\hat{\psi}\| - \|v\|}{\Delta t \|\hat{\psi}\|} \cdot v + \right.
\]

\[
\left( 1 - \frac{\|v\|}{\|\hat{\psi}\|} \right) \nabla \tilde{u}_n v
\]

\[
= \lim_{\Delta t \to 0} \left( \nabla \tilde{u}_n v - \frac{\|\hat{\psi}\|^2 - \|v\|^2}{\Delta t \|\hat{\psi}\| (\|\hat{\psi}\| + \|v\|)} \cdot v + \right.
\]

\[
\left( 1 - \frac{\|v\|}{\|\hat{\psi}\|} \right) \nabla \tilde{u}_n v
\]

\[
= \lim_{\Delta t \to 0} \left( \nabla \tilde{u}_n v - \frac{2\Delta t \cdot (\nabla \tilde{u}_n v)}{\Delta t \|\hat{\psi}\| (\|\hat{\psi}\| + \|v\|)} \cdot v + \right.
\]

\[
\left( 1 - \frac{\|v\|}{\|\hat{\psi}\|} \right) \nabla \tilde{u}_n v
\]

\[
= \lim_{\Delta t \to 0} \left( \nabla \tilde{u}_n v - \frac{2v \cdot (\nabla \tilde{u}_n v) + \Delta t \|\nabla \tilde{u}_n v\|^2}{\|\hat{\psi}\| (\|\hat{\psi}\| + \|v\|)} \cdot v + \right.
\]

\[
\left( 1 - \frac{\|v\|}{\|\hat{\psi}\|} \right) \nabla \tilde{u}_n v = \nabla \tilde{u}_n v - \frac{v \cdot (\nabla \tilde{u}_n v)}{\|v\|^2} \cdot v.
\]

After having discussed the different kinds of quantities, the evolution of the quantities caused by $O$ can be summarized by the combined material derivative $D_O O$ of $O$. If $a$ is an intensive property, it reads as

\[
\frac{D_O}{D t} O \left( \begin{array}{c} x \\ a \\ v \end{array} \right) = \left( \begin{array}{c} \tilde{u}_n \\ -a (\nabla \cdot (\tilde{u}_n)_{T,M}) \\ \nabla \tilde{u}_n v - v \cdot \nabla \tilde{u}_n v / \|v\|^2 \end{array} \right).
\]

Note that, if $a$ is an extensive property, it is just advected alongside $x$, i.e., $\frac{D_O}{D t} a = 0$. Next, it is discussed how friction between the outer process and inner dynamics is modeled.
6.2.3 Coupling

So far, it was assumed that the outer process influences the inner dynamics solely in normal direction. Next, it is discussed how the tangential part \( \mathbf{u}_t = \mathbf{u} - \mathbf{u}_n \) of the outer process will influence the inner dynamics. That implies that friction-less coupling is modeled, i.e., matter slides on the surface, and no adhesion is present. Adhesive forces are modeled to reduce the relative velocities \( \mathbf{v}^{\text{rel}} = \mathbf{v} - \mathbf{u}_t \) between the outer process and inner dynamics as

\[
\frac{D_c}{Dt} \mathbf{v} = -s_1 \mathbf{v}^{\text{rel}},
\]

where \( s_1 \) is a user-defined coefficient and \( \frac{D}{Dt} \) denotes the rate of change induced by coupling effects. Similar to Section 6.2.2, \( \mathbf{v} \) can be considered to be a small linear material line element exposed to the velocity \( \mathbf{u}_t \) and its rate of change would then read as \( (\nabla_{T,M} \mathbf{u}_t) \mathbf{v} \). Both views are combined and the total rate of change of \( \mathbf{v} \) caused by coupling effects is modeled as

\[
\frac{D_c}{Dt} \mathbf{v} = -s_1 \mathbf{v}^{\text{rel}} + s_2 (\nabla_{T,M} \mathbf{u}_t) \mathbf{v}.
\]

The transfer of intensive physical quantities (e.g., density) from the outer process to the inner dynamics is modeled by sinks and sources on the surface. Inspired by Fick’s laws of diffusion, this transfer is modeled in a way that the concentration \( a^{\text{out}} \) from the outer process and the concentration \( a \) of the inner dynamics will align. In other words, the concentration of the substance on the surface will be changed by

\[
\frac{D_c}{Dt} a = -s_3 a^{\text{rel}},
\]

to reduce the concentration difference \( a^{\text{rel}} = a - a^{\text{out}} \). The speed of the reduction is determined by the factor \( s_3 \). If only sources shall be modeled, the concentration difference is restricted to negative values, i.e., \( a^{\text{rel}} \leq 0 \). Sinks can be modeled when restricting \( a^{\text{rel}} \geq 0 \).

6.2.4 Modeling the Dynamics on the Surface

After discussing the surface evolution and coupling, it is discussed how dynamics on the surface can be modeled. To describe such a process a set of PDEs is used, e.g.,

\[
\frac{D_s}{Dt} \mathbf{v} = F_1 (t, a, \mathbf{v}, \partial_1 a, \partial_2 a, ..., \partial_1 \mathbf{v}, \partial_2 \mathbf{v}, ...),
\]

and

\[
\frac{D_s}{Dt} a = F_2 (t, a, \mathbf{v}, \partial_1 a, \partial_2 a, ..., \partial_1 \mathbf{v}, \partial_2 \mathbf{v}, ...),
\]
Figure 6.3: Different stages of the method for a two-phase input simulation. From left to right: Rare input simulation, the result including only mass transfer of the base process, the result including fluid flow on the surface, and high-quality rendering of the simulation result. The different stages show how new details on the surface can be added on top of an existing input simulation.

where the functions \( F_1 \) and \( F_2 \) characterize the modeled phenomena on the surface. The term \( \partial_i := \left\{ \frac{\partial^{\vert \alpha \vert}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \partial x_3^{\alpha_3}} \right\} \) \( \vert \alpha \vert = i \) denotes the set of all partial derivatives of order \( i \), with \( \alpha \) being a multi-index. For example, if fluid flow is modeled, \( F_1 \) describes the Navier-Stokes momentum equation and \( F_2 \) the continuity equation.

To solve Eqs. 6.24 and 6.25 on the surface, tangential deviations are allowed only. To this end, the PDEs are solved locally in the corresponding tangent spaces \( T_xM \). This implies that, instead of using the ordinary spatial derivatives, they are projected onto the surface, e.g., the operator \( \nabla \) is replaced by \( \nabla_{T_xM} \).

