

# EFFICIENT RENDERING AND SIMULATION OF FLUID TRANSPORT AND PHASE TRANSITIONS IN SPH-BASED FLUIDS

DISSERTATION  
zur Erlangung des Grades eines  
Doktors der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von  
Dipl.-Inf. Hendrik Hochstetter

eingereicht bei der Naturwissenschaftlich-Technischen Fakultät  
der Universität Siegen  
Siegen 2018

Betreuer und erster Gutachter  
Prof. Dr. Andreas Kolb  
Universität Siegen

Zweiter Gutachter  
Prof. Dr. Rüdiger Westermann  
Technische Universität München

Tag der mündlichen Prüfung  
22. Mai 2019

# Abstract

Particle based fluid simulation using the smoothed particle hydrodynamics (SPH) method has gained much attention in recent years. Due to its flexible discretization, inherent mass preservation and its ability to stably simulate free surface flows and complex interactions with boundaries, SPH has found its way into many fields of research and application. While many phenomena including the transport of substances such as salt or dye, and melting and solidification can already be described in SPH, interactions with the air phase are commonly omitted. Typically, SPH liquids are rendered only in terms of their surface. However, with the growing complexity of simulations there arises a need for complementary rendering and visualization techniques that take transport of substances in the fluid's bulk into account. The goal of this work is to improve fluid animation of surface dynamics and the rendering and visualization of fluid transport.

First, a simulation of evaporation and condensation of SPH based fluids is introduced. Therefore, the air phase is simulated on a coarse grid and exchanges mass with the particle based liquid phase. Condensation only takes place on surfaces of rigid objects and is realized using textures into which mass can be condensed and from which particles can be generated. In order to achieve high visual detail of condensed liquids at surfaces, an implicit surface model is developed that allows to render moving liquid droplets at sub-particle detail including dynamic contact angles.

Second, an efficient adaptive volume ray casting of SPH-based scalar fields is developed. In order to achieve fast spatial access to particle data, particles are mapped to cells of a view-aligned perspective grid. By applying a sampling error analysis to the volume rendering equation inside of each grid cell, the sampling rate can locally be adjusted according to a user-defined screen space error tolerance yielding substantial speedups without sacrificing image quality.

Third, this work presents a vector field visualization of advective-diffusive flows of scalar quantities. Therefore, the advective, diffusive and total flow are each decomposed into a scalar quantity and a velocity component of transport. By introducing the novel visual metaphor of stream feathers, all flow components can be simultaneously visualized allowing for an intuitive insight into complex flow scenarios.



# Zusammenfassung

**P**artikelbasierte Flüssigkeitssimulation mit der Smoothed Particle Hydrodynamics (SPH) Methode hat in den letzten Jahren große Aufmerksamkeit erfahren. Aufgrund ihrer flexiblen Diskretisierung, ihrer inhärenten Masseerhaltung und ihrer Stärke, Strömungen an freien Oberflächen und komplexe Fluid-Struktur-Kopplung stabil zu simulieren, hat die SPH-Methode Einzug in viele Forschungs- und Anwendungsbereiche gehalten. Während viele Phänomene wie der Transport von Substanzen wie Salz oder Farbstoff und das Schmelzen und Erstarren bereits in SPH umgesetzt werden, werden Wechselwirkungen mit der umgebenden Luftphase im Allgemeinen vernachlässigt. Üblicherweise werden nur Oberflächenrenderings von SPH-basierten Flüssigkeiten zur Darstellung verwendet. Mit der wachsenden Komplexität von Simulationen geht allerdings auch der Bedarf an entsprechenden Techniken des Renderings und der Visualisierung einher, die auch den Transport von Substanzen im Inneren des Fluids miteinbeziehen. Das Ziel dieser Arbeit ist es, sowohl die Fluidanimation von Oberflächenprozessen als auch das Rendering und die Visualisierung von Fluidtransportphänomenen weiterzuentwickeln.

Zuerst wird eine Simulation der Verdunstung und Kondensation SPH-basierter Flüssigkeiten vorgestellt. Dazu wird die Luftphase auf einem groben Gitter simuliert, über das ein Masseaustausch mit der partikelbasierten Flüssigkeitsphase erfolgt. Kondensation findet ausschließlich an Oberflächen von Festkörpern statt und wird mittels Texturen realisiert, in die Masse kondensieren kann und von der Partikel erzeugt werden können. Um kondensierte Flüssigkeit in hoher visueller Auflösung zu erreichen, wird ein implizites Oberflächenmodell entwickelt, das es ermöglicht, sich bewegende Partikel in Subpartikelauflösung mit dynamischen Kontaktwinkeln darzustellen.

Zweitens wird ein effizientes adaptives Volumen-Raycasting für SPH-basierte Skalarfelder entwickelt. Um dabei schnellen räumlichen Zugriff auf Partikel-daten zu erhalten, werden Partikel auf Zellen eines perspektivischen Gitters abgebildet, das mit dem Sichtfrustum ausgerichtet ist. Durch eine Analyse des Samplingfehlers der Volumenrenderinggleichung innerhalb jeder Gitterzelle kann die Samplingrate lokal an eine benutzerdefinierte Bildfehlertoleranz angepasst werden. Dadurch lässt sich eine große Beschleunigung erreichen, ohne die Bildqualität zu beeinträchtigen.

Drittens wird in dieser Arbeit eine Vektorfeldvisualisierung von advektiv-diffusiven Flüssen von skalaren Größen vorgestellt. Dazu werden der advective, der diffusive und der totale Fluss jeweils in eine skalare und eine Geschwindigkeitskomponente des Transports zerlegt. Mit Hilfe der neuen graphischen Metapher der Stream Feathers können alle Komponenten von Flüssen gleichzeitig dargestellt werden, was einen intuitiven Zugang zu komplexen Fluss Szenarien erlaubt.

# Acknowledgments

**F**irst of all, I would like to thank my supervisor Prof. Dr. Andreas Kolb for giving me the opportunity to do research in the highly exciting field of SPH-based fluids, for his steady support and working late into the night before deadlines. I am also very thankful that Prof. Dr. Rüdiger Westermann from the Technical University of Munich agreed to read and review this thesis.

During my research in Siegen, I was surrounded by a group of highly competent and friendly colleagues. It has been a great pleasure working with them.

I am deeply grateful to Dr. Jens Orthmann for co-authoring two great papers and commenting on large parts of this thesis. If it weren't for Jens' support and his contagious enthusiasm, I probably would have overlooked much of the beauty and joy of doing research in SPH.

I am very thankful to Markus Kluge and Dr. Martin Pätzold who helped producing two videos.

Special thanks go to Ulrich Schipper for his steady technical support even on vacation.

I am very grateful to Willi Gräfrath for his assistance in all administrative tasks, regular proofreading and for turning videos into something special by lending them his voice.

Last but not least, I want to express my deep gratitude to my girlfriend, parents, family, and friends for their encouragement and support.





# Contents

<b>Notations</b>	<b>xiii</b>
<b>List of Figures</b>	<b>xviii</b>
<b>List of Tables</b>	<b>xix</b>
<b>List of Algorithms</b>	<b>xxi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Methodology . . . . .	3
1.3 Challenges . . . . .	4
1.4 Contributions . . . . .	6
1.5 Overview . . . . .	7
<b>2 Foundations</b>	<b>9</b>
2.1 Theoretical Background of Fluid Transport . . . . .	10
2.1.1 Conservation of Momentum . . . . .	10
2.1.2 Eulerian vs. Lagrangian Point of View . . . . .	11
2.1.3 Concentration Transport . . . . .	12
2.1.4 Heat Transport . . . . .	12
2.2 Overview of Fluid Simulation Approaches . . . . .	13
2.2.1 Mesh-based Approaches . . . . .	14
2.2.2 Particle-based Approaches . . . . .	15
2.2.3 Hybrid Approaches . . . . .	16
2.2.4 Comparison Between Different Approaches . . . . .	17
2.3 Time Integration . . . . .	18
2.4 Rendering and Visualization . . . . .	19
2.4.1 Surface Rendering . . . . .	19
2.4.2 Volume Rendering . . . . .	20
2.4.3 Vector Field Visualization . . . . .	22

---

<b>3</b>	<b>SPH-based Simulation of Fluid Transport</b>	<b>25</b>
3.1	SPH Interpolation and Kernel Functions . . . . .	26
3.1.1	Derivatives in SPH . . . . .	28
3.1.2	Calculating the Fluid Density and Volume . . . . .	29
3.1.3	Corrected SPH Interpolation . . . . .	29
3.2	Discretizing Fluid Transport in SPH . . . . .	30
3.2.1	Non-pressure Accelerations . . . . .	31
3.2.2	Pressure Acceleration and Incompressibility . . . . .	33
3.2.3	Transport of Concentrations and Heat . . . . .	34
3.3	Adaptive Simulation . . . . .	35
3.4	Efficient and Parallel Implementation . . . . .	36
<b>4</b>	<b>Simulation of Evaporation and Condensation</b>	<b>39</b>
4.1	Foundations and Prior Work . . . . .	40
4.1.1	Equations of Fluid Flow . . . . .	41
4.1.2	Evaporation and Condensation . . . . .	42
4.2	Algorithm Overview . . . . .	42
4.2.1	Particle Simulation . . . . .	43
4.2.2	Grid Simulation . . . . .	44
4.2.3	Texture-based Rigid Surface Representation . . . . .	45
4.2.4	Neighborhood Search . . . . .	45
4.3	Heat Transfer . . . . .	45
4.3.1	Heat Transfer Between Grid and Particles . . . . .	46
4.3.2	Heat Transfer to Texture . . . . .	46
4.4	Evaporation and Condensation . . . . .	47
4.4.1	Modeling Evaporation and Condensation . . . . .	47
4.4.2	Texel Evaporation and Condensation . . . . .	48
4.4.3	Evaporation and Condensation of Particles . . . . .	48
4.4.4	Dynamic Particle Adjustment . . . . .	49
4.5	Surface Rendering . . . . .	50
4.6	Results . . . . .	53
4.7	Conclusions . . . . .	57
<b>5</b>	<b>Adaptive Volume Ray Casting</b>	<b>59</b>
5.1	Foundations and Prior Work . . . . .	61
5.1.1	Adaptive Volume Rendering . . . . .	61
5.1.2	Interval Arithmetic . . . . .	62

5.2	Proposed Adaptive Ray Casting Pipeline . . . . .	63
5.3	Sampling Error Analysis Framework . . . . .	65
5.3.1	Lateral Quantity and Sample Bounds . . . . .	66
5.3.2	Cell Bounds . . . . .	68
5.3.3	Screen Space Error Analysis . . . . .	69
5.3.4	Greedy Optimization of Sampling Levels . . . . .	70
5.4	Analysis and Performance Optimizations . . . . .	70
5.4.1	Relaxed Error Estimation . . . . .	71
5.4.2	Lateral Adaptive Sampling Using Super-Pixels . . . . .	72
5.4.3	Combined Greedy Optimization . . . . .	73
5.5	Implementation Details . . . . .	73
5.5.1	Particle Access via Perspective Grids . . . . .	73
5.5.2	Cell Merging . . . . .	74
5.5.3	Adaptive Ray Casting . . . . .	75
5.6	Results and Discussion . . . . .	77
5.7	Conclusions . . . . .	82
<b>6</b>	<b>Visualization of Advective-Diffusive Flows</b>	<b>83</b>
6.1	Foundations and Prior Work . . . . .	85
6.1.1	Advective and Diffusive Flux . . . . .	85
6.1.2	Visualization of Advective-Diffusive Flow . . . . .	86
6.1.3	Visualization of SPH Fluid Simulations . . . . .	86
6.2	Overview . . . . .	87
6.3	A Framework for Tracing Advective-Diffusive Fluxes . . . . .	88
6.3.1	Decomposition of Diffusive Flux . . . . .	88
6.3.2	Unified Model of Advective-Diffusive Flux . . . . .	90
6.4	Visualization of Fluxes Using Stream Feathers . . . . .	92
6.5	Advective-Diffusive Fluxes in SPH . . . . .	93
6.6	Results and Discussion . . . . .	93
6.7	Conclusions . . . . .	100
<b>7</b>	<b>Conclusions</b>	<b>101</b>
7.1	Summary . . . . .	101
7.2	Future Work . . . . .	102
	<b>Bibliography</b>	<b>105</b>



# Notations

This nomenclature gives all necessary symbols as used throughout this thesis. Despite striving for uniqueness, some symbols are still used redundantly in different meanings. This is true, e.g., for  $T$  which in general denotes temperature but denotes transparency in the context of ray casting in Chapter 5. Thus, the nomenclature below is grouped into general quantities and special quantities that are used only in specific contexts and chapters. In case of redundancy, the specific meaning should always be clear from context.

## Acronyms and abbreviations

APIC	Affine Particle-in-Cell method
CFL	Courant-Friedrichs-Lewy
CSPH	Corrected SPH
EOS	Equation of state
FLIP	Fluid-Implicit-Particle method
IISPH	Implicit Incompressible SPH
MAC	Marker-and-Cell
PBF	Position-based fluids
PCISPH	Predictive Corrective Incompressible SPH
PIC	Particle-in-Cell method
RK4	The Runge-Kutta method (of order 4)
SPH	Smoothed Particle Hydrodynamics method
WCSPH	Weakly Compressible SPH

## Mathematical and Physical Notations

$\nabla$	Gradient operator $\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)^T$
$\nabla \cdot$	Divergence operator $\left(\frac{\partial}{\partial x} + \frac{\partial}{\partial y} + \frac{\partial}{\partial z}\right)$
$\nabla^2$	Laplace operator $\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)$
$\tau, \Delta\tau$	Time, time step

$m$	Mass
$A$	Area
$V$	Volume
$\rho$	Density
$p$	Pressure
$T$	Temperature
$c$	Concentration
$\mathbf{x}$	Position
$\vec{v}$	Velocity
$\vec{a}$	Acceleration
$\vec{F}, \vec{f}$	Force, force density $\vec{f} = \frac{\vec{F}}{V}$
$\vec{g}, \hat{g}$	Gravitational acceleration and its normalized direction
$\eta, \nu$	Dynamic viscosity, kinematic viscosity
$\kappa$	Surface curvature
$\delta$	Surface delta function
$\sigma$	Surface tension
$\Sigma$	Surface stress tensor
$\zeta$	Viscous stress
$D$	Diffusivity
$k_B$	Boltzmann constant
$C$	Specific heat capacity
$\kappa$	Heat conductivity
$U$	Overall heat transfer coefficient
$R_w$	Specific gas constant of water

### SPH notations

$\mathbf{x}_i$	Position of particle $i$
$\vec{\mathbf{x}}_{ij}, x_{ij}$	Vector $\mathbf{x}_i - \mathbf{x}_j$ between particles $i$ and $j$ and its length
$\vec{v}_{ij}$	Velocity difference vector $\vec{v}_i - \vec{v}_j$ between particles $i$ and $j$
$W$	Kernel function
$\hat{W}$	Corrected SPH kernel function
$h$	Kernel support radius
$n_i$	Particle number density
$Q_i$	Quantity $Q$ , e.g., temperature or concentration, of particle $i$
$\phi(\mathbf{x})$	Implicit surface function
$\delta_i$	Smoothed surface delta

**Evaporation and Condensation**

$c$	Cell index
$t$	Texel index
$i, j$	Particle index
$s$	Surfel, i.e., either a surface particle or a texel index
$w_i$	Weighting term of particle $i$ , $w_i \in [0, 1]$

**Volume Ray Casting**

$I_e, \sigma_\alpha$	Emitted radiance, absorption of light
$x^\perp$	Interval $x^\perp = [x^\perp, x^\top]$
$w(x^\perp)$	Width of interval $x^\perp$
$E_I$	Error tolerance for radiance in the final image $\in [0, 1]$
$E_{\text{relax}}$	Relaxation parameter for relaxed error estimation $\in [0, 1]$
$Q_i, Q_k$	Quantity of particle $i$ , of ray sample $k$
$Q_k^\perp$	Quantity bounds at sampling depth $k$
$i_k^{\perp, l}, t_k^{\perp, l}$	Radiance, transparency sample bounds at sampling depth $k$ and level $l$
$I_c^{\perp, l}, T_c^{\perp, l}$	Radiance, transparency cell bounds of cell $c$ and level $l$
$I_{\text{SP}}, T_{\text{SP}}$	Radiance, transparency of a super-pixel
$\mathbb{I}_L^{\perp, \vec{l}}, \mathbb{T}_L^{\perp, \vec{l}}$	Radiance, transparency ray bundle bounds for cell sequence $l_0, \dots, l_L$
$c$	Cell
$C$	Cell indexing function
$l_c^{\text{max}}$	Maximum sampling level of cell $c$
$\vec{l}$	Vector of sampling levels of cell sequence $l_0, \dots, l_{\text{last}}$
$D_{xy}, D_z$	Resolution of a cell in xy- and z-directions
$\Delta i_l = 2^l$	Discrete sampling step size at level $l$
$\text{tf}$	Transfer function

**Advection-Diffusion Visualization**

$c_a, c_d, c_t, c_m$	Advective, diffusive, unified mean and unified maximum concentration
$\vec{v}_a, \vec{v}_d, \vec{v}_t, \vec{v}_m$	Advective, diffusive, unified mean and unified maximum velocity
$\vec{j}_a, \vec{j}_d, \vec{j}_t, \vec{j}_m$	Advective, diffusive, unified mean and unified maximum concentration fluxes





# List of Figures

2.1	A mesh-based simulation using a staggered grid . . . . .	14
2.2	A Lagrangian simulation using particles . . . . .	15
2.3	A hybrid simulation using both particles and grid cells . . . . .	16
2.4	Different simplifications of the volume rendering model . . . . .	21
2.5	Volume rendering using a Riemann sum approximation . . . . .	22
2.6	Vector field visualization using streamlines . . . . .	23
3.1	SPH-interpolation . . . . .	26
3.2	The poly6, spiky and cohesion kernel functions . . . . .	27
3.3	Comparison of standard and corrected SPH-interpolation . . . . .	30
4.1	Mass and heat transfer between different systems . . . . .	44
4.2	Adjustment of distance function according to advancing $\alpha_{adv}$ and receding $\alpha_{rec}$ angles . . . . .	51
4.3	A drop of one particle is modified using the proposed approach . . . . .	52
4.4	Uncorrected surface rendering (left) and the proposed modification to render dynamic contact angles (right). . . . .	53
4.5	A spherical drop of water evaporates on a hot surface . . . . .	54
4.6	A mirror is steamed by humid air revealing the impregnated SCA logo. . . . .	55
4.7	Blowing moist air onto the impregnated SCA logo causes particles to condense. . . . .	56
4.8	A glass filled with cold liquid surrounded by moist air (left) causes water to condense at the glass surface (right). . . . .	57
5.1	The fluid letters HPG 2016 are dropped into a basin . . . . .	59
5.2	Sparse access structure . . . . .	65

5.3	A radially increasing concentration profile rendered with a complex transfer function . . . . .	67
5.4	Calculation of quantity bounds and sample bounds . . . . .	68
5.5	Computation of radiance bounds . . . . .	69
5.6	Back to front exchange of sampling levels . . . . .	71
5.7	Two iterations of the cell merging . . . . .	75
5.8	3D checker board of increasing concentrations . . . . .	77
5.9	A mixer is causing a stream of green dye to mix with solvent in the fluid tank . . . . .	78
5.10	A frame of the flubber scene and the respective sparse grid . . .	79
5.11	Errors of the simulation frames of the Mixer scene . . . . .	79
5.12	Timings of the simulation frames of the Mixer scene . . . . .	81
6.1	A drop of green dye is dripped into water . . . . .	83
6.2	Diffusion follows concentration gradients . . . . .	89
6.3	Construction of the direction of unified maximum velocity flux .	92
6.4	Stream feathers . . . . .	92
6.5	Stream feather visualization of advective and diffusive fluxes corresponding to the unified mean velocity flux in Fig. 6.1, right. . .	94
6.6	Advective, unified mean velocity and diffusive fluxes after impact of dye in solvent . . . . .	95
6.7	Advective, diffusive and unified mean velocity fluxes at impact of a solvent drop in a tank of dye . . . . .	96
6.8	Advective, diffusive and unified fluxes of the flow in a t-sensor .	97
6.9	Unified mean velocity flux of the checker board scene over three time steps . . . . .	99

# List of Tables

1.1	Methodological foci and directions in different fields of research	4
2.1	Comparison of different fluid solvers . . . . .	17
4.1	Scene characteristics and run times . . . . .	54
5.1	GPU timings, speed-ups and errors of adaptive and non-adaptive volume rendering . . . . .	80
5.2	Error statistics of the adaptive sampling . . . . .	81
6.1	GPU timings of streamline integration and stream feather rendering	99



# List of Algorithms

3.1	A generic SPH simulation loop . . . . .	31
4.1	Overview over the proposed simulation of evaporation and condensation . . . . .	43
4.2	Evaporation from and condensation into textures . . . . .	49
5.1	The proposed sampling error analysis . . . . .	66
5.2	Thread-coherent volume ray casting . . . . .	76



# Introduction

*This chapter briefly describes the challenges of fluid simulation and rendering in the context of computer graphics and motivates the investigation of complex phenomena involving fluid transport and thermodynamic processes. The chapter closes by outlining the contributions that have been made to different fields of computer graphics.*

---

**I**n the field of computer graphics, a wide range of topics is studied. This includes 3D modeling as it is used in computer-aided design, animation and rendering of dynamic scenes including fluid simulation, and visualization of simulated or scanned data. Especially in rendering and animation, many developments are driven by the computer games and film industries which strive for ever-growing levels of realism in physical simulations and visual effects. For games, real-time constraints additionally have to be fulfilled which require the development of highly efficient methods that often exploit massively parallel processors like GPUs. Animation includes the simulation of liquids and gases which are rendered to produce high-quality visual results. In visualization, computer graphical methods are developed that focus on visually conveying insight into often large amounts of data instead of striving only for realism. Visualization most often addresses medical and scientific applications, e.g., the visual presentation of simulated scalar fields by means of volume rendering.

Research in  
computer graphics

In animation, rendering and visualization, liquids are usually only described in terms of their convective motions. Additional effects like advective-diffusive fluid transport, interactions with the air phase and thermodynamic processes have only rarely been considered. The goal of this thesis is to fill some of these gaps and to introduce a simulation approach for the thermodynamic processes of evaporation and condensation, an efficient rendering of volumetric scalar data, and a vector field visualization method that intuitively conveys advective-diffusive transport phenomena.

Fluid animation and  
rendering

## 1.1 Motivation

### CFD methods in computer graphics

Every new generation of hardware leads to an increase in performance and memory space and thus allows for simulations and renderings at higher resolutions as well as with more realistic and more detailed models. While early fluid simulation models in computer graphics rather focused on imitating the visual appearance of fluids, it is desirable to derive simulation models in close accordance with the laws of physics. The parameters involved, e.g., density, viscosity and surface tension, can be interpreted very intuitively. Thus, techniques in computer graphics are increasingly adopted from the field of computational fluid dynamics (CFD). For efficiency reasons, models in computer graphics are often reduced and only encompass a small subset of underlying physical laws.

### Fluid transport

In fluid transport, the fluid acts as a carrier of substances which are transported by two distinct modes of transport. On the microscopic level, substances are transported by diffusion, i.e., through random collisions between neighboring molecules. On the macroscopic level, transport is due to the advective flux which follows the velocity field of the fluid and hence its bulk movement. Both processes occur simultaneously and are key processes in a large number of engineering problems such as chaotic microscopic mixing. A lot of interesting phenomena both from the fluid simulation and engineering perspective occur mainly at the fluid surface and the interface with rigid objects. These include the effect of wetting in combination with surfactants, i.e., soluble substances that have a higher affinity to the fluid surface than the fluid volume.

### Modeling complex phenomena

Many phenomena encountered in day to day life are due to an intricate interplay between different influences. For example, the tears of wine inside a glass are due to the mixture of water and alcohol which both have different dew points at which they turn into a gaseous state and due to the fact that the surface concentration of alcohol in water reduces surface tension. Because alcohol evaporates faster than water, a gradient in surface tension is maintained that is able to pull liquid up the glass wall against the influence of gravity. Simulating such visually appealing effects using physically-based models strongly motivates the contributions of this thesis.

### Efficient parallel simulation and visualization

Even with physically-based models it is, however, not trivial to achieve a desired fluid behavior. Animators often have to spend a large amount of time in tweaking artificial parameters. Efficient simulations in which results can be visually inspected on-the-fly and parameters can be interactively adjusted are thus invaluable. These visual tools encompass surface, volume and flow visualizations that are able to convey the shape of the fluid, as well as the distribution and transport of concentrations and other fluid properties.



## 1.2 Methodology

In fluid simulation, there have traditionally been two main approaches, i.e., grid-based approaches [FSJ01, Sta99] that follow the Eulerian point of view and grid-free approaches like smoothed particle hydrodynamics (SPH) [Luc77, GM77] which follow the Lagrangian point of view. Eulerian models describe fluids with respect to a fixed frame of reference, while in Lagrangian methods the points of discretization, i.e., the particles, are moved along with the fluid. Thus, particle-based discretizations allow fluids of arbitrary shape and especially free surface flows to be very naturally described. In SPH, physical processes are described in terms of interactions between particles which cause particles to change their physical properties, e.g., their position. Due to its simplicity, its inherent mass-preservation and its suitability to be efficiently parallelized on GPUs, this work adopts the Lagrangian point of view using SPH as method of choice for simulating liquids.

Fluid simulation methods

Smoothed particle hydrodynamics

Depending on the area of research, the requirements both to the simulation and the rendering or visualization of results can differ considerably. In CFD, the focus usually lies on a precise modeling of the underlying phenomena and on quantitatively evaluating results. As CFD simulations often generate large amounts of data [CCB\*08], visualization techniques are employed that allow for an interactive navigation through data sets at the cost of offline preprocessing [FSW09].

CFD simulation and visualization

For simulations in the area of computer graphics, coarser simplified simulation models are often preferred that can be evaluated more efficiently. By not simulating the air phase and assuming isothermal conditions, the simulation complexity is significantly reduced. While in computer animation, the goal usually is to derive physically plausible models, the focus in real-time fluids [MMCK14, CMK15] lies on achieving the simulation and rendering at interactive frame rates. Thus, physical plausibility often has to be traded for coarse approximations that yield only visually plausible results. Physical models have even been replaced by predicting particle positions and velocities using machine learning techniques and massive training from precomputed offline simulations [LJS\*15].

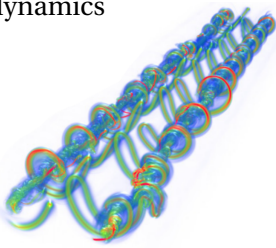


Fluids in computer graphics

The visual presentation of fluids in computer graphics follows similar goals. The rendering of real-time fluids has to obey strict frame rate constraints, thus, approximate methods like screen space splatting [vdLGS09] of particles are mainly employed. For computer animation purposes, fluids are often rendered offline using mesh based intermediate representations that allow for photo-realistic rendering like ray tracing [AIAT12]. Nonetheless, interactive rendering methods are still important tools for an on-the-fly investigation of

Rendering of fluids

simulation runs and an interactive steering of simulation parameters. Tab. 1.1 shows examples of particle-based simulations from different areas of research and compares the respective goals of the applied methods with respect to simulation and graphical presentation of results.

**Table 1.1:** Methodological foci and directions of the simulation (following Orthmann [Ort14]) and visual presentation of fluids in different fields of research.

Computational Physics	Computer Graphics	
Computational fluid dynamics 	Fluid animation 	Real-time fluids 
Chatelain et al. [CCB*08] <sup>1</sup>	© 2014 IEEE Ihmsen et al. [ICS*14]	© 2015 IEEE Chentanez et al. [CMK15]
Simulation: <ul style="list-style-type: none"> <li>• Physical accuracy</li> <li>• Numerical precision</li> <li>• Quantification</li> </ul>	Simulation: <ul style="list-style-type: none"> <li>• Physical basis</li> <li>• Numerical robustness</li> </ul>	Simulation: <ul style="list-style-type: none"> <li>• Visual plausibility</li> <li>• Numerical efficiency</li> <li>• Interactivity</li> </ul>
Visual presentation: <ul style="list-style-type: none"> <li>• Volume visualization</li> <li>• Flow visualization</li> </ul>	Visual presentation: <ul style="list-style-type: none"> <li>• High-quality offline rendering</li> <li>• Photo-realism</li> </ul>	Visual presentation: <ul style="list-style-type: none"> <li>• Approximate, fast rendering</li> <li>• Interactivity</li> </ul>

## 1.3 Challenges

Although SPH-based fluid animation is already able to simulate and render highly realistic and astonishingly beautiful artistic scenes, numerous challenges still remain, both in simulation and rendering of particle-based fluids.

Scalability of parameters

Especially for the movie industry, a physically plausible behavior of fluids is desirable because scenes are not always rendered from scratch but instead only

<sup>1</sup>Reprinted from Computer Methods in Applied Mechanics and Engineering, 197, Chatelain, Philippe; Curioni, Alessandro; Bergdorf, Michael; Rossinelli, Diego; Andreoni, Wanda; Koumoutsakos, Petros “Billion vortex particle direct numerical simulations of aircraft wakes”, pp. 1296–1304, © 2008, with permission from Elsevier.

parts of recorded scenes may be simulated. At different spatial and temporal resolutions, simulations may yield substantially different fluid motion due to the fact that SPH parameters are not invariant to scaling [PICT15, WHK17]. Although the behavior can always be adjusted by manual tweaking, parameters should be intuitive and independent of the resolution.

In order to achieve high visual detail, a high particle resolution is usually preferred. Large particle counts, however, lead to computationally expensive simulations. Adaptive simulations in which high resolution is limited to areas of interest save computational resources [SG11, OK12, WHK17]. Although fluids can already be described with highly adaptive particle systems [WHK17], the boundary conditions need to be adjusted as well in order to guarantee for versatile simulations including complex interactions like cleansing or depositing paint [OHB\* 13] on static and dynamic bodies. Moreover, manual tweaking of scale dependent parameters gets even more difficult as adaptive simulations can contain particles of different sizes that dynamically change over time.

Robust adaptivity

For most simulations in computer graphics, the assumption of isothermal conditions is made and the air phase is neglected although a large number of physical phenomena is only due to temperature differences and the interaction with an air phase. While some works have taken heat transport into account in order to simulate, e.g., melting and solidification [SSP07], a dynamic coupling with an air phase suffers from instabilities due to the large density difference between air and liquid which have only partly been alleviated [SP08, IOS\* 14]. The air phase has only been mimicked by sampling ghost particles around the liquid surface [SB12] or as a constant external velocity field that causes friction at the liquid surface [GBP\* 17a, GBP\* 17b]. For the simulation of phase changes of water to steam, i.e., the simulation of evaporation and condensation, both temperature and the air phase have to be taken into account. Moreover, the air phase has to be modeled such that it allows for a conservative transfer of mass and heat between air and liquid.

Comprehensive simulation

Research in fluid rendering is mainly concerned with conveying the geometric shape of the fluid, i.e., with surface rendering. Increasing the particle resolution at the surface does, of course, also increase the detail in rendering. Yet, lower resolutions are preferable in interactive or real-time applications [MMCK14]. Although surface reconstruction is able to yield smooth surfaces [YT10, AIAT12], the collision of liquids with rigid boundaries still causes problems. For mesh based fluid representations, vertices of the fluid surface can be shifted to closest point on the boundary to resolve intersections [HKK07b], which, however, can cause self-intersections in the resulting mesh. By mirroring particles to the opposite side of boundaries, static contact angles can be modeled [MWE16]. This, however, induces a costly sampling of additional particles. By taking rigid boundaries into account, surface rendering

Surface rendering

quality can be increased and interactions of the liquid with rigid objects can be conveyed that don't even have to be present in the simulation data.

On-the-fly volume rendering

The more parameters are involved in simulations, the more sophisticated methods of visually presenting the data are necessary. Volume rendering of additional parameters like concentrations of substances can create visually appealing effects and, moreover, allows to examine simulation data for their plausibility. Even though there are methods that address the volume rendering of particle-based fluids, they mostly depend on preprocessing and rely on rasterization hardware [FAW10] or they are built around object space data access structures which limits parallelism due to traversal logic [OKK10]. For interactive applications that allow for a direct steering of parameters, however, an on-the-fly presentation of volumetric scalar data is of great importance.

Vector field visualization

Apart from volume rendering, applying vector field visualization also adds an invaluable tool to inspect the time-dependent behavior of fluid flows. While there has been research addressing the visualization of pathlines that are directly derived from particle trajectories [FW12], vector field visualization of concentration and heat transport has not yet been investigated.

## 1.4

## Contributions

In this thesis, three major contributions to the field of computer graphics in fluid animation, volume rendering of particle-based data and in flow visualization are presented. The contributions address the challenges of comprehensive simulation, surface rendering, on-the-fly volume rendering and vector field visualization and are components of a larger framework that combines fluid simulation with rendering and visualization capabilities and allows for efficient GPU-based computation of incompressible SPH fluids and their visual presentation. The contributions are summarized in the following.

**Evaporation and Condensation** are both thermodynamic processes and, thus, depend on the temperature of the involved substances. As heat makes liquid water turn into a gaseous state, an air phase has to be simulated explicitly. To that end, the air phase is efficiently simulated on a coarse regular grid in which vapor mass is transported. Using a novel physically-based model of evaporation and condensation, liquid particles can be evaporated by transferring their mass to the air phase. Condensation takes place on surfaces of rigid objects using textures that store liquid mass. If sufficient mass is accumulated, it is transformed back to new liquid particles. The small-scale details of condensation using textures are complemented by an improved implicit

surface definition for rendering the particle-based liquid phase. By taking rigid surfaces into account, dynamic contact angles of moving particles can be rendered. Even at low particle counts, the method is able to achieve convincing high-quality results.

**Adaptive Volume Ray Casting:** Most fluid rendering focuses on surface rendering techniques. The depiction of scalar quantities like concentration and temperature, however, requires a volumetric rendering. While in general, particle quantities can be mapped onto grids and rendered using standard volume rendering methods, this introduces interpolation errors, requires additional memory and also limits the rendering performance due to the resampling. In this thesis, a novel volume rendering method is introduced that operates directly on raw particle data. In order to speed up rendering, a hierarchical error analysis based on interval arithmetic is proposed that allows to locally adjust sampling rates while obeying a user-controlled screen space error tolerance. The volume ray casting is also used to ray cast surfaces yielding an efficient high-quality on-the-fly rendering.

**Vector Field Visualization of Advective-Diffusive Flows:** Although a combination of surface and volume rendering techniques is well able to convey static distributions of scalar quantities, their dynamic behavior can only be revealed by time sequences. Especially in advective-diffusive transport processes, it is desirable not only to visualize the quantity distribution but also the direction in which transport takes place, and which mode of transport, advection or diffusion, dominates. In this thesis, a novel method to visualize advective-diffusive flows is developed. By introducing a new visual metaphor, the stream feather, superposed advective-diffusive flows and their respective components can be intuitively presented. Although it is demonstrated only for SPH-based fluids, the method describes a general framework and can be applied to simulation data from different sources.

## 1.5

## Overview

Most of the content presented in the following chapters has already been published. The structure of this work thus follows these publications which are complemented by two chapters that cover foundations and a final chapter that concludes this thesis.

Chapter 2 introduces the theoretical foundations of fluid transport. General concepts of mesh-based and particle-based fluid simulations are outlined as

necessary to understand later chapters. As a visual representation of fluid simulations is usually desired, general rendering and visualization techniques for fluid surfaces, volumetric data, and general vector field visualization techniques for flow visualization are reviewed.

Chapter 3 introduces the theoretical foundations of SPH-based simulation and describes how fluid transport can be modeled with SPH. As SPH simulations can be computationally expensive, different adaptive methods are presented as well as algorithmic means to efficiently solve them on parallel platforms like GPUs.

Chapter 4 explains how the “Simulation of Evaporation and Condensation of SPH-based Fluids” is realized. It is based on the work presented at the Symposium on Computer Animation 2017 [HK17] and includes both the physically-based model of evaporation and condensation and the improved implicit surface definition that takes interactions of the liquid with rigid surfaces into account and allows for a rendering of dynamic contact angles.

Chapter 5 describes an “Adaptive Sampling for On-The-Fly Ray Casting of Particle-based Fluids” presented at High Performance Graphics 2016 [HOK16]. The adaptive sampling is shown to speed up volume rendering while preserving visual features. The screen space error of the resulting renderings is limited by a user-controlled error tolerance.

Chapter 6 presents a “Vector Field Visualization of Advective-Diffusive Flows” that has been presented at EuroVis 2015 and published in Computer Graphics Forum [HWK15]. The combined visualization of advection and diffusion using stream feathers as a novel visual metaphor allows for an expressive and intuitive graphical presentation of complex flow scenarios.

Chapter 7 summarizes this thesis, draws conclusions and gives hints for possible directions of future works.

## Foundations

*This chapter introduces the governing equations of fluid transport including advective-diffusive flows of concentration and heat transport. In the following, particle-based, grid-based and hybrid simulation approaches are presented and compared. Subsequently, a brief discussion of time integration schemes is presented. As simulation results will be visually conveyed, foundations of surface rendering, volume rendering and vector field visualization are outlined.*

*For further details the reader is referred to the following textbooks on fluid simulation by Bridson [Bri08], transport phenomena by Bird et al. [BSL07], volume rendering by Engel et al. [EHK\* 06], and visualization by Johnson and Hansen [JH04].*

---

**T**ransport phenomena typically encompass three main components, i.e., fluid dynamics, the transport of heat, and mass transport. All these components are very closely related [BSL07]. While the governing equations are described assuming a continuous medium, the fundamental processes behind these models occur on a molecular level. Sec. 2.1 covers both the molecular intuition and the mathematical formalisms that constitute fluid transport.

Fluid transport can be described from two distinct viewpoints. In the Eulerian viewpoint, the fluid moves according to a fixed coordinate system and in the Lagrangian viewpoint, the fluid elements themselves follow the flow. Both viewpoints lead to distinct simulation methods as presented and discussed in Sec. 2.2. In order to simulate the time evolution of fluid transport phenomena, different time integration schemes can be applied as will be discussed in Sec. 2.3.

At the end of each simulation pipeline, renderings and visualizations of the simulated data are usually generated to present simulation results. In liquid simulations, these are mainly surface renderings and in case of volumetric scalar fields, such as the concentration of solutes, also volume rendering and vector field visualization as outlined in Sec. 2.4.

## 2.1 Theoretical Background of Fluid Transport

In general, a fluid volume can be described by a set of smaller parcels of fluid volumes that interact with each other. Each of these parcels carries mass  $m$ , has volume  $V$  and thus a density  $\rho = \frac{m}{V}$  and is centered at a position  $\mathbf{x}$  in space. The fluid motion is described by the velocity  $\vec{v}$  that evolves over time according to the momentum equation (see Sec. 2.1.1). Additionally concentrations of substances and heat can be transported along with the fluid flow and be diffused between different fluid parcels (see Secs. 2.1.3 and 2.1.4).

### 2.1.1 Conservation of Momentum

Momentum equation

Fluid parcels are moved through space by velocity  $\vec{v}$ , which changes over time  $\tau$  through the influence of other fluid parcels and external forces as described by the momentum equation

$$\rho \frac{D\vec{v}}{D\tau} = -\nabla \cdot \Sigma + \vec{f}^{\text{ext}} \quad (2.1)$$

where  $\Sigma$  denotes the surface stress, and  $\vec{f}^{\text{ext}}$  external forces.  $\frac{D\mathcal{O}}{D\tau}$  denotes the material derivative and will be discussed in Sec. 2.1.2.

Surface stress

The surface stress  $\Sigma$  transfers momentum across the fluid and is due to molecular collisions that depend on the free space between molecules and their respective velocities. The stress can be subdivided into viscous stress  $\zeta$  and stress due to pressure  $p$  as  $\Sigma = \zeta - p\mathbb{1}$ , where  $\mathbb{1}$  denotes the  $3 \times 3$  identity matrix. In incompressible fluids, the pressure term ensures a divergence free velocity field

Incompressibility

$$\nabla \cdot \vec{v} = 0 \quad (2.2)$$

Viscosity

and the viscous stress can further be simplified to  $\nabla \cdot \zeta = \nabla \eta \nabla \vec{v} = \eta \nabla^2 \vec{v}$  assuming a constant viscosity  $\eta$  throughout the medium. The viscosity acts as diffusion of momentum which in turn locally smoothes the velocity field.

Gravity

Surface tension

External forces usually comprise gravitation  $\vec{g}$  and in case of liquids surface tension as  $\vec{f}^{\text{ext}} = \rho \vec{g} + (\sigma \kappa \hat{\mathbf{n}} + \nabla_{\hat{\mathbf{n}}} \sigma) \delta$ , where  $\kappa$  is the surface curvature,  $\sigma$  the strength of surface tension,  $\hat{\mathbf{n}}$  the unit surface normal, and  $\delta$  the surface delta function which is one exactly at the surface and zero everywhere else. Surface tension acts to minimize the surface area of the liquid phase. Additionally, gradients in surface tension can cause Marangoni convection that induces a fluid motion tangentially to the surface as described by the surface gradient  $\nabla_{\hat{\mathbf{n}}} = \nabla - \hat{\mathbf{n}}(\hat{\mathbf{n}} \cdot \nabla)$ , i.e., the gradient's orthographic projection onto the surface.

Marangoni convection



While the formulation above was derived using force densities  $\vec{f} = \frac{\vec{F}}{V}$ , it is more easy to reformulate it to accelerations according to Newton's second law  $\vec{F} = m\vec{a} = m\frac{D\vec{v}}{D\tau}$ . Plugging the simplified stress into Eq. (2.1) and dividing by  $\rho$ , gives the well-known Navier-Stokes equation [Bri08]

Navier-Stokes equation

$$\frac{D\vec{v}}{D\tau} = \underbrace{\nu\nabla^2\vec{v}}_{\text{Viscosity}} - \underbrace{\frac{1}{\rho}\nabla p}_{\text{Pressure Acceleration}} + \underbrace{\vec{a}^{\text{ext}}}_{\text{External Acceleration}}, \quad (2.3)$$

where  $\nu = \frac{\eta}{\rho}$  is the kinematic viscosity, and  $\vec{a}^{\text{ext}} = \frac{\vec{F}^{\text{ext}}}{m} = \frac{\vec{f}^{\text{ext}}}{\rho}$  describes external acceleration.

## 2.1.2

## Eulerian vs. Lagrangian Point of View

There are two distinct viewpoints from which fluid transport can be described. In the Eulerian viewpoint, a fixed frame of reference is adopted that can be compared to describing the flow of a river while standing on a bridge. More formally speaking, the fluid is described in terms of volume elements that are fixed in space but exchange physical quantities over their faces that connect them to neighboring volume elements. In the Lagrangian viewpoint, the frame of reference always moves with the fluid flow. Imagine the flow of a river described from a boat that is carried along with the flow. More formally, the fluid is again discretized as volume elements, however, in the Lagrangian viewpoint, the fluid elements themselves are moved to describe the flow.

Eulerian viewpoint

Lagrangian viewpoint

Mathematically, both viewpoints are related by the material derivative  $\frac{D0}{D\tau}$ . Using the global frame of reference of the Eulerian viewpoint, the material derivative reads  $\frac{D0}{D\tau} = \frac{\partial 0}{\partial \tau} + \nabla() \cdot \frac{dx}{d\tau} = \frac{\partial 0}{\partial \tau} + \nabla() \cdot \vec{v}$  and comprises an unsteady and a convective derivative. In the Lagrangian viewpoint  $\frac{D0}{D\tau}$  equals the total differential  $\frac{d0}{d\tau}$  because it is calculated according to a local frame of reference that moves with the flow.

Material derivative

From the Eulerian viewpoint, the convective derivative has to be calculated explicitly as

$$\frac{\partial \vec{v}}{\partial \tau} + \underbrace{\vec{v} \cdot \nabla \vec{v}}_{\text{Convective Derivative}} = \underbrace{\nu \nabla^2 \vec{v}}_{\text{Viscosity}} - \underbrace{\frac{1}{\rho} \nabla p}_{\text{Pressure Acceleration}} + \underbrace{\vec{a}^{\text{ext}}}_{\text{External Acceleration}}, \quad (2.4)$$

while in the Lagrangian viewpoint in which fluid parcels travel with the flow, the momentum equation simplifies to

Navier-Stokes in Lagrangian frame

$$\frac{d\vec{v}}{d\tau} = \nu \nabla^2 \vec{v} - \frac{1}{\rho} \nabla p + \vec{a}^{\text{ext}} \quad (2.5)$$

as the convective derivative vanishes.

Throughout this thesis, both the Lagrangian and the Eulerian point of view will be adopted in different places.

### 2.1.3 Concentration Transport

Advection-diffusion equation

While the fluid motion can be described by the momentum equation (cf. Eq (2.3)), additional quantities can also be transported with the flow. Transport of concentration  $c$  of solved substances inside the fluid generally follows the advection-diffusion equation as [BSL07]

$$\frac{\partial c}{\partial \tau} = \underbrace{\nabla \cdot (D \nabla c)}_{\text{Diffusion}} - \underbrace{\nabla \cdot (\vec{v} c)}_{\text{Advection}} + \underbrace{R_c}_{\text{Sinks and Sources}}, \quad (2.6)$$

Fick's law

where  $R_c$  denotes source and sink terms due to, e.g., chemical reactions.  $D$  denotes the diffusivity which is either scalar in case of isotropic diffusion or a  $3 \times 3$ -matrix for anisotropic diffusion.  $D$  depends on the material components as well as the temperature. The diffusive term of Eq. (2.6) follows Fick's law [ALS09] and is due to the random movements of molecules that causes concentration gradients to vanish over time.

Concentration transport in Lagrangian frame

Considering a Lagrangian frame of reference, Eq. (2.6) simplifies to

$$\frac{dc}{d\tau} = \nabla \cdot (D \nabla c) + R_c, \quad (2.7)$$

so that the advection term vanishes. In Chapter 6, a vector field visualization to simultaneously convey advection and diffusion of field quantities is presented.

### 2.1.4 Heat Transport

Radiation

The transport of heat is usually made up of three modes of transport: conduction, convection and radiation. Heat radiation takes place without a connecting medium due to emission and absorption of electromagnetic waves. In the following, however, only convection and conduction of heat will be considered that obey similar laws like concentration transport.

Fourier's law

Heat  $Q$  is transported following an advection-diffusion equation. For the diffusive part, i.e., the conduction, Fourier's law [BSL07] relates the heat flux  $q$  to the temperature difference as  $q = -\kappa \nabla T$ . On a molecular level, conduction is due to collision of molecules, atoms and electrons by which internal energy, i.e., heat, is dispersed and temperature differences are equalized.

Heat convection can generally describe the fluid movement induced by temperature differences. These cause for differences in density and hence for buoyancy. Here, however, only the passive advection of heat is considered in which volumes of different heat are transported with the velocity field.

Heat advection

Assuming a homogeneous medium, heat  $Q$  can be related to temperature  $T$  by the simple relation  $T = \frac{Q}{\rho C}$ , where  $C$  is the specific heat capacity of the medium so that only temperatures appear in the following equations. The time rate of change of temperature in the Eulerian frame can then be expressed as

Heat transport in Eulerian frame

$$\frac{\partial T}{\partial \tau} = \frac{1}{\rho C} \nabla \cdot \underbrace{(\kappa \nabla T)}_{\text{Conduction}} - \nabla \cdot \underbrace{(\vec{v} T)}_{\text{Advection}} + \underbrace{R_T}_{\text{Sinks and Sources}}, \quad (2.8)$$

where  $R_T$  describes sources and sinks of heat. In the Lagrangian frame the heat transport equation simplifies to

Heat transport in Lagrangian frame

$$\frac{dT}{d\tau} = \frac{1}{\rho C} \nabla \cdot (\kappa \nabla T) + R_T \quad (2.9)$$

as the advective component vanishes again because the temperature is carried by the moving fluid parcels.

## 2.2 Overview of Fluid Simulation Approaches

While the underlying equations of fluid motion have been discussed, the methodology to discretize the equations is yet to be described. Mainly, there exist mesh-based methods that take on the Eulerian viewpoint and mesh-free or particle-based methods that follow the Lagrangian viewpoint. Additionally, there are hybrid methods that try to take advantage of the strengths of both representations.

Although simulation approaches can vary considerably, they all should fulfill a common set of requirements in order to allow for convincing interactive applications:

Requirements to fluid solvers

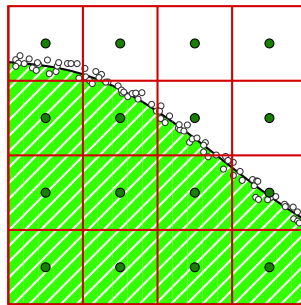
- conservation of quantities, especially in free surface flows,
- ability to enforce incompressibility,
- computational efficiency,
- results have to be visually plausible

In the following, common mesh-based, particle-based and hybrid fluid simulation approaches will be outlined and discussed. There exist many optimizations to each simulation method that are able to alleviate specific shortcomings that are out of the scope of this thesis.

## 2.2.1 Mesh-based Approaches

Fluid animation started out with Eulerian methods that discretize the simulation domain using regular grids [FM96]. Staggered grids store fluid properties in cell centers and velocities at cell faces in order to increase stability [Har64, FSJ01]. Most solvers follow the concept of splitting [Bri08], i.e., velocities are first updated according to external forces and viscosity before quantities and velocities are advected. Lastly, velocities are projected onto a divergence-free field by efficiently solving the pressure Poisson equation. While a direct forward advection [FM96] is easy to implement, the unconditionally stable semi-Lagrangian advection (SLA) alleviates time step restrictions by tracing the velocity back in time to calculate the advection [Sta99]. Advection is still highly dissipative and leads to a visible damping of the fluid motion and loss of mass and volume. The blurring of sharp features causes small droplets and splashes to disappear.

To alleviate volume loss at free surfaces, the surface resolution should generally not be restricted to the grid resolution [FM96]. In order to track liquid surfaces, massless marker particles that are passively advected with the flow have been employed [Har64, FM96]. A combined surface tracking using particles and a level-set representation (PLS) [FF01, EMF02] more stably advects liquid surfaces. Moreover, numerical dissipation has been addressed by using higher order [SFK\*08] and conservative advection schemes [LAF11, CM14]. Fig. 2.1 shows a schematic simulation using semi-Lagrangian advection and the particle level-set surface representation.



**Figure 2.1:** A mesh-based simulation using a staggered grid. Quantities are stored in cell centers (dark green points) while velocity components are stored on cell faces (red). The liquid surface is tracked using the particle level-set method (white points).

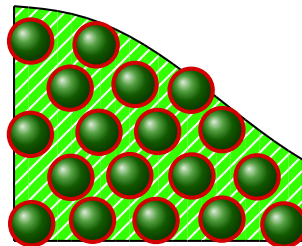
Although semi-Lagrangian advection suffers from numerical dissipation, it is in widespread use due to its simplicity especially when simulating gaseous substances [FM97, FSJ01, PTC\*10]. Vorticity confinement is used to inject fine

turbulent swirling motion back to the simulation that has been damped [FSJ01]. Marker particles are often employed for rendering smoke [Bri08] and turbulence particles increase the details of simulations on coarse grids [PTC\* 10].

### 2.2.2 Particle-based Approaches

Lagrangian methods like SPH have become very popular in computer graphics [DC96, MCG03] due to their flexibility and their ability to stably handle free surfaces and dynamic boundaries. In SPH, particles are used as carriers of mass that is automatically preserved throughout the simulation. Advection is solved correctly by moving particles according to the velocity field. Fig. 2.2 shows a schematic particle-based simulation.

SPH



**Figure 2.2:** A Lagrangian simulation using particles. The particles carry all information and are moved with velocity field in order to calculate the advection.

While grid-based methods can efficiently access neighborhoods as they are implicitly defined by the mesh, in particle-based methods neighborhoods have to be tracked explicitly. To that end, background grids [HKK07c, Gre09], compact spatial hashing [IABT11] or hierarchical [ATO16] data access structures are employed into which the particles get sorted. As interpolations of quantities and their derivatives are based on the particle neighborhood, irregular sampling and underresolved neighborhoods can cause instable behavior [Mon05]. In order to increase stability, artificial viscosity is usually employed, which, however, damps the fluid motion [Mon05, IOS\* 14]. In order to achieve more turbulent fluids, vorticity confinement [MMCK14] or a micropolar model [BKKW17] in which particles also carry a rotational component have been employed.

Particle neighborhoods

Turbulence modeling

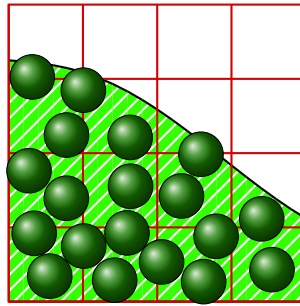
Although the advection term is easily solved in SPH, solving the pressure term depends on derivatives and is more involved. It can either be solved for compressible fluids using an equation of state [BT07, MCG03] or for incompressible fluids using iterative [SP09b] and implicit [ICS\* 14, BK15] or constraint-based solvers [MM13]. Pressure has also been solved by coupling SPH to a grid [RWT11] or using a coarse incompressible SPH simulation that is coupled to finer FLIP particles [CIPT14].

Incompressibility

### 2.2.3 Hybrid Approaches

PIC Hybrid approaches, originating from the Particle-in-cell (PIC) method [EH57], solve the pressure projection on a grid and use particles to solve the advection step. Therefore, quantities and velocities are interpolated between particles and grid cells. While this alleviates the loss of mass and numerical dissipation of grid-based advection, direct interpolation between particles and grid introduces a damping of the dynamics and a loss of momentum. Fig. 2.3 shows a schematic hybrid particle-grid-based approach. By updating particle veloci-

Loss of momentum



**Figure 2.3:** A hybrid simulation using both particles and grid cells. Quantities and velocities are advected using particles (green points) while the velocity is projected onto a divergence free field using the grid (red) and interpolated back to the particles.

FLIP ties according to the time rate of change of grid velocities the Fluid-Implicit-Particle method (FLIP) [BR86] eliminates the damping of PIC simulations. This, however, comes at the cost of particle resampling [ATW13] due to clustering or void space between particles as there is no interaction between particles. Using a combination of PIC and FLIP interpolation is widely used in computer graphics [ZB05] as it increases stability compared to pure FLIP at lower numerical damping than PIC. The numerical damping due to interpolating between grid and particles in PIC has also been addressed by storing locally affine (APIC) [JSS\*15] or polynomial (PolyPIC) [FGG\*17] descriptions of the velocity field. While conservation of momentum can well be improved, both methods have large memory requirements. In narrow-band FLIP [FAW\*16], the use of FLIP particles is restricted to a small region around the liquid surface to reduce both the memory footprint and the computational cost. Chentanez et al. [CMK15] couple a shallow water simulation for large scale fluid scenarios with a grid and a particle-based surface simulation to locally resolve fine details.

Particle resampling

Extensions to PIC

Narrow-band FLIP

Vortex methods

In contrast to vorticity confinement which only amplifies existing vortices, vortex methods are based on the curl of the Navier-Stokes equations [SRF05, CCB\*08] and can simulate highly turbulent phenomena like explosion. Vortex

particles carry the vorticity while the pressure is solved on a grid. The approach, however, is quite expensive as the Poisson equation becomes vector valued.

### 2.2.4 Comparison Between Different Approaches

Simulation methods perform differently well according to the requirements formulated in Sec. 2.2. Tab. 2.1 gives a short summary of the general strengths and weaknesses as discussed above.

**Table 2.1:** Comparison of different fluid solvers. Advantages of solvers with respect to the requirements are highlighted in green, disadvantages in red and general properties in black.

Method \ Requirement	SLA	PIC-FLIP/APIC	SPH
Conservation	Numerical dissipation	Conservative advection	
Incompressibility	Efficiently solvable on grid		Stable gradients are difficult
Efficiency	Unconditionally stable, efficient incompressibility	Interpolation between grid and particles	Costly search for neighbors and incompressibility
Plausibility	Visible loss of mass and momentum	Damped dynamics, particle clustering	Plausible if incompressible

A recent user study of the visual plausibility of different methods [UHT17] revealed that incompressible SPH simulation was superior to grid-based and hybrid methods at comparable spatial resolutions. Hybrid solvers, especially APIC, however, yielded more plausible results when the simulation resolution was adjusted to a given time budget. There was no simulation method that generally outperformed the remaining approaches, yet, comparisons of more specialized simulation methods under more versatile scenarios still have to be conducted.

Visual plausibility study

In the following, all liquids are simulated using incompressible SPH while the gas phase used in Chapter 4 is simulated using a grid-based solver using semi-Lagrangian advection and vorticity confinement.

## 2.3 Time Integration

As analytic solutions cannot be found in most cases, the time evolution of, e.g., a particle's quantities like its position due to its changing velocity

$$\mathbf{x}^{\tau+\Delta\tau} = \mathbf{x}^\tau + \int_{\tau}^{\tau+\Delta\tau} \vec{v}(s, \mathbf{x}^s) ds \quad (2.10)$$

Explicit numerical integration

has to be numerically solved using time discretization schemes. The simplest method is a direct numerical integration using a forward Euler scheme as

$$\mathbf{x}^{\tau+\Delta\tau} = \mathbf{x}^\tau + \vec{v}(\tau, \mathbf{x}^\tau) \Delta\tau, \quad (2.11)$$

Runge-Kutta method

where the derivative  $\vec{v} = \frac{\partial \mathbf{x}}{\partial \tau}$  is approximated as a piecewise constant function. By predicting intermediate steps, the Runge-Kutta method achieves a consistency of fourth order (RK4). The final approximation of a function

$$\mathbf{x}^{\tau+\Delta\tau} = \mathbf{x}^\tau + \frac{\Delta\tau}{6} (\vec{v}_1 + 2\vec{v}_2 + 2\vec{v}_3 + \vec{v}_4) \quad (2.12)$$

is a weighted sum of individual approximations  $\vec{v}_1 = \vec{v}(\tau, \mathbf{x}^\tau)$ ,  $\vec{v}_2 = \vec{v}(\tau + \frac{\Delta\tau}{2}, \mathbf{x}^\tau + \frac{\Delta\tau}{2} \vec{v}_1)$ ,  $\vec{v}_3 = \vec{v}(\tau + \frac{\Delta\tau}{2}, \mathbf{x}^\tau + \frac{\Delta\tau}{2} \vec{v}_2)$ ,  $\vec{v}_4 = \vec{v}(\tau + \Delta\tau, \mathbf{x}^\tau + \Delta\tau \vec{v}_3)$  that is able to reduce the accumulated error [BZBP09].

CFL conditions

Generally, the propagation of a signal, i.e., the displacement of a particle in one time step may not exceed the spatial resolution, i.e., the particle size. To prevent such cases, Courant-Friedrichs-Lewy (CFL) conditions can be defined that limit time steps according to the current particle configuration. Usually the current forces and velocities are used in order to adjust  $\Delta\tau$  so that

$$\Delta\tau \leq \lambda_{\vec{a}} \sqrt{\frac{h}{\|\vec{a}^{\max}\|}} \quad \text{and} \quad \Delta\tau \leq \lambda_{\vec{v}} \frac{h}{\|\vec{v}^{\max}\|}, \quad (2.13)$$

where  $\vec{v}^{\max}$  and  $\vec{a}^{\max}$  are the maximum velocity and acceleration of all particles and  $\lambda_{\vec{v}} = 0.4$  and  $\lambda_{\vec{a}} = 0.25$  are empirically determined constants [Mon05, IAGT10]. CFL conditions for heat and concentration transport can also be defined, however, in most cases they are not necessary, because the particle movement is the limiting factor at the scales of interest [CM99].

Implicit numerical integration

Another family of numerical integration schemes, i.e., implicit integration, has already been mentioned in the context of pressure solvers. Although implicit integration does not increase the rate of convergence, it yields unconditionally stable approximations and thus allows for large time steps in simulations. The backward Euler scheme is simply expressed as

$$\mathbf{x}^{\tau+\Delta\tau} = \mathbf{x}^\tau + \vec{v}^{\tau+\Delta\tau} \Delta\tau, \quad (2.14)$$



where the derivative  $\vec{v}$  of the next time step is used instead of the derivative of time  $\tau$  in the explicit Euler scheme [BZBP09].

For the integration of equations of motion, semi-implicit methods like the Euler-Cromer scheme are often used. Therefore, the velocity is first integrated using a forward Euler step as  $\vec{v}^{\tau+\Delta\tau} = \vec{v}^\tau + \Delta\tau \vec{a}^\tau$ . The position is then implicitly updated using the already updated velocity as  $\mathbf{x}^{\tau+\Delta\tau} = \mathbf{x}^\tau + \Delta\tau \vec{v}^{\tau+\Delta\tau}$  [IOS\* 14].

Euler-Cromer  
scheme

In this thesis, the Euler-Cromer scheme is adopted for SPH simulations while the Runge-Kutta method is used in Chapter 6 to trace streamlines.

## 2.4 Rendering and Visualization

Simulation results are displayed via rendering and visualization. In rendering, the focus usually lies on the visually plausible appearance of a scene. Fluid rendering can be divided into surface and volume rendering techniques. Visualization, in contrast, aims at revealing important processes, e.g., the spatial distribution or temporal evolution of quantities, that otherwise would be missed. Visualization typically uses scalar or vector fields that are visualized as volumes or as glyphs, characteristic lines and surfaces. Sections 2.4.1 to 2.4.3 describe surface and volume rendering as well as vector field visualization techniques with a focus on particle data.

### 2.4.1 Surface Rendering

Rendering of particle-based surfaces can be subdivided into high-quality offline methods and into interactive screen space methods.

Fluid surfaces are usually implicitly described and rendered indirectly using a Marching Cubes triangulation [LC87]. Early descriptions used meta-balls [Bli82] or iso-surfaces of the color field [MCG03] which, however, yield quite blobby surfaces. Taking particle radii  $r_i$  into account and calculating the distance between the weighted average particle position  $\bar{\mathbf{x}}$  and the current position  $\mathbf{x}$  as [ZB05]

Implicit surface  
reconstruction

$$\phi(\mathbf{x}) = \|\mathbf{x} - \bar{\mathbf{x}}\| - \bar{r}(\mathbf{x}) \quad (2.15)$$

$$\bar{r} = \hat{w}_i(\mathbf{x}) r_i \quad (2.16)$$

$$\bar{\mathbf{x}} = \hat{w}_i(\mathbf{x}) \mathbf{x}_i, \quad (2.17)$$

yields a smooth signed distance function, where  $\hat{w}_i(\mathbf{x}) = \frac{w_i(\mathbf{x})}{\sum_j w_j(\mathbf{x})}$  with  $w_i(\mathbf{x}) = w(\|\mathbf{x} - \mathbf{x}_i\|)$  is a corrected kernel function (see Sec. 3.1.3). The zero level set of  $\phi$  yields the fluid surface. Between small splashes that are separated by a

**Spurious artifacts**  
**Artifact correction** distance of about  $\bar{r}$ , spurious fluid artifacts can appear. These can be alleviated by modifying the distance  $\bar{r}$  according to the largest Eigenvalue of the Jacobian of  $\bar{\mathbf{x}}$  which is large in proximity to artifacts [SSP07], by density decay functions [OCD11] or by using anisotropic smoothing kernels [YT13]. The resulting level set can be tessellated using uniform Marching Cubes [LC87, AIAT12] or with adaptive methods [AAOT13]. Direct ray casting [GSSP10, OCD11] and ray tracing [MWE16, BSS\*18, WTYH18] of iso-surfaces has also been presented but is limited to small sets of particles if interactive frame rates are desired.

**Screen space rendering** In order to achieve smooth surfaces in interactive applications, screen space approaches are usually employed. Particles are either splatted onto the screen as spheres [vdLGS09] or as ellipsoids that are calculated according to the particle's anisotropy [YT13, MM13]. Smoothing is achieved by applying separable binomial filtering [MSD07] or screen space curvature flow [vdLGS09] to the splatted depth values. However, as only the foremost surface can be displayed, depth perception of transparent renderings is limited especially in complex fluid scenes. For moderate particle numbers, multiple surface layers can be rendered using a perspective grid of binary voxels that is constructed on-the-fly and smoothed in screen space [ZD15, ZD17].

**Large data sets** Large data sets of opaque particles can be efficiently rendered using P-k-d-trees [WJP14]. Hierarchical binary volume representations have been used to efficiently render surfaces of very large particle data which however have to be preprocessed offline [RCSW14]. Particles have also been mapped to sparse voxel structures that are used for volume and iso-surface ray tracing [Hoe16].

An implicit surface definition that is able to render dynamic contact angles for liquids in contact with rigid surfaces will be presented in Chapter 4.

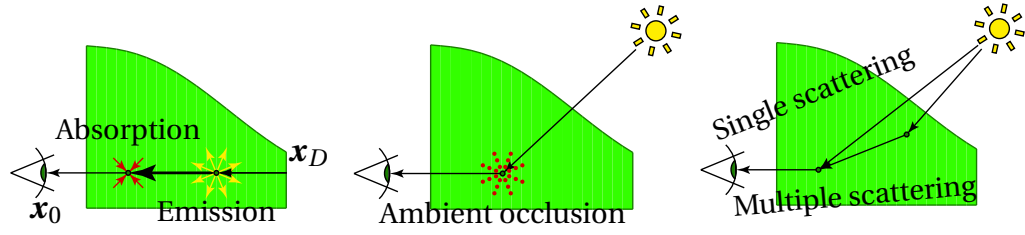
## 2.4.2 Volume Rendering

**Volume rendering integral** In direct volume rendering (DVR), a physically-based model of light transport through a participating medium is evaluated that can comprise emission, adsorption and scattering. While the former two do not change the direction of light, scattering causes light rays to be diffracted and sent off in different directions. The volume rendering integral assumes viewing rays are cast from the viewer at  $\mathbf{x}_0$  through the medium to exit point  $\mathbf{x}_D$  as

$$I(\mathbf{x}_D, \vec{\omega}) = \int_{\mathbf{x}_0}^{\mathbf{x}_D} I(\mathbf{x}', \vec{\omega}) \cdot T(\mathbf{x}_0, \mathbf{x}') d\mathbf{x}' + I_{\text{bg}}(\mathbf{x}_0, \vec{\omega}) T(\mathbf{x}_0, \mathbf{x}_D), \quad (2.18)$$

$$T(\mathbf{x}_i, \mathbf{x}_j) = e^{-\int_{\mathbf{x}_i}^{\mathbf{x}_j} \sigma_\alpha(\mathbf{x}') d\mathbf{x}'},$$

where  $I(\mathbf{x}, \vec{\omega}) = I_e(\mathbf{x}) + I_s(\mathbf{x}, \vec{\omega})$  models emitted and scattered radiance in direction  $\vec{\omega}$  towards the viewer,  $I_{\text{bg}}$  the background radiance, and  $T$  the transparency



(a) At each point, light is emitted and absorbed on the way to the viewer  
 (b) Light is attenuated by absorption in the local neighborhood  
 (c) External and internal light emission is scattered towards the viewer

**Figure 2.4:** Different simplifications of the volume rendering model. The volume is shown in green. Black arrows indicate light traveling towards the viewer on the left. Emission is highlighted in yellow and absorption in red.

that is attenuated due to the medium's absorption  $\sigma_\alpha$  [JSYR14].

Multiple scattering means to recursively evaluate Eq. (2.18) with  $I_s(\mathbf{x}, \vec{\omega}) = \int_{\Omega} s(\mathbf{x}, \vec{\omega}_i, \vec{\omega}) I(\mathbf{x}, \vec{\omega}_i) d\vec{\omega}_i$  taking scattered contributions  $s$  from all directions  $\vec{\omega}_i$  on the unit sphere  $\Omega$  into account. Due to its computational effort, scattering is usually simplified or omitted. Single scattering of external light approximates  $I_s$  by local illumination models like BRDFs using the gradient of the scalar field as normal vector [Lev88, EHK\*06, HLSR08]. Local ambient occlusion attenuates light contributions according to the absorption of surrounding samples [HLY10] and deep shadow maps approximate transparency as seen from a light source in a multi-layer texture to render shadows [HKSB06]. Global illumination can be approximated by simulating light transport as a convection-diffusion problem [ZM13]. Fig. 2.4 illustrates different simplifications of volume rendering integral. For a detailed overview, the reader is referred to the survey papers by Max and Chen [MC10] and Jönsson et al. [JSYR14].