The governing equations for the overall model are then given by:

\[
\frac{D}{Dt} \mathbf{v} = \frac{D_O}{Dt} \mathbf{v} + \frac{D_c}{Dt} \mathbf{v} + \frac{D_s}{Dt} \mathbf{v} \quad \text{and} \quad \frac{D}{Dt} \mathbf{a} = \frac{D_O}{Dt} \mathbf{a} + \frac{D_c}{Dt} \mathbf{a} + \frac{D_s}{Dt} \mathbf{a}.
\]

While the outer process is ignored in Equation 6.24 and Equation 6.25, it is included in Equation 6.26 and Equation 6.27, i.e., they govern the overall dynamics on the surface.

Before providing examples in Section 6.4.1, technical details of the method are given next.

6.3 Method

The above described governing equations are solved on a narrow-band. To obtain a simulation on the surface, the CPM [108] is used. The resulting algorithm can be described in seven steps. A schematic overview of it is given
Figure 6.4: Method overview as a data flow chart. First, the input data is read (Step 1). Next, the narrow-band is generated, and the input data is transferred onto it (Step 2 and 3). Afterwards, the surface can be evolved (Step 4), and the inner dynamics are coupled to the outer process (Step 5). Lastly, the physical phenomenon on the surface is modeled by solving a set of PDEs (Step 6). Lastly, the fields are advected (Step 7).
in Figure 6.4. Step 1 is to read the input data, which can be, e.g., particle data resulting from an SPH simulation, or a key-frame animated surface.

Afterward, the input data is converted into a signed distance field (Step 2). For polygonal meshes, there exist a variety of methods (e.g., [117], [126], or [76]). When dealing with particle data, the particles are converted into an implicit surface first [130]. Based on the implicit surface, the distance field is created. Subsequently, a narrow band grid is created around the surface (Step 3). All needed scalar and vector fields are defined on this narrow-band grid, e.g., the outer velocity \( \mathbf{u} \), the inner velocity \( \mathbf{v} \), and a fluid density \( \rho \). By using the CPM, it can be ensured that the quantities do not change in normal direction of the surface. In the surface evolution step (Step 4), the quantities are adjusted to compensate for surface changes (as described in Section 6.2.2). Next, the outer process is coupled with the inner dynamics (Step 5). Details of this step can be found in Section 6.2.3. Having accounted for changes due to the outer process, the set of PDEs can be solved on the surface (Step 6). Last, the quantities are advected using a fixed time step size \( \Delta t \) (Step 7). Steps 3 to 7 are repeated until the desired state is reached.

Some steps are also given in Figure 6.3. The left image shows the input simulation consisting of two different phases. In the second image from left shows a result, where only the transfer of intensive quantities is modeled. The second image from right depicts a simulation result, with fluid flow modeled on the surface. The rightmost image shows a resulting high-quality rendering with a thin film shader, reproducing the effect of oil poured into water.

### 6.4 Results

To show the versatility of the model, different sets of PDEs are used to model physical phenomena on an evolving surface. They are discussed in Section 6.4.1. Subsequently, characteristics of model are evaluated in Section 6.4.2.

#### 6.4.1 Examples

Three physical phenomena are used to test the versatility of the model, namely fluid flow, buoyancy–induced flow, and reaction–diffusion. They are simulated on base processes with different characteristics, which vary from static (Figure 6.6) to hand animated meshes (Figure 6.5), or from coarse scale SPH-based fluid simulations (including sudden topology changes, Figure 6.7) to high-resolution multi-phase SPH simulations (Figure 6.3). In the following, the sets of equations are given, and the examples are shortly discussed.
Figure 6.5: Fluid flow on a rotating sphere resulting in a flow similar to the von Kármán vortex street. The outer velocity is coupled to the internal simulation using $s_1 = 0.01 \frac{1}{s}$ and $s_2 = -0.002$. Vortex confinement is used to add fine-scaled vortices.

Fluid Flow

One application of our model is the simulation of fluid flow on an evolving surface. To model viscous fluid flow on the surface the Navier-Stokes momentum equation is used:

$$\frac{Ds}{Dt} \mathbf{v} = -\frac{1}{\rho} \nabla_{T_M} p + \nu \nabla^2_{T_M} \mathbf{v} + \mathbf{a}_b.$$  \hspace{1cm} (6.28)

To obtain additional fine-scale details, vorticity confinement was added, as presented by Fedkiw et al. [42].

The evolution of density is modeled by using the general continuity equation. In its differential form, it states that the change $\frac{D\rho}{Dt}$ of density $\rho$ of the fluid can be defined as

$$\frac{Ds}{Dt}\rho = \sigma - \rho \left( \nabla_{T_M} \cdot \mathbf{v} \right),$$  \hspace{1cm} (6.29)

where $\sigma$ is the generation rate of the substance per unit volume, i.e., it is used to model sinks ($\sigma < 0$) or sources ($\sigma > 0$). Incompressible fluid flow on the surface is modeled and density changes only occur due to surface divergence. As a result, Equation 6.29 simplifies to a volume conservation law:

$$\nabla_{T_M} \cdot \mathbf{v} = \frac{\sigma}{\rho}.$$  \hspace{1cm} (6.30)
6.4. RESULTS

Figure 6.6: Thermal convection on a hemisphere is simulated using the Boussinesq approximation [28]. To induce heat convection some areas are heated, and others are cooled down, modeled as an outer temperature constraint and coupled using temperature transfer as defined in Equation 6.23. As a result, buoyancy-driven flow arises due to temperature differences. The temperature field is visualized, where the temperature rises from blue over white to red.

If no sinks or sources are present (i.e., $s_3 = 0$), it becomes $\nabla T \cdot v = 0$. Note that if the outer process is divergence-free, i.e., $\nabla \cdot u = 0$, an incompressible flow is obtained as $\frac{D\rho}{Dt} = 0$ holds.

Examples of fluid flow are given in Figures 6.3, 6.5, and 6.7b. In Figure 6.5, the surface rotates, and due to friction forces, the fluid on the surface starts to move. A secondary fluid simulation on the surface of an SPH-based fluid simulation is shown in Figures 6.3 and 6.7b.

Buoyancy-induced Flow

Buoyancy-driven flow (or natural convection) is not generated by external forces but arises due to internal variations in, e.g., density or temperature. It can be modeled using the so-called Boussinesq approximation [28]. The Boussinesq approximation assumes incompressible flow and that variations of density only occur due to temperature differences: $\rho = \rho_0 - \beta \rho_0 (T - T_0)$, where $\beta$ is the coefficient of thermal expansion, $T_0$ the reference temperature, and $\rho_0$ the reference density. An ideal gas is assumed and, therefore, $\beta = \frac{1}{T_0}$. If
gravity is present as the only body-force, Equation 6.28 becomes

$$\frac{Ds}{Dt} \mathbf{v} = -\frac{1}{\rho} \nabla T_x M p + \nu \nabla^2 T_x M \mathbf{v} + \left( 1 - \frac{T}{T_0} \right) \mathbf{g}, \quad (6.31)$$

with $\mathbf{g}$ being the gravitational constant. To model changes in the temperature field the convection-diffusion equation is used:

$$\frac{D_s}{Dt} T = \mu_d \nabla^2 T_x M, \quad (6.32)$$

where a constant diffusion coefficient $\mu_d$ is assumed.

An example simulation with buoyancy-driven flow is given in Figure 6.6. It shows a static hemisphere. Some areas on the surface are heated, others cooled by outer temperature constraints. We model temperature transfer according to Equation 6.23 and, therefore, the fluid on the surface changes its temperature. Natural convection arises due to temperature differences in the fluid.

**Reaction-diffusion**

Reaction-diffusion is commonly used to model chemical reactions of one or more substances (e.g., a substance is transformed into another due to chemical reactions) and the diffusion of the substance(s) in space. Figure 6.7a shows such a process. Two chemical substances diffuse and react with each other. In this example, a two-component reaction-diffusion process is given by

$$\frac{D_s}{Dt} f_1 = R_1(f_1, f_2) + \mu_{d_1} \nabla^2 T_x M f_1 \quad \text{and} \quad (6.33)$$

$$\frac{D_s}{Dt} f_2 = R_2(f_1, f_2) + \mu_{d_2} \nabla^2 T_x M f_2. \quad (6.34)$$
6.4. Results

Figure 6.8: Conservation test for the total amount of intensive quantities. On a sphere with a periodically oscillating radius, the mean concentration is measured over time. Left: The mean concentration plotted over time together with the analytic solution. Right: The mean absolute error for varying grid sizes.

The functions $R_1$ and $R_2$ are the reaction terms and characterize the system. There are many possibilities to define these reaction terms and depend on the specific reactions which shall be modeled. In this thesis, the model from Gray and Scott [52] is used to define how morphogens react:

$$R_1(f_1,f_2) = -f_1 f_2^2 + \beta (1 - f_1),$$

$$R_2(f_1,f_2) = f_1 f_2^2 - (\beta + \gamma) f_2.$$  

(6.35)  

(6.36)

To generate the example illustrated in Fig. 6.7a, the parameters were set to $\beta = 0.03$, $\gamma = 0.06$, $\mu_{d_1} = 0.1$, and $\mu_{d_2} = 0.1$.

6.4.2 Evaluation

The accuracy of the model with respect to the conservation of the total amount of intensive properties is investigated using a test-case where it is possible to calculate the ground-truth analytically. It is a sphere with a periodically oscillating radius. On the surface, a substance with homogeneous concentration is placed, and the mean concentration is measured over time. Note that in this case, no PDE is solved on the surface. Two tests were conducted in this scenario: first, the mean density over time was measured and, secondly, the aggregated mean absolute error for different grid-resolutions. On the left of Figure 6.8, the mean concentration over time is plotted. It can be observed that it is possible to conserve the overall amount of any intensive quantity when using the presented model. Only minor deviations from the analytic solution can be observed at the smallest radius (time $t = 2.6$ s). This error depends on the grid resolution used in the simulation. As observable in Figure 6.8 on the right, the mean absolute error reduces for finer grid resolutions.
Figure 6.9: Velocity drift on curved surfaces. As a result of the CPM, the substance experiences a drift toward the equator if no coupling is present (left). With coupling to the outer process, this effect is reduced and can completely vanish (center) if the no-slip condition \( s_1 = \frac{1}{\Delta t} \) and \( s_2 = 0 \) is used. For the friction coefficients \( s_1 = 0 \) and \( s_2 = 20 \), even an inverted drift can be observed (right).

Coupling of the inner dynamics to the outer process is modeled based on adhesive forces (Equation 6.21) and the Jacobian of the tangential velocity. The user-specified parameter \( s_1 \) is defined to control the strength of the coupling. For \( s_1 = 0 \), the substance slips over the surface, and no friction is employed. In contrast, for \( s_1 = \frac{1}{\Delta t} \), the no-slip boundary condition is obtained as the substance will have the same velocity as the one resulting from the outer process. Therefore, it is recommended restricting \( s_1 \) to \( s_1 \in [0, \frac{1}{\Delta t}] \). Using the Jacobian of the tangential velocity, the coupling is controlled by scaling the constant \( s_2 \).

Another interesting fact can be observed. On curved surfaces, a substance experiences a drift. For example, in the rotating sphere example (Figure 6.5), the fluid drifts from the poles to the equator. To investigate this fact, the rotating sphere example is used, and the system on the surface is initialized with the outer velocity. Again, no PDEs are solved on the surface. Only coupling and advection are present. To investigate the evolution of the substances' velocity, tracer particles are placed, and their pathlines are visualized (Figure 6.9). If no coupling is present \( (s_1 = 0, s_2 = 0) \), the matter on the surface drifts toward the equator, resulting in a velocity field directing toward the equator of the sphere (Figure 6.9, left). The drift effect could be interpreted as a centripetal force. Nonetheless, it is a direct consequence of the CPM. Due to the advection, the substance slightly drifts away from the surface in tangential direction. In the projection step, it is projected back onto the surface. As it has a certain distance to the surface, a natural consequence of the curved surface is that it will be projected to a point closer to the equator. With coupling present, this effect is reduced. Setting \( s_1 = \frac{1}{\Delta t} \) and \( s_2 = 0 \), the drift is completely vanished as the no-slip condition is obtained and the velocity of the substance equals the
outer velocity (Figure 6.9, center). For \( s_1 = 0 \) and \( s_2 = 20 \), an inverted drift can be observed (Figure 6.9, right). The substance now experiences a slight drift away from the equator toward the poles.

## 6.5 Discussion and Future Work

With the presented model, a variety of secondary effects can be modeled on top of an evolving surface. As discussed, any kind of phenomena can be modeled on the surface, as long as it can be defined with a set of PDEs. Testing further sets of PDEs could be a future research direction.

Using the CPM, the model can deal with surfaces generated by any kind of time dependent process. Nonetheless, only free surface flow was considered, and no collision objects were included. A next step could be the implementation of different boundary conditions. For the simulation of secondary effects on surfaces resulting from SPH-based fluid animations, this would help to obtain more plausible movements around collision objects.

In the current version, the secondary simulation does not alter the surface itself. By now, concentrations on the surface could be used to generate a displacement field, and the results can be rendered accordingly. An interesting future research direction could be the investigation on how to change the surface topology with the simulation on it. Combined with a two-way coupling altering the outer velocity field in a narrow-band, it could be used to model wind-induced swirl on the surface of an animated liquid.