A widespread simplification for interactive applications only considers emission and absorption of the medium. As Eq. (2.18) is analytically solvable only for few cases [MC10], it is often expressed as a Riemann sum as shown in Fig. 2.5. The ray path  $0 \dots D$  is therefore subdivided into  $N$  segments of equal length  $\Delta s = D/N$ , where the  $i$ -th segment spans the interval  $[s_i, s_{i+1}]$ . The radiance of segment  $i$  is approximated as  $I_i = I_e(s_i)\Delta s$  and the transparency as  $T_i = e^{-\sigma_\alpha(s_i)\Delta s}$ , so that  $T(0, D)$  simplifies to  $e^{-\int_0^D \sigma_\alpha(t) dt} \approx e^{-\sum_{i=0}^{N-1} \sigma_\alpha(s_i)\Delta s} = \prod_{i=0}^{N-1} e^{-\sigma_\alpha(s_i)\Delta s} = \prod_{i=0}^{N-1} T_i$  [Max95]. The approximation of Eq. (2.18) then reads

$$I(D) \approx \sum_{i=0}^{N-1} I_i \prod_{j=0}^{i-1} T_j + I_{bg} \prod_{j=0}^{N-1} T_j. \quad (2.19)$$

While higher order integration techniques can be applied, their accuracy is

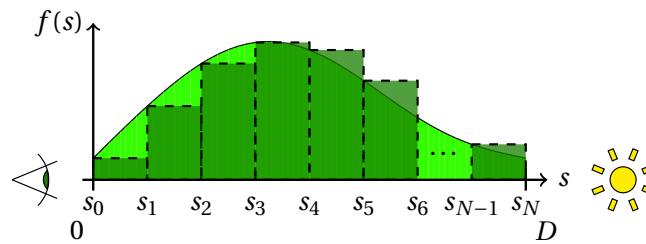
Volume illumination

Volume shadowing

Emission-absorption model

Discretization

Adaptive volume rendering



**Figure 2.5:** Volume rendering of a signal using a Riemann sum approximation that subdivides the interval  $[0, D]$  into  $N$  segments. The viewer is placed in the origin and background illumination is emitted from outside the medium.

restricted due to discontinuities, e.g., at material boundaries. Thus, sampling techniques in which segment lengths are adapted to the properties of the integrand are preferred [EHK\*06, MC10].

Transfer functions  
and classification

Visual attributes, i.e., radiance and transparency, are calculated from the medium's scalar quantities via transfer functions. Pre-classification directly maps quantities at each data point to visual attributes. As features of non-linear transfer functions can be missed when interpolating pre-classified values at ray samples, post-classification applies transfer functions after interpolating quantities [EHK\*06]. Moreover, the volume rendering integral can be pre-integrated for each pair of quantities and stored in lookup tables [EKE01].

Pre-integration

Application to SPH

Particle splatting

While SPH simulations evaluate field quantities only at particle positions, direct volume rendering requires a continuous sampling along rays. To prevent costly sampling, particles can be splatted in any order with an emission-only model [FSW09] or using depth-sorted particles to approximate the emission-absorption model [HE03]. When directly splatting particles onto the screen, only pre-classification can be applied. Particle quantities can also be splat into an intermediate volumetric grid for ray casting [KC05, FAW10], which allows for post-classification, but can cause interpolation artifacts and imposes severe memory traffic. Direct rendering of unstructured particle data requires efficient access structures. Object space data structures like octrees [OKK10, RTW13, HE03] can be used as well as perspective data structures that align with the viewing rays [FAW10].

Particle access

The emission-absorption volume rendering presented in Chapter 5 uses a perspective data structure, adaptive sampling step sizes and post-classification.

### 2.4.3 Vector Field Visualization

While surface and volume rendering are able to convey mass and concentration distributions, vector field visualization techniques are necessary in order to

convey directional information of the fluid flow.

General vector field visualizations use regularly sampled geometric primitives like arrows and lines that may vary in size according to the vector magnitude [JH04]. In line integral convolution (LIC) noise textures are convolved with vector fields to yield a dense representation [CL93]. By mapping vector fields to scalar data, volume rendering can be applied as discussed in Sec. 2.4.2.

General  
visualization  
primitives

Flow visualization can be expressively conveyed through integral lines, i.e., streamlines, pathlines, and streaklines. Integral lines are based on the velocity field  $\vec{v}(\mathbf{x}, \tau)$  that maps positions to velocities for each point in time  $\tau$ . By tracing massless marker particles from a position  $\mathbf{x}_0$  at starting time  $\tau_0$  through the velocity field, an arbitrary position on a streamline is found as [JH04]

Integral lines

$$\mathbf{x}^{\text{stream}}(\tau, \mathbf{x}_0, \tau_0) = \mathbf{x}_0 + \int_{\tau_0}^{\tau} \vec{v}(\mathbf{x}^{\text{stream}}(s, \mathbf{x}_0, \tau_0), \tau_0) ds, \quad (2.20)$$

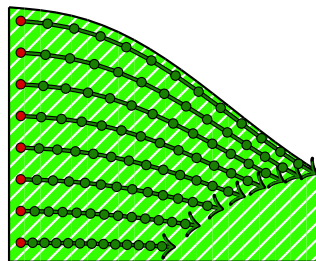
and on a pathline as

$$\mathbf{x}^{\text{path}}(\tau, \mathbf{x}_0, \tau_0) = \mathbf{x}_0 + \int_{\tau_0}^{\tau} \vec{v}(\mathbf{x}^{\text{path}}(s, \mathbf{x}_0, \tau_0), s) ds. \quad (2.21)$$

For a streakline, marker particles are released at subsequent points in time  $\tau \in [\tau_0, \tau_{\text{end}}]$  and integrated to time  $\tau_{\text{end}}$  as

$$\mathbf{x}^{\text{streak}}(\tau, \mathbf{x}_0, \tau_{\text{end}}) = \mathbf{x}_0 + \int_{\tau}^{\tau_{\text{end}}} \vec{v}(\mathbf{x}^{\text{path}}(s, \mathbf{x}_0, \tau), s) ds. \quad (2.22)$$

Connecting subsequent positions over time yields the actual integral lines. While streamlines assume a steady velocity field, pathlines and streaklines are traced assuming an unsteady flow. Fig. 2.6 shows a schematic visualization of



**Figure 2.6:** Vector field visualization using streamlines. Lines are seeded from the red points and traced along the velocity field as depicted by the green points.

a set of streamlines that are seeded from a line of seed points. The precision of the results and the rendering performance depend both on the time integration scheme and the time-step. Thus, adaptive sampling of integral lines can be employed [CPK09].

Adaptive integration

Integral lines are usually visualized by geometric means like tubes and ribbons [MLP\*10]. Additionally, illustrative techniques enhance renderings by adding directional information, by reducing cluttering or by improving depth perception [BCP\*12]. Surfaces of integral lines can be interactively rendered by connecting neighboring lines to meshes [BFTW09] or by densely seeding lines according to the current viewpoint [MSE14].

Stream surfaces

Occlusion and cluttering

While it is desired to capture all important flow features, especially in 3D flows, dense representations cause occlusion and cluttering. Several techniques have been proposed to improve the visual presentation by automatically adjusting opacities in LIC [FW08] and in line-based rendering according to the current viewport [GRT13, GTG17]. By automatically seeding lines or surfaces only in important flow regions [ELM\*12] and by a hierarchical clustering and splitting of streamline bundles [HCCC12] followed by an adjustment of the line thickness in screen space [KFW16] cluttering can be avoided.

Automatic line seeding

In Chapter 6 a vector field visualization for advective-diffusive flows is presented which is based on streamlines.

## SPH-based Simulation of Fluid Transport

*This chapter describes the theoretical foundations of SPH-based simulations and presents models to discretize the fluid transport equations as they are used throughout the remainder of this thesis. The chapter closes by presenting adaptive methods and implementation strategies for efficient and parallel simulations.*

*For a comprehensive introduction to the SPH method, the reader is referred to the survey papers by Monaghan [Mon05] and Ihmsen et al. [IOS\* 14].*

---

The Lagrangian smoothed particle hydrodynamics method (SPH) has been introduced by Lucy [Luc77] and Gingold and Monaghan [GM77] for the simulation of astrophysical problems. While the first simulations of liquids in fluid animation were based on compressible fluid models [MCG03, BT07], SPH has progressed rapidly. Since then, the simulation of highly deformable bodies [DC96], incompressible fluids [SP09b, MM13, ICS\* 14, BK15], multi-phase flows [SP08, RLY\* 14] and surface tension effects [BT07, AAT13] have been realized and SPH is easily able to interact with static [HKK07c] and dynamic rigid [BTT09, AIA\* 12] and elastic bodies [MMCK14, YCL\* 17]. The principle idea of SPH is to represent a continuous medium in terms of a discrete number of particles which act as carriers of physical quantities. Continuous fields are reconstructed at arbitrary positions from quantities of neighboring particles using a weighting kernel. The momentum and transport equations can then be discretized in terms of interactions between neighboring particles.

In the following, the theoretical foundations of SPH simulations are discussed in Sec. 3.1 before describing how fluid transport equations are discretized using SPH in Sec. 3.2. In Sec. 3.3, adaptive simulation methods are presented that aim at reducing the computational overhead by adapting particle sizes or time steps and Section 3.4 discusses algorithmic means to achieve efficient simulation especially on massively parallel platforms like GPUs.

## 3.1 SPH Interpolation and Kernel Functions

The derivation of the SPH interpolation starts by assuming an infinite number of sampling points  $\mathbf{x}$  in the simulation domain  $\Omega$ . A function  $Q$  can then be described in integral form as

$$Q(\mathbf{x}) = \int_{\Omega} Q(\mathbf{x}') \delta(\|\mathbf{x} - \mathbf{x}'\|) d\mathbf{x}', \quad \delta(x) = \begin{cases} \infty & x = 0 \\ 0 & \text{else,} \end{cases} \quad (3.1)$$

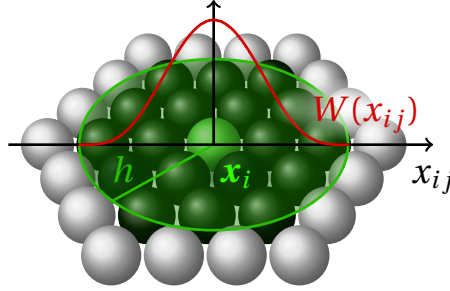
where  $d\mathbf{x}'$  is the differential volume element and  $\delta$  the Dirac delta function. For computations, the Dirac delta function is approximated by a weighting kernel  $W$  as

$$Q(\mathbf{x}) \approx \int_{\Omega} Q(\mathbf{x}') W(\|\mathbf{x} - \mathbf{x}'\|, h) d\mathbf{x}', \quad (3.2)$$

where  $h$  denotes the smoothing length. Finally, the field is discretized for an arbitrary position  $\mathbf{x}$  by replacing the volume integral with a sum over a finite number of interpolation points, the particles, using particle quantities  $Q_j$  as

$$Q(\mathbf{x}) \approx \langle Q(\mathbf{x}) \rangle = \sum_j Q_j V_j W(\|\mathbf{x} - \mathbf{x}_j\|, h), \quad (3.3)$$

where  $V_j$  is a particle's dynamic volume and  $\langle \cdot \rangle$  denotes the SPH-interpolation. The discretization around another particle  $i$  as shown in Fig. 3.1 is usually



**Figure 3.1:** SPH-interpolation for a particle  $i$  weights contributions of neighboring particles (green) inside its support radius  $h$  (green circle) by the kernel function  $W_{ij}$  according to their distance.

written as  $\langle Q_i \rangle = \langle Q(\mathbf{x}_i) \rangle = \sum_j Q_j V_j W(\|\mathbf{x}_i - \mathbf{x}_j\|, h)$ . In the following, however, angle brackets will be omitted as is usually done in literature [Mon05, IOS\* 14].

The kernel function

$$W(x_{ij}, h) = W_i(\mathbf{x}_j, h) = W_{ij} \quad (3.4)$$

weights contributions of particles  $j$  in the neighborhood of particle  $i$  depending on their distance  $x_{ij} = \|\tilde{\mathbf{x}}_{ij}\| = \|\mathbf{x}_i - \mathbf{x}_j\|$ . The kernel function usually should satisfy the following properties [Mon05]

Kernel properties



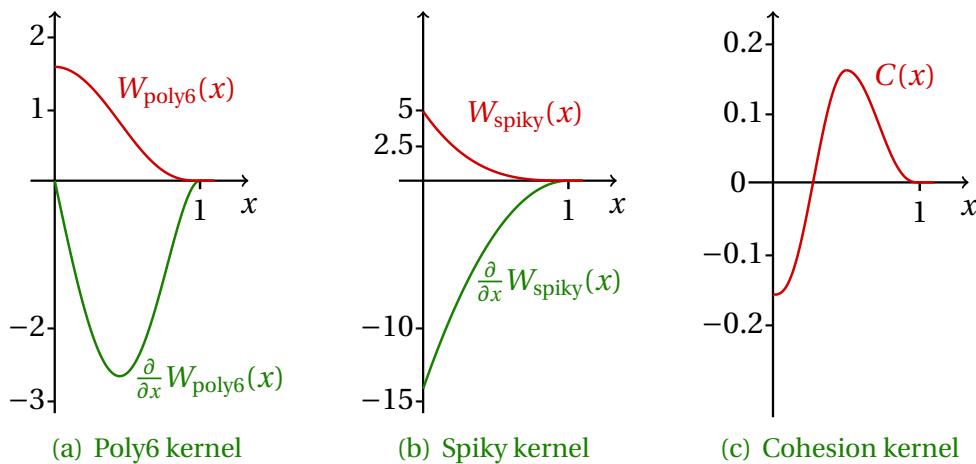
- Normalization, i.e.,  $\int_{-h}^h W(x, h) dx = 1$
- Even function, i.e.,  $W(x, h) = W(-x, h)$
- Converges to Dirac delta, i.e.,  $\lim_{h \rightarrow 0} W(x, h) = \delta(x)$
- Non-negativity, i.e.,  $W(x, h) \geq 0$
- Compact support, i.e.,  $x > h \implies W(x, h) = 0$ .

While for physical correctness it is best to assume a Gaussian kernel [Mon92, GM77], efficient simulations require a compact support radius as the computational cost of simulations is directly linked to the number of neighbors that have to be considered. The number of neighbors, however, determines the stability of interpolation and, in particular, derivatives are prone to errors if there are only few neighbors [IOS\* 14].

Small support radii

In practice, spline functions, e.g., the cubic spline, the poly6 or the spiky kernels are preferred. Especially in computer graphics the poly6 kernel is very popular because the particle distance only appears squared so that no square roots have to be calculated [MCG03]. Computing pressure forces, however, causes particles to clump together because its derivative vanishes as  $x$  approaches zero so that there is too little repulsion between close particles. Thus, for pressure calculations the spiky kernel has been proposed [DC96]. Fig. 3.2 shows the poly6 and spiky kernels and their first derivatives that will be used throughout this thesis if not stated otherwise. Depending on the simulated physical quantity, more specialized kernel functions have also been proposed. The cohesion kernel shown in Fig. 3.2(c) is employed to simulate surface tension effects (see Sec. 3.2.1).

Common kernel functions



**Figure 3.2:** The poly6 and spiky kernel functions (red) and their first derivatives (green) and the cohesion kernel (right). The particle distance is plotted on the  $x$ -axis, the kernel weight on the  $y$ -axis. The support radius is  $h = 1$ .

### 3.1.1 Derivatives in SPH

**First derivatives** As SPH interpolates field quantities only in terms of particle quantities  $Q$ , spatial derivatives only have to be applied to the kernel function as

$$\nabla Q_i = \sum_j Q_j V_j \nabla W_{ij}, \quad (3.5)$$

where  $\nabla W_{ij}$  is short for  $\nabla_{\mathbf{x}_i} W(\|\mathbf{x}_i - \mathbf{x}_j\|, h) = \frac{\bar{x}_{ij}}{x_{ij}} \frac{\partial}{\partial x_{ij}} W(x_{ij}, h)$ . While this formulation is a valid first derivative, it does not vanish for constant field values and also the sum of gradients of all particles is not zero which, however, is necessary if, e.g., pressure accelerations should be calculated in a momentum preserving way.

**Vanishing gradient for constant fields**

Monaghan [Mon05] therefore proposed two different discretizations of gradients in SPH by introducing an additional term in the derivative as  $\nabla(\Phi Q)$ , where  $\Phi$  is any differentiable function. After applying the product rule and rearranging, the gradient can be written as  $\nabla Q = \frac{1}{\Phi}(\nabla(\Phi Q) - Q\nabla\Phi)$ . By applying the standard SPH gradient from Eq. (3.5) on the right hand side and simply assuming  $\Phi = 1$ , the gradient definition

$$\nabla Q_i = \sum_j (Q_j - Q_i) V_j \nabla W_{ij} \quad (3.6)$$

**Antisymmetric first derivatives**

is obtained that vanishes for constant fields [Mon05]. For deriving the acceleration due to pressure differences,  $\Phi = \frac{1}{\rho}$  can be used which after using the quotient rule yields  $\frac{\nabla Q}{\rho} = \nabla\left(\frac{Q}{\rho}\right) + \frac{Q\nabla\rho}{\rho^2}$  and by applying SPH interpolation and rearranging reads

$$\nabla Q_i = \rho_i \sum_j m_j \left( \frac{Q_j}{\rho_j^2} + \frac{Q_i}{\rho_i^2} \right) \nabla W_{ij}. \quad (3.7)$$

This antisymmetric formulation has been shown to conserve both angular and linear momentum when used to calculate pressure acceleration, i.e., when  $Q = p$  is used [Mon05].

**Second derivatives**

For the standard Laplace operator  $\nabla^2$  which in SPH formulation is given as

$$\nabla^2 Q_i = \sum_j Q_j V_j \nabla^2 W_{ij}, \quad (3.8)$$

**Antisymmetric second derivatives**

a similar problem arises, i.e., the Laplacian does not vanish for constant  $Q$ . This causes loss or gain of quantities if it is applied to diffusion equations. Additionally, the formulation is very sensitive to particle disorder [Mon05]. Instead of directly using the second derivative of the kernel, Morris et al. [MFZ97]

proposed a more stable formulation:

$$\nabla^2 Q_i = 2 \sum_j \left( \frac{Q_i - Q_j}{x_{ij}} \right) V_j \frac{\mathbf{x}_i - \mathbf{x}_j}{x_{ij}} \cdot \nabla W_{ij} = 2 \sum_j (Q_i - Q_j) V_j \frac{\|\nabla W_{ij}\|}{x_{ij}}, \quad (3.9)$$

in which a first derivative of the kernel and a finite difference scheme for the quantities is used.

### 3.1.2 Calculating the Fluid Density and Volume

While the particle's mass  $m_i$  is an intensive property of each particle  $i$ , it does usually not change during a simulation so that mass preservation is implicitly guaranteed. However, the standard SPH interpolation does depend on the volume of each particle which is an extensive property and thus has to be recomputed every time the particle configuration changes. According to Monaghan, the volume is generally expressed as  $V_i = \frac{m_i}{\rho_i} = \frac{m_i}{\sum_j m_j W_{ij}}$  [Mon05].

If, however, multiple fluids of different rest densities are simulated, this formulation leads to severe clumping at fluid interfaces [SP08]. Hence, a dimensionless formulation based on the *particle number density*

Particle number density

$$n_i = \sum_j v_j W_{ij} \quad (3.10)$$

can be used to calculate the dynamic volume  $V$  as

Dynamic volume

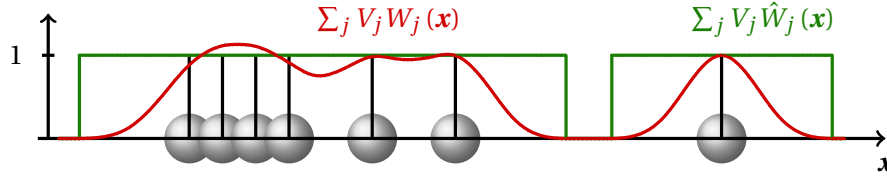
$$V_i = \frac{v_i}{n_i}, \quad (3.11)$$

where  $v$  is the rest volume of a fluid particle [OHB\*13]. The density is then calculated as  $\rho_i = \frac{m_i}{V_i}$ . This formulation will be assumed in all later SPH interpolations. In the case of constant rest density, however, the above formulations are equal.

### 3.1.3 Corrected SPH Interpolation

Although the standard SPH interpolation (see Eq. (3.3)) is commonly used in simulations, it is not even able to reconstruct a constant field if the particle distribution is not perfectly regular. The evaluation of field quantities, especially at the fluid surface or at arbitrary non-particle positions, can result in severe under- or overestimation of quantities as shown in Fig. 3.3. In order to be able

Irregular neighborhoods



**Figure 3.3:** Comparison of the reconstruction of a constant quantity field using standard SPH-interpolation (red) and corrected SPH-interpolation (green).

to reconstruct constant quantity fields, *Corrected SPH (CSPH)* interpolation uses a normalized kernel [BK02]

$$\hat{W}_i(\mathbf{x}) = \frac{W_i(\mathbf{x})}{\sum_j V_j W_j(\mathbf{x})} \quad (3.12)$$

based on the Shepard filter [She68]. In the following, the corrected SPH interpolation will be used whenever field quantities have to be calculated for non-particle positions as done in Chapter 5 for volume rendering and in Chapter 6 for vector field visualization.

## 3.2 Discretizing Fluid Transport in SPH

In the following, the discretization of the Navier-Stokes equation (see Eq. (2.5)) and the fluid transport equations (see Eq. (2.7) and Eq. (2.9)) in terms of interactions between SPH particles will be described. In order to allow for an efficient evaluation of the incompressibility, which is mandatory for visually plausible fluid motion, non-pressure and pressure accelerations are usually calculated separately following the splitting approach [Bri08]. The non-pressure accelerations are first evaluated and used to predict particle velocities

$$\vec{v}^* = \vec{v}(\tau) + \Delta\tau \vec{a}^{\text{non-pressure}}, \quad (3.13)$$

which are used to derive pressure values to reduce the density deviations from the rest density. The pressure acceleration is used to calculate the final velocity as

$$\vec{v}(\tau + \Delta\tau) = \vec{v}^* + \Delta\tau \vec{a}^{\text{pressure}}. \quad (3.14)$$

The incompressibility can be enforced using any of the pressure solvers that will later be described. Additionally, transport of heat and concentrations can be simulated. Alg. 3.1 gives a description of a splitting-based SPH simulation.

---

1: <b>while</b> Simulating <b>do</b>		
Particle neighborhoods and volumes		
2:	<b>for all</b> Particle $i$ <b>do</b>	
3:	$N_i \leftarrow$ Neighborhood of particle $i$	▷ see Sec. 3.4
4:	<b>end for</b>	
5:	<b>for all</b> Particle $i$ <b>do</b>	
6:	$V_i \leftarrow$ Calculate dynamic volume	▷ see Sec. 3.1.2
7:	<b>end for</b>	
Particle interactions		
8:	<b>for all</b> Particle $i$ <b>do</b>	
9:	$\vec{a}_i^{\text{non-pressure}} \leftarrow$ Gravity, viscosity, etc.	▷ Non-pressure acceleration
10:	$\vec{v}_i^* \leftarrow \vec{v}(\tau) + \Delta\tau \vec{a}_i^{\text{non-pressure}}$	▷ Predict velocity
11:	$\frac{dQ_i}{d\tau} \leftarrow$ Fluxes due to, e.g., diffusion	▷ Heat and concentration fluxes
12:	<b>end for</b>	
Calculate pressures		
13:	<b>for all</b> Particle $i$ <b>do</b>	
14:	$\vec{a}_i^{\text{pressure}} \leftarrow$ Resolve compression	▷ Pressure acceleration
15:	<b>end for</b>	
Time integration		
16:	<b>for all</b> Particle $i$ <b>do</b>	
17:	$\vec{v}_i \leftarrow \vec{v}_i^* + \Delta\tau \cdot \vec{a}_i^{\text{pressure}}$	▷ Correct velocities
18:	$\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta\tau \cdot \vec{v}_i$	▷ Advect particles
19:	$Q_i \leftarrow Q_i + \Delta\tau \cdot \frac{dQ_i}{d\tau}$	▷ Integrate fluxes
20:	<b>end for</b>	
21:	<b>end while</b>	

---

**Algorithm 3.1:** A generic SPH simulation loop following the splitting approach. The neighborhood search allows for efficient interpolation of particle quantities. Accelerations (green) and quantity fluxes (red) are calculated and finally integrated.

### 3.2.1 Non-pressure Accelerations

Non-pressure accelerations mainly comprise gravity, viscosity, surface tension, and interactions with rigid bodies.

**Viscosity:** Although viscosity is quite low for liquids like water, it is an integral component of the equations of fluid motion. The most widespread model of viscosity is the *artificial viscosity* [Mon05] which has been introduced to

Artificial viscosity computer graphics by Becker and Teschner [BT07] as

$$\vec{a}_i^{\text{viscosity}} = \begin{cases} -\sum_j m_j \nu \frac{\vec{v}_{ij} \cdot \vec{x}_{ij}}{x_{ij} + 0.01h^2} \nabla W_{ij} & \vec{v}_{ij} \cdot \vec{x}_{ij} < 0 \\ 0 & \vec{v}_{ij} \cdot \vec{x}_{ij} \geq 0 \end{cases}, \quad (3.15)$$

where  $\nu$  is an arbitrary parameter that loosely corresponds to the kinematic viscosity and where  $0.01h^2$  in the denominator is used to prevent divisions by zero. Even though there are SPH models of viscosity that are more physically plausible [MFZ97], the artificial viscosity is often preferred because it increases stability [IOS\*14]. For highly viscous fluids like, e.g., tooth paste, implicit viscosity formulations have been derived which are computationally expensive and are not applicable to simulate fluids of low viscosity [PICT15].

Inter-particle  
interaction forces

Continuum surface  
forces

**Surface Tension:** Surface tension is mostly simulated using inter-particle interaction forces (IIF) or continuum surface forces (CSF). IIF models [BT07, TM05, YML\*17] are motivated by molecular attraction between particles that cancels out in the fluid volume but causes a net force at fluid surfaces. CSF models, in contrast, aim at minimizing the surface curvature (CSF) [BKZ92, MCG03, BPHK13] and often only influence particles at the fluid surface which can be detected, e.g., by thresholding a color field gradient [MCG03]. According to a comparison of different surface tension models, no model is equally suited for all use cases and models should be chosen according to the desired fluid effects [HRWE15].

Throughout this thesis, the combined model of Akinici et al. [AAT13] is employed as

$$\vec{a}_i^{\text{tension}} = -\sum_j K_{ij} \gamma m_j C(x_{ij}) \frac{\vec{x}_{ij}}{x_{ij}} - \sum_j K_{ij} \gamma (\vec{n}_i - \vec{n}_j), \quad (3.16)$$

where surface normal  $\vec{n}_i = h \sum_j V_j \nabla W_{ij}$ ,  $K_{ij} = \frac{2\rho_0}{\rho_i + \rho_j}$  is a density dependent weighting term, and  $\gamma$  a user-defined parameter. The first term uses the specially designed cohesion kernel  $C$  (see Fig. 3.2(c) in Sec. 3.1) that causes a repulsion for close particles and an attraction between farther particles with a maximum attraction at rest distance. The second term aims at minimizing the surface curvature.

Static rigid  
boundaries

**Boundary Handling:** Static rigid boundaries such as the scene geometry can be efficiently modeled using signed distance functions [HKK07c] or by explicitly calculating distances between particles and triangle geometries that are efficiently accessed through, e.g., OpenVDB [ATO16]. Distance-based penalty forces can

then be applied to particles [Mon05] or particle velocities and positions can be adjusted by direct forcing approaches [BTT09, IAGT10].

Particle-based representations of rigid objects allow for a consistent dynamic two-way coupling. Rigid objects can be represented as a single layer of particles. The dynamic particle volume is calculated as the inverse number density as  $V_k = \frac{1}{\sum_l W_l}$ , where  $k$  and  $l$  are only rigid particles. The fluid-rigid interaction is realized by mirroring fluid particle pressures to the rigid particles so that these exert a pressure force on the fluid and vice versa. Additionally, friction between rigid and fluid particles can be modeled using artificial viscosity (see Eq. (3.15)) [AIA\*12]. Also for secondary effects like heat transfer between the fluid and rigid objects [SSP07], a consistent particle-based representation is advantageous. In this thesis, rigid objects are thus represented as particles.

Two-way rigid coupling

Although the air phase is commonly ignored, randomly sampled *ghost particles* around the fluid surface [SB12] or an external pressure [HWZ\*14] have been used to mimic an air phase. Gissler et al. [GBP\*17a, GBP\*17b] simulate liquid-air interaction in terms of a constant external velocity field that causes friction at the fluid's surface.

Fluid-air interaction

Many substances like surfactants [FAB\*11] behave differently on the fluid surface than in the bulk. Orthmann et al. [OHB\*13] introduced a consistent surface model that assigns a value to each particle estimating its surface area. By applying a one-sided tent kernel

Consistent surface model

$$\delta_i = \begin{cases} \frac{1}{d} \left(1 + \frac{\phi_i}{d}\right) & \text{if } \phi_i < 0 \\ 0 & \text{else} \end{cases} \quad (3.17)$$

to the implicit surface definition  $\phi_i = \phi(\mathbf{x}_i) = \|\mathbf{x}_i - \sum_j \mathbf{x}_j V_j \hat{W}_{ij}\| - d$  [ZB05], a smoothed surface delta value can be derived. The surface distance  $d$  matches the particle radius. Using corrected SPH interpolation yields the particle's surface area as

$$A_i = V_i \sum_j \delta_j V_j \hat{W}_{ij}. \quad (3.18)$$

This allows to formulate conservative transport processes at the surface like cleansing of rigid objects [OHB\*13] and to stably detect surface particles and will be used in Chapter 4 for the simulation of evaporation and condensation.

### 3.2.2

## Pressure Acceleration and Incompressibility

The last acceleration  $\vec{a}_i^{\text{pressure}} = \frac{\nabla p_i}{\rho_i}$  is due to pressure gradients which are commonly calculated using the antisymmetric formulation of Eq. (3.7) in order to guarantee conservation of linear and angular momentum [Mon05, IOS\*14]

Momentum conservation

(see Sec. 3.1.1). Although this answers the question how pressures lead to an acceleration of fluid particles, the question remains, how to obtain pressure values to enforce incompressibility.

Equation of state solvers

First methods to calculate pressure used equations of state that relate particle densities to pressures. Equations of state for compressible fluids are based on the ideal gas equation [MCG03] and for weakly compressible fluids on Tait's equation [BT07]. Even though equations of state are easily implemented, reducing compressibility requires very small time steps due to their stiffness.

Predictive-corrective incompressibility

Implicit pressure solvers

Solenthaler and Pajarola [SP09b] alleviated the time step restrictions by introducing predictive-corrective incompressible SPH (PCISPH) in which the pressure is iteratively accumulated until particle densities converge to the rest density. In implicit incompressible SPH (IISPH) [ICS\*14] the density invariance condition is discretized by predicting densities due to non-pressure forces and solving for pressures that restore constant rest density after time integration [ICS\*14]. Instead of the density invariance, the divergence-free condition of the velocity field is addressed in divergence-free SPH (DFSPH) [BK15]. In position-based fluids (PBF) [MM13] particle positions are directly optimized assuming a quasi-static problem to resolve density constraints. The reader is referred to the survey by Ihmsen et al. [IOS\*14] for a thorough discussion of pressure solvers.

Position-based fluids

Throughout this thesis, incompressibility in SPH-based fluids is enforced using PCISPH.

### 3.2.3

### Transport of Concentrations and Heat

While the above models only considered the equations of fluid motion, additional fluid properties can be advected with and diffused inside the fluid as discussed in Secs. 2.1.3 and 2.1.4.

Homogeneous diffusion

The time rate of change of concentration  $c$  due to diffusion follows Fick's second law [ALS09]. Assuming a homogeneous medium it can be expressed as [Mon05, CM99]

$$\frac{dc_i}{d\tau} = 2D \sum_j (c_i - c_j) V_j \frac{\|\nabla W_{ij}\|}{x_{ij}}. \quad (3.19)$$

Heterogeneous materials

Heat transfer follows the very similar law of Fourier. However, heat transport is usually not restricted to a homogeneous liquid phase but also encompasses heat transport in rigid objects. Cleary and Monaghan [CM99] arrive at

$$\frac{dT_i}{d\tau} = \frac{V_i}{m_i C_i} \sum_j \frac{4\kappa_i \kappa_j}{\kappa_i + \kappa_j} (T_i - T_j) V_j \frac{\|\nabla W_{ij}\|}{x_{ij}} \quad (3.20)$$



to model heat conduction, where  $C$  denotes the specific heat capacity and  $\kappa$  the heat conductivity. By using  $\frac{4\kappa_i\kappa_j}{\kappa_i+\kappa_j}$  as the effective heat conductivity, heterogeneous materials with substantially varying conductivities can be simulated in SPH [CM99]. In both cases, the discretization of the Laplacian of Eq. (3.9) has been applied.

### 3.3 Adaptive Simulation

Apart from the proper modeling of the fluid transport equations, visually plausible simulations also depend on the spatial resolution. As larger numbers of particles increase both the memory consumption and the computation time, adaptive methods have been introduced that limit the particle refinement to certain areas of interest, e.g., the fluid's surface.

**Spatial Adaptivity:** A simple but effective method to achieve high resolution at the surface and lower resolution in the fluid's bulk is to use separate simulations in each area and indirectly couple the resolutions by artificial forces [SG11, HS13]. However, separate simulations can diverge and mass-preservation is not guaranteed. Instead, splitting and merging of directly interacting particles has been applied in order to adjust particle sizes [APKG07, DC99]. Direct replacement of particles, however, causes instabilities in incompressible fluids. These have been alleviated with the concept of Temporal Blending [OK12] by smoothly adjusting resolutions over time. Recently, the concept of continuous adjustment of particle masses has been introduced [WHK17], where local importance criteria are mapped to an optimal mass  $m_i^{\text{opt}}$  each particle should take on. The ratio  $m_i^{\text{rel}} = \frac{m_i}{m_i^{\text{opt}}}$  is then used to classify particles into categories. Particles that are far too large can be split. Far too small particles can be merged with neighbors, and among other particles, mass can be smoothly redistributed to meet the optimal particle mass [WHK17]. A derived approach is used in Chapter 4 to continuously evaporate particles.

Two-scale simulation

Splitting and merging

Continuous particle mass

**Temporal Adaptivity:** As discussed in Sec. 2.3, time steps should be chosen according to CFL conditions that depend on the current particle configuration. CFL conditions can also be defined to vary in space according to the configuration of the local particle neighborhood. By locally adjusting time steps, computational resources can be saved. Locally adaptive time stepping can either freeze particles in quiet areas [GP11] or use integer multiples of a global time step [GB14] or freely adjustable time steps [RHEW17] that locally adapt to the fluid properties. Local time stepping methods, however, have not

Global vs. local time steps

been applied to incompressible fluids because they cause instabilities. In order to improve efficiency in incompressible fluids, time steps are often globally adapted [IAGT10].