The conservation laws described in Section 6.2 are motivated by physical laws. As observable in the evaluation, the conservation of the amount of an intensive property not only accounts for changes in the overall surface area but also local changes. A possible direction would be to test the overall influence in combination with, e.g., fluid flow on converging and diverging regions on a surface. A quick test for absolute momentum conservation did not reveal any noticeable differences in this test case. In that case, \( \frac{\partial}{\partial t} \mathbf{v} = \nabla \hat{u}_n \mathbf{v} \) was used instead of \( \frac{\partial}{\partial t} \mathbf{v} = \nabla \hat{u}_n \mathbf{v} - \mathbf{v} \cdot \nabla \hat{u}_n \mathbf{v} \frac{\mathbf{v}}{||\mathbf{v}||^2} \). This test case was not discussed, as there could no differences be observed. Nonetheless, if the surface rotates in normal direction instead of tangential, it may make a difference. A more in-depth study of the differences would also be a possible research direction.

## 6.6 Summary

In this chapter, a possible answer to Research Question 4 was given by discussing a method to solve PDEs on evolving surfaces. It can be used to model
secondary effects on animated surfaces, e.g., resulting from an SPH-based fluid animation. The model is derived, using physical conservation laws. The coupling of the surface dynamics to the outer process is also based on physical principles. After discussing previous work and the simulation model, a quick overview of the actual method was given. Different phenomena were tested on top of various types of base processes to evaluate the model. Additionally, the quality of the model concerning the conservation of intensive properties was evaluated, and its most important characteristics were discussed.
In this thesis, computational methods were presented to improve SPH-based fluid animations. They mainly concentrate on the spatial and temporal discretization process. The presented methods were developed to meet the special requirements present in computer graphics applications focusing on the following three aspects:

- **Efficiency**
  One key aspect of computer graphics applications is the efficiency of the simulation method, not only for real-time applications but also for offline simulations. It was of central concern when investigating Research Questions 1, 2, and 4.

- **Accuracy**
  Improving the accuracy of a discretization method helps improve different issues. For example, increasing the accuracy may improve the stability of the simulation method, which, in turn, helps improve efficiency. In addition, an accurate representation of the physical phenomena always helps obtain a realistic result. Particular focus was paid to this aspect in Research Questions 1 and 3.

- **Visual Quality**
  Producing high quality and realistic results is always important for computer graphics applications. Therefore, it was always of central concern when investigating the research questions. A special focus was given to visual quality in Research Question 4.

In this final chapter, the contributions are summarized (Section 7.1) and discussed (Section 7.2). The discussion focuses on the formulated research questions. Finally, possible future research directions are given in Section 7.3.
7.1 Summary

This work comprises computational methods improving SPH-based fluid animations. In particular, the presented research focuses on enhancing the numerical simulation aspect. Different methods were developed to improve the efficiency, accuracy, and visual quality of the simulation and provide answers to Research Questions 1–4. Before discussing these answers, the presented methods are summarized.

Visual Debugging of SPH Simulations. Chapter 3 presented a technique that supports the development of computational methods for SPH-based fluid animation. It is an approach for visual analysis and debugging of SPH simulations. The general idea is to use visual analysis tools from the field of information visualization to inspect the simulation result. To this end, the most important requirements were identified to provide visual assistance. Abstract views of the data are enabled using scatter plots and PCPs. They are complemented with a spatial view, which includes coloring of particles according to user-definable attribute values. Interactive exploration of the dataset was enabled, and the individual views were coupled using brushing and linking. Case studies were conducted to evaluate the environment, showing how the application can support challenges arising during the development process. In particular, these are debugging of implementation errors, investigating attribute correlations, identifying regions of interest, and comparing the characteristics of competing methods.

Fully Asynchronous SPH. An asynchronous time stepping model was presented in Chapter 4. In contrast to previous methods, no synchronization barriers are introduced, and the time step restriction is handled for each particle individually. As there is no global simulation state, the neighborhood needs to be synchronized locally. An export barrier is introduced to enable the visualization of the simulation states. It is discussed how the simulation can be evolved globally. To this end, a particle processing scheme is introduced, using a priority queue. This queue is sorted using the individual particle time to ensure that the particle lagging most behind is processed next. Additionally, a parallelization scheme was discussed, including a thread synchronization method to prevent race conditions. A simple but effective load balancing scheme was also presented. The method was evaluated with respect to performance, showing that all previous time adaptive methods were outperformed. Characteristics of the method (especially in combination with the used boundary handling model) were investigated by visual comparison, and the findings were quantified with a quantitative velocity study.
7.2 Discussion

**Consistent Shepard Interpolation.** In Chapter 5, a kernel correction scheme was discussed that reduces errors arising in the spatial discretization process and, therefore, increases its accuracy. It is based on the Shepard kernel correction scheme but removes inconsistencies arising when applying it to SPH. An iterative solver to compute the correction factors using the corrected densities was presented, achieving constant completeness. To this end, a linear formulation of the problem was used, and a stable solver was developed using the power method. Its convergence is proved in Appendix B. In addition, a kernel gradient correction scheme was presented that satisfies the constant completeness reproducing condition. It was discussed how the boundary handling model is adjusted to be compatible with the correction scheme. The method was evaluated on a variety of scenarios with different characteristics. The smoothness of the density field is inspected visually, and the observations were substantiated by measuring the smoothness of the density field locally and globally. For this purpose, a seminorm was defined that measures the smoothness of the density field locally. Moreover, two different norms were defined to provide a global measurement of the density field smoothness.

**Solving PDEs on Evolving Surfaces.** A method to describe secondary effects on an animated surface was presented in Chapter 6. A generic model was developed, following physical laws. It enables the simulation of arbitrary real-world phenomena as long as they are describable by a set of PDEs. The model allows for one-way coupling of the outer process to modify the simulation on the surface. A short overview of the method was given, describing the technical realization. Three different phenomena were used on a variety of animated surfaces to show the versatility of the model. The surfaces were either key-frame animated or resulted from an SPH simulation. Finally, the accuracy of the method and conspicuous characteristics were discussed.

7.2 Discussion

In this thesis, computational methods were presented that improve the numerical simulation aspect of SPH-based fluid animation. To investigate different aspects of the improvement process, research questions were formulated. The presented methods were designed to provide possible answers to them and improving previously available techniques. In the following, the presented methods are discussed in the context of Research Questions 1-4 and an overall conclusion is drawn.
Research Question 1

How to support the development process of computational methods for SPH-based fluid simulation?

This research question concerns a general problem that applies to the development of all computational methods for SPH-based fluid animation. Using only conventional software debugging is generally not sufficient when designing such new techniques. In Chapter 3, a visual analysis tool was presented approaching this problem. It complements classical debugging and is tailored for the development of computational methods for SPH-based fluid animation. In the development process, recurring evaluation is very important. Therefore, special attention was given to the efficiency aspect of the views, to enable interactive exploration of the data. Allowing for the investigation of parameter combinations, and comparison of competing models supports the improvement process. Spatial visualization of the dataset facilitates the visual investigation of the fluid’s behavior and the quality of the simulation; therefore, it helps the development process. Simultaneously, the analysis of parameters and their relationships using scatter plots, and PCPs helps investigate the accuracy of the model of interest. The presented research shows that typical questions arising in the development process can be answered using the presented visual debugging environment. Additionally, it is possible to identify a broad range of issues in a fast and convenient way.

Nonetheless, the presented environment can support the process only to some extent. A visual debugging environment cannot cover all aspects. For example, detecting conceptual errors in the mathematical model is only possible if they result in conspicuous attribute correlations. In addition, model analysis concerning consistency or – in case of an iterative solver – convergence is left to the developer and conventional debugging applications.

Research Question 2

How to improve the efficiency of SPH-based fluid animation with adaptive time discretization?

With this question, efficiency improvements of SPH-based fluid animation are investigated by locally varying time step sizes. One possible solution is proposed in Chapter 4. With spatially varying time stepping, the efficiency of the time discretization is improved. The maximum possible time step size per particle is achieved by introducing an individual timeline for each particle. Therefore, all global restrictions are removed, which could possibly reduce the individual time step size. In essence, the simulation is fully asynchronous, and computational resources are focused on important regions. It is shown that the presented method is more efficient than previous approaches.
Nevertheless, there are some restrictions. It is difficult to incorporate modern iterative solvers, which, e.g., effectively handle the incompressible constraint, as only local dependencies are allowed. In its current form, the presented model only allows for using the EOS to compute the pressure. In scenarios that induce a small positive density error, non-iterative solvers that use an EOS are very efficient. Nonetheless, in typical scenarios, a large stiffness constant is needed to overcome the compressibility issue and to ensure realistic looking animations. In addition, there exist many ways to improve the efficiency of SPH-based fluid animations, tailored for different aspects and difficult to compare. For example, improving the accuracy of the temporal discretization (e.g., by using higher-order time integration schemes) may improve the stability of the solver and, hence, the efficiency of the simulation as larger time step sizes may be possible.

**Research Question 3**

*How to improve the accuracy of the spatial discretization process?*

Employing numerical methods to animate a fluid body using SPH requires a spatially discrete representation of the continuous fluid body. The stated question addresses the errors introduced in the spatial discretization process. These errors manifest in a non-exact representation of the fluid quantities. To measure them, consistency and completeness are often analyzed. In Chapter 5, an answer is discussed, resulting in constant completeness (i.e., constant functions are exactly represented). A typical approach for constant completeness is Shepard interpolation. Nonetheless, when applied to SPH, this method yields an inconsistent formulation. These inconsistencies result in an error-prone approximation. The discussed method resolves the inconsistencies using an implicit formulation and accounting for corrections in the neighborhood. The result is a smooth density field. It is a generalization of the Shepard kernel correction. In essence, it respects that the density (which is needed for the discrete approximation of the fluid body) undergoes a correction as well. By using a stable and efficient iterative solver to compute the correct correction factors the introduced overhead is kept very small.

The presented solution is consistent with the SPH discretization of the fluid body, but only obtains constant completeness. However, the vector and scalar fields representing the fluid quantities are typically not constant. To represent them exactly, higher-order correction schemes would be needed, such as the reproducing kernel particle method (RKPM). Using the moving least square reproducing kernel particle method (MLS RKPM) presented by Liu et al. [84], a polynomial function of arbitrary degree could be represented exactly. Nonetheless, higher orders come with non-neglectable computational costs, and a trade-off between accuracy and efficiency has to be chosen.
Research Question 4

How to model fine-scaled effects on the surface of an SPH-based fluid animation?

For computer graphics, the surface of an animated fluid body is of particular interest. It is typically used to visualize the fluid body in high quality and realistic renderings. The more fine-scaled details are observable on the surface, the more realistic is its appearance. This question was investigated in Chapter 6. The idea is to enrich the surface appearance by modeling secondary effects on the surface. Using the CPM and performing the simulation only on a narrow band around the surface creates a very efficient system to model fine details, and effects with very high spatial resolution on the surface. The generalized model provides much flexibility, i.e., a variety of secondary effects can be modeled. One-way coupling was chosen deliberately, allowing for post-processing of coarse-scale base simulations. With the presented solution, many effects could be modeled that increase the visual quality, and level of realism of SPH-based fluid animation.

One issue of the presented method is that it does not alter the base surface in any way as only one-way coupling is employed. That restricts the variety of effects that can be modeled, to ones that do not have any depth. For example, enhancing turbulence of a fluid animated with SPH, as presented by Wang et al. [124], is only possible to some extent with the current formulation of the model. Performing a secondary fluid simulation on the surface (driven by an outer fluid simulation), and using the result to create a displacement field for the surface would create fine-scale details on the surface itself.

Overall Discussion

The presented computational methods were developed to investigate Research Questions 1–4, which, in turn, were designed to pursue the overall goal:

Improving the numerical simulation aspect of an SPH-based fluid animation.

Recurring evaluation of a computational method is an important step in the process of improving the numerical simulation aspect. The presented visual debugging environment was designed to support this development and complements classical debugging. By allowing for deeper insights into the simulation methods it helps improve the effectiveness of the development process. As mentioned, the research goal was pursued while accounting for special requirements present in computer graphics applications. Therefore, special attention was given to the efficiency and accuracy of the fluid solver and the visual quality of its outcome. One key aspect of the aforementioned
accuracy and efficiency of the numerical simulation is the spatial and temporal discretization of the physical model. To improve the accuracy of the simulation or, more precisely, the spatial discretization of the fluid body, Research Questions 3 was discussed. Using the method developed in this context, fluid properties are represented more accurately. An efficient fluid solver employing spatio-temporal adaptive discretization of the body was discussed in the context of Research Questions 2. In the field of computer graphics, efficient fluid solvers are very important as they enable the user to perform more iterations of the same scenario in less time. Additionally, it allows to increase the fluids’ resolution resulting in a more accurate representation and enables the modeling of finer details; therefore, it increases the degree of realism and the visual quality of the simulation. A very efficient way to improve the visual quality of the simulation was discussed in Research Question 4. Overall, the developed methods help advance SPH-based fluid solver and improve the overall process of creating an SPH-based fluid animation.

Nevertheless, the formulated research questions are only one out of many possible ways to pursue the overall research goal. Additionally, the presented methods are also only one answer to the research question itself, and there exist numerous other techniques that help advance the animation process and its outcomes. The presented research mostly focuses on the numerical simulation part, while improving the efficiency, accuracy, and visual quality of SPH-based fluid animations. Improving the visual quality and the degree of realism could also be achieved by enhancing the physical model, e.g., add spray, foam, and bubbles on top of the base simulation [121]. Adding such effects can also be achieved very efficiently by extending the visualization aspect [8].

One major drawback of the presented methods is that some are not entirely compatible with each other. It is, e.g., not possible to combine ATS (Chapter 4) with consistent Shepard interpolation (Chapter 5) as ATS only allows for local constraints; therefore, it is not possible to combine ATS with iterative solvers. Moreover, the presented methods cannot be implemented in a fully modular way. For example, the neighborhood search algorithm needed adjustments to work with ATS. Similarly the boundary model needed to be altered with consistent Shepard interpolation.