The reader is referred to the survey of Manteaux et al. [MWN\*16] for further details on adaptive methods.

## 3.4 Efficient and Parallel Implementation

**Complexity of SPH** When using a Gaussian kernel as proposed by Monaghan [Mon05], SPH simulations can have quadratic cost in case every particle has to interact with every other particle like in astrophysical gravitational problems. In case of incompressible fluids and compact support radii, however, pressure forces keep particles apart so that the problem can be reduced to linear cost as only a maximum number of particles is in the neighborhood of every other particle. As interactions between neighboring particles can be formulated independently for each particle, SPH can be trivially parallelized and is well-suited to massively parallel architectures like GPUs [Gre09].

**Neighborhood search** The question of how to compute dynamically changing particle neighborhoods so that no unnecessary particle pairs have to be considered lies at the heart of every efficient simulation. In simulations with varying particle sizes, hierarchical data structures like kd-trees [KAG\*06, APKG07] or OpenVDB [ATO16] are often used to sort and efficiently access particles according to their spatial position. For uniform particle sizes, uniform grids are preferred because of their simplicity [Gre09, GSSP10, IABT11, WHK16]. For every uniform grid cell, memory needs to be reserved even if it is empty, thus, for large scale simulations, compact hashing can be employed in which memory is only occupied for non-empty cells [THM\*03, IABT11]. In order to efficiently simulate FLIP on GPUs, voxel data structures are applied to calculate pressures and access particles [WTYH18].

**Neighborhood lists** These data structures can either be accessed in every SPH interpolation to iterate over particle neighborhoods on-the-fly or neighborhood lists can be computed. Interpolations then only have to loop over each particle's list of neighbors and the access data structure can be discarded after the lists have been generated [DCGG13]. Neighborhood lists are able to increase performance considerably, especially if many interpolations are necessary like in iterative pressure solvers, however, they also drastically increase memory consumption [WHK16]. As the calculation of neighborhood lists itself imposes a computational overhead, the fact that particles usually don't move much between time steps can be exploited by recomputing neighborhood lists only

every  $n$ -th time step [IABT11, ATO16].

For GPU-based implementations there are two distinct options: particles can either be scattered onto evaluation points which incurs write collisions or they are gathered at sampling positions which requires access to particle neighborhoods. In early GPU-based approaches, scattering of particle data into textures was usually applied as it maps well to the programmable rasterization pipeline [HKK07c, HKK07a, ZSP08]. Scattering does not rely on acceleration data structures and is still attractive for rendering [vdLGS09, FAW10, FGE10]. Gathering approaches, in contrast, are better able to exploit massive parallelism but rely on generic APIs like OpenCL or CUDA in order to calculate data structures for neighborhood search [Gre09, GSSP10, MMCK14]. Neighborhood search is usually realized using uniform grids and spatial indexing [Gre09, WHK16] in combination with data-parallel sort [SHG09], and scan and compact primitives [SHZO07]. Neighborhood lists can be calculated in one pass using preallocated memory assuming a fixed maximum number of neighbors per particle [WHK16] or in two-passes by first counting the number of neighbors and in the second pass collecting the indices of neighboring particles to reduce memory consumption [OK12]. In constrained neighborhood lists [WHK16], particle support radii are iteratively reduced to achieve a user-defined number of particle neighbors. This both increases performance and reduces memory consumption as necessary on GPUs due to limited VRAM.

GPU-based  
implementations

Scattering

Gathering

Neighborhood lists



# 4

## Simulation of Evaporation and Condensation

*This chapter presents a method to simulate evaporation and condensation of liquids. Therefore, both the air and liquid phases have to be simulated. A coarse grid is employed for the air phase, as a carrier of vapor, and mass-preservingly coupled to an SPH-based liquid and rigid body simulation. Since condensation only takes place on rigid surfaces, it is captured using textures that carry water to achieve high surface detail. The textures can exchange water with the air phase and are used to generate new particles due to condensation effects yielding a full two-way coupling of air phase and liquid. In order to allow gradual evaporation and condensation processes, liquid particles can take on variable sizes.*

*In order to improve the rendering of liquids in contact with rigid surfaces, e.g., of condensed droplets, a modified implicit surface definition is proposed that is able to render dynamic contact angles for moving droplets and yields highly detailed fluid renderings.*

*The methods described in this Chapter have been published [HK17] and presented at the Symposium of Computer Animation (SCA) 2017 in Los Angeles, USA.*

---

**E**vaporation and condensation are ubiquitous phenomena encountered in everyday life, yet, they have not been modeled and simulated in a comprehensive way. While there are some works in which liquid and gaseous fluids are coupled, e.g., to simulate burning oil [LSSF06, MMCK14], no two-way coupling of water and its vapor in air has been proposed so far. In the context of SPH-based fluids there have been models of evaporation due to boiling [MSKG05, PCPW15, VTT\* 18] while Tillmann and Bohn [TB15] simulated condensation on rigid surfaces by generating explicitly meshed droplets from vapor transported in a grid.

In this chapter, a simulation of evaporation and condensation is proposed that uses a coarse Eulerian grid to simulate the air phase while the liquid phase is simulated using SPH particles. Condensation only takes place on rigid surfaces and is realized in sub-particle detail using textures to deposit mass into, and to generate new particles from. In order to allow for gradual evaporation, particles use variable sizes. In reality, fluids in close contact with

rigid surfaces take on distinct contact angles. In order to allow for a rendering of SPH fluids with dynamic contact angles a modified implicit surface description is proposed.

#### Contributions

In summary the contributions are

- a physically-based model of evaporation and condensation that allows for
- mass-preserving coupling of grid, and particle system to simulate evaporation and condensation,
- sub-particle detail of surface wetting using textures to contain mass and
- rendering of dynamic contact angles using an improved implicit surface representation for SPH fluids.

The remainder of this chapter is structured as follows. Sec. 4.1 discusses related work and introduces necessary foundations for the proposed algorithm that is outlined in Sec. 4.2. The heat coupling between different systems is discussed in Sec. 4.3, and Sec. 4.4 presents the method to simulate evaporation and condensation. In Sec. 4.5 the improved implicit surface definition is introduced. Results are presented in Sec. 4.6, before Sec. 4.7 draws final conclusions.

## 4.1 Foundations and Prior Work

#### SPH-based simulation

This section briefly describes closely related works to the contributions in this Chapter and gives necessary foundations of fluid flow and evaporation and condensation that have not been covered in Chapter 2. As discussed in Chapter 3, there is a wide array of phenomena that have successfully been described in SPH, like the thermodynamic processes of melting and solidification [SSP07]. Evaporation has been modeled by stochastically transforming liquid to air particles after reaching a critical temperature [MSKG05] or using a physically-based model of boiling [PCPW15]. Villa Salazar et al. [VTT\*18], published after the work presented here, simulate boiling and condensation by directly transforming liquid to gas particles and vice versa using a latent heat model. The model, however, does not take the surrounding air phase into account and is not able to address other evaporative phenomena than boiling. Ren et al. [RLY\*14] allow particles to carry volume fractions of different fluid phases that can mix, unmix and react. Although the model is able to simulate chemical reactions of two liquids to form a gaseous product, all particles have a constant mass so that the volume of gas particles drastically increases which impairs simulation stability. Moreover, the fluid mixture is not incompressible, hence, the pressure term has to rely on an equation of state. Yang et al. [YCL\*17]

extend the approach by including heat transport and model continuous phase changes using a volume fraction model. Although the model is supposed to capture phase changes between solid, liquid and gaseous states, only melting and solidification is considered.

In the context of mesh-based methods, versatile interactions of liquids, solids and gases have been realized by Losasso et al. [LSSF06] and very detailed interaction of water with surfaces has been simulated by Wang et al. [WMT05]. Tillmann and Bohn [TB15] first used a grid to simulate an air phase that transports vapor which can be condensed to static droplets on rigid surfaces using explicit meshes.

Grid-based simulation

High-quality rendering of SPH fluids is mainly achieved using implicit surface definitions [ZB05, SSP07, YT10, OCD11] as discussed in Sec. 2.4.1. Although these methods are able to yield smooth liquid surfaces, interactions with rigid objects can cause visually unplausible interpenetration. For mesh based surface representations, vertices can just be moved to closest point on the rigid surface to resolve intersections [HKK07b]. This however can cause self-intersections in the resulting mesh and is, like all mesh optimization strategies, not applicable to direct rendering approaches. Approaches that directly modify the implicit surface definition, in contrast, can also be applied in direct rendering. In that way, Huber et al. [HEW15] successfully prevent the liquid surface from penetrating cloth and Morgenroth et al. [MWE16] enforce a prescribed static contact angle between liquid and solid surfaces which greatly enhances optical realism. However, dynamic contact angles that are formed between moving liquids and rigid surfaces have not yet been addressed.

Surface rendering

Interaction with rigid objects

### 4.1.1 Equations of Fluid Flow

For grid simulations, usually an incompressible, inviscid fluid is assumed which can be described using the Navier-Stokes equations (see Eq. (2.4) in Sec. 2.1). Temperature  $T$  and density  $\rho$  are just advected with the flow according to

$$\frac{\partial T}{\partial \tau} = -(\vec{v} \cdot \nabla) T \quad (4.1)$$

$$\frac{\partial \rho}{\partial \tau} = -(\vec{v} \cdot \nabla) \rho. \quad (4.2)$$

In order to make warm air rise and dense air sink, an explicit buoyancy can be introduced into the external acceleration as an additional term

Buoyancy

$$\vec{a}_{\text{buoy}} = -\alpha \rho \hat{\mathbf{g}} + \beta (T - T_{\text{amb}}) \hat{\mathbf{g}}, \quad (4.3)$$

where  $\hat{\mathbf{g}}$  is the normalized direction of gravity,  $\rho$  the density and  $\alpha$  and  $\beta$  are user-defined parameters [FSJ01].

### 4.1.2 Evaporation and Condensation

Evaporation and condensation depend on the states of the liquid and air phase and follow the difference between the saturation vapor pressure at the surface temperature and the partial pressure of the vapor in the air phase, i.e., the vapor pressure [SLJ94, Sha14]. At saturation vapor pressure, the liquid and gaseous state of water are just at an equilibrium so that no net phase change occurs. If the actual vapor pressure is larger, condensation sets in, else evaporation takes place. The vapor pressure can easily be described using the ideal gas equation [RY96] as

Vapor pressure

$$p_c = \frac{m_c R_w (T_c - 273.15 \text{ }^\circ\text{C})}{V_c}, \quad (4.4)$$

where  $m_c$  is the water vapor mass in the air,  $V_c$  its volume,  $T_c$  the temperature in  $^\circ\text{C}$  and  $R_w$  the specific gas constant of water. The saturation vapor pressure at surface temperature  $T_s$  can be calculated from Magnus' formula [RY96] as

Saturation vapor pressure

$$p_s^{\text{sat}} = 611.2 \text{ Pa} \cdot \exp\left(\frac{17.62 \cdot T_s}{243.12 \text{ }^\circ\text{C} + T_s}\right). \quad (4.5)$$

To calculate the time rate of change of mass of a liquid surface due to evaporation, Smith et al. [SLJ94] proposed the following formula based on empirical data

Evaporation rate

$$\frac{\partial m_s}{\partial \tau} = A_s (a + b \cdot \|\vec{v}\|) (p_s^{\text{sat}} - p_c). \quad (4.6)$$

It depends on the surface of contact  $A_s$ , the saturation vapor pressure of the liquid surface  $p_s^{\text{sat}}$ , the vapor pressure of the air phase  $p_c$ , the air velocity at the interface  $\vec{v}$ , and two parameters  $a$  and  $b$ .

## 4.2 Algorithm Overview

The proposed algorithm can be subdivided into three phases.

**Phase 1** comprises a standard SPH-simulation including rigid particles and adaptive particle sizes.

**Phase 2** is a standard grid-based simulation of the air phase with vapor and temperature advection.

**Phase 3** realizes the couplings between the grid based air phase, the SPH-based liquid phase and rigid objects. The coupling comprises heat transfer (see Sec. 4.3) and mass transfer due to evaporation and condensation (see Sec. 4.4). In order to add sub-particle details of evaporation and



condensation, rigid objects use textures into which water can condense and evaporate from.

---

1:	<b>while</b> Simulating <b>do</b>
<hr/>	
Phase 1 - Simulate particles (liquid and rigid)	
<hr/>	
2:	Move particles according to forces of last iteration
3:	Neighborhood search
4:	Heat transport
5:	External forces
6:	Pressure solve
<hr/>	
Phase 2 - Simulate grid (air phase)	
<hr/>	
7:	Classify cells as air or liquid/rigid
8:	Add forces, enforce boundary conditions
9:	Advection of velocity, vapor and temperature
10:	Diffusion of heat, vapor
11:	Pressure projection
<hr/>	
Phase 3 - Coupling of grid, particles, texture	
<hr/>	
12:	Neighborhood search: <b>texture</b> ↔ <b>(rigid) particles</b> ↔ <b>grid</b> ↔ <b>texture</b>
13:	Heat transfer: <b>(rigid) particles</b> ↔ <b>grid</b>
14:	Temperature interpolation: <b>rigid particles</b> → <b>texture</b>
15:	Particle evaporation: <b>particles</b> → <b>grid</b>
16:	Sub-particle evaporation and condensation: <b>grid</b> ↔ <b>texture</b>
17:	Particle condensation: <b>texture</b> → <b>particles</b>
18:	Mass redistribution and merging of particles
19:	Particle path force
20:	<b>end while</b>

---

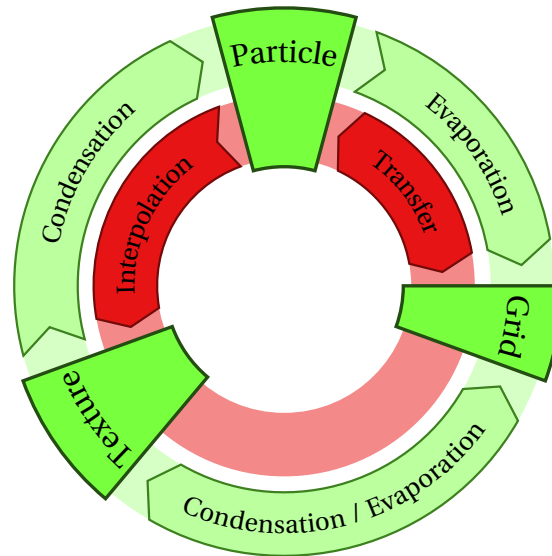
**Algorithm 4.1:** Overview over the proposed simulation of evaporation and condensation. Colored parts are added for the approach. Red color denotes heat transport and green color denotes mass transport. Coupling directions are highlighted in bold face.

Alg. 4.1 gives an overview over the proposed algorithm. Fig. 4.1 shows the paths of heat transport and mass transport in more detail. Except for the deposition of mass, no dynamic behavior of water inside textures is simulated. However, there are different approaches to simulate texture-based water flows on surfaces which could be included [WMT07, EJGP09].

### 4.2.1

## Particle Simulation

The SPH-simulation is based on standard components and comprises a PCISPH pressure solver [SP09b], rigid particles [AIA\*12] and surface tension and adhe-



**Figure 4.1:** Mass transfer (outer green circle) and heat transfer (inner red circle) between different systems. Arrows indicate in which direction transfer can take place.

sion [AAT13]. As evaporation and condensation takes place at surfaces, the approach depends on a stable measure of a particle's surface area  $A_i$  which is calculated as proposed by Orthmann et al. [OHB\*13] (see Eq. (3.18) in Sec. 3.2.1).

Adaptive particle  
mass

As the simulation of evaporation causes particles to lose mass, particles are allowed to have adaptive sizes. In literature, particle sizes are usually adjusted by modifying the support radius. If particles of different support radii interact, this can, however, lead to severe instabilities [OK12, WHK17]. Thus, the support radius is left unchanged and only particle masses are scaled using a weighting term  $w_i \in [0, 1]$ . Due to the fact that the weighted mass  $m_{w_i} = w_i m_i$  is used in the density calculation, the fluid volume is automatically scaled accordingly. In order to achieve an efficient simulation it is desirable to use larger particles of  $w = 1$ , therefore, smaller particles are allowed to merge.

## 4.2.2

### Grid Simulation

The grid is mainly used as a means to simulate the air phase and transport water vapor, thus, a coarse grid with a cell size larger than two times the particle support radius is employed. The grid simulation uses semi-Lagrangian advection [Sta99], a simple conjugate gradient solver to enforce a divergence-free velocity field and vorticity confinement [FSJ01] to retain fine details of the flow dynamics. The liquid and rigid particles are considered as boundary condition

for the grid simulation, i.e., cells are classified as air or solid/liquid.

### 4.2.3 Texture-based Rigid Surface Representation

To allow for a sub-particle detail evaporation and condensation at rigid surfaces, textures are used as a means to describe condensed water mass. For the approach to work, several static textures are precomputed, i.e., textures that store the world position  $\mathbf{x}_t$  of each texel  $t$  and surface area spanned by the texel  $A_t$ . In order to transfer mass from the texture to particles, a texture that stores particle seeding positions is used. These are created using Poisson disk sampling [Bri07] to prevent overlapping seeding positions which cause instabilities if particles are simultaneously emitted at very close positions. Additionally, a texture that stores the maximum amount of mass that is allowed to be deposited in each texel  $m_t^{\max}$  is used.

Particle seed texture

The only dynamic textures are the mass texture which stores the actual amount of water at each texel  $m_t$ , the temperature texture  $T_t$  and a wetting history [WMT05]. The latter allows particle paths along surfaces to be rendered and to prevent emitting particles in the neighborhood of texels that are already covered by particles.

### 4.2.4 Neighborhood Search

As depicted in Fig. 4.1, the simulation includes direct interactions between every pair of systems, thus, neighborhoods for grid-cells, texels and rigid and liquid particles have to be calculated. As rigids are static in the simulation and textures are fixed to their rigid bodies, neighborhoods between rigid particles, grid cells and texels have to be calculated only once. For liquid particles, however, neighborhoods have to be recomputed in every time step. In order to calculate neighborhoods between grid cells, particles and texels, the cell-based approach of Green [Gre09] is adopted.

## 4.3 Heat Transfer

Thermodynamic processes like evaporation and condensation depend on the temperatures of the different phases. Thus, all components of the simulation first have to be coupled in terms of heat transport to achieve realistic behavior. Heat transport between particles is realized according to Cleary and Monaghan [CM99] (see Eq. (3.20) in Sec. 3.2.3). Heat, however, has to be transferred

across phase transitions between grid cells and particles and to the surface textures.

### 4.3.1 Heat Transfer Between Grid and Particles

The transfer of heat between grid cells and particles follows a form of Fourier's law that resembles Newton's law of cooling and reads

$$\frac{dT}{d\tau} = \frac{\frac{dQ}{d\tau}}{\rho \cdot C} = \frac{UA\Delta T}{\rho \cdot C}, \quad (4.7)$$

where  $A$  is the interface area over which heat is transferred, and  $U$  is the overall heat transfer coefficient which is a constant for each pair of materials and can be calculated from the respective materials' heat conductivities  $\kappa$  and the distance the heat transfer has to bridge [LL16, BSL07].

Time rate of change  
of temperature

Particles  $i$  can interact with more than one neighboring cell  $c$ , and cells interact with more than one particle, thus, neighborhoods have to be iterated in order to calculate heat transfer. The time rate of change of temperature due to heat transfer across the air-liquid or air-rigid interface then can be expressed as

$$\frac{dT_i}{d\tau} = \frac{1}{\rho_i C_i} A_i \sum_c U_{ic} \alpha_{ic} (T_i - T_c) \quad (4.8)$$

for particles  $i$  and

$$\frac{dT_c}{d\tau} = \frac{1}{\rho_c C_c} \sum_j A_j U_{jc} \alpha_{jc} (T_c - T_j) \quad (4.9)$$

for cells  $c$ . The coefficients  $\alpha_{ic}$  denote trilinear interpolation weights for position  $\mathbf{x}_i$  that are used to weight cell-particle interactions. The heat transfer coefficient is calculated in accordance with the heat transfer between particles (see Eq. (3.20)) as  $U_{ic} = \frac{2\kappa_i\kappa_c}{h(\kappa_i+\kappa_c)}$  by assuming the transfer has to bridge the support radius  $h$  until the air phase is reached<sup>2</sup>. Note that the temperature flux is not antisymmetric as the actual physical quantity transferred is heat.

### 4.3.2 Heat Transfer to Texture

Textures are only used as a means to increase the surface detail of rigid bodies. Thus, they should not add too much computational overhead. To that end,

<sup>2</sup>In the original manuscript [HK17], the heat conductivity  $\kappa$  was directly used whereas the distance over which heat is transported erroneously was neglected. This is corrected by taking the overall heat transfer coefficient  $U = \frac{\kappa}{\Delta x}$  [LL16]. Note, the presented simulation results were not affected as  $\Delta x$  corresponds to the particle support which was set to  $h = 1$ .

instead of directly including textures in heat transport, rigid particle temperatures are only interpolated onto the texels in each time step using a corrected SPH interpolation, see Eq. (3.12).

Interpolation of  
texel temperatures

## 4.4 Evaporation and Condensation

As evaporation and condensation only take place at rigid and liquid surfaces, the simulation model is based on the general notion of surface elements or surfels. Surfels will be denoted with  $s$  which can mean texel or liquid particle and cell values are denoted with  $c$ . First, the general model of evaporation and condensation is described before the specific algorithmic solution is detailed that allows for a mass-preserving coupling between texels, grid cells and particles.

In order to allow evaporation and condensation to transfer variable amounts of mass between air and liquid, particles of variable size are used. However, in order to prevent a system of a lot of very tiny particles and to prevent too small particles to penetrate rigid surfaces [WHK17], particles are kept as close as possible to  $w_i = 1$ . This is done by allowing particles to exchange mass among each other and by merging neighboring particles.

### 4.4.1 Modeling Evaporation and Condensation

Although the time rate of change of mass in Eq. (4.6) originally only described evaporation, it will also be used to model condensation. Depending on the sign of  $\frac{\partial m_s}{\partial \tau}$ , either evaporation or condensation takes place which allows to separately describe both rates as

Initial evaporation  
and condensation  
rates

$$\text{evaRate}(c, s) = \max\left(\frac{\partial m_s}{\partial \tau}, 0\right), \quad (4.10)$$

$$\text{condRate}(c, s) = -\min\left(\frac{\partial m_s}{\partial \tau}, 0\right), \text{ with } b = 0, \quad (4.11)$$

for a cell  $c$  and a surfel  $s$ . To get spatially smooth values, trilinear interpolation of cell quantities at position  $\mathbf{x}_s$  is used. As the velocity term in Eq. (4.6) introduces energy, it can only cause an increase of evaporation but not of condensation, thus, its contribution is removed in case of a condensation process by setting  $b = 0$ . In order to guarantee mass preservation, the same amount of mass that is taken from the liquid has to be transferred to the air phase and vice versa, i.e.,  $m_{s \leftarrow c} = -m_{c \leftarrow s}$  has to be enforced.

### 4.4.2 Texel Evaporation and Condensation

As the model of evaporation and condensation is based mainly on temperatures, it can easily calculate evaporation or condensation rates that would cause either negative masses or make texels exceed their maximum allowed mass  $m_t^{\max}$ . Thus, a balancing step is necessary that guarantees that both the cell's vapor mass and the surfel's water mass stay non-negative. Additionally, the maximum amount of mass  $m_t^{\max}$  of texels must not be exceeded.

Correction of initial rates

The proposed approach works by taking the evaporation and condensation rates above only as initial estimates and correcting them using scaling factors. Alg. 4.2 gives a detailed description of evaporation and condensation using textures. First the texel mass is bounded. Evaporation is scaled by factor  $w_t^{\text{eva}} \in [0, 1]$  which enforces non-negativity of texel masses while the factor  $w_t^{\text{cond}} \in [0, 1]$  is applied to condensation and enforces a maximum texel mass of  $m_t^{\max}$ . After properly scaling the mass transfer to respect the texel bounds, a third scaling factor  $w_c^{\text{cond}}$  is applied to condensation that enforces non-negativity of vapor mass in cells. After scaling, the masses of cells and texels are updated. Alg. 4.2 ensures that the total mass in the simulation is preserved.

### 4.4.3 Evaporation and Condensation of Particles

While particles can directly evaporate by transferring mass to the grid, condensation is only realized through the texture. In order to generate particles from texels due to condensation, the precomputed texture that stores particle seeding positions is used. Particle condensation directly takes mass from texels and only takes place if sufficient mass is present, thus, no special balance has to be calculated.

The same is not true for particle evaporation for which a similar correction as outlined in Alg. 4.2 is applied. However, adaptive particle masses are determined using weighting terms  $w_i$ , which thus have to be adjusted instead of changing the particle mass, directly.

Wetting history

When a texel has just been in contact with a particle, the texel is excluded from condensation for a small amount of time. This achieves two goals: Firstly, it prevents fluctuating evaporation and condensation of particles back and forth into the texture and, secondly, it allows for a realistic rendering of particle paths down surfaces. To keep track of excluded texels, the time of the last contact between texels and particles is stored in a wetting history texture [WMT05].

---



---

Phase 1 - Initial estimate

---

```

1: for all Texel t do
2:   for all Cell c do
3:      $m_{c \leftarrow t}^{\text{eva}} \leftarrow \text{evaRate}(c, t) \cdot d\tau$  ▷ see Eq. (4.10)
4:      $m_{t \leftarrow c}^{\text{cond}} \leftarrow \text{condRate}(c, t) \cdot d\tau$  ▷ see Eq. (4.11)
5:   end for
6: end for

```

---

Phase 2 - Balance

---

```

7: for all Texel t do
8:    $w_t^{\text{eva}} \leftarrow \min\left(\frac{m_t}{\sum_c m_{c \leftarrow t}^{\text{eva}}}, 1\right)$  ▷ Keep texel non-negative
9:    $w_t^{\text{cond}} \leftarrow \min\left(\frac{m_t^{\text{max}} - m_t}{\sum_c m_{t \leftarrow c}^{\text{cond}}}, 1\right)$  ▷ Keep texel mass below  $m_t^{\text{max}}$ 
10:  for all Cell c do
11:     $m_{c \leftarrow t}^{\text{eva}} \leftarrow m_{c \leftarrow t}^{\text{eva}} \cdot w_t^{\text{eva}}$  ▷ Scale mass transport
12:     $m_{t \leftarrow c}^{\text{cond}} \leftarrow m_{t \leftarrow c}^{\text{cond}} \cdot w_t^{\text{cond}}$ 
13:  end for
14: end for
15: for all Cell c do
16:    $w_c^{\text{cond}} \leftarrow \min\left(\frac{m_c - \sum_t m_{c \leftarrow t}^{\text{eva}}}{\sum_t m_{t \leftarrow c}^{\text{cond}}}, 1\right)$  ▷ Keep cell non-negative
17: end for

```

---

Phase 3 - Update mass

---

```

18: for all Texel t do
19:    $m_t \leftarrow m_t + \sum_c (m_{t \leftarrow c}^{\text{cond}} \cdot w_c^{\text{cond}} - m_{c \leftarrow t}^{\text{eva}})$ 
20: end for
21: for all Cell c do
22:    $m_c \leftarrow m_c - \sum_t (m_{t \leftarrow c}^{\text{cond}} \cdot w_c^{\text{cond}} - m_{c \leftarrow t}^{\text{eva}})$ 
23: end for

```

---

**Algorithm 4.2:** Evaporation from and condensation into textures. Because the initial evaporation and condensation rates in Phase 1 may cause negative mass or may exceed mass bounds, they have to be corrected in Phase 2 before updating the masses Phase 3.

#### 4.4.4

#### Dynamic Particle Adjustment

As particles can shrink due to evaporation, particles of different sizes may interact. Recently, an adaptive SPH simulation with continuously adjustable particle sizes has been presented [WHK17] that builds on particle merging and redistribution of mass among neighboring particles in order to arrive at prescribed sizes. In order to prevent particles from interacting that have strongly different sizes, a similar idea is adopted.

In this approach, however, particles of uniform size  $w_i = 1$  should be achieved, thus, particles are allowed to merge if the sum of their weights does

not exceed 1, else mass between particles of different size can be redistributed in order to make particle sizes more uniform. Uniform particles achieve better stability and are able to prevent liquid particles from penetrating rigid objects that are also uniformly sampled.

**Particle merging** In order to merge neighboring particles  $i$  and  $j$  to form a new particle  $n$ , their weights are used. The new weight is the sum of the old weights  $w_n = w_i + w_j$ . All other quantities, except for the mass which remains constant, are determined using a weighted average as

$$\mathbf{x}_n = \frac{\mathbf{x}_i w_i + \mathbf{x}_j w_j}{w_n} \quad (4.12)$$

to guarantee conservation of momentum [WHK17]. After merging only particle  $n$  remains and  $i$  and  $j$  are removed. In case of mass redistribution, only the fraction  $w_i - \frac{w_i + w_j}{2}$  of the larger particle  $i$  is redistributed to the smaller particle  $j$ . And only the remaining quantities of particle  $j$  are updated like above.

## 4.5 Surface Rendering

In order to achieve a realistic fluid appearance, it is crucial to have a proper interaction of fluids with rigid surfaces. This can either be achieved by simulating the interaction of liquid particles with the rigid at costly high particle resolution [BPHK13] or by approximating the appearance only in the rendering step. Although fluid rendering that uses implicit surface definitions is not new, only recently Morgenroth et al. [MWE16] proposed a method to achieve prescribed static contact angles at rigid surfaces. Starting from the original zero level-set surface definition described in Sec. 2.4.1 in the following two subsequent correction steps to  $\phi(\mathbf{x})$  are proposed in order to achieve dynamic contact angles depending on the velocity of the fluid.

**Parametric fluid meniscus** Deserno [Des04] derived a parametric description for the fluid meniscus at a vertical solid wall for arbitrary angles between the horizontal plane and the fluid surface  $\psi_0$  as

$$\begin{aligned} x(s, \psi_0) &= l \frac{\frac{s}{l} \cosh \frac{s}{l} + (\frac{s}{l} \cos \frac{\psi_0}{2} - (1 - \cos \psi_0)) \sinh \frac{s}{l}}{\cosh \frac{s}{l} + \cos \frac{\psi_0}{2} \sinh \frac{s}{l}} \\ y(s, \psi_0) &= l \frac{2 \sin \frac{\psi_0}{2}}{\cosh \frac{s}{l} + \cos \frac{\psi_0}{2} \sinh \frac{s}{l}}, \end{aligned} \quad (4.13)$$

where  $x(s, \psi_0)$  describes the distance to the wall in normal direction,  $y(s, \psi_0)$  the vertical offset from the horizontal plane and  $s$  is the arc length of the

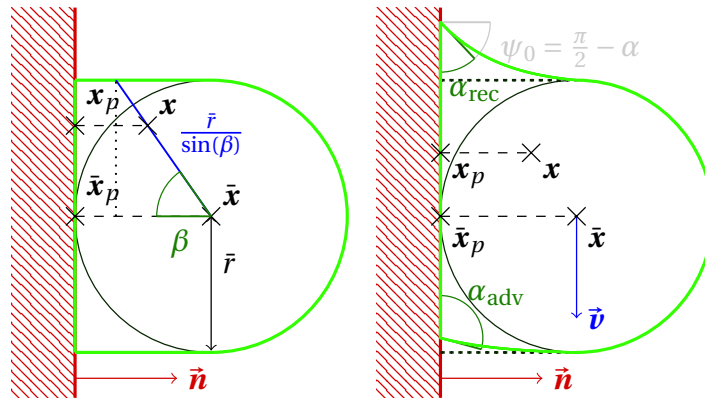


meniscus starting at the wall with  $s = 0$ .  $l$  is the capillary length which in this work is set to the particle radius.  $x(s, \psi_0)$  is close to identity, i.e.,  $s \approx x(s, \psi_0)$  for moderate angles  $\psi_0$  so that  $y(s, \psi_0)$  can be used to correct the surface distance function as proposed by Morgenroth et al. [MWE16].

As the derivation in Eq. (4.13) only describes an offset from the horizontal plane, in practice, the correct initial contact angle between fluid and rigid has to be enforced. Therefore, Morgenroth et al. [MWE16] mirror ghost particles across solid walls which, however, can cause problems if fluid particles are on both sides of thin rigid walls and is costly due to the sampling of additional particles. Instead, in this work an explicit correction is presented which in turn extrudes a projected footprint of the fluid onto the wall. Fig. 4.2 (left) shows a schematic of the proposed approach. Assume a fluid is in close vicinity to a wall with unit normal  $\vec{n}$ , the weighted average position is  $\bar{\mathbf{x}}$  and the grid point to evaluate the signed distance for is  $\mathbf{x}$ . Then the distance function is adjusted according to

$$\begin{aligned}\phi(\mathbf{x}) &= \|\mathbf{x} - \bar{\mathbf{x}}(\mathbf{x})\| - \left( \frac{\bar{r}(\mathbf{x})}{\sin(\beta)} + d(\mathbf{x}) \right), \text{ with} \\ \beta &= \arccos\left(\max\left(\frac{\bar{\mathbf{x}} - \mathbf{x}}{\|\bar{\mathbf{x}} - \mathbf{x}\|} \cdot \vec{n}, 0\right)\right).\end{aligned}\quad (4.14)$$

As particle sizes vary, the CSPH interpolated radius  $\bar{r}(\mathbf{x})$  is used (see Eq. (3.12)). The max in  $\beta$  restricts the correction to work in the direction of the solid wall. In a next step the term  $d(\mathbf{x})$  is calculated which allows to enforce dynamic



**Figure 4.2:** Adjustment of distance function according to advancing  $\alpha_{\text{adv}}$  and receding  $\alpha_{\text{rec}}$  angles yields nicely rendered details at the surface (red) and dynamic appearance even for single particles (green outline).

wetting angles as depicted in Fig. 4.2 (right). In general, there is a dependency between the capillary number, which relates viscosity and velocity to surface tension, and the third power of the dynamic contact angle [Kis93]. As surface

Static contact angle

Projected footprint

Dynamic contact angle

tension and viscosity are constant, the receding and advancing contact angles can be described as functions of the velocity as

$$\begin{aligned}\alpha_{\text{rec}}(\vec{v}) &= \frac{\pi}{2} - A_{\text{rec}} \|\vec{v}\|^{\frac{1}{3}} \\ \alpha_{\text{adv}}(\vec{v}) &= \frac{\pi}{2} + A_{\text{adv}} \|\vec{v}\|^{\frac{1}{3}},\end{aligned}\quad (4.15)$$

where  $A_{\text{rec}}$  and  $A_{\text{adv}}$  control the influence of the velocity. Using two different control parameters allows to capture hysteresis effects that can cause advancing and receding angles to differ.

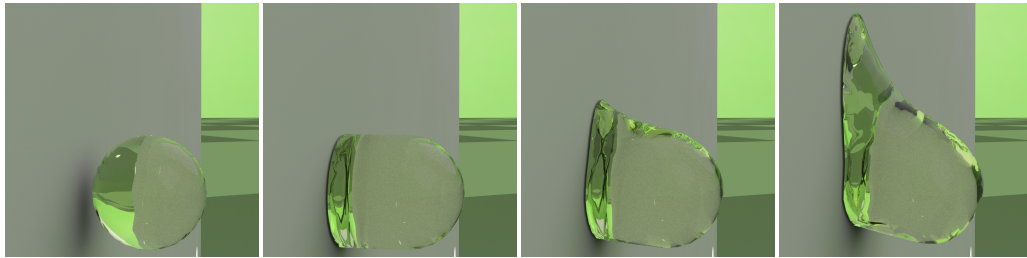
The actual dynamic contact angle to achieve is determined by the relative projected positions of the weighted average particle position  $\bar{\mathbf{x}}$  and the grid position of evaluation  $\mathbf{x}$ . These are orthogonally projected onto the surface to yield  $\bar{\mathbf{x}}_p$  and  $\mathbf{x}_p$ . The distance between these points is projected onto the velocity vector as  $p = \left( \frac{(\mathbf{x}_p - \bar{\mathbf{x}}_p) \cdot \vec{v}}{\|(\mathbf{x}_p - \bar{\mathbf{x}}_p)\| \cdot \|\vec{v}\|} \right)$ . The sign of  $p$  determines if  $\mathbf{x}$  is located at the receding or advancing front of the droplet. The contact angle  $\alpha$  is then interpolated between the static contact angle  $\alpha_{\text{base}}$  and the receding and advancing angles  $\alpha_{\text{rec}}$  and  $\alpha_{\text{adv}}$  as

$$\alpha(p) = \begin{cases} p^4 \cdot \alpha_{\text{rec}}(\vec{v}) + (1 - p^4) \cdot \alpha_{\text{base}} & \text{if } -1 \leq p < 0 \\ p^{\frac{1}{4}} \cdot \alpha_{\text{adv}}(\vec{v}) + (1 - p^{\frac{1}{4}}) \cdot \alpha_{\text{base}} & \text{if } 0 \leq p \leq 1 \end{cases}. \quad (4.16)$$

The power terms force the receding angle to form a narrow tail and the advancing front to form a more drop-like shape. Finally the corrections

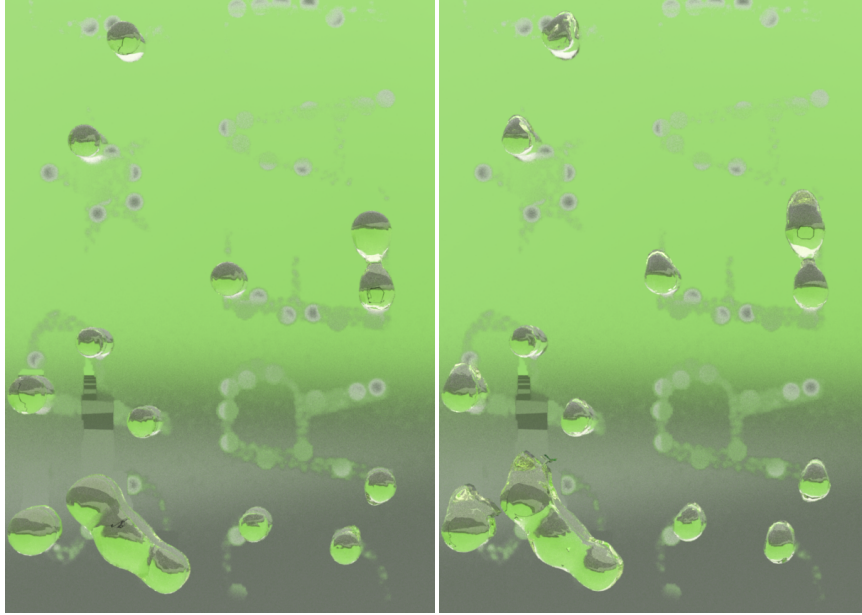
$$d(\mathbf{x}) = y \left( \|\mathbf{x}_p - \mathbf{x}\|, \frac{\pi}{2} - \alpha(p) \right) \quad (4.17)$$

are plugged into Eq. (4.14). Fig. 4.3 shows how a droplet consisting of a single particle is modified using the proposed method. In order to not make the



**Figure 4.3:** A drop of one particle is modified using the proposed approach. Left: Unmodified Drop, middle left: drop with projected footprint, middle right: drop at moderate downward velocity, right: drop at fast downward velocity.

correction terms are restricted to distances  $\|\bar{\mathbf{x}} - \bar{\mathbf{x}}_p\| < \bar{r}$  which has, however, been omitted in Fig. 4.3 to make the corrections more articulate. Fig. 4.4 shows a comparison of an uncorrected surface extraction and the proposed method in one of the demo scenes.



**Figure 4.4:** Uncorrected surface rendering (left) and the proposed modification to render dynamic contact angles (right).

## 4.6

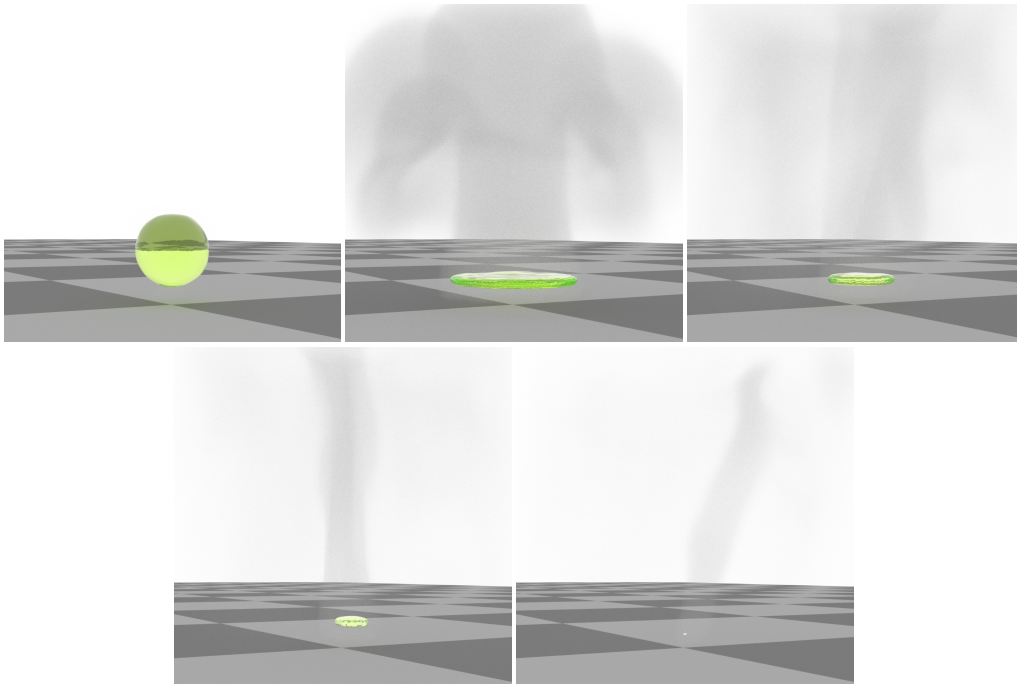
## Results

The proposed simulation was tested on four different scenes. In order to evaluate evaporation, a liquid drop was put on a hot surface (see Fig. 4.5). In the SCA Logo Tex scene, humid air is blown onto an impregnated mirror. It uses purely texture-based evaporation and condensation (see Fig. 4.6) while in the SCA Logo scene, particles are emitted from the texture seeding points (see Fig. 4.7). In the Glass scene, the outside of a glass filled with a cold liquid is steamed with a stream of warm vapor from the left hand side while the liquid inside remains at rest (see Fig. 4.8). All simulations have been carried out using an NVIDIA GeForce GTX Titan with 6 GiB VRAM. The simulation framework has been implemented using C++, CUDA 8.0. Tab. 4.1 shows a summary of the resolutions and timings of the demo scenes. All scenes have been rendered using Mitsuba [Jak10] and the proposed modified surface extraction. Surface steaming effects have been rendered using a rough material where the rough-

**Table 4.1:** Scenes with particle '#Ptcl' and rigid particle '#RigPtcl' counts, grid '#Cell' and texture '#Texel' resolution and run times. 'Ptcl' gives the time spent for the SPH simulation, 'Grid' the run time of the grid simulation. The remaining timings describe the coupling: 'Neigh' denotes the time for neighborhood searches, 'Heat' for heat transfer and interpolation, and 'Eva' for evaporation and condensation,  $d\tau$  denotes the corresponding simulation time step.

Scene	Resolution				Run time per step (ms)					
	#Ptcl	#RigPtcl	#Cell	#Texel	Ptcl	Grid	Neigh	Heat	Eva	$d\tau$
Drop (no tex)	1207	5108	$64^3$	–	18	65	2	6	1	4
Drop	1207	5108	$64^3$	$512^2$	18	65	11	6	26	4
SCA Logo Tex	–	28 K	$64^3$	$1024^2$	–	60	–	16	15	5
SCA Logo	135	28 K	$64^3$	$1024^2$	13	60	8	17	31	5
Glass	174 K	89 K	$64^3$	$1024^2$	210	31	32	23	40	2

ness was controlled by the water mass textures. The impregnation of the SCA logo has been realized by adjusting the maximum mass texture.



**Figure 4.5:** A spherical drop of water is dripped onto a hot surface, evaporates and transfers its mass into the grid simulation. The sequence progresses from left to right and from top to bottom.

Evaporation of a drop

In the drop scene (see Fig. 4.5), the initial liquid and air temperatures were

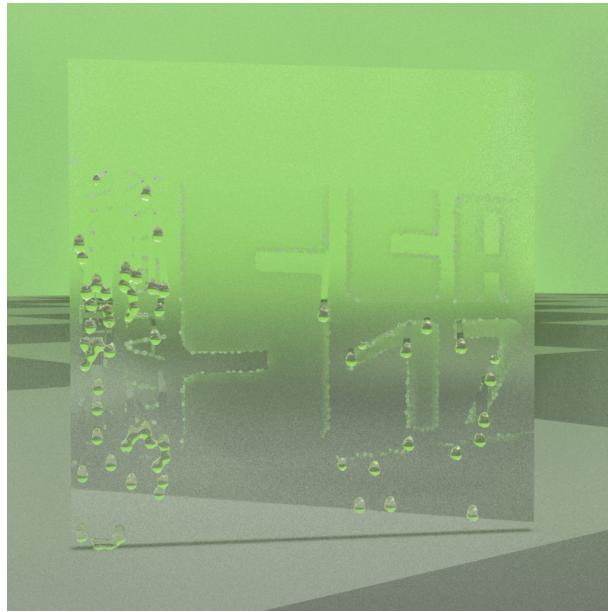
set to 20°C while the ground plane was set to a temperature of 150°C causing a fast evaporation of the liquid. In order to assess the overhead of using textures, two versions of the drop scene were simulated, while the general simulation outcome was not changed. Due to the comparably low number of particles, the run time was mainly determined by the grid solver which took 65 *ms* per time step while the particle simulation only took 18 *ms* and the coupling between air and liquid phase only 9 *ms* (see line 'Drop (no tex)' in Tab. 4.1). When adding textures to the simulation (see line 'Drop' in Tab. 4.1), the timings for the coupling increased to 53 *ms*. However, due to the fact that heat is only interpolated from rigid particles to texels, the run time for heat transfer does not measurably increase when using textures.



**Figure 4.6:** A mirror is steamed by humid air revealing the impregnated SCA logo.

Also for the SCA Logo scene two versions were simulated, one that uses only textures and no liquid particles (SCA Logo Tex, see Fig. 4.6) and one that uses the full simulation including liquid particles (see Fig. 4.7). In the

Condensation on a mirror



**Figure 4.7:** Blowing moist air onto the impregnated SCA logo causes particles to condense.

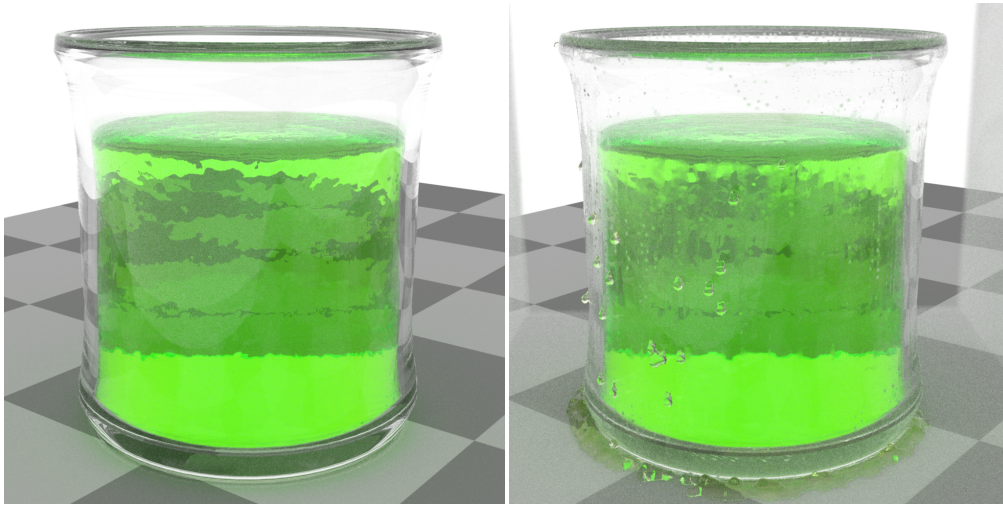
logo scene, moist air at  $20^{\circ}\text{C}$  is blown onto a cold mirror at  $4^{\circ}\text{C}$  which causes condensation to set in. The SCA Logo Tex scene demonstrates that very fine surface detail can be captured using the proposed coupling. The run time was again dominated by the grid solver with  $60\text{ ms}$  per time step while the coupling only took  $31\text{ ms}$  in total. As the neighbor search for rigid-texel coupling is done in a preprocessing step it did not introduce any additional cost per time step. When additionally introducing liquid particles into the scene, neighbor search has to be performed in each step increasing the run time for the coupling to  $56\text{ ms}$ .

Balanced  
evaporation and  
condensation

The glass scene (see Fig. 4.8) demonstrates that the proposed method is able to properly work in a well balanced way, i.e., allowing to simulate a cold fluid at rest that does not evaporate while particles in the vicinity condense at the outside of the glass. The liquid simulation for the glass scene took  $210\text{ ms}$  per time step and dominated the overall run time. The grid solver only took  $31\text{ ms}$  and the coupling took  $95\text{ ms}$  in total.

Surface rendering

The marching cubes [LC87] based renderer runs solely on the CPU. Rendering times for the Glass scene are  $31.7\text{ s}$  using the original surface definition and  $53.8\text{ s}$  using the proposed approach, i.e., the proposed modification lead to an overhead of 70%. For the SCA Logo scene rendering times were  $9.3\text{ s}$  per frame without and  $14.6\text{ s}$  with modification, i.e., an overhead of 60%.



**Figure 4.8:** A glass filled with cold liquid surrounded by moist air (left) causes water to condense at the glass surface (right).

**Limitations** As evaporating particles can become very small, they sometimes are able to penetrate rigid bodies. This is a known limitation of current adaptive approaches [WHK17]. Since explicit meshes of large triangle counts are used to represent rigid surfaces, the overhead for the proposed implicit surface definition was quite big. It could be reduced by using more efficient representations for rigid surfaces.

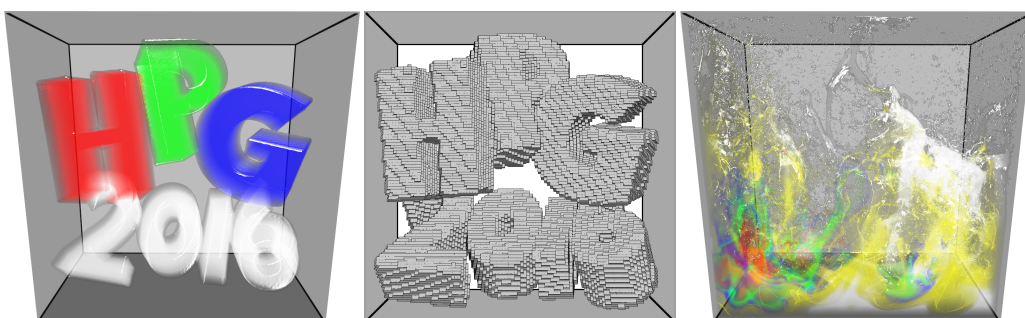
## 4.7 Conclusions

This chapter described the modeling and simulating of the evaporation and condensation of SPH-based fluids. The air phase is modeled using a coarse Eulerian grid solver while the liquid phase is based on an adaptive SPH solver. In order to achieve very fine surface detail, textures are used on rigid objects into which mass can be transferred. The texels exchange mass with the grid solver and if sufficient mass has been gathered, are used to generate particles. Particles can evaporate by transferring mass into grid cells which in turn yields a mass preserving cycle of evaporation and condensation. In order to achieve realistic high-quality surface renderings of fluids, a modified implicit surface definition has been presented that is able to enforce static and dynamic contact angles of fluids at rigid surfaces. The use of textures in combination with the novel implicit surface model allows for a rendering of rigid-fluid interactions at very high spatial resolution even at moderate particle sizes.





## Adaptive Volume Ray Casting



**Figure 5.1:** The fluid letters HPG 2016 (left) are dropped into a basin and cause a splashing effect (right). The sparse, perspective particle access grid adapts to the scene (middle).

*This chapter presents a fast and accurate ray casting technique for unstructured and dynamic particle sets. The technique focuses on efficient, high quality volume rendering of fluids for computer animation and scientific applications. The novel adaptive sampling scheme allows to locally adjust sampling rates both along rays and in lateral direction and is driven by a user-controlled screen space error tolerance. In order to determine appropriate local sampling rates, a sampling error analysis framework based on hierarchical interval arithmetic is proposed. The approach leads to significant rendering speed-ups with controllable screen space errors. Efficient particle access is achieved using a sparse view-aligned grid which is constructed on-the-fly without any pre-processing. The basic principle of the adaptive ray casting approach has first been described by Orthmann [Ort14] and is introduced in Sec. 5.2 and 5.3. In order to achieve competitive rendering performance, the approach was significantly improved as presented in Sec. 5.4. The resulting method has been published [HOK16] and presented at High Performance Graphics (HPG) 2016 in Dublin, Ireland.*

**P**article-based methods like smoothed particle hydrodynamics (SPH) offer several advantageous properties. Due to their high spatial flexibility, convection-driven free surface flows, and interactions with dynamic objects

can naturally be described [IOS\*14]. Moreover, the flow of concentrations and heat, i.e., advective-diffusive transport can easily be incorporated [CM99, Mon05, OHB\*13]. However, ad hoc rendering of large unstructured and dynamic particle sets is still a challenging task.

As discussed in Sec. 2.4, there are several methods to reconstruct smooth surfaces from particle-based data [SSP07, vdLGS09, OCD11, AIAT12, YT13, RCSW14, ZD15, ZD17]. While surface rendering is able to convey the fluid's geometric shape, it does, however, not provide any information about internal fluid structures, e.g., in advection-diffusion scenarios. In order to provide insight into fluid transport phenomena, volume rendering [EHK\*06, HLSR08] has to be applied. Volume rendering of particle data has been realized using hybrid splatting-slicing approaches [FGE10, FAW10] that scatter particle contributions [Wes90] onto axis-aligned [SP09a] or view-aligned [FGE10, FAW10, NMM\*06] texture slices. Compositing these slices in front-to-back order yields the final image. Because texture-slicing approaches are tailored for the rasterization pipeline of the GPU, incorporating adaptive sampling techniques is very difficult. Ray casting, in contrast, constitutes a more generic volume rendering approach for which, however, efficient means to directly access particle data are necessary. Moreover, in case of dynamic particle sets, these data structures must not rely on any kind of costly preprocessing. In the following an on-the-fly volume ray casting for unstructured particle data sets is proposed. The ray casting makes use of an adaptive sampling scheme to allow for an efficient rendering of large dynamic particle sets as shown in Fig. 5.1. In detail, the proposed rendering approach incorporates the following contributions:

#### Contributions

- An on-the-fly sampling error analysis framework based on hierarchical interval arithmetic for ray bundles which is able to derive strict bounds to the screen space error resulting from locally adapting sampling rates in lateral and viewing directions.
- A greedy algorithm that optimizes the degree of adaptivity both in viewing and lateral directions to yield significant speed-ups for a user-controlled screen space error.
- The perspective, view-aligned grid known from texture-slicing [FAW10] is enhanced to an efficient sparse access structure for particle data that is constructed on-the-fly.

The remainder of this chapter is structured as follows: Sec. 5.1 discusses foundations and prior work in adaptive volume ray casting and introduces interval arithmetic. Sec. 5.2 gives an overview of the ray casting pipeline. Sec. 5.3 and 5.4 describe the proposed sampling error analysis framework and how it is able to determine local sampling rates. Implementation details are given in Sec. 5.5. Sec. 5.6 discusses the results before conclusions are drawn in Sec. 5.7.

## 5.1 Foundations and Prior Work

### 5.1.1 Adaptive Volume Rendering

As introduced in Sec. 2.4.2, volume ray casting evaluates a physically-based model of light transport by treating quantity fields as a participating medium. Each viewing ray  $\mathbf{x}(s_k)$  is sampled at integer coordinates  $k = 0, \dots, N - 1$  which are distributed at distances  $s_k \in [s_{\text{near}}, s_{\text{far}}]$  from the camera, where  $s_{\text{near}}$  and  $s_{\text{far}}$  are the distances to the near and far clipping planes of the view frustum. The SPH-based quantity field is evaluated using corrected interpolation [BK02] (see Eq. (3.12)). Interpolated samples  $Q_k = Q(\mathbf{x}(s_k))$  along rays are mapped to radiance and transparency values as  $i_k = \text{tf}_I(Q_k)$  and  $t_k = \text{tf}_T(Q_k)$ , respectively, where  $\text{tf}_I, \text{tf}_T : [0, 1] \rightarrow [0, 1]$  are material dependent transfer functions. The volume rendering equation composites radiance and transparency samples to yield the ray's radiance and transparency as [EHK\*06]

$$I = \sum_{k=0}^{N-1} i_k \prod_{j=0}^{k-1} t_j^{\Delta s_j}, \text{ and } T = \prod_{k=0}^{N-1} t_k^{\Delta s_k}. \quad (5.1)$$

In contrast to the previously presented volume rendering equation (2.19) which assumed uniform step sizes, the transparency values given for a unit reference length are corrected in Eq. (5.1) to match the sampling step size  $\Delta s_k$ . In the following, opacity correction terms will be omitted to improve readability but in practice they have to be applied appropriately.

**Adaptive Rendering of Grid Structures** Adaptive sampling is mainly applied for data on regular grids [BHMF08, GS04, KHW\*09]. Danskin and Hanrahan use importance sampling in order to locally adapt sampling rates [DH92]. Although it is possible to substantially speed up rendering using importance sampling, it is a stochastic approach and hence it is difficult to calculate explicit bounds for the screen space error. Ledergerber et al. [LGM\*08] introduce a Moving-Least-Squares (MLS) approach to reconstruct higher order continuous field functions, which can also be applied to irregular grids. Similar to the proposed approach, Guthe and Strasser [GS04] estimate screen space errors due to adaptive sampling when uncompressing wavelet representations. However, their approach requires a costly wavelet transform as precomputation, which is unfeasible for on-the-fly visualization of dynamic particle data sets.

**Volume Rendering for Particle Sets** In contrast to sampling in regular grids, efficiently accessing unstructured particle data that contribute to a sampling

Object-aligned data structures

position poses quite a challenge. This is commonly addressed using spacial subdivision structures such as object-aligned octrees [OKK10]. As all object-aligned access structures require cell finding logic and may introduce thread divergence, parallelism can be drastically reduced [PGSS07].

#### Particle upsampling

In order to increase rendering performance, the particle count can be reduced by upsampling operators [APKG07, ZSP08, HHK08] that approximate particle subsets by fewer larger particles [HE03, FSW09, FAW10]. As upsampling can introduce visual artifacts [BOT01, KAG\*06], it should only be used to prevent under-sampling in case the particle size falls below the pixel size.

#### View-aligned grids

Perspective, view-aligned grids can be employed to achieve memory coherence and to remove traversal efforts [HM08]. Starting from a precomputed multiresolution particle representation Fraedrich et al. [FAW10] resample particle sets to a perspective grid for further texture-based ray casting. Their approach adjusts the sampling step size in viewing direction to be consistent with the perspective increasing lateral resolution (see also Sec. 5.5.1). In a follow up work, Reichl et al. [RTW13] preprocess the SPH particle set by resampling it onto an object-aligned octree hierarchy before ray casting. Orthmann [Ort14] used a perspective grid to directly access particle data and applied an adaptive ray casting. As the error analysis was very pessimistic, speed ups could only be achieved for very large error tolerances. By deriving screen space error bounds from interval arithmetic and separating lateral from longitudinal contributions, the sampling error analysis proposed in the following very tightly follows the error tolerance and yields large speed-ups.

### 5.1.2 Interval Arithmetic

Interval arithmetic is a common tool that helps to put bounds on, e.g., numerical approximations of mathematical calculations [MKC09]. Instead of working on real valued variables, a whole interval is used in calculations in order to bound all possible results of computations for a range of input values. In the following, interval variables will be marked with  $\mathbb{I}$  and the corresponding upper and lower bounds with superscripts  $\top$  and  $\perp$ , respectively. The width of an interval  $x^{\mathbb{I}} = [x^{\perp}, x^{\top}]$  will be denoted as

$$w(x^{\mathbb{I}}) := x^{\top} - x^{\perp}. \quad (5.2)$$

#### Arithmetic operations

In order to perform calculations using intervals, arithmetic operations have to be extended, accordingly. For any operation  $\bullet \in \{+, -, \cdot, /\}$ , the lower bound of  $x_1^{\mathbb{I}} \bullet x_2^{\mathbb{I}}$  can be determined as  $\min \{x_1^{\perp} \bullet x_2^{\perp}, x_1^{\perp} \bullet x_2^{\top}, x_1^{\top} \bullet x_2^{\perp}, x_1^{\top} \bullet x_2^{\top}\}$ . The upper bound is determined accordingly as the maximum. Some operations can, however, be expressed more efficiently. The addition operation can be defined

by adding the lower and upper bounds separately

$$x_1^\perp + x_2^\perp = [x_1^\perp + x_2^\perp, x_1^\top + x_2^\top].$$

As, in general, negative bounds have to be considered, the multiplication operation can not be simplified. In the case of only non-negative values, which is true for radiance and transparency values, it can however be simplified to

$$x_1^\perp x_2^\perp = [x_1^\perp x_2^\perp, x_1^\top x_2^\top].$$

As neither subtractions nor divisions are used in the following, they are omitted.

Calculating output intervals of general functions can be very costly as results to all potential inputs have to be considered as  $f(x^\perp) = \{f(x) | x \in x^\perp\}$ . In contrast, monotonically increasing functions are efficiently evaluated as [MKC09]

Monotonically increasing functions

$$f(x^\perp) = [f(x^\perp), f(x^\top)]. \quad (5.3)$$

As the volume rendering equation (5.1) takes only non-negative radiance and transparency values as input and is composed only of addition and multiplication operations, it is monotonically increasing in each argument. For a set of input radiance and transparency intervals it, thus, has to be evaluated only twice in order to calculate interval bounds. This makes interval arithmetic a perfect tool for the proposed screen space error analysis.

Application to volume rendering

## 5.2 Proposed Adaptive Ray Casting Pipeline

Even though the proposed on-the-fly ray casting is used for dynamic SPH data sets in this work, the scheme can be applied to any kind of unstructured particle sets based on local operators for recovering continuous quantity fields. However, particle sizes are assumed to not fall below pixel size in screen space, thus, particle upsampling cannot be applied.

The proposed ad hoc on-the-fly ray casting of dynamic SPH particle sets uses the original particle set “as is” to prevent any additional error due to resampling or interpolating particle quantities onto intermediate data structures such as grids or coarser particles. Also, any other kind of prohibitive and costly precomputation is omitted.

The proposed ray casting pipeline comprises five components:

**Sparse View-Aligned Grid Structure:** An enhanced perspective grid that subdivides the view frustum into cells that are aligned with view rays. In contrast to Fraedrich et al. [FAW10], who resample particle quantities

into a dense grid, a sparse data structure to access the original particle data is used. Figure 5.2 shows the resulting sparse grid structure. The grid is built from scratch in every frame using only raw particle data as input. The particles are assigned to all cells that intersect their volume and only cells that contain particles are present in the final grid. During ray casting each cell is traversed by a *ray bundle* covering  $D_{xy} \times D_{xy}$  pixels in screen space. As the cells are aligned with the view rays, all rays of a bundle traverse the same set of cells in viewing direction. Using an inverse perspective mapping, the perspective distorted sampling positions in view space are described in uniform sample space (see Sec. 5.5.1).

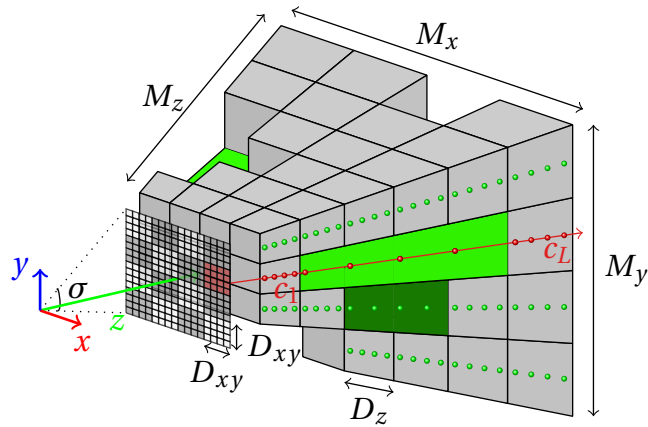
Initially each cell contains  $D_{xy} \times D_{xy} \times D_z$  samples, with  $D_{xy} > 1$  and  $D_z > 1$ . Each ray samples  $D_z$  positions inside of each cell, however, the sampling error analysis allows to locally reduce the number of samples in powers of 2. Thus, the *sampling level*  $l_c$  corresponds to  $D_z/2^{l_c}$  sampling positions per ray in cell  $c$ .

**Sampling Error Analysis:** To locally adapt the sampling rate for each cell, a formulation of the rendering equation based on hierarchical interval arithmetic is introduced. Inside of each cell, upper and lower bounds to radiance and transparency values due to adaptive sampling are determined. This is efficiently realized by mapping particle data onto one representative ray that is cast through the cell instead of sampling all  $D_{xy}^2$  rays. The cell bounds are then composited in front-to back to efficiently predict screen space errors due to adaptive sampling and to compute an optimal combination of per-cell sampling levels (see Sec. 5.3).

The interval arithmetic is further extended so that adaptive sampling in viewing and lateral directions can be combined. To that end, a representative radiance and transparency is calculated for each cell. Lateral adaptivity is then realized by rendering the cell's radiance and transparency as a *super-pixel* that spans all pixels the cell covers in screen space (see Sec. 5.4).

**Greedy Optimization:** In order to achieve an efficient adaptive ray casting, the degree of adaptivity has to be maximized for each cell for a user-defined screen space error tolerance (see Sec. 5.3.4). To yield higher speed-ups, the error prediction can be relaxed by removing the lateral error in the sampling error analysis. This leads to a performance optimized greedy algorithm which practically still satisfies the error bounds while allowing for higher sampling levels in viewing direction and for an alternative super-pixel rendering (see Sec. 5.4).

**Cell Merging:** Consecutive cells in viewing direction that support higher sampling levels are merged to reduce the number of particles to be sampled



**Figure 5.2:** Sparse access structure. Cell merging ensures a constant number of  $D_{xy} \times D_{xy} \times D_z$  samples per cell. The image plane is split into ray bundles of size  $D_{xy} \times D_{xy}$  each storing cells  $c_1$  and  $c_L$  in a look-up table.  $M_x \times M_y \times M_z$  is the maximum number of cells present in a dense grid.

and keep a constant number of samples per cell (see Sec. 5.5.2).

**Adaptive Ray Casting:** The final volume ray casting algorithm simplifies to an entirely thread-coherent front-to-back traversal of cells. Cells are either rendered as super-pixels or else all rays only sample the necessary subset of particles that has been assigned to the cell (see Sec. 5.5.3).

## 5.3 Sampling Error Analysis Framework

The goal of the sampling error analysis is to determine sampling levels for each cell so that a user-defined error tolerance  $E_I$  is not exceeded by the screen space error. Therefore, a hierarchical interval arithmetic scheme is proposed to determine upper and lower bounds to the radiance and transparency on the level of ray bundles. These bounds determine the screen space error via the volume rendering equation (Eq. (5.1)).

Error tolerance

The error analysis works hierarchically on the level of samples, of cells and finally of whole ray bundles. Inside each cell, bounds to the quantity field at each sampling depth along the ray bundle are first calculated. The quantity bounds  $Q^\perp$  are mapped to radiance and transparency sample bounds  $i^\perp, t^\perp$  that bound the radiance and transparency at each sampling depth for all rays of the bundle (see Sec 5.3.1). Sample bounds are composited inside cells to cell bounds  $I^\perp, T^\perp$  (see Sec. 5.3.2). The screen space error analysis (see Sec. 5.3.3) composites bounds of traversed cells in viewing direction to ray bounds  $\mathbb{I}^\perp, \mathbb{T}^\perp$

Error analysis overview

using different combinations of cell sampling levels. Note that only  $D_z$  lower and upper bounds have to be composited inside each cell to yield cell bounds and only one upper and lower cell bound has to be composited per cell along the cell sequence to efficiently calculate the bounds for all rays of a ray bundle.