7.3 Future Research Directions

In this thesis, a particular focus is given to the discretization process of the underlying physical and numerical model that is used in the SPH-based fluid animation. Improving – both the spatial and temporal – discretization of the fluid body has great potential for future research.
In Chapter 4, time stepping models were discussed that improve the efficiency of the simulation process using adaptive time step sizes. The efficiency is improved at the cost of accuracy. A more in-depth accuracy analysis could be a direction for future work. Also, a higher-order time integration scheme, such as the classic Runge–Kutta method, may help improve the accuracy again. Additionally, it may help increase the stability of the simulation and, therefore, allow for larger coefficients of the CFL-condition, which would again improve the efficiency. Another direction for future research would be combining model for spatial-adaptive resolution (e.g., Horvath and Solenthaler [61]) with spatially varying time-adaptive ones like ITS or ATS. In areas with homogeneous flow, it could not only be possible to use larger particle sizes but also larger time steps then. It would also be interesting to model complex effects only close to the surface in a high resolution and perform a coarse-scale simulation with solely basic effects, inside the fluid body.

How to model effects on the surface was discussed in Chapter 6. Performing such a simulation on a high-resolution narrow band of a spatial adaptive SPH-simulation as a two way coupled system would also be interesting. In this way, topology changes of the surface could be modeled as well as two-way coupling could be used to model effects like additional splashes. Incorporating the ideas of the modeling of surface effects into the simulation itself may help to create more realistic looking simulations even at a coarse base spatial resolution.

Using consistent Shepard interpolation to improve the accuracy of the simulation process showed promising results. However, only constant completeness can be achieved using this correction scheme. There exist many methods to obtain higher-order completeness. One class is RKPM, such as MLS RKPM [84]. This method can achieve an arbitrary degree of completeness. Nonetheless, it has the same issues as classical Shepard interpolation and ends up with an inconsistent scheme. Applying the consistency ideas to RKPM would be another topic for future work.

In general, improving the flexibility of the presented methods would be an interesting topic for future research. For example, to implement the fully asynchronous time integration presented in Chapter 4, the neighborhood search algorithm needed to be adjusted. With consistent Shepard (Chapter 5), the boundary model needed adjustments as densities had to be computed on the boundary objects. Thinking of other ways to combine the presented computational methods with others (e.g., neighborhood search or boundary handling) without the need to alter the other ones would improve their potential.

As mentioned, the methods were developed focusing on efficiency, accuracy, and visual quality of SPH-based fluid animations. These are key aspects of computational methods in the field of computer graphics. Nonetheless, there are also other important aspects that determine the usefulness and applica-
bility of a method, such as flexibility or stability. Especially when it shall be integrated into a simulation tool or large software packages. Flexibility is crucial to enable the interaction between different methods. It also helps enable the artistic controllability of the simulation, which is important especially in a visual effects application scenario. While the stability aspect was partly covered in Chapter 5, it was not explicitly discussed. Focusing on these aspects might be a good next step for future research.

In summary, there are further challenging research directions for the efficient and versatile creation of SPH-based fluid animations. Artistic controllability could be an interesting topic, e.g., through force fields. As machine learning shows promising results in other areas of fluid mechanics [30], it would also be an interesting research direction to advance SPH-based fluid animations.
Co-authored references


Supplemental Images for Visual Comparison – ATS

In Chapter 4, an asynchronous time integration model (ATS) is presented to simulate fluids using a non-iterative EOS-based SPH solver. Additional images for comparison to previous methods shall complement the Visual Comparison Study. Results are presented for fixed time stepping (FTS), globally adaptive time stepping (GTS) [39], individual time stepping (ITS) [13], and ATS.

Figure A.1: Color map for the examples in Appendix A.

Throughout Appendix A, velocities are color-coded using the color map in Figure A.1.
A.1 Corner dam break

Figure A.2: Side-by-side comparison of the different time stepping methods for the corner dam break at different simulation times: there are only slight differences in the animations and ATS achieves a speedup of a factor of 6.8 in parallel, and a factor of 7.3 in serial execution. Parameters for this scenario are given in Chapter 4.
A.2 Radial flow

A.2.1 Side view

Figure A.3: Side-by-side comparison of the different time stepping methods in the radial flow scenario at different simulation times: the overall flow looks similar for all tested integration methods. Using GTS, minor instabilities occur when particles hit the collision object, which can be observed at \( t_e = 3.5 \) s. No explicit shock handling was applied and the maximum possible values for the factors \( \lambda_v \) and \( \lambda_a \) were used. There are also slight differences in the velocity field behind the collision object. At \( t_e = 3.5 \) s, it is observable that ATS damps velocities at fluid interaction with the collision object. A top view of this scenario is provided in Figure A.4 focusing on this difference.
A.2.2 Top view

Figure A.4: Top view of the radial flow scenario: only slight differences can be observed in the radial flow behind the collision object. Using ATS, velocities are slightly damped compared to FTS at fluid interaction with the collision object. Using GTS or ITS, small differences are observed as well compared to FTS, especially some instabilities resulting in isolated particles.
A.3 Fountain

Figure A.5: Side-by-side comparison of the different methods in the fountain scenario at different simulation times: noticeable differences can be observed in the overall shape of the fluid fountain. As mentioned in Chapter 4, ATS damps fluid velocities at interaction with collision objects. The velocities of the particles dropping on the top of the cone are damped with ATS and the particles pouring down the fountain form a different shape compared to FTS, GTS, and ITS. Again, instabilities can be observed using GTS. This results in many spraying particles, e.g. at $t_e = 2s$. Using ITS, slight instabilities can be observed, resulting in a large horizontal movement of the particles on top of the fountain. In the experiments, values twice as large as proposed by Ban et al. [13] were used for $\lambda_v$ and $\lambda_a$. 
In Chapter 5, a technique to reduce errors induced by the discretization process to achieve constant (or 0th order) completeness was presented. In the following, additional information is provided for the presented examples.