#### Greedy optimization

Based on the error analysis, cell sampling levels  $l_c^{\text{opt}}$  are determined for all cells  $c$  for the final adaptive ray casting. As a direct or analytic identification of the optimal sampling levels is not possible, sampling levels are greedily optimized while keeping the width of the error bound below a user defined error tolerance  $E_I$  (see Sec. 5.3.4). Alg. 5.1 gives pseudocode of the sampling error analysis.

---

```

1: for all cell  $c \in$  perspective grid do
Sample bounds
2:   for all Sample  $0 \leq k < D_z$  do
3:      $Q_k^\perp =$  lateral_quantity_bounds(particle quantities  $Q_{j_c}$ )
4:      $[i_k^\perp, t_k^\perp] =$  sample_bounds( $Q_k^\perp$ )
5:   end for
Cell bounds
6:   for all Sampling level  $l$  do
7:      $[I_c^{\perp,l}, T_c^{\perp,l}] =$  cell_bounds( $\{i_k^\perp\}, \{t_k^\perp\}, l$ )
8:     if  $w(I_c^{\perp,k}) > E_I$  or  $w(T_c^{\perp,k}) > E_I$  then
9:        $l_c^{\text{max}} = \max(l - 1, 0)$   $\triangleright$  Error tolerance exceeded, last level is maximum
10:      break
11:    end if
12:  end for
13: end for
Greedy optimization
14: for all Ray bundle  $b$  do
15:    $\vec{l} =$  greedy_optimization( $E_I, \{I_c^{\perp,l}\}, \{T_c^{\perp,l}\}, \{l_c^{\text{max}}\}$ )
16: end for

```

---

**Algorithm 5.1:** The proposed sampling error analysis determines bounds for particle quantities and samples in cells, composites sample bounds to cell bounds and finally composites cell bounds along ray bundles to determine ray bundle bounds and thus to the screen space error. The greedy algorithm finally returns appropriate cell sampling levels  $\vec{l}$ .

### 5.3.1 Lateral Quantity and Sample Bounds

First, the maximum and minimum of all  $D_{xy}^2$  samples at each sampling depth  $s_k$  inside of cells are determined. Since CSPH quantities are affine combina-

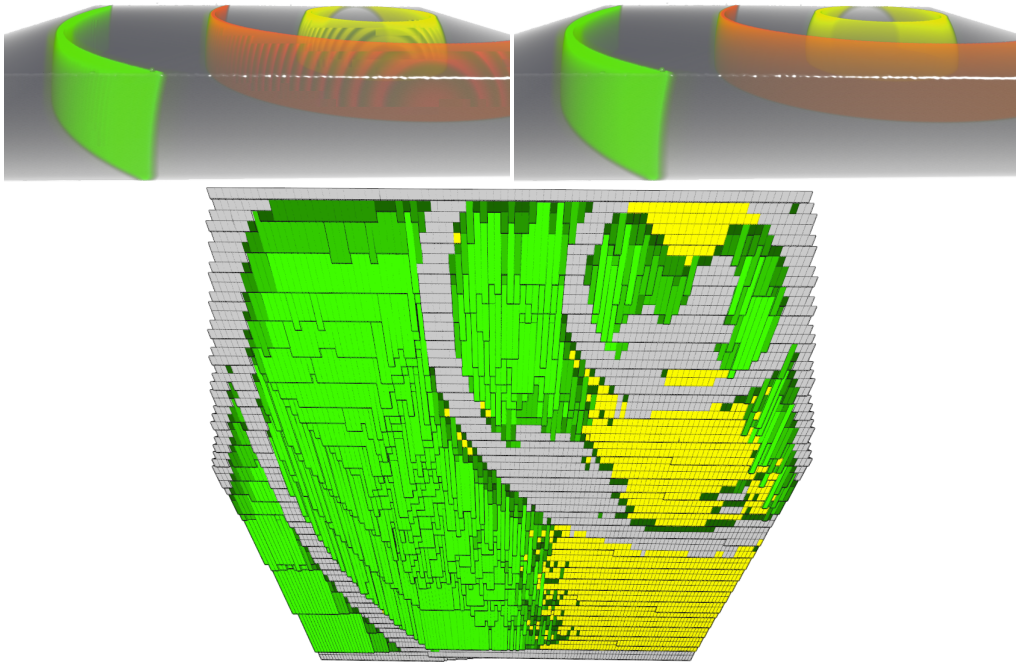


tions of particle quantities (cf. Sec. 3.1.3), sample quantities are bounded by the particle quantities contributing to the lateral neighborhood at depth  $s_k$ , i.e.

Lateral quantity bounds

$$Q_k^\top = \max_{|z_j - s_k| < h_j} Q_j, \quad Q_k^\perp = \min_{|z_j - s_k| < h_j} Q_j,$$

where  $z_j$  is the z-coordinate of particle  $j$  in view space and  $h_j$  its radius. As transfer functions can introduce high frequencies (see Fig. 5.3), the bounds  $Q_k^\perp$



**Figure 5.3:** A radially increasing concentration profile rendered with a complex transfer function. The center of the concentration profile is in the upper right of all images. High-frequency transfer functions introduce visible artifacts if sampling rates are naïvely reduced (left). The proposed screen space error analysis locally adjusts sampling rates to retain visible features (right). Only the volume covered by gray cells has to be sampled at the highest sampling rate to give correct rendering results, yellow cells are rendered as super-pixels and green cells by reducing the sampling rate in viewing direction (bottom).

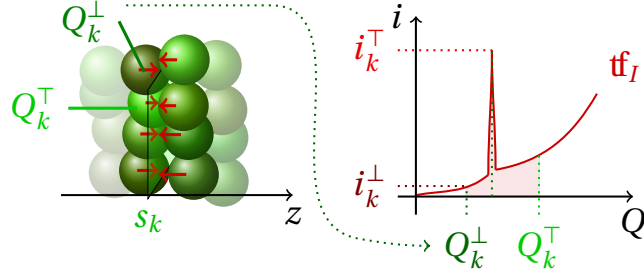
of the quantity field are mapped to radiance bounds using exhaustive search for extremes in  $\text{tf}_I(Q)$  as shown in Fig. 5.4:

Radiance sample bounds for level 0

$$i_k^\top = \max_{Q \in [Q_k^\perp, Q_k^\top]} \text{tf}_I(Q), \quad i_k^\perp = \min_{Q \in [Q_k^\perp, Q_k^\top]} \text{tf}_I(Q).$$

In practice,  $i_k^\top, i_k^\perp$  are accessed from a dynamically calculated 2D-texture using  $Q_k^\perp$  and  $Q_k^\top$  as lookup coordinates. Analogously, transparency bounds  $t_k^\top$  are

computed and stored by analyzing  $\text{tf}_T(Q)$ . Note that only  $D_z$  sample bounds are calculated each of which strictly bounds the  $D_{xy}^2$  samples of the ray bundle passing through the cell. By reducing the  $D_{xy}^2$  ray samples to sample bounds, the following stages of the sampling error analysis has to be performed only on the  $D_z$  representative radiance and transparency sample bounds.



**Figure 5.4:** For each sampling depth  $s_k$ , quantity bounds  $Q_k^T, Q_k^\perp$  are determined by finding the maximum and minimum quantities of the particles that contribute to  $s_k$  depicted by the red arrows (left). The transfer function's maximum and minimum  $i_k^T, i_k^\perp$  is searched in the parameter range  $[Q_k^\perp, Q_k^T]$  (right).

### 5.3.2 Cell Bounds

The second stage of the hierarchical interval arithmetic calculates bounds for cell  $c$  at sampling level  $l$ . For  $l = 0$ , cell radiance and transparency bounds  $I_c^{\perp,0}, T_c^{\perp,0}$  are found by compositing sample bounds  $i_k^\perp, t_k^\perp$  using the volume rendering Eq. (5.1). Note that the sample bounds already include the lateral variation of the ray bundle.

If a cell is sampled at a higher level  $l$ , less samples are used. Each of these coarser samples, however, represents a larger span of  $\Delta_l = 2^l$  samples of level 0 along the ray. Thus, samples of higher levels are bounded by the minimum and maximum of the original samples at level 0 they span. For samples  $k = 0, \dots, D_z - 1$  this yields

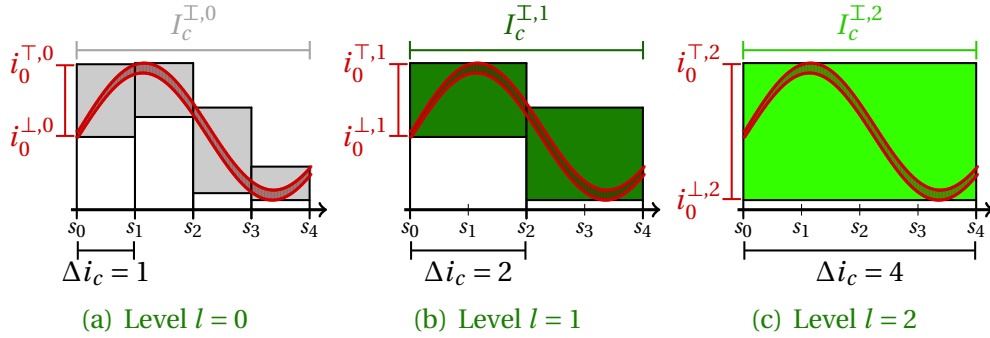
$$i_k^{\perp,l} = \max_{j \in \left\{ \left\lfloor \frac{k}{\Delta_l} \right\rfloor \cdot \Delta_l, \dots, \left\lceil \frac{k}{\Delta_l} \right\rceil \cdot \Delta_l \right\}} \left\{ i_j^\perp \right\}.$$

For  $i_k^{\perp,l}$ , min is used instead of max. Analogously,  $t_k^{\perp,l}$  is calculated using transparency samples. After determining the proper set of sample bounds for level  $l$  in cell  $c$ , the samples are composited using the volume rendering Eq. (5.1), to yield cell bounds  $I_c^{\perp,l}$  and  $T_c^{\perp,l}$  as shown in Fig. 5.5.

As large radiance variations, i.e.  $w(I_c^{\perp,l}) \gg 0$ , strongly influence the error bounds, the sampling level of each cell  $c$  is limited according to the cell's

Sampling level and step size

Radiance sample bounds for higher levels



**Figure 5.5:** Computation of radiance bounds of cell  $c$  for three sampling levels. With larger sampling step sizes  $\Delta i_c = 4$ , bounds (right, green area) differ from accurate bounds at  $\Delta i_c = 1$  (left, gray area) to the signal (red area). This may lead to sampling errors. Compositing sample bounds  $i_k^{\perp,l}$  and  $i_k^{\top,l}$  yields cell radiance and transparency bounds  $I_c^{\perp,l}, T_c^{\perp,l}$ .

maximum potential error to  $l_c^{\max} = \operatorname{argmax}_l \{w(I_c^{\perp,k}) < E_l\}$ . Preventing that coarser sampling of a single cell exhausts the error tolerance, yields a better distribution of the error tolerance between cells. The more cells are sampled coarsely the higher the speed-up.

An important aspect is the handling of cells  $c$  which contain surface particles. As particles only model the fluid but not the air phase, radiance and transparency bounds cannot be computed correctly. Surface particles are detected using the approach of Orthmann et al. [OHB\*13]. Assume one particle covers all sampling depths inside a cell but is only hit by some rays while all other rays pass through empty space, then the cell bounds yield  $w(I_c^{\perp,0}) = 0$  although errors are introduced into the image. As only the particle's z-component is used to calculate sample bounds, empty space in the lateral neighborhood is not detected. In order to prevent erroneous adaptive sampling in surface cells,  $l_c^{\max} = 0$  is set. Apart from this, surface particles are treated like bulk particles.

Surface cells

### 5.3.3 Screen Space Error Analysis

The final stage of the hierarchical interval arithmetic computes bounds for ray bundles by compositing cell bounds. As ray bundle bounds automatically bound each single ray of the bundle, they also naturally bound the screen space error. To calculate ray bundle bounds over a cell sequence  $c_1, \dots, c_L$  with cell sampling levels  $\vec{l} = (l_1, \dots, l_L)$ , the volume rendering equation is applied as

Interval volume rendering

$$\mathbb{I}_L^{\perp, \vec{l}} = \sum_{c=1}^L I_c^{\perp, l_c} \prod_{D=1}^{c-1} T_D^{\perp, l_D}, \quad \mathbb{T}_L^{\perp, \vec{l}} = \prod_{c=1}^L T_c^{\perp, l_c}. \quad (5.4)$$

The width of the ray bundle bounds  $w(\mathbb{I}_L^{\mathbb{I}, \vec{l}})$  is an upper bound to the screen space error that sampling at cell sampling levels  $\vec{l}$  may introduce. However, still an optimal choice of  $\vec{l}$  has to be found that satisfies the user's error tolerance  $w(\mathbb{I}_L^{\mathbb{I}, \vec{l}}) < E_I$ . Therefore, the following greedy algorithm is proposed.

### 5.3.4 Greedy Optimization of Sampling Levels

The algorithm starts by compositing bounds from Eq. (5.4) using  $l_c = 0$  for all cells  $c$ , i.e.,  $\vec{l} = \vec{0} = (0, \dots, 0)$ . Walking backwards from cell  $c_L, \dots, c_1$ , sampling levels are then greedily increased, while the error tolerance  $E_I$  for the ray bundle is not exceeded. While finding an optimal combination of sampling levels on-the-fly is infeasible, greedily increasing sampling levels back-to-front allows to skip many samples without introducing visible errors into the final image. Given bounds  $\mathbb{I}_L^{\mathbb{I}, \vec{0}}, \mathbb{T}_L^{\mathbb{I}, \vec{0}}$  over the full cell sequence, Eq. (5.4) is decomposed by splitting off the last cell and replacing it by coarser sampling level  $l_L$  to get the new bounds  $\mathbb{I}_L^{\mathbb{I}, (0, \dots, 0, l_L)}$ . The sampling level is increased while  $w(\mathbb{I}_L^{\mathbb{I}, (0, \dots, 0, l_L)}) < E_I$  until level  $l_L^{\max}$  is reached. Hence, it sets

$$l_L^{\text{opt}} = \arg \max_{l_L \leq l_L^{\max}} \left\{ w(\mathbb{I}_L^{\mathbb{I}, (0, \dots, 0, l_L)}) < E_I \right\}. \quad (5.5)$$

Having decided on the sampling level for cell  $c_L$ , the final results are collected for cell  $c_L$  in background radiance  $\mathbb{I}_{\text{back}}^{\mathbb{I}} = I_L^{\mathbb{I}, l_L^{\text{opt}}}$ . As shown in Fig. 5.6, the algorithm proceeds backwards sequentially testing coarser sampling levels  $l_c$  in cell  $c$  in back-to-front order:

$$\mathbb{I}_L^{\mathbb{I}, (0, \dots, 0, l_c, l_{c+1}^{\text{opt}}, \dots, l_L^{\text{opt}})} = \mathbb{I}_{c-1}^{\mathbb{I}, \vec{0}} + \mathbb{T}_{c-1}^{\mathbb{I}, \vec{0}} \cdot (I_c^{\mathbb{I}, l_c} + T_c^{\mathbb{I}, l_c} \cdot \mathbb{I}_{\text{back}}^{\mathbb{I}}).$$

Analogously to Eq. (5.5) the optimal level  $l_c^{\text{opt}}$  is computed and, finally, the background radiances are updated:

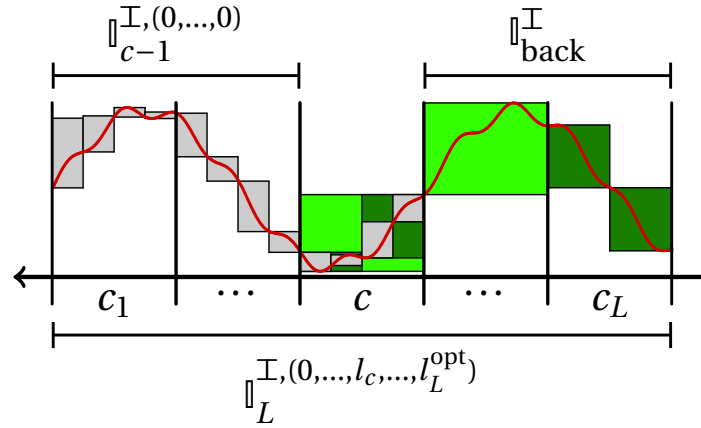
$$\mathbb{I}_{\text{back}}^{\mathbb{I}} \leftarrow I_c^{\mathbb{I}, l_c^{\text{opt}}} + T_c^{\mathbb{I}, l_c^{\text{opt}}} \cdot \mathbb{I}_{\text{back}}^{\mathbb{I}}$$

Strict screen space  
error bounds

The derivation of error bounds using interval arithmetic guarantees, if  $w(\mathbb{I}_L^{\mathbb{I}, \vec{l}}) < E_I$ , the screen space error stays below  $E_I$  for levels  $\vec{l}$ .

## 5.4 Analysis and Performance Optimizations

Although the sampling error analysis framework of Orthmann [Ort14] described in Secs. 5.2 and 5.3 reveals proper bounds to the screen space error, it suffers from severe limitations; see [Ort14] and Tab. 6.1 in there:



**Figure 5.6:** Back to front exchange of sampling levels for an unknown signal (red). In the depicted step, the greedy algorithm estimates the screen space error (green and gray areas) for different sampling levels  $l_c$  of cell  $c$ , i.e. by compositing cell bounds  $I_c^{I, l_c}$  with foreground radiances  $I_c^{I, \bar{0}}$  and the already adaptively sampled background  $I_{back}^I$ .

- Due to the pessimistic estimation, the error tolerance has to be large in order to achieve any speedups.
- The large difference between the screen space error and the error tolerance leads to poor error control.
- Speedups are very limited as adaptivity is restricted to the viewing direction.

In this section, major improvements to the error analysis framework are proposed that are necessary to alleviate these shortcomings and achieve a competitive rendering. By relaxing the error estimation, the error tolerance is utilized more exhaustively which leads to large gains in performance and more precise control over the screen space error. Moreover, combining adaptive sampling in viewing direction with lateral adaptivity using super-pixels yields additional large speedups at controlled screen space error.

### 5.4.1

#### Relaxed Error Estimation

Cells with a large width  $w(I^{\mathbb{I}, 0})$  strongly increase the error estimation along rays. However, they do not introduce any screen space error if they are sampled at the highest sampling rate. Thus, instead of interpreting the width on level  $l = 0$  as a measure of error introduced due to adaptive sampling, the width  $w(I_c^{\mathbb{I}, 0})$  can be more appropriately considered as a measure of lateral variation due to the reduction of  $D_{xy}^2$  samples to sample bounds (see Sec. 5.3.1). By re-

Remove influence of lateral variation

moving  $w(I_c^{\top,0})$  from the interval widths of higher levels, the effects of adaptive sampling in viewing direction can be separated from the influence of lateral variation. The proposed relaxation approach partially subtracts  $w(I_c^{\top,0})$  from the lower bounds of  $l = 0$  as

$$I_{c,\text{relax}}^{\perp,0} := I_c^{\top,0} - (1 - E_{\text{relax}})w(I_c^{\top,0}),$$

and for higher levels  $l > 0$  as

$$I_{c,\text{relax}}^{\perp,l} := I_{c,\text{relax}}^{\perp,0} + (I_c^{\perp,l} - I_c^{\perp,0}).$$

The user-defined relaxation parameter  $E_{\text{relax}} \in [0, 1]$  controls the influence of cell bounds on level  $l = 0$  on the error estimation of ray bundles.  $E_{\text{relax}} = 1$  completely removes the influence of the width of level  $l = 0$  and  $E_{\text{relax}} = 0$  gives a strict error estimation without relaxation. Transparency is adjusted analogously.

As relaxing, i.e. decreasing, the upper transparency bound could lead to severe underestimates for the errors behind the current cell, the upper error bounds are left unchanged. Thus, it is ensured that errors in the background remain visible and are adequately accounted for in the error analysis.

Note, that the relaxation approach does not affect the maximum sampling levels  $l_c^{\text{max}}$  per cell, as they also account for the lateral error in the ray bundle (cf. Sec. 5.3.2).

## 5.4.2

### Lateral Adaptive Sampling Using Super-Pixels

In cells that contain large differences in viewing direction but sample bounds  $i^{\top}$  of small width, reducing the sampling rate in viewing direction causes large errors. In these cases, adapting the lateral sampling rate instead, i.e., the number of cast rays, is a promising alternative. To this end, the non-relaxed cell bounds can directly be used by rendering the mean radiance  $\frac{I_c^{\top,0} + I_c^{\perp,0}}{2} =: I_{\text{SP}}$  and transparency  $\frac{T_c^{\top,0} + T_c^{\perp,0}}{2} =: T_{\text{SP}}$  as super-pixels in the final image. Instead of sampling any particles,  $(I_{\text{SP}}, T_{\text{SP}})$  is just composited to the accumulated image for all pixels covered by the cell. This approach optimally reuses the previously calculated cell bounds and can, thereby, drastically reduce the number of sampling operations of the final ray casting.

Super-pixel radiance and transparency

Error estimation of super-pixels

Apparently, super-pixel rendering is by far faster than sampling particles but causes larger screen space errors. To estimate this error, the relaxation scheme cannot be used. By relaxing bounds, the lateral error is removed from the error estimation, however, rendering super-pixels introduces exactly these lateral errors. To properly bound errors due to the reduced lateral resolution of super-pixels, the original cell bounds of level 0 have to be used.

### 5.4.3 Combined Greedy Optimization

The lateral sampling optimization using super-pixels can be combined efficiently with the adaptivity in viewing direction using the proposed error estimation. In order to decide between super-pixel rendering and adaptive sampling in viewing direction, the relaxed cell bounds  $I_{c,\text{relax}}^{\perp,l}$ ,  $l = 0, \dots, l^{\text{max}}$  and the non-relaxed cell bound  $I_c^{\perp,0}$  are sorted in ascending order. The greedy algorithm just works as described in Sec. 5.3.4, only if  $I_c^{\perp,0}$  is found to be the largest acceptable error, the super-pixel will be used during ray casting.

In Fig. 5.3, a radially increasing concentration profile was rendered using both optimizations. Areas where concentration gradient and viewing rays run perpendicularly have small sample bound widths, hence, they are rendered using super-pixels (yellow cells). In other areas only adaptive sampling in viewing direction is applicable (green cells).

## 5.5 Implementation Details

The algorithm was implemented using NVIDIA CUDA 7.5 and OpenGL. First, the details of the particle-to-cell mapping are given which in fact constructs the perspective grid as particle access structure (Sec. 5.5.1). Then, the cell merging approach will be presented which allows to reduce the sampling rate at a constant number of samples per cell and prevents unnecessary duplicate particle accesses (Sec. 5.5.2). Last, some details of the ray casting are presented (Sec. 5.5.3).

### 5.5.1 Particle Access via Perspective Grids

In the perspective grid structure [Ort14], each cell  $c$  stores references to particles that overlap the cell's volume  $\Omega_c := \{\mathbf{x} \mid C(\mathbf{x}) = c\}$  which is defined via the indexing function  $C : \mathbb{R}^3 \rightarrow \mathbb{N}_0$

$$C(\mathbf{x}) = (C_x(\mathbf{x}) M_y + C_y(\mathbf{x})) M_z + C_z(\mathbf{x}), \quad (5.6)$$

which subdivides the view space into  $M_x \times M_y \times M_z$  view-aligned cells. The cell coordinates  $(C_x(\mathbf{x}), C_y(\mathbf{x}), C_z(\mathbf{x}))$  at position  $\mathbf{x} = (x, y, z)^T$  in view space are given as

$$\left( \left\lfloor \frac{x}{\gamma \alpha(z)} + \frac{M_x}{2} \right\rfloor, \left\lfloor \frac{y}{\alpha(z)} + \frac{M_y}{2} \right\rfloor, \left\lfloor M_z \left( \frac{\ln\left(\frac{z}{s_{\text{near}}}\right)}{\ln\left(\frac{s_{\text{far}}}{s_{\text{near}}}\right)} \right) \right\rfloor \right). \quad (5.7)$$

Perspective cell index

Here,  $C_z$  is derived using the inverse of the following perspective transformation that maps samples from uniform sampling space to non-uniform view space [FAW10]:

$$s_i = s_{\text{near}} \left( \frac{s_{\text{far}}}{s_{\text{near}}} \right)^{\frac{i}{N}}. \quad (5.8)$$

$C_x$  and  $C_y$  are defined by the window's aspect ratio  $\gamma$ .  $\alpha(z) = \frac{2z}{M_y} \tan\left(\frac{\sigma}{2}\right)$  is the cell height at distance  $z$ , which depends also on the current field of view  $\sigma$  of the view frustum.

Assigning particles  $j$  to cells  $c$  yields a list of particle-cell-pairs  $(j, c)$ . Sorting the particle-cell pairs by the cell index  $c$  yields cell-particle-pairs that describe the perspective access structure. Only cells are referenced to which at least one particle contributes and only particles are present that contribute to at least one cell. Thus, empty cells are skipped implicitly and frustum culling is performed (cf. Fig. 5.2).

As ray bundles always sample a whole cell sequence in viewing direction, the indices of the first and last cell of the sequence are also stored. To allow for a fast access to particles inside cells, the first index  $j_c$  and the number  $N_c$  of the relevant cell-particle-pairs for each cell are stored. During error estimation and ray casting, the “traversal logic” reduces to a simple loop over the relevant cells in the sequence of cell-particle pairs.

## 5.5.2 Cell Merging

Although the adaptive sampling can increase the sampling level along rays inside cells, this results in a different number of samples per cell and complicates the ray casting. Furthermore, particles in subsequent cells often are redundantly sampled. Thus, a pairwise merging of neighboring cells [Ort14] is performed that allows for sampling at a higher level. Thus, overlapping particle references in the newly merged cell can be removed and, instead of adjusting the number of samples per cell, only the sampling distance is adjusted. This both simplifies the ray casting and reduces the number of particles that have to be sampled redundantly.

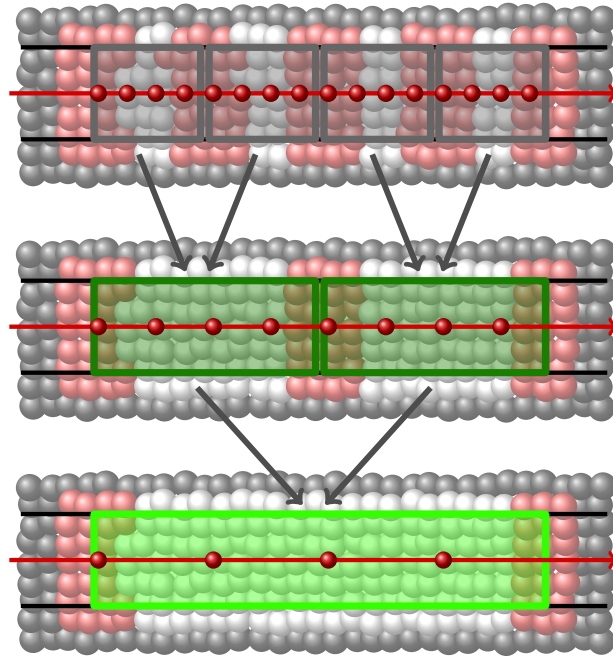
Constant number of  
cell samples

Merging of cells is realized level by level, starting from level 0. Figure 5.7 shows two subsequent merging steps. Neighboring cells are only merged if they both allow for sampling at the next higher level. During the process of setting up the grid, the linear cell index  $c$  is bit-shifted one bit to the left and the least significant bit is set iff a particle contributes to samples of the subsequent cell  $c + 1$ . The cell index then also directly indicates if a particle is redundantly referenced in the subsequent cell. After sorting the particle-cell-

Remove redundant  
particles



pairs, all redundantly referenced particle indices of a cell contiguously lie in memory and can be removed easily.



**Figure 5.7:** Two iterations of the cell merging. The number of samples remains constant inside the merged cells, however, the distance between samples (red) is doubled. Redundant particle references (rose) between cells are removed and uniquely referenced in the merged cell (white).

Only cells are merged that are sampled adaptively in viewing direction. Cells that are rendered as a super-pixel would not benefit from merging because no particles are accessed during ray casting.

### 5.5.3 Adaptive Ray Casting

Alg. 5.2 depicts the pseudocode of the proposed ray casting algorithm as described by Orthmann [Ort14] that has been extended to support the rendering of super-pixels. Since consecutive cells in viewing direction are neighbors in memory, no specific cell finding logic is required. Either a super-pixel can be rendered or particles have to be sampled. As all rays then have to sample the same set of particles, particle data for all  $D_{xy} \times D_{xy}$  adjacent rays can efficiently be cached using shared memory (cf. red lines in Alg. 5.2) and each particle has to be read only once per cell. Using a small thread-local cache, particles scatter their data to  $D_z$  samples of a ray at once to further reduce memory traffic. In

the GPU-implementation of Alg. 5.2,  $D_{xy} = 8$  and  $D_z = 16$  were used so the maximum sampling level is  $\log_2 D_z = 4$ .

---

```

1:  $I = \mathbf{0}, T = 1,$  ▷ initialize ray radiance and transparency
2:  $\mathbf{x}[D_z] = \mathbf{0}$  ▷ sampling positions
3:  $Q[D_z] = 0, V[D_z] = 0$  ▷ sampled quantities and volumes
4: for all Cell  $c$  in  $c_1, \dots, c_L$  do
5:    $j_c, N_c$  ▷ index to particle array and number of particles (cf. Sec. 5.5.1)
6:    $i_c, \Delta i_c$  ▷ linear cell index (cf. Eq. (5.6)) and cell sampling distance
7:    $[j_c, N_c, i_c, \Delta i_c] = \text{read\_celldata}(c)$ 

```

---

Super-pixel Rendering

---

```

8:   if  $c$  can be rendered as Super-pixel then
9:      $[I, T] = \text{composite}(I, T, I_{\text{SP}}, T_{\text{SP}})$  ▷ cf. Sec. 5.4.2
10:    continue
11:   end if

```

---

Sampling

---

```

12:  for all Sample  $k$  in  $0, \dots, D_z - 1$  do
13:     $Q[k] = V[k] = 0$  ▷ initialize sampled quantity and volume
14:     $\mathbf{x}[k] = \mathbf{x}(s(i_c + \Delta i_c k))$  ▷ get ray sampling position  $\mathbf{x}$  in view space
15:    end for
16:    for all Particle  $j$  in  $j_c, \dots, j_c + N_c - 1$  do
17:       $[\mathbf{x}_j, h_j, Q_j, V_j] = \text{read\_particledata}(j)$ 
18:      for all Sample  $k$  in  $0, \dots, D_z - 1$  do
19:         $Q[k] = Q[k] + Q_j V_j W_j(\mathbf{x}[k])$ 
20:         $V[k] = V[k] + V_j W_j(\mathbf{x}[k])$ 
21:      end for
22:    end for
23:    for all Sample  $k$  in  $0, \dots, D_z - 1$  do
24:      if  $(V[k] > 0)$   $Q[k] = Q[k] / V[k]$  ▷ CSPH normalization (cf. Eq. (3.12))
25:    end for

```

---

Compositing

---

```

26:  for all Sample  $k$  in  $0, \dots, D_z - 1$  do
27:     $\Delta s_k = s(i_c + \Delta i_c (k + 1)) - s(i_c + \Delta i_c k)$  ▷ view space distance
28:     $[I, T] = \text{composite}(I, T, \text{tf}_I(Q[k]), \text{tf}_T(Q[k])^{\Delta s_k})$ 
29:  end for
30: end for

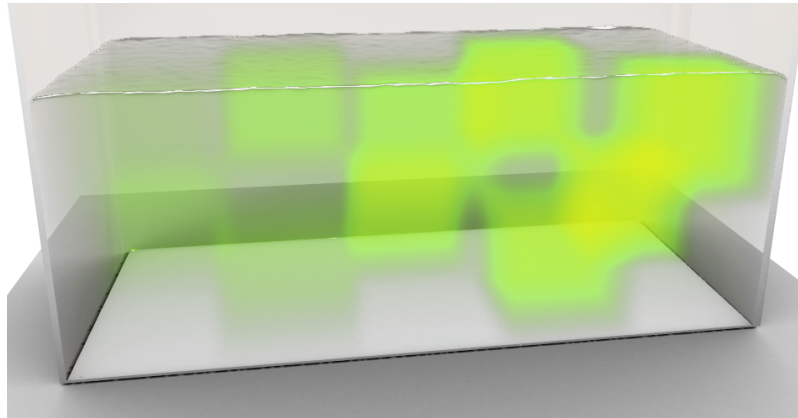
```

---

**Algorithm 5.2:** Thread-coherent volume ray casting with adaptive sampling step sizes (green), super-pixel rendering and efficient shared memory access (red). Particles contribute to  $D_z$  ray samples at once using a thread-local cache.