**Figure B.1**: Linear color map used in the examples throughout Appendix B.

**Figure B.2**: Pictograms to understand the orientation of the camera in the test scenes.
B.1 Fluid Block

B.1.1 WCSPH

\[ t = 0.0 \text{s} \]
\[ t = 0.2 \text{s} \]
\[ t = 0.4 \text{s} \]
\[ t = 0.6 \text{s} \]
\[ t = 0.8 \text{s} \]

No correction  Classical Shepard  Consistent Shepard

Figure B.3: Selected renderings from the fluid block scenario simulated with WCSPH. From left to right: Simulation conducted without any kernel correction, with classical Shepard correction, and with the presented consistent Shepard correction. Particles are colored with respect to density, where the lowest value (blue) corresponds to \( 997.5 \text{kg/m}^3 \), and the highest value (red) to \( 1002.5 \text{kg/m}^3 \). A significantly improved smoothness of the density field can be observed. Especially at time \( t = 0.2 \text{s} \), it is noticeable that with consistent Shepard one smooth wave front appears (second row, right), whereas both other simulations result in a noisy density field.
B.1.2 DFSPH

Figure B.4: Side-by-side comparison of the different methods in the fluid block scenario simulated with DFSPH. From left to right: Simulation conducted without any kernel correction, with classical Shepard, and with consistent Shepard correction. Similar to Figure B.3, particles are colored with respect to density. The lowest value (blue) again corresponds to $997.5 \frac{\text{kg}}{\text{m}^3}$ and the highest value (red) to $1002.5 \frac{\text{kg}}{\text{m}^3}$. It is recognizable that the improvements are less significant employing consistent Shepard, but still a smoother density field is achieved in comparison to classical Shepard.
B.1.3 Quantitative Evaluation

Figure B.5: Density variance measurements of the fluid block scenario. The norm \( \| \sigma(\bar{\rho}) \|_{\sigma,1} \) representing the mean density variance is shown on the left and the max norm \( \| \sigma(\bar{\rho}) \|_{\sigma,\infty} \) on the right. In the upper row, the measurements of simulations conducted with WCSPH are shown, and in the lower row, the ones with DFSPH. Lower density variance corresponds to a smoother density distribution. In all cases, it is observable that the smoothness of the density field is significantly improved using consistent Shepard. Considering the maximum density variance, it is observable that, in simulation performed with classical Shepard correction, the local density variance is even increased at some points in time. This could be interpreted that the classical Shepard correction introduces noise at some points in space and time.
B.2 Fluid Pillar

B.2.1 WCSPH and DFSPH

Figure B.6: Selected snapshots from the fluid pillar scenario conducted with WCSPH and DFSPH. The simulations were conducted without kernel corrections (left), with classical Shepard correction (middle), and with consistent Shepard correction (right). The particles are color-coded with respect to density: the lowest value (blue) was set to $998 \frac{kg}{m^3}$ and the highest value (red) to $1002 \frac{kg}{m^3}$ for DFSPH (right column). For the simulations conducted with WCSPH, the value range is adjusted so the lowest value corresponds to $970 \frac{kg}{m^3}$ and the highest to $1030 \frac{kg}{m^3}$. Again, the same behavior as in the fluid block scenario is observable (see Figure B.3 and Figure B.4). In this case, classical Shepard kernel introduces a coarse-scale noise into the density field for both kind of simulations (second row left and right as well as third row on the right).
B.2.2 Quantitative Evaluation

Figure B.7: Density variance measurements of the fluid block scenario. Similar to Figure B.5, the norm $\|\sigma(\bar{\rho})\|_{\sigma,1}$ shown on the left represents the mean density variance and the max norm $\|\sigma(\bar{\rho})\|_{\sigma,\infty}$ is shown on the right. The diagrams in the upper row contain the measurements of simulations conducted with WCSPH and the lower row the ones with DFSPH. Again the same overall behavior is observable. Additionally, an interesting effect can be recognized: when the fluid is repelled after the moment of highest compression (at $t \approx 1.75$ s for WCSPH and $t \approx 0.75$ s for DFSPH), the mean density variance increases for uncorrected and classical Shepard correction, whereas it stays at lowest levels when using consistent Shepard. Considering the maximum density variance, this effect is even more significant.
B.3 Corner Dam Break

B.3.1 WCSPH

Figure B.8: Selected renderings of the corner dam break scenario conducted with WCSPH. From left to right: no kernel correction, classical, and consistent Shepard correction. The particles are color-coded with respect to density, where the lowest value (blue) corresponds to $997 \text{ kg/m}^3$ and the highest value (red) to $1003 \text{ kg/m}^3$. To look into the fluid body, a plane is placed diagonally through the simulation domain and the particles on one side of it are hidden. The same behavior is observed previously. Additionally, it is observable that less splashes of single particles occur (second last and last row) when using consistent Shepard correction.
Figure B.9: Selected renderings of the corner dam break scenario conducted with WCSPH. From left to right: no kernel correction, classical, and consistent Shepard correction. The particles are color-coded with respect to local density variance, where the lowest value (blue) corresponds to 0 and the highest value (red) to 3000. Again, it can be observed that using consistent Shepard correction the noise in the density field is reduced, which is equivalent to reducing the local density variance. The cut through the fluid shows that not only the smoothness of the density field is improved at the surface but also beneath.
B.3.2 DFSPH

Figure B.10: Selected renderings of the corner dam break scenario conducted with DFSPH. From left to right: no kernel correction, classical, and consistent Shepard correction. The particles are color-coded wrt. density: the lowest value (blue) corresponds to 997 kg m$^{-3}$ and the highest (red) to 1003 kg m$^{-3}$. Even when simulating with DFSPH, some noise in the density field can be observed (left column). It is slightly reduced with classical Shepard correction but fine-scale density noise is still present (see, e.g., $t_e = 0.6s$). Even though consistent Shepard correction does not completely remove the noise as with WCSPH, it still improves the smoothness of the density field.
Figure B.11: Selected renderings of the corner dam break scenario conducted with DFSPH. From left to right: no kernel correction, classical, and consistent Shepard correction. The particles are color-coded with respect to local density variance, where the lowest value (blue) corresponds to 0 and the highest value (red) to 3000. This shows that still the smoothness of the density field is improved significantly, as the local noise is removed almost completely.
B.3.3 Quantitative Measurements

Figure B.12: Density variance measurements of the corner dam break scenario to quantify the observations. Again, the norm $\| \sigma(\bar{\rho}) \|_{\sigma, 1}$ shown on the left represents the mean density variance and the max norm is shown on the right $\| \sigma(\bar{\rho}) \|_{\sigma, \infty}$. The upper row contains the measurements of simulations conducted with WCSPH and the lower row the ones with DFSPH. The same overall behavior as in the other scenarios can be observed.
B.4 Fountain

Figure B.13: Selected renderings of the fountain scenario, which was conducted with DFSPH. Particles are again color coded with respect to density, where the lowest value (blue) corresponds to $995 \text{ kg m}^{-3}$ and the highest (red) to $1005 \text{ kg m}^{-3}$. Again, a smooth density field is obtained. This example was used to stress test the presented algorithm with regard to convergence of the power method. Particles are continuously added and initialized with $c_i = 1$. This could affect the convergence rate of the algorithm as the temporal coherence of the correction factors is destroyed. Yet, it could be observed that the algorithm is very robust and only needed in average 2.12 iterations and never more than 4.
B.4.1 Quantitative Measurements

Figure B.14: Measurements of the mean and max local density variance of the fountain scenario. Notice again the low values using consistent Shepard correction.