## 5.6 Results and Discussion

In order to demonstrate the performance and to evaluate the image quality, the proposed adaptive ray casting has been tested in five scenarios: The *Checker Board scene* with cubes of varying concentrations that diffuse over time (see Fig. 5.8), the *Mixer scene* simulating the mixing of solvent with dye streaming

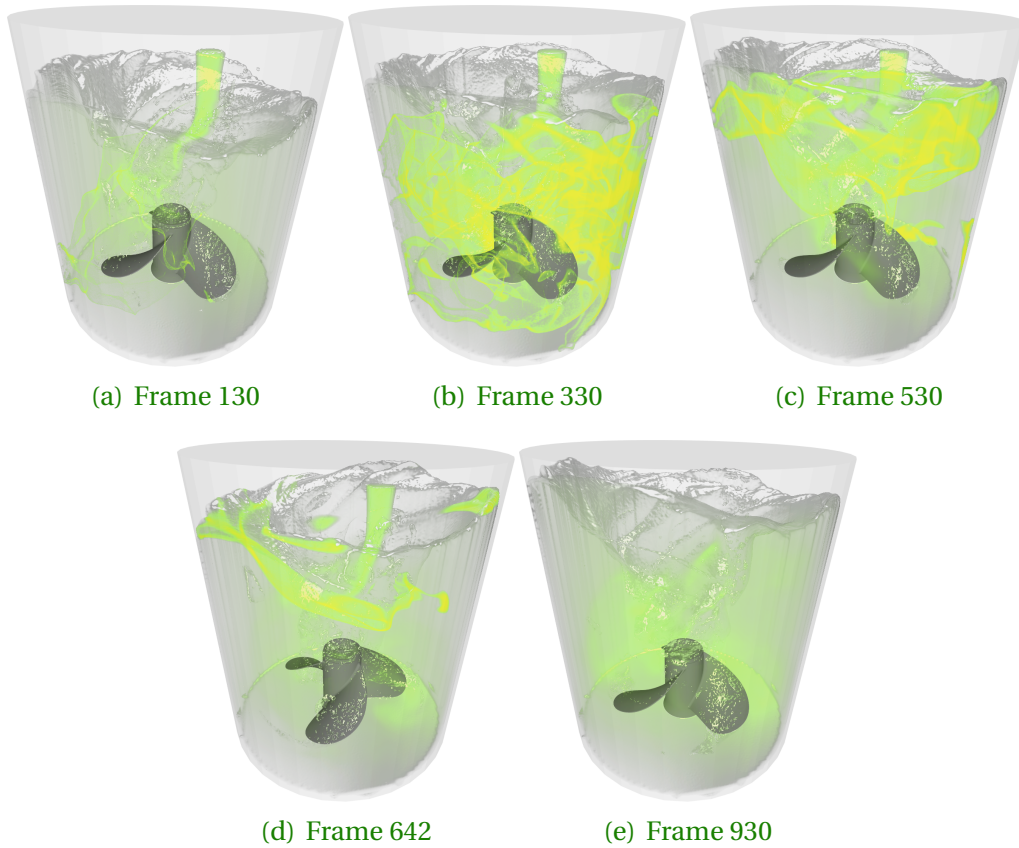


**Figure 5.8:** 3D checker board of increasing concentrations from left to right and front to back.

from an inlet (see Fig. 5.9), the *HPG 2016 scene* with fluid letters splashing to the ground (see Fig. 5.1), the static *Radial Concentration scene* of a fluid with radially increasing concentrations rendered with an extreme transfer function (see Fig. 5.3), and the *Flubber scene* with two fluids mixing while orbiting a virtual center of gravity (see Fig. 5.10). The relaxation factor for the error analysis was set to  $E_{\text{relax}} = 1$  in all examples and super-pixel rendering was enabled if not stated otherwise. Simulations and renderings were carried out on an NVIDIA GeForce GTX Titan with 6 GB of VRAM. All scenes were rendered on a  $1024^2$  viewport. Surfaces were also ray cast using the view-aligned access structure. However, to get smoothed surfaces, an increased particle support radius was employed. As the focus of this chapter lies on the adaptive volume ray casting, timings for surface rendering are not included.

Tab. 5.1 shows particle counts, timings and speed-ups as well as errors using different tolerances  $E_I$  averaged over all frames of the respective scene. Errors are given as maximum absolute difference over all pixels in any color or alpha channel  $\in [0, 1]$  compared to the non-adaptive rendering  $E_I = '-'$ . Values of  $\{0.001, 0.004, 0.01\}$  relate to error values  $\{0.255, 1.02, 2.55\}$  in the respective 8-bit value range  $[0, 255]$ , respectively.

The following discussion addresses the image quality and the performance

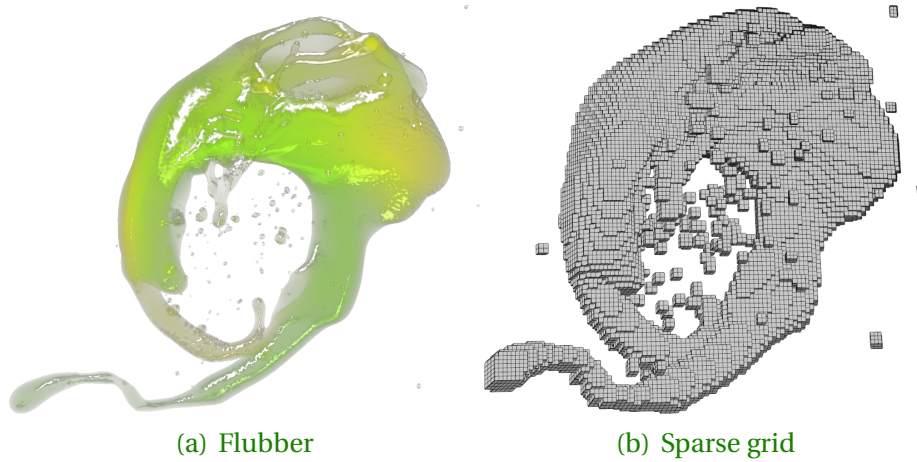


**Figure 5.9:** A mixer is causing a stream of green dye to mix with solvent in the fluid tank. Above, five of the 1000 frames that were rendered are shown. The sequence progresses from left to right and from top to bottom.

characteristics of the adaptive volume ray casting approach and will also compare the method to previous work.

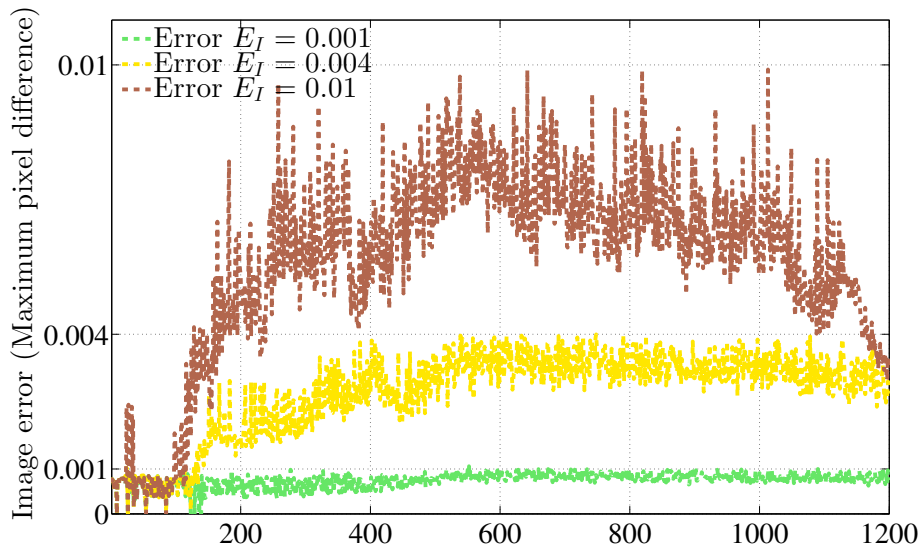
#### Image quality

Regarding *image quality*, the errors due to the adaptive volume ray casting of the Flubber, the Checker Board, and the Radial Concentration scenes always stayed below the error tolerance. For both the Mixer and HPG 2016 scenes, errors of all frames stayed below the error tolerance for  $E_I = 0.01$  and  $E_I = 0.004$ . For  $E_I = 0.001$ , however, the error exceeded the tolerance for 1 of the 1200 frames of the Mixer scene and for 1 of the 1500 frames of the HPG 2016 scene. Table 5.2 summarizes the errors of the approach for all frames of all scenes. Figure 5.11 shows the error behavior for the Mixer scene over the full 1200 simulation frames. The sampling error analysis allowed the adaptive sampling to very precisely exhaust the error tolerance  $E_I$ . Additionally, the Mixer scene was rendered using  $l_c^{\max}$  for all cells  $c$  (cf. Sec. 5.3.2), i.e., the sampling level for each cell was limited separately without applying the greedy algorithm to



**Figure 5.10:** A frame of the Flubber scene and the respective sparse grid showing the large surface to volume ratio of this scene.

control the screen space error along cell sequences of rays. The error of 0.016 exceeded the user-defined tolerance  $E_I = 0.004$  (cf. row ‘No greedy alg.’ in Tab. 5.1) which indicates that the error estimation has to consider the whole cell sequence along ray bundles.



**Figure 5.11:** Errors of the 1200 simulation frames (x-axis) of the Mixer scene. Errors are displayed as dotted lines against the y-axis.

Considering *performance*, the adaptive sampling yields total speed-up factors between 1.08 and 1.62 in all scenes (cf. Tab. 5.1).

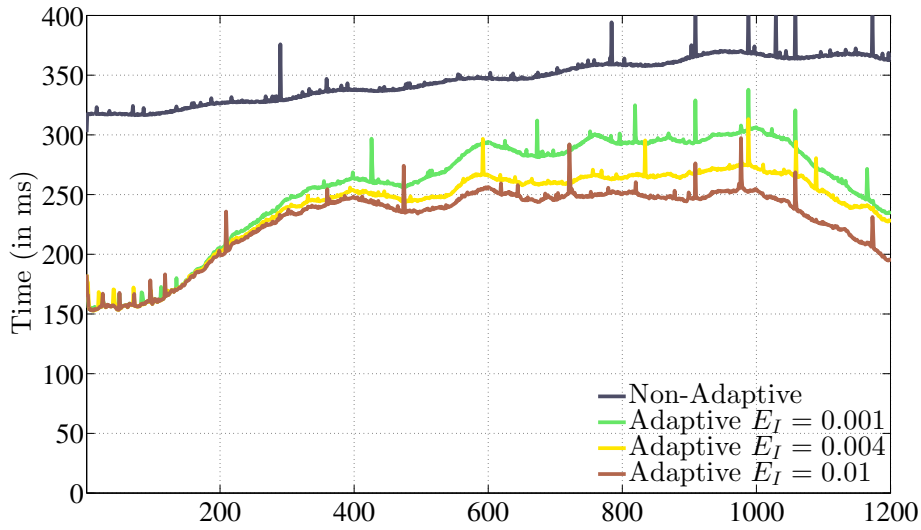
Rendering  
performance

**Table 5.1:** GPU timings and speed-ups of adaptive ( $E_I \geq 0$ ) and non-adaptive ( $E_I = '-'$ ) volume rendering. 'Grid' is the particle-cell assignment and setting up the view-aligned grid, 'Adapt' is the sampling error analysis and the merging of cells, 'RC' is the ray casting and 'Total' gives the total time to render. 'Error' is given as maximum absolute single pixel difference in any color or alpha channel compared to the non-adaptive rendering. 'Speedup' relates the timing to timings for  $E_I = '-'$ . Results are averaged over all frames of the given scenes.

Scene (#Part.)	Image quality		Timing (in ms)			Speedup	
	$E_I$	Error $\in [0, 1]$	Grid	Adapt	RC (Total)	RC (Total)	
Flubber (500 K) No surface	-	-		-	<b>66</b> (73)	<b>1</b> (1)	
	0.004	7.2e-04	7	5	<b>44</b> (56)	<b>1.5</b> (1.3)	
	0.004	$\gg E_I$	7	8	<b>19</b> (34)	<b>3.5</b> (2.15)	
Checker Board (1.2 M)	-	-		-	<b>134</b> (157)	<b>1</b> (1)	
	0.001	3.3e-05	23	14	<b>109</b> (146)	<b>1.23</b> (1.08)	
	0.004	2.1e-04		15	<b>87</b> (125)	<b>1.54</b> (1.26)	
	0.01	0.0011		16	<b>75</b> (114)	<b>1.79</b> (1.38)	
Mixer (2.5 M) No greedy alg. Object space	-	-		-	<b>297</b> (346)	<b>1</b> (1)	
	0.001	7.4e-04	49	28	<b>181</b> (254)	<b>1.64</b> (1.36)	
	0.004	0.0027		29	<b>164</b> (242)	<b>1.81</b> (1.43)	
	0.01	0.0056		29	<b>152</b> (230)	<b>1.95</b> (1.5)	
	0.004	0.016	49	23	<b>121</b> (193)	<b>2.45</b> (1.79)	
	0	75	-	<b>981</b> (1056)	<b>0.3</b> (0.33)		
HPG2016 (5.3 M)	-	-		-	<b>317</b> (388)	<b>1</b> (1)	
	0.001	5.4e-04	71	31	<b>217</b> (319)	<b>1.46</b> (1.21)	
	0.004	0.002		32	<b>191</b> (294)	<b>1.66</b> (1.32)	
	0.01	0.0035		33	<b>173</b> (280)	<b>1.83</b> (1.38)	
Radial Concentrations (10 M) Only z-Adapt	-	-		-	<b>598</b> (677)	<b>1</b> (1)	
	0.001	6.2e-04	79	49	<b>460</b> (588)	<b>1.3</b> (1.15)	
	0.004	0.0013		50	<b>340</b> (469)	<b>1.76</b> (1.44)	
	0.01	0.002		50	<b>290</b> (419)	<b>2.06</b> (1.62)	
	0.004	0.0014		59	<b>389</b> (527)	<b>1.54</b> (1.28)	

Figure 5.12 shows the detailed time analysis of the Mixer scene. Speed-up factors of about 2 were observed for all values of  $E_I$  within the first 100 frames. With increasing scene complexity (appearance of iso-surfaces and spreading of dye) the speed-up factors for  $E_I = 0.001$  drop to 1.15 and for  $E_I = 0.01$  to between 1.18 and 1.4. Towards the end of the sequence, the scene complexity decreases again (cf. Fig. 5.9, right) allowing for increasing speed-ups.

Apparently, the variation in the speed-up factors depends on the homogeneity of the radiance and transparency and the given error tolerance. Also, large surface to volume ratios impair speed-up factors as surface cells are excluded



**Figure 5.12:** Timings of the 1200 simulation frames (x-axis) of the Mixer scene. Timings are displayed using solid lines against the y-axis.

**Table 5.2:** Error statistics of the adaptive sampling for different scenes and tolerances  $E_I$ . ‘Err<sub>max</sub>’ gives the maximum single pixel error of all frames, ‘%Rays> $E_I$ ’ gives the maximum percentage of erroneous rays over the total number of cast rays for the respective frame and ‘#Frames> $E_I$ ’ gives the number of frames of the scene that exceeded the error tolerance.

Scene	$E_I$	$E_{relax}$	Err <sub>max</sub>	%Rays> $E_I$	#Frames> $E_I$
Flubber	*	1	< $E_I$	0	0
Checker Board	*	1	< $E_I$	0	0
Mixer	0.001	1	0.00106	0.0125 %	1/1200
	0.004, 0.01	1	< $E_I$	0	0
HPG 2016	0.001	1	0.00101	0.0004 %	1/1500
	0.004, 0.01	1	< $E_I$	0	0
Radial Concentrations	*	1	< $E_I$	0	0

from adaptive sampling. To show this effect, the Flubber scene was rendered with surface detection disabled (cf. row ‘No surface’ in Tab. 5.1) and achieved a speed-up of 2.21 for  $E_I = 0.004$ . Although this causes visible artifacts where super-pixels are rendered for cells that are not fully covered with particles, it indicates that there is potential for further speed-ups.

In order to demonstrate the benefits of combined lateral adaptivity and adaptivity in viewing direction, the Radial Concentration scene was rendered using only adaptivity in viewing direction with  $E_{relax} = 1$  and  $E_I = 0.004$  and achieved a speed-up of 1.28 (cf. row ‘Only z-Adapt’ in Tab. 5.1). The speed-up with lateral adaptivity using super-pixels was 1.44 at an even lower screen space

error.

Comparison to  
previous works

A *comparison to previous work* was done for the Mixer scene which was rendered using an object space particle access structure [OKK10] (cf. row ‘Object space’ in Tab. 5.1). A speed-up of about 3 was observed using only the view-aligned access structure.

A comparison to the adaptive ray casting of Orthmann [Ort14] was made in order to assess the benefits of the optimizations of Sec. 5.4. Although different scenes were used, they were of similar complexity and should serve as a fair basis for comparison. While the screen space error and the error tolerance differed at least by an order of magnitude in all scenes as shown Tab. 6.1 [Ort14]<sup>3</sup>, the optimizations of Sec. 5.4 gave more precise control over the screen space error and, thus, yielded higher speedups for lower values of  $E_I$  (see Tab. 5.1). Even for large tolerances of  $E_I = 0.2$ , the maximum speedup<sup>4</sup> was only between 0.94 and 1.04 and rendering was even slowed down in one scene (see Tab. 6.1 [Ort14]). In contrast, the relaxation in combination with super-pixels yielded speedups of at least 1.28 for  $E_I = 0.004$  (see Tab. 5.1).

## 5.7 Conclusions

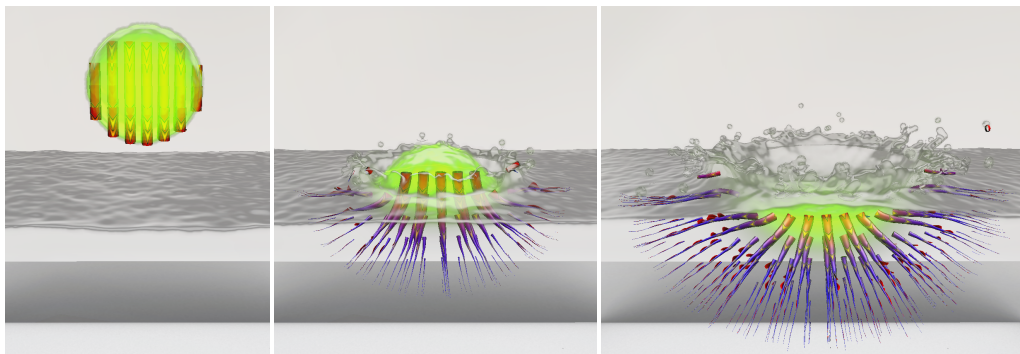
In this chapter an adaptive on-the-fly volume ray casting for unstructured particle data has been presented. The approach employs a sparse perspective, view-aligned grid as access structure for particles and does not require any precomputations. Inside each grid cell, sampling rates can locally be adapted both in viewing and lateral direction. The presented on-the-fly sampling error analysis for volume rendering of SPH-based quantity fields allows to precisely estimate screen space errors due to adaptive sampling. A greedy algorithm optimizes the adaptive sampling for each cell according to a user-defined error tolerance. The per-cell sampling information is used during ray casting to shift computational resources to salient regions of the fluid volume. The proposed algorithm leads to significant rendering speed-ups without sacrificing image quality.

<sup>3</sup>Table 6.1 appears on page 101 in the thesis of Orthmann which can be retrieved from the University Library Siegen under the following link: [http://dokumentix.uni-siegen.de/opus/volltexte/2015/911/pdf/Dissertation\\_Jens\\_Orthmann.pdf](http://dokumentix.uni-siegen.de/opus/volltexte/2015/911/pdf/Dissertation_Jens_Orthmann.pdf).

<sup>4</sup>The speedup is calculated as the ratio of non-adaptive rendering time over adaptive rendering time.



## Visualization of Advective-Diffusive Flows



**Figure 6.1:** A drop of green dye is dripped into water. Before impact, only advection plays a role (left). Directly after impact, the drop's velocity, i.e. advection, still dominates concentration transport (middle), but diffusion increasingly becomes the main mode of transport (right). Red and blue indicate advection and diffusion dominated flow, respectively.

*In this chapter a framework for unified visualization of advective and diffusive concentration fluxes is presented. These play a key role in many phenomena like, e.g. Marangoni convection and microscopic mixing. The main idea is to decompose fluxes into their concentration and velocity parts. Using this flux decomposition, advective-diffusive concentration transport can be conveyed using integral lines. In order to visualize superposed flux effects, a new graphical metaphor, the stream feather, is introduced which adds extensions to stream tubes pointing in the directions of deviating fluxes. The resulting unified visualization of macroscopic advection and microscopic diffusion allows for deeper insight into complex flow scenarios that cannot be achieved with current volume and surface rendering techniques alone.*

*The approach presented in this Chapter has been published in Computer Graphics Forum [HWK15] and presented at EuroVis 2015 in Cagliari, Italy.*

**U**nderstanding the behavior of concentration transport in fluid flows is a challenging task. There exists a wide array of scientific visualizations to aid in gaining insight into experimental and simulated flow data. The most

important approaches are volume rendering of concentration fields [EHK\*06], vector field visualization of advection by means of line integral convolution [CL93], by tracing integral lines, and also by flow based surfaces [MLP\*10]. However, so far mainly advective flows have been investigated while the complex flow of concentrations inside fluids, which additionally depends on diffusive flux, has not been considered in the context of integral line visualizations but only in direct volume rendering [KSW\*12] and visualization of topological features [SKE14] of dye-advection. Concentrations in the fluid have an impact on important physical quantities like surface tension, which, e.g., leads to the effect of Marangoni convection. Many real-world applications depend on the complex interplay between advection and diffusion although their respective contributions to an observed behavior are not always clear. The presented visualization of advective-diffusive transport aims at filling this gap, enabling researchers to identify and understand the driving forces in complex transport scenarios encountered in, e.g. microscopic mixing [KWFY99], and dynamic wetting [FAB\*11].

In this chapter a visualization framework for advection and diffusion is proposed that is based on tracing integral lines. The main idea is to provide insight into the intrinsic structure of the concentration transport consisting of both, the diffusive and the advective component. In order to combine both components to a unified flux of concentration, concepts for decomposing the diffusive flux into a velocity and a concentration part are proposed. The visualization then uses integral lines that provide means for disclosing the intrinsic diffusive and advective components of the combined flux. The main

**Contributions** contributions of the presented approach are:

- A generic framework to describe arbitrary types of fluxes that is based on the decomposition of fluxes into velocity and concentration.
- A decomposition of diffusive flux based on the concept of mean and maximum diffusion velocity allows for integral line visualization of diffusive transport.
- Vector field visualization of combined advection-diffusion processes introducing *stream feathers* to visualize diffusion and advection simultaneously.

The proposed visualization approach is applied to *Smoothed Particle Hydrodynamics (SPH)* simulations of incompressible fluids. The contribution in this context is

- a stable reconstruction of continuous advective and diffusive flux fields in SPH as required for stream feather visualization.

Note that the proposed decomposition of fluxes and thus the visualization approach for advective-diffusive flows can be applied to any kind of (simulation)

data as long as velocity and concentration data are available.

The remainder of the chapter is structured as follows: Sec. 6.1 discusses the relevant theory and related work. Sec. 6.2 gives a general overview of the visualization framework. Sec. 6.3 shows how to decompose fluxes to make them available for integral line visualization. The resulting fields can be rendered using the proposed novel stream feather metaphor introduced in Sec. 6.4. Details of the SPH-based simulation and visualization framework are described in Sec. 6.5. Results are presented and discussed in Sec. 6.6. Sec. 6.7 concludes the chapter.

## 6.1 Foundations and Prior Work

In this section, a brief overview of advective and diffusive flux (Sec. 6.1.1) and on respective visualization techniques (Sec. 6.1.2) is given. As the proposed generic advective-diffusive flux visualization will later be applied to SPH-based flow simulations, visualization techniques applied to SPH fluids are also discussed (Sec. 6.1.3).

### 6.1.1 Advective and Diffusive Flux

*Advective flux* carries concentration  $c(\mathbf{x})$  with the velocity field  $\vec{v}(\mathbf{x})$  through unit area per unit time at position  $\mathbf{x}$  as

$$\vec{\mathbf{j}}_a(\mathbf{x}) = c(\mathbf{x})\vec{v}(\mathbf{x}). \quad (6.1)$$

In the presence of concentration gradients, a net transport from areas of higher concentrations to areas of lower concentrations takes place. This *diffusive flux* is calculated according to Fick's law as

$$\vec{\mathbf{j}}_d(\mathbf{x}) = -D\nabla c(\mathbf{x}), \quad (6.2)$$

where  $D$  is the molecular diffusivity. The *total flux* of concentration through unit surface per unit time at position  $\mathbf{x}$

Total flux of concentration

$$\vec{\mathbf{j}}_t(\mathbf{x}) = \vec{\mathbf{j}}_d(\mathbf{x}) + \vec{\mathbf{j}}_a(\mathbf{x}) \quad (6.3)$$

is the sum of advective and diffusive fluxes and follows the direction of maximum transport [BSL07].

### 6.1.2 Visualization of Advective-Diffusive Flow

While general vector field visualization methods have been covered in Sec. 2.4, the focus here lies on advective-diffusive flows. LIC [CL93] has been extended by a model of non-linear diffusion which, however, is not part of the simulation data but is, for example, applied to segment the resulting flow fields [BPR01, DPR00]. Advective-diffusive flows have been visualized using surface renderings of clouds of concentration spreading. However, the diffusive part does not follow a gradient but just extends streamlines that follow advection to cone-shaped clouds [MS93]. Several approaches have been proposed for visualization of diffusion tensors that describe the behavior of anisotropic diffusion. Hyperstreamlines follow the direction of the major eigenvector of the diffusion tensor field [DH93] and have been extended to Tensorlines to increase the stability in isotropic regions where all eigenvalues are nearly identical [WKL99]. Other approaches have employed tensor glyphs [KW06] and tensor volumes [KWH00] to visualize diffusion tensors. None of these approaches, however, visualized actual transport of concentrations.

Advective-diffusive flow has been visualized using direct volumetric visualizations of dye-advection [KSW\*12]. Topological features of advective-diffusive flows have been examined, however, the advection-diffusion equation has only been solved in form of a secondary simulation step on top of a purely advective flow [SKE14].

The proposed visualization approach is based on the geometric construction of integral lines from advection-diffusion simulation data. To the best of the author's knowledge, neither the effects of diffusion nor the combined advective-diffusive transport have been considered in the context of integral line based visualization, so far.

### 6.1.3 Visualization of SPH Fluid Simulations

In the context of SPH-based fluid simulations, concentrations can be visualized using volume rendering as presented in Chapter 5 whereas surface rendering reveals the fluid's geometric shape [AIAT12]. SPH data can always be visualized by sampling field quantities on a grid and applying standard techniques. However, resampling can introduce artifacts in undersampled regions and increases computational complexity in case of unnecessary oversampling [SFBP09]. Pure advection has been visualized by directly rendering particle trajectories in combination with space-time hierarchical clustering to reduce visual clutter [FW12]. This, however, only allows to render path lines at the resolution of particles. For higher spatial and temporal resolution, interpolation between particle posi-

tions of subsequent time steps can be employed [COJ15]. Vortex core lines have been visualized directly from SPH-data using Hermite splines to interpolate between particle positions in time [SFBP09].

As the time dependent behavior of fluids and the distinct roles of advection and diffusion to concentration transport cannot be captured by current visualization techniques, we propose an integral line based approach to simultaneously visualize advective and diffusive concentration transport. In order to achieve an interactive visualization that does not rely on any preprocessing or resampling, the flux-related quantities are directly computed within the SPH simulation. In this context, it is not sufficient to trace SPH particles to deduce concentration transport comprising diffusion and advection. As diffusion is a microscopic phenomenon modeled as concentration exchange between SPH particles, concentration transport has to be traced along arbitrary, i.e. inter-particle spacial positions.

## 6.2 Overview

The proposed visualization framework for advection and diffusion using integral lines requires a velocity field and a scalar concentration field which both may be unsteady. Firstly, Sec. 6.3 discusses how to compute advective and diffusive fluxes directly from velocity and concentration data without any preprocessing. One main challenge here is the requirement to express a flux  $\vec{j}$  as decomposition of velocity  $\vec{v}$  and concentration  $c$ , i.e.

$$\vec{j} = c \cdot \vec{v} \quad (6.4)$$

Flux decomposition

in order to trace integral lines of fluxes. The advective flux is already given in this form. For diffusion, however, this kind of decomposition is not unique. Two different decomposition schemes of diffusive flux are proposed, according to the *mean velocity of molecules* [Ein05] and to the *maximum velocity*, as for instance applied in environmental sciences in the context of the spreading of toxic waste [Sch96], which are common interpretations to diffusive processes (see Sec. 6.3.1).

Using the flux decomposition in Eq. (6.4) and the interpretations for the diffusive flux, the unified flux is calculated consisting of the advective and the diffusive component. The mean diffusive velocity, which follows the *direction of maximum transport*, yields the so-called total flux  $\vec{j}_t$ , and the maximum diffusive velocity, which follows the direction of *fastest advancing concentration front*, yields the maximum velocity flux (see Sec. 6.3.2).

Based on the velocity components for the advective, diffusive and unified fluxes the tracing of integral lines over time is demonstrated (see Sec. 6.4).

Integral lines are visualized geometrically using stream tubes the thickness of which can be varied, e.g. according to the transported concentration in order to convey the actual magnitude of flux. As the goal is to visualize the relation of all three, potentially divergent fluxes, the geometric primitive of the stream tube is extended by introducing *stream feathers*. Stream feathers are appendages of the integral line indicating the directions and magnitude of deviating fluxes. In this thesis, only streamlines are visualized. However, in general, the presented approach is able to visualize any kind of integral lines.

## 6.3 A Framework for Tracing Advective-Diffusive Fluxes

The core concept of the proposed visualization approach is an extension of the concept of integral lines, as introduced in Sec. 2.4.3, in order to achieve insight into multi-component, i.e. advective-diffusive fluxes. Standard integral lines are *streamlines*, *pathlines* and *streaklines*. Even though the visualization concept directly applies to all types of integral lines, the focus lies on streamlines in this chapter.

The position  $\mathbf{x}^{\text{stream}}(\tau, \mathbf{x}_0, \tau_0)$  of samples moving along a streamline seeded at position  $\mathbf{x}_0$  at time  $\tau_0$  is determined by time integration of the velocity field  $\vec{\mathbf{v}}(\mathbf{x}, \tau_0)$  (see Eq. (2.20) in Sec. 2.4.3). As the advective flux  $\vec{\mathbf{j}}_a$  already is in a separated form, i.e.  $\vec{\mathbf{j}}_a = c \cdot \vec{\mathbf{v}}_a$ , it has a velocity component that can directly be used to trace integral lines over time [JH04].

Ambiguous  
decomposition of  
diffusion

The same does not hold for diffusive flux as there is no unique definition of a velocity which could be used to determine streamlines. In order to make diffusive flux  $\vec{\mathbf{j}}_d$  and total flux  $\vec{\mathbf{j}}_t$  available for integral line visualizations, they have to be expressed in a decomposed form just as the advective flux, i.e. as  $\vec{\mathbf{j}}_d = c_d \cdot \vec{\mathbf{v}}_d$  and  $\vec{\mathbf{j}}_t = c_t \cdot \vec{\mathbf{v}}_t$ .

Note that although the flux is unchanged by the way it is decomposed, the resulting integral lines may depend on the decomposition, i.e. on the velocity part.

In the following, the decomposition of the diffusive flux is discussed in Sec. 6.3.1. Afterwards, Sec. 6.3.2 describes how to derive a similar form of the unified flux.

### 6.3.1 Decomposition of Diffusive Flux

One major challenge with diffusion is its random nature. As diffusion arises from Brownian molecular motion, there is no unique way to define a diffusion

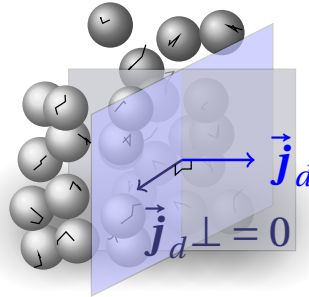
velocity for the decomposition of diffusive flux [Cus09].

Assuming that solute molecules move at a mean velocity  $v_d^{\text{mol}}$  in solution and randomly change their direction due to collisions with neighboring molecules and assuming further that the mean free path until a collision with neighboring molecules occurs is  $\Delta x$ , then the position of molecules after some time  $\tau$  can only be described in terms of density distributions. Based on this consideration, the *molecular diffusivity* is defined as (see also Eq. (6.2))

$$D = v_d^{\text{mol}} \Delta x. \quad (6.5)$$

Even though movement occurs as a random process, transport from areas of higher concentrations to areas of lower concentrations is more likely than in the opposite direction causing a net diffusive flux  $\vec{j}_d$  in the presence of concentration gradients as depicted in Fig. 6.2.

There are two major interpretations of diffusion related velocity. The *mean velocity* considers the average speed of diffusing molecules, whereas the *maximum velocity* seeks to capture the advancing front of diffusion transport.



**Figure 6.2:** Diffusion follows the random movement of molecules, here depicted as the black molecule trajectories. The high concentration left of the blue plane leads to a net flux to the right. However, there is no net flux through the gray plane as the concentration on either side is the same.

**Mean Velocity:** According to Einstein, the mean velocity depends on the local concentration as well as the concentration gradient [Ein05]. At low Reynolds numbers, which is the case for small solute molecules in solvents like water, the mean molecular velocity of diffusion is proportional to the force due to the gradient of the chemical potential  $\mu$  as

$$\vec{v}_d^{\text{mean}} = -\sigma \nabla \mu = -\sigma \frac{k_B T}{c} \nabla c = -\frac{D}{c} \nabla c = \frac{\vec{j}_d}{c}, \quad (6.6)$$

where  $\sigma$  is a temperature-dependent friction coefficient.  $k_B$  is the Boltzmann constant and  $T$  the temperature which is also considered constant, here. As

Brownian molecular motion

Decomposition of mean velocity diffusion

the force due to the chemical potential gradient acts the same on all molecules, all molecules are assumed to move at mean velocity in Einstein's model, i.e. the full concentration  $c$  is diffused resulting in  $c_d^{\text{mean}} = c$ . Nevertheless, in reality there will always be a fraction of molecules that move faster than the mean.

Einstein's model relates the mean diffusion velocity inversely proportional to the concentration, see Eq. (6.6), thus, the practical problem arises that for  $c \rightarrow 0$  velocity diverges. Therefore, the mean velocity is bounded by the user-defined maximum velocity  $v_d^{\text{max}} > 0$ . The value  $v_d^{\text{max}}$  will also be used for the maximum diffusion velocity.

**Maximum Velocity:** The maximum velocity obviously depends on the diffusivity, see Eq. (6.5). However, even if the molecular velocity  $v_d^{\text{mol}}$  is known, it cannot be taken as maximum diffusion velocity as the free length until molecular collision  $\Delta x$  needs to be taken into account, i.e. no molecule travels without collision. Practically, this length can hardly be determined.

Thus, the user can control the magnitude of diffusion velocity  $v_d^{\text{max}}$ , which is the same already used to clamp the mean velocity in case of low concentration values. Assuming a fraction of molecules moves always at this speed in the flux direction, a corresponding concentration is deduced as

$$c_d^{\text{max}} = \frac{\|\vec{\mathbf{j}}_d\|}{v_d^{\text{max}}}, \quad \vec{\mathbf{v}}_d^{\text{max}} = \frac{\vec{\mathbf{j}}_d}{c_d^{\text{max}}} = \frac{-D\nabla c}{c_d^{\text{max}}}. \quad (6.7)$$

Comparing the mean and the maximum velocity interpretation for the diffusive flux, the main difference is that the mean velocity sets the concentration to the maximum value, i.e. to the total concentration, whereas the maximum velocity approach fixates the velocity. For the diffusive flux the choice of the maximum velocity  $v_d^{\text{max}}$  affects the speed a sample travels along the integral line, the line itself does not change, except for numerical integration errors. The user controlled maximum velocity does actually enhance the evaluation, as for a fixed integration time the length and thickness of the diffusion streamline can be adapted. Figs. 6.8(c) and 6.8(d) show flow fields with the diffusive flux using Einstein's mean velocity and maximal diffusive velocity, respectively. However, considering the unified maximum velocity flux as defined in Sec. 6.3.2, the choice of  $v_d^{\text{max}}$  also influences the direction of maximum velocity; see also Fig. 6.3.