B.5 Convergence Proof

In Section 5.3, a fixed-point problem was derived in a linear formulation $Ac = c$, where $A \in \mathbb{R}^{N \times N}$ and $c \in \mathbb{R}^N$ with

$$a_{ij} = \frac{m_i W_{ij}}{\sum_{k \in N_j} m_k W_{jk}}.$$  

From a mathematical point of view, the formulation corresponds to an eigenvalue equation to the eigenvalue 1. Therefore, it is reasonable to analyze the structure of the matrix $A$ to prove existence and uniqueness (up to scaling) of a corresponding eigenvector. These results will be the key for the convergence proof of the power method.

For the derivation, it is assumed that the masses are equal, i.e., $m = m_i$ for all $i = 1, \ldots, N$. Before starting the investigation of the matrix $A$, the concept of connected fluid bodies with regard to Lagrangian approaches is introduced.

**Definition B.1** – A fluid body with positions $x_1, \ldots, x_N \in \mathbb{R}^3$ (or $\mathbb{R}^2$) is called **connected**, if and only if for all pairs $x_i, x_j$ there exists a $L \in \mathbb{N}$ and $x_{k_1}, \ldots, x_{k_L}$ such that $x_i \in B_{k_1}^c$, $x_j \in B_{k_L}^c$, and $x_{k_l} \in B_{k_{l+1}}^c$ for all $l = 1, \ldots, L - 1$, where $B_{k}^c = \{x \in \mathbb{R}^3 | ||x - x_k|| < h\}$ is the open ball around $x_k$ with radius $h$. This means that $x_{k_1}$ and $x_{k_L}$ are connected by a path $x_{k_1}, \ldots, x_{k_L}$ of particles that are chained by overlapping smoothing kernels.
The next lemma proves algebraic properties of the matrix $A$ and links the previous definition of connection to the matrix $A$.

**Lemma B.1** – The matrix $A$ has the following properties:

1. It is a left stochastic matrix, i.e.,
   - $0 \leq a_{ij} \leq 1$ for $i, j = 1, \ldots, N$,
   - $\sum_{i=1}^{N} a_{ij} = 1$ for $j = 1, \ldots, N$.

2. The diagonal is positive, i.e., $a_{ii} > 0$ for $i = 1, \ldots, N$.

3. It is irreducible if and only if the fluid body is connected.

**Proof.** If $j \notin N_i$, then $W_{ij} = 0$ and, hence, $a_{ij} = 0$. If $j \in N_i$, it follows that

$$
0 \leq \frac{m_j W_{ij}}{\sum_{k \in N_j} m_k W_{jk}} \leq \frac{m_j W_{ij}}{m_j W_{ji}} = 1,
$$

as $m_j = m > 0$ and $W_{ij} \geq 0$ for $i, j = 1, \ldots, N$. Furthermore, summing up the columns of the matrix results in

$$
\sum_{i=1}^{N} a_{ij} = \sum_{i=1}^{N} \frac{m_j W_{ij}}{\sum_{k \in N_j} m_k W_{jk}} = \frac{\sum_{i \in N_j} m_j W_{ij}}{\sum_{k \in N_j} m_k W_{jk}} = 1.
$$

The second statement follows from the properties of a smoothing kernel. To prove the last statement the fluid body is considered to be an undirected graph $G$, where particle positions are vertices and (open) neighborhood relations correspond to edges. Denoting the associated adjacency matrix by $\tilde{A}$, the following relationship is obtained:

$$
\tilde{a}_{ij} = \begin{cases} 
1 & \text{if } a_{ij} > 0 \text{ and } i \neq j, \\
0 & \text{else.}
\end{cases}
$$

Since the irreducibility of non-negative matrices is equivalent to $(I + A)^{N-1} > 0$ [60, Theorem 6.2.24], the equivalence

$$
A \text{ irreducible } \iff \tilde{A} \text{ irreducible}
$$

follows. A well-known fact from graph theory is that the adjacency matrix $\tilde{A}$ is irreducible if and only if the corresponding graph $G$ is connected. By construction, the graph is connected if and only if the fluid body is connected. \(\square\)

The following lemma shows the spectral-theoretic properties of the matrix $A$, characterizing the existence and uniqueness of an eigenvector to the eigenvalue $1$ and the location of other eigenvalues.
Lemma B.2 – For matrix $A$ with eigenvector $\lambda$, the following properties hold:

1. The value 1 is an eigenvalue.
2. If $\lambda \neq 1$, then $|\lambda| < 1$.
3. If the matrix $A$ is irreducible, then the eigenvalue $\lambda = 1$ is simple and the corresponding eigenvector can be chosen to be positive, i.e., every entry of the vector is positive.

Proof. The first statement is a direct consequence of Lemma B.1 and the second follows from the fact that $A$ is a left stochastic matrix with a positive diagonal \cite{45, Theorem 2.6}. The last statement is a well-known result from the Perron-Frobenius theory for non-negative irreducible matrices \cite{46}. \hfill $\square$

The next theorem is the final result proving convergence of the power method.

Theorem B.1 – For the matrix $A$, the following statements hold:

1. The power method converges for every initial value, i.e., the limit $c^* = \lim_{k \to \infty} A^k c_0 / \|A^k c_0\|$ exists for every $c_0 \in \mathbb{R}^N$.
2. For a non-negative initial value, the power method converges to a non-negative eigenvector to the eigenvalue 1, i.e., for $c_0 \geq 0$, the power method converges to an eigenvector $c^* \geq 0$ to the eigenvalue 1.
3. Let the fluid body be connected. For a non-negative initial value, the power method converges to a unique positive eigenvector (up to scaling).

Proof. By Lemma B.1 and Lemma B.2, the matrix $A$ is a stochastic matrix with $\lambda = 1$ as the only eigenvalue with $|\lambda| = 1$. Hence, the series $A^k$ converges \cite{45, Theorem 3.4} and, therefore, the limit of the power method $\lim_{k \to \infty} A^k c_0 / \|A^k c_0\|$ exists for every initial value $c_0$, especially for an initial value $c_0 \geq 0$. For $c = \lim_{k \to \infty} A^k c_0$, it follows that

$$Ac = A \lim_{k \to \infty} A^k c_0 = \lim_{k \to \infty} A^{k+1} c_0 = c,$$

which implies that the power method converges to a (normalized) eigenvector $c^*$ to the eigenvalue 1. As the initial value is chosen as $c_0 \geq 0$, the limit has to be $c^* \geq 0$.

Lemma B.1 states that the matrix $A$ is irreducible, if and only if the fluid body is connected. Moreover, Lemma B.2 implies the simplicity of the eigenvalue 1 and the existence of a positive eigenvector. Therefore, the limit of the power method has to be positive because the initial value is non-negative and the eigenspace is one-dimensional. \hfill $\square$