Decomposition of  
maximum velocity  
diffusion

### 6.3.2 Unified Model of Advective-Diffusive Flux

As the goal is to visualize advection and diffusion in a unified approach, the *unified flux* has to be decomposed in the same way as the diffusive flux. Simi-



larly to the mean and maximum diffusion velocities decompositions for the unified fluxes for both interpretations of diffusion are derived.

**Unified Flux of Mean Velocity / Total Flux:** In the case of mean diffusive velocity, the whole concentration at a point in space  $c_t = c_a = c_d^{\text{mean}}$  is transported at the same velocity which is just the superposition of the advection and diffusion velocities. The resulting *unified mean velocity flux* or *total flux* can thus directly be decomposed as

Decomposition of unified fluxes

$$\vec{\mathbf{j}}_t = c_t \vec{\mathbf{v}}_t = c_t (\vec{\mathbf{v}}_a + \vec{\mathbf{v}}_d^{\text{mean}}) \quad (6.8)$$

and follows the direction of maximum transport of concentration.

**Unified Flux of Maximum Velocity:** In the case of a constant maximum diffusion velocity, the direction of maximum transport of concentration is actually not of interest but rather the direction of preferred concentration spreading. Thus, instead of visualizing the total flux as  $\vec{\mathbf{j}}_t = \vec{\mathbf{j}}_a + \vec{\mathbf{j}}_d$ , the direction of flux with maximum velocity is followed as

$$\vec{\mathbf{v}}_m = \vec{\mathbf{v}}_a + \vec{\mathbf{v}}_d^{\text{max}}. \quad (6.9)$$

To find the decomposition of constant velocity flux

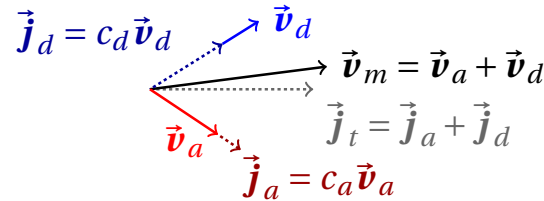
$$\vec{\mathbf{j}}_m = c_m \vec{\mathbf{v}}_m, \quad (6.10)$$

the amount of transported concentration  $c_m$  in direction  $\vec{\mathbf{v}}_m$  still has to be determined. As  $c_m$  should be physically plausible,  $c_m \geq 0$  is required as well as compliance with the total flux in Einstein's mean velocity consideration, i.e. for  $c_t = c_a = c_d^{\text{mean}} = c$ ,  $c_m = c$  must hold, too. Considering Fig. 6.2, it is obvious that the diffusive flux changes its magnitude depending on the considered direction of flow through a local plane. If a linear model is applied, the effective concentration flow through a unit area with normal  $\hat{\mathbf{n}}$  with respect to the flux direction  $\hat{\mathbf{v}}$  is proportional to  $(\hat{\mathbf{n}} \cdot \hat{\mathbf{v}})$  [BSL07]. All of the required properties are fulfilled by defining

$$c_m = \frac{\max(0, \hat{\mathbf{v}}_a \cdot \hat{\mathbf{v}}_m) c_a \|\vec{\mathbf{v}}_a\| + \max(0, \hat{\mathbf{v}}_d \cdot \hat{\mathbf{v}}_m) c_d^{\text{max}} \|\vec{\mathbf{v}}_d\|}{\max(0, \hat{\mathbf{v}}_a \cdot \hat{\mathbf{v}}_m) \|\vec{\mathbf{v}}_a\| + \max(0, \hat{\mathbf{v}}_d \cdot \hat{\mathbf{v}}_m) \|\vec{\mathbf{v}}_d\|}. \quad (6.11)$$

The clamping of the inner product in Eq. (6.11) guarantees positive contributions to concentration transport. Fig. 6.3 shows the different directions of unified mean and maximum velocity fluxes for  $c_a \neq c_d$ .

This decomposition allows to trace integral lines of advective, diffusive, and unified fluxes in the same way.



**Figure 6.3:** Construction of the direction of unified maximum velocity flux. If  $c_d \neq c_a$ , the directions of unified mean velocity flux  $\vec{j}_a + \vec{j}_d$  and maximum velocity  $\vec{v}_a + \vec{v}_d$  are different.

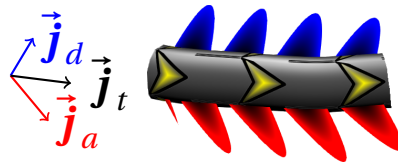
## 6.4

## Visualization of Fluxes Using Stream Feathers

The final visualization uses the velocity and concentration values deduced from the advective, diffusive and unified fluxes. Integral lines can now be calculated which follow either of the fluxes using numerical integration. The resulting polylines are visualized using *stream tubes* which are constructed in an OpenGL geometry shader.

In case advection and diffusion carry concentrations in different directions, the stream tube is extended to a novel visualization metaphor, the *stream feathers*. Stream feathers provide a convenient way to visualize additional, diverging fluxes in a unified manner. As the unified fluxes are a combination of diffusive and advective flux,  $\vec{j}_t$  ( $\vec{j}_m$ ),  $\vec{j}_a$ , and,  $\vec{j}_d$  are coplanar. If  $\vec{j}_d$  or  $\vec{j}_a$

Simultaneous visualization of diverging fluxes



**Figure 6.4:** Stream feathers are able to capture different flux components in an intuitive combined view. The barbs of the feathers point in the direction of fluxes strongly deviating from the flux visualized as stream tube.

strongly deviate from  $\vec{j}_t$  or  $\vec{j}_m$  in case of the mean or maximum diffusion velocity, respectively, small planar appendages like barbs of a feather are drawn to the stream tubes pointing in the direction of the deviating fluxes as shown in Fig. 6.4. This way, attention can be steered to areas of strongly divergent advective and diffusive fluxes.

Illustration of flux direction

In order to show the flux direction, stream tubes are textured with arrows. The size and spacing of arrows is proportional to the velocity of transport. The stream tube thickness is scaled according to the transported concentration.

By mapping velocity and concentration to different visual qualities, not the magnitude but also the ratio of velocity and concentration of the flux can be conveyed. To prevent visual clutter, feather barbs are only displayed if the angle between fluxes exceeds a user-defined threshold  $\epsilon > 1 - \|\hat{\mathbf{j}}_i \cdot \hat{\mathbf{j}}_d\|$ . The length of barbs is scaled with their respective flux magnitude. Additionally, stream tubes can be colored according to the flux that most contributes to the unified flux so that the driving transport mechanism can easily be identified as shown in Fig. 6.1. A fix color pattern is applied to highlight advective (red) and diffusive (blue) contributions.

Presentation of flux magnitude

## 6.5 Advective-Diffusive Fluxes in SPH

The proposed advective-diffusive flux visualization framework was applied to SPH-based simulation. While the simulation was carried out using the formalisms outlined in Chapter 3, the reconstruction of fluxes from simulated data has to be derived for arbitrary points in space. The advective flux is reconstructed from simulated data at arbitrary positions as

$$\vec{\mathbf{j}}_a(\mathbf{x}) = \left( \sum_j c_j V_j \hat{W}_i(\mathbf{x}) \right) \left( \sum_j \vec{v}_j V_j \hat{W}_i(\mathbf{x}) \right). \quad (6.12)$$

In order not to underestimate field quantities in areas of neighborhood deficiency, corrected SPH (CSPH) interpolation is applied (see Eq. (3.12)).

The diffusive flux first is evaluated at particle positions as

$$\vec{\mathbf{j}}_d(\mathbf{x}_i) = -D \sum_j (c_i - c_j) \frac{V_i + V_j}{2} \nabla W_j(\mathbf{x}_i) \quad (6.13)$$

and is then approximated at arbitrary positions using corrected SPH

$$\vec{\mathbf{j}}_d(\mathbf{x}) = \sum_j \vec{\mathbf{j}}_d(\mathbf{x}_j) V_j \hat{W}_j(\mathbf{x}). \quad (6.14)$$

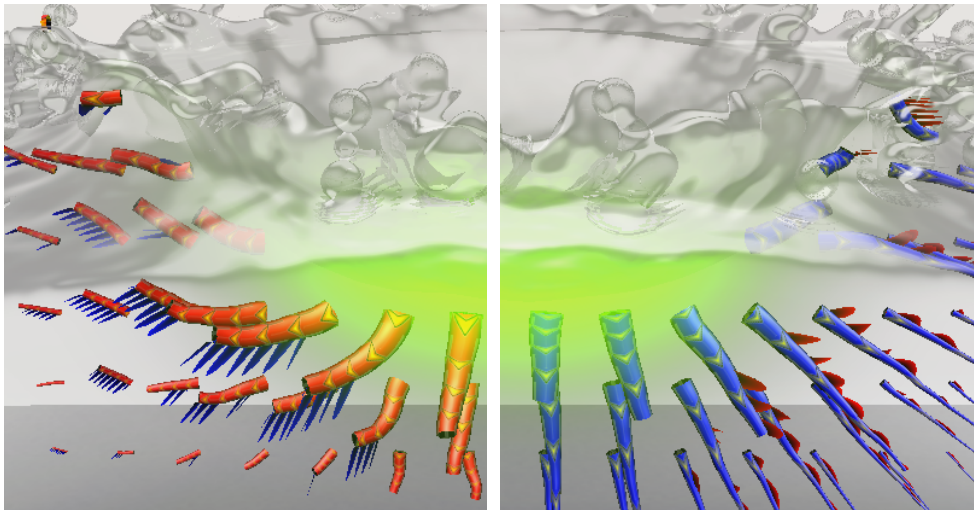
Note that while the simulation uses Fick's second law of diffusion (see Sec. 2.1.3) in order to find the time rate of change of concentrations, here, Fick's first law of diffusion has to be calculated as it yields the diffusive flux [ALS09].

## 6.6 Results and Discussion

In order to demonstrate the benefits of the proposed novel advection-diffusion visualization, several example scenes were set up.

Advection and diffusion in similar directions

The first scene simulates dripping a solute dye into a tank of solvent (see Figs. 6.1, 6.5, 6.6). At first, diffusion and advection of dye work in the same direction. After impact, advection slows down due to water pressure and the water begins to bounce back around the site of impact. Fig. 6.5 shows the advective and diffusive fluxes shortly after impact, corresponding to the unified mean velocity flux on the right hand side of Fig. 6.1. Diffusion transports concentration perfectly radially away from the site of impact. Stream feathers nicely accentuate areas of divergent advective and diffusive fluxes which would have not been revealed otherwise.



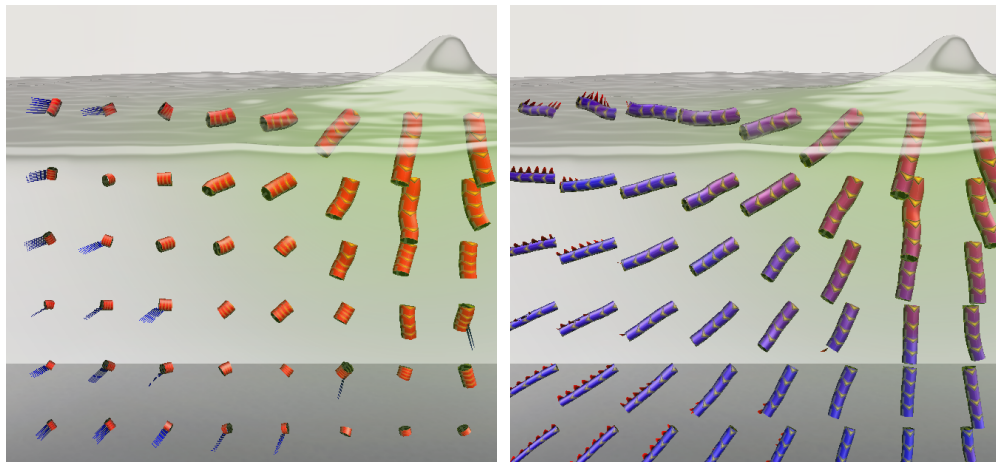
(a) Advective flux with diffusion feathers (b) Mean velocity diffusive flux with advection feathers

**Figure 6.5:** Stream feather visualization of advective and diffusive fluxes corresponding to the unified mean velocity flux in Fig. 6.1, right.

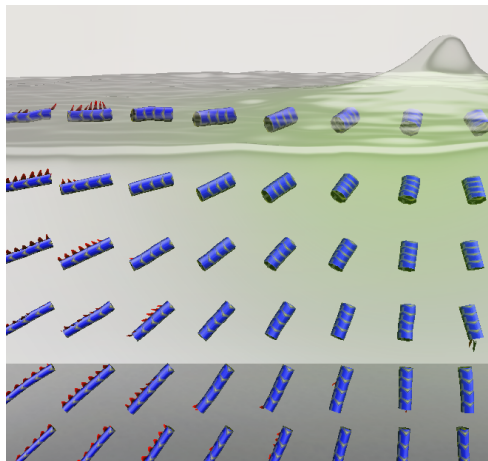
Fig. 6.6 shows the situation about 6 s after impact when the water is still bouncing. The advective movement clearly slows down and diffusion dominates the total transport. At this point in time, advection and diffusion work in the same direction around the site of impact, hence, stream feathers are not visible in that region. The coloring according to the dominating flux is still able to convey the respective contributions of advection and diffusion to the total transport.

Opposite advection and diffusion

In the second scene pure solvent is dripped into a tank of solute dye as an example for counteracting advection and diffusion. Fig. 6.7 shows the situation shortly after impact of the solvent drop. At that point in time, diffusion transports dye into the drop (Fig. 6.7(b)) at nearly the same velocity as the drop's downward advective motion (Fig. 6.7(a)) so that the unified mean velocity flux



(a) Advective flux with diffusion feathers (b) Unified mean velocity flux with advection and diffusion feathers



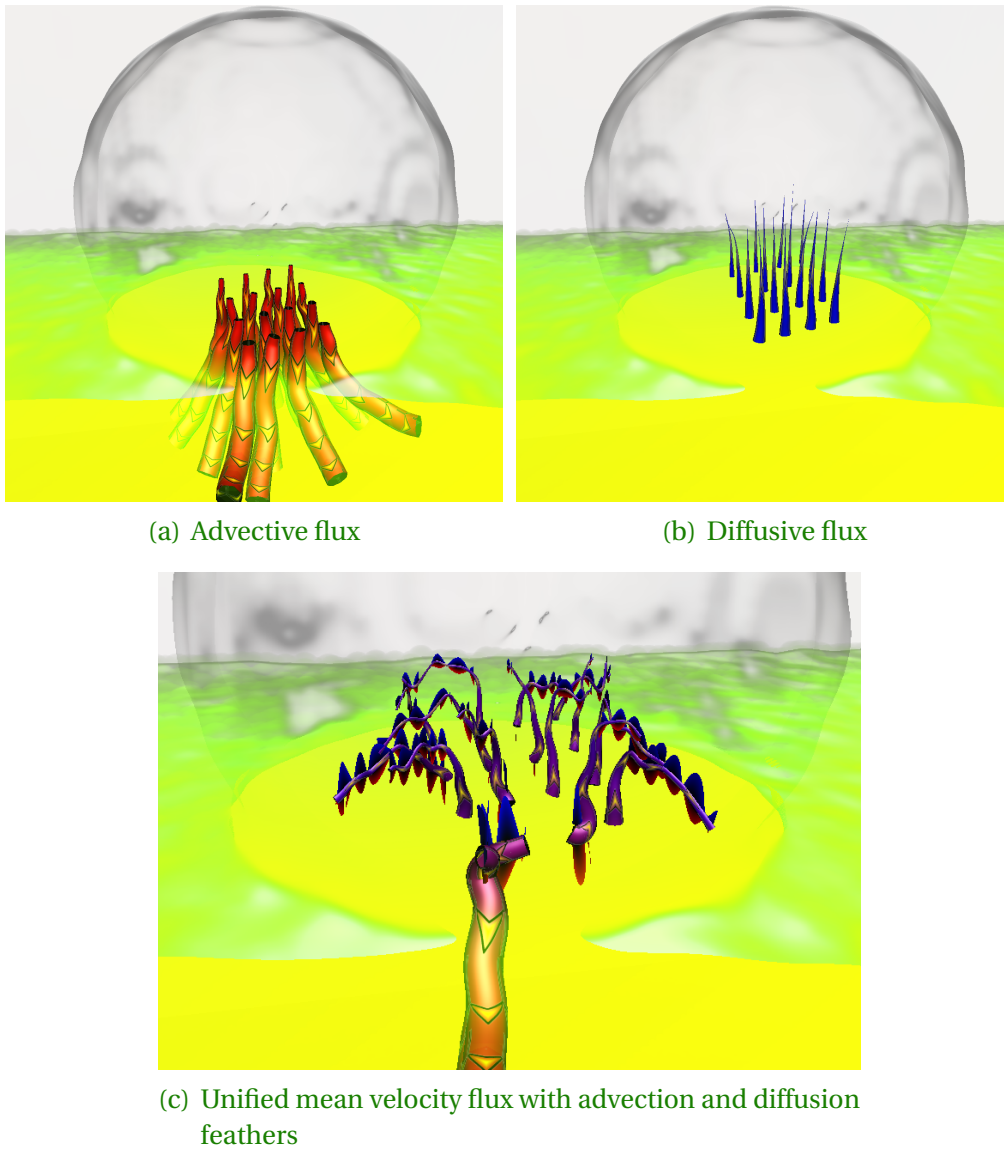
(c) Diffusive flux with advection feathers

**Figure 6.6:** Advective, unified mean velocity and diffusive fluxes after impact of dye in solvent (see Fig. 6.1). Diffusion radially transports concentration away from the site of impact and dominates the flow farther away from the impact site. Advection due to bouncing water dominates the flow near the impact site.

nearly gets perpendicular to both the advective and diffusive fluxes as shown in Fig. 6.7(c). The stream feathers and the colored stream tubes are able to intuitively convey this situation of opposite fluxes.

The third scene shown in Fig. 6.8 demonstrates the utility of the visualization in a real-world application. The mixing of solute dye and solvent in a t-sensor [KWFY99] was simulated. In the t-sensor the dye and solvent streams on the left are accelerated by pressure. The two streams meet at the junction of the t-sensor and merge to one stream which can be analyzed. In the lower left

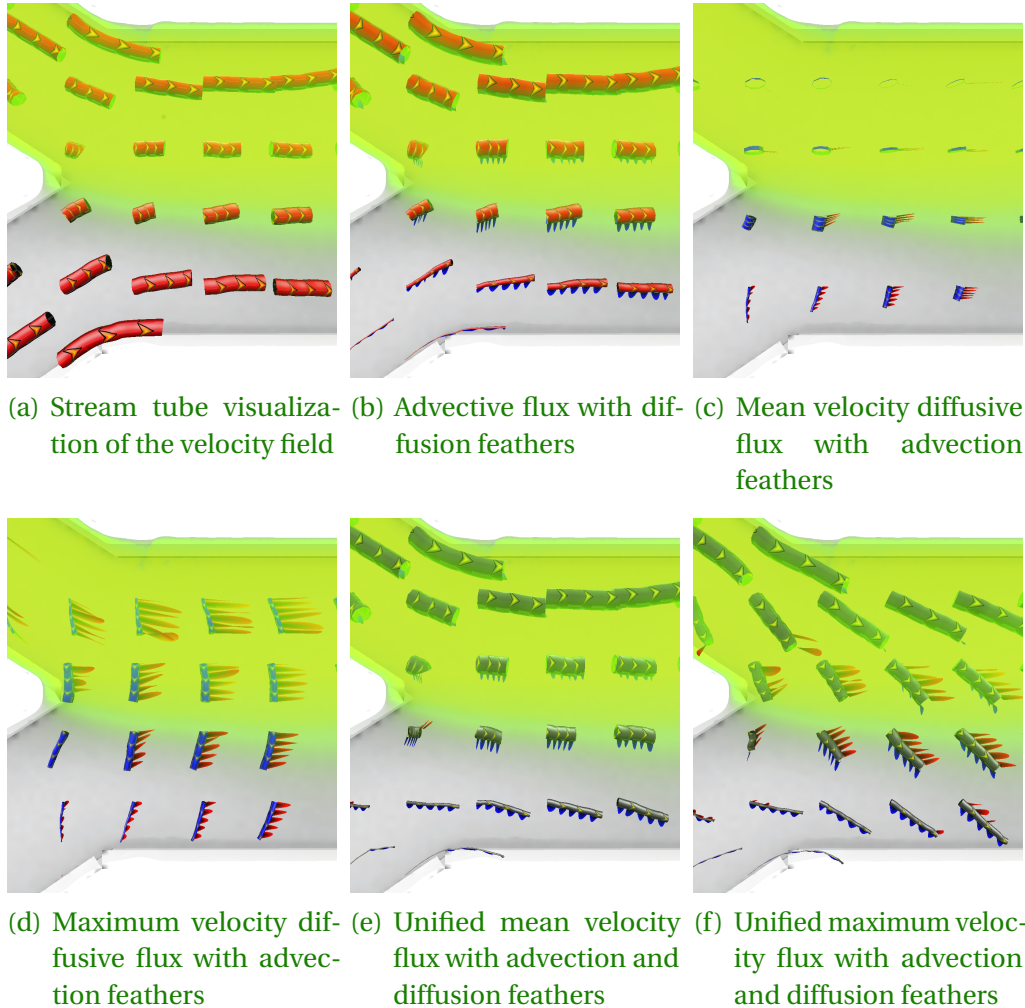
Complex superposed fluxes



**Figure 6.7:** Advective, diffusive and unified mean velocity fluxes at impact of a solvent drop in a tank of dye. Advection and diffusion work in opposite direction.

of the t-sensor, back-diffusion of concentration in the opposite direction of the advection of the solvent stream takes place.

A standard stream tube visualization of the velocity field (Fig. 6.8(a)) can not convey the magnitude of advective flux. In contrast, the presented visualization of advection (Fig. 6.8(b)) intuitively shows the magnitude of transport by scaling the tube thickness according to the transported concentration while



**Figure 6.8:** Advective, diffusive and unified fluxes of the flow in a t-sensor. Note how advective and diffusive fluxes transport concentration in nearly perpendicular directions.

the velocity is captured by the arrow size and spacing. The additional stream feathers show the diverging direction of diffusive flux. The visualization of maximum velocity diffusion (Fig. 6.8(d)) can give insight into diffusive transport in regions in which the Einstein model smoothes velocities (Fig. 6.8(c)) causing stream tubes to degenerate. In both visualizations of diffusion, stream feathers intuitively reveal the diverging direction of advection. The unified maximum velocity flux (Fig. 6.8(f)) more clearly reveals the back-diffusion that takes place in the lower left stream compared to the visualization of unified mean velocity flux (Fig. 6.8(e)). The stream feathers are scaled according to the magnitude of their respective fluxes and nicely capture the nearly perpendicular directions

of advection and diffusion.

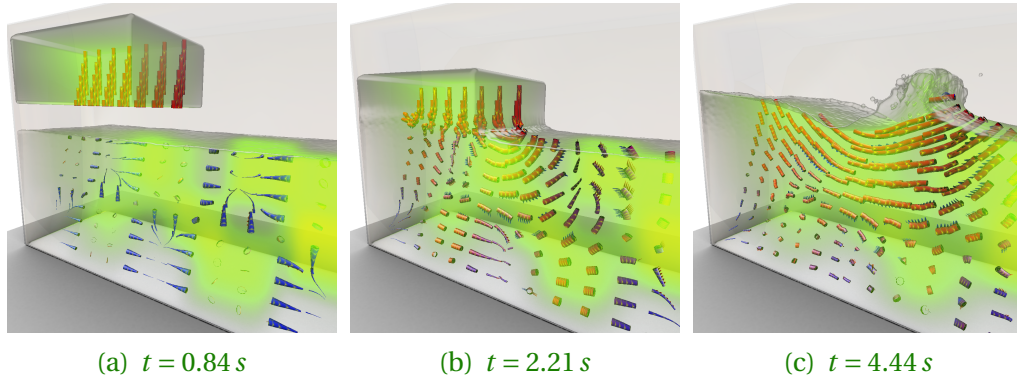
Compared to simple glyph-based renderings the stream tubes do not only encode magnitude and direction of flux but can reveal the ratio of velocity and concentration of transport by mapping these entities to stream tube thickness and arrow spacing, respectively. Important additional information is added by simultaneously showing deviating flux directions as feathers. A drawback of the approach, however, is the fact that the length of stream feathers, i.e. the flux magnitude, and the spacing between arrows, i.e. the velocity, cannot directly be compared.

The last scene is an artistic setup of a tank containing a 3D checker board pattern of different dye concentrations in solvent. From above, a quad of solute dye drops into the tank. Thus, the advection follows a classic dam break scenario. The checker board pattern of concentrations in the fluid bulk causes a distinct pattern of diffusion. Fig. 6.9 shows the development of the scenario over time for three different time steps. At first (Fig. 6.9(a)), diffusion and advection take place in separated areas: Diffusion is restricted to the fluid bulk and advection to the falling fluid quad. As soon as the fluid quad hits the tank (Fig. 6.9(b)), fluid in the tank gets displaced. Hence, the diffusive flux in the tank is superposed by a strong advective flux. The regions of strongly diverging fluxes at the edges of the checker board pattern are again emphasized by the stream feather rendering. The red coloring of the stream tubes clearly indicates that advection is the driving force in the upper half of the fluid. In the lower half, however, diffusion still is dominant. The movement of the quad continues to displace fluid and effectively creates a wave that travels to the right (Fig. 6.9(c)). The stream feathers and the coloring still clearly indicate that diffusion plays an important role for total transport and should not be neglected in visualization.

#### Visualization performance

All integral lines are directly computed from raw simulation data using a CUDA implementation. Time integration of streamlines has been realized using a fourth order Runge-Kutta scheme with fixed time step for a fixed duration of time. Streamlines have been generated from a planar grid of seed points that can be interactively controlled by the user. Calculations have been carried out on an Intel Core i7 930 at 2.8 GHz with 24 GiB RAM and on an NVIDIA GTX Titan with 6 GiB of VRAM. Table 6.1 gives timings of both the integral line calculations and the renderings of stream feathers. The timings denoted by 'Drop' in the table apply to both drop scenarios. The timings clearly show that in all scenes interactive frame rates were achieved both for the generation of streamlines and the rendering of stream feathers allowing for a direct exploration of simulation data. As the approach operates directly on raw data, the visualization can already be used during simulation. The computation time for streamlines is mainly dominated by the amount of samples along a





**Figure 6.9:** Unified mean velocity flux of the checker board scene over three time steps. The quad of solute dye (Fig. 6.9(a)) moves downward and hits the surface of the tank (Fig. 6.9(b)) causing a superposition of the initially separated advective and diffusive fluxes. The impact causes a wave to form (Fig. 6.9(c)) that travels to the right causing a strong advective flux.

**Table 6.1:** GPU timings of streamline integration and stream feather rendering for the test scenarios. The first column names the scenario and gives the number of simulated particles (in Mio.). Res. is the seeder plane resolution. The duration (integration time) of the streamline and the step size for the Runge-Kutta integration are given in Dur. and Step, respectively. Int. denotes the calculation time for the streamlines and Vis. is the frame rate of rendering stream feathers.

Scene (#Mio. ptcls)	Res.	Dur. (s)	Step (ms)	Int. (ms)	Vis. (fps)
T-Sensor (0.58)	$64 \times 64$	0.75	2.5	270	32
Drop (1.16)	$4 \times 4$	2	5	240	60
	$64 \times 64$	1		145	32
Checker Board (1.34)	$32 \times 32$	1	7.5	171	60
		2		300	60
	$64 \times 64$	1		171	46
		2		305	23

streamline and not by the number of streamlines which is due to the parallel CUDA implementation.

## 6.7 Conclusions

This chapter presented a framework for simultaneous visualization of advective, diffusive and unified fluxes based on integral lines. The main goal is to provide insight into the intrinsic structure of the concentration transport and the relation between the diffusive and the advective components. As integral lines require a velocity field, the diffusive and the unified flux are decomposed into a velocity and a concentration part based on the concept of mean and maximum diffusion velocity, yielding the unified mean and maximum velocity fluxes, respectively. Using the new metaphor of stream feathers, combined advection-diffusion processes can simultaneously be visualized.

The proposed decomposition of fluxes and the visualization for advective-diffusive flows can be applied to any kind of (simulation) data which provides velocity and concentration data. It was applied to SPH-based flow simulations and methods for a stable reconstruction of continuous advective and diffusive flux fields in SPH were presented.

**Benefits:** Evaluating the new visualization approach shows that it nicely reveals situations with divergent advective and diffusive fluxes. Both approaches, the unified mean and maximum velocity fluxes, give clear hints to the intrinsic nature of the superposed concentration transport mechanisms. In cases of high concentrations the Einstein model strongly decreases velocities. Here, the unified maximum velocity flux more clearly reveals effects like back-diffusion. Thus, both approaches complement each other in their expressiveness.

**Limitations:** By tracing either mean or maximum velocity diffusive fluxes, the random nature of diffusion is not fully reflected, as it statistically spreads concentrations uniformly. In areas of high concentration and small diffusion gradients, stream tubes of mean diffusive flux can degenerate. In this case using the maximum velocity diffusion model can still reveal all relevant information of diffusion.

## Conclusions

*This chapter summarizes the contributions and concludes the thesis. A brief discussion of limitations also directly opens up possible directions for future work.*

SPH-based fluid transport poses highly exciting research topics. In this thesis new methods for simulating and rendering SPH-based fluids have been presented. These include the physically-based simulation of phase changes between the liquid and gaseous states, adaptive volume rendering and vector field visualization of advective-diffusive flows. Although the contributions leave room for improvement, they also fill major gaps of current research. Especially surface dynamics and the visual presentation of transport phenomena have a large potential. Hopefully, this work will find applications and inspire further research.

### 7.1 Summary

Although phase-changes between solid and liquid states have been discussed in many works, this work is the first to realize evaporation and condensation in SPH-based fluids. The air phase is therefore simulated on a coarse Eulerian grid into which mass evaporates. Evaporation is realized by adaptively scaling particles according to the amount of transferred mass. As condensation takes place on surfaces in very fine spatial detail, textures are employed as an intermediate medium to condense liquid mass into and to generate particles from. Using a consistent SPH surface model, mass and energy preserving phase changes and heat transport have been realized between liquid and air phase.

Grid and particle coupling

Conservative phase changes

Interactions of liquids and rigid surfaces at small scales require a high spatial resolution to yield convincing results. Using textures as an intermediate medium allows for straight-forward data-parallel simulation and direct rendering of wetted surfaces at sub-particle resolution. By taking rigid boundaries into account, the presented implicit surface definition allows even single particles to be rendered with dynamic contact angles yielding high visual details at

Rendering at sub-particle detail

moderate particle counts.

Efficient volume rendering

Rendering of particle-based fluids is often restricted to surfaces, although volume rendering enhances the visual impression of fluids and is mandatory in order to convey concentration transport. A performance-critical aspect of volume rendering is the number of sampling operations to consider. Previously, particle data sets have been either re-sampled onto grids and rendered by view-aligned slicing using rasterization hardware or have been directly rendered through ray casting using object space particle access structures. While object space structures incur a complex ray traversal logic, rasterization prohibits adaptive sampling and often leads to a memory bottleneck. Employing a view-aligned particle access data structure, in contrast, allows for an on-the-fly data-parallel thread-coherent traversal of rays and locally adaptive sampling. The introduced screen space error analysis applies hierarchical interval arithmetic to the volume rendering equation in order to establish local sampling rates while enforcing a user-defined error tolerance. Adaptive ray casting avoids unnecessary sampling operations and is able to substantially speed up rendering without sacrificing image quality.

Adaptive sampling

The visual presentation of fluid flow can naturally be captured by characteristic lines. Fluid transport, however, follows a superposition of advection and diffusion and thus eludes direct vector field visualization. The presented decomposition of fluxes is able to derive separate velocity and concentration components for the superposed, advective, and diffusive fluxes making them available to visualization. By introducing the novel metaphor of stream feathers, superposed concentration transport and its components can intuitively be captured. The presented visualization works on raw simulation data and allows to interactively inspect concentration transport.

Decomposition of fluxes

Stream feathers

Data-parallel implementation

Without being restricted to specific hardware features, the contributions of this thesis were all designed such that they can be efficiently implemented in a data-parallel fashion. Thus, interactive simulation and on-the-fly rendering and visualization as well as interactive steering of parameters can easily be realized on GPUs.

## 7.2

## Future Work

There is a wide array of directions for further research based on the contributions presented in this work.

Consistent unified simulation

Instead of relying on a coarse grid to simulate the air phase, recent developments in adaptive SPH fluid simulation [WHK17] might be adopted in order to simulate the air phase in a consistent manner. Although the presented coupling

between air and liquid phases features conservative heat and mass transfer, transfer of momentum could lead to interesting new effects like waves and air turbulence. It is, however, a challenging problem to dynamically couple two phases of such large density differences as water and air [SP08]. In a first step, this could be solved by simulating both phases separately and by applying a more loose coupling as has been previously done to simulate liquids at locally different spatial resolutions [SG11, HS13], albeit at the cost of physical plausibility. Employing an efficient GPU-based implementation, this could yield a unified adaptive SPH simulation with coupled air, liquid and solid phases as well as phase transitions at interactive frame rates.

Transfer of momentum

Furthermore, small-scale scenarios like the condensation of droplets depend on strong surface tension forces for which the influence of temperature and concentrations so far has been neglected [BT07, AAT13]. Therefore, a new surface tension model has to be devised. The model should be able to yield controllable contact angles between liquid surfaces and rigid object. Additionally, the model has to capture Marangoni effects [FAB\*11] that cause a convective motion on the fluid surface and follows gradients in surface tension. A fully dynamic coupling with the air phase would also allow for a more natural modeling of bubbles and surface tension related effects like tears of wine.

Surface tension with Marangoni effects

Although the adaptive sampling presented in this thesis is able to considerably speed up volume ray casting, it has only been demonstrated using the emission-absorption model. Extending adaptive ray casting to support light scattering from external light sources would greatly enhance the visual appearance [JSYR14]. Furthermore, by adaptively sampling transparency as seen from the light source, an efficient rendering of shadows might be incorporated, e.g., using Deep Shadow Maps [HKSB06] that can also be included in the error analysis.

Adaptive volume illumination

Recently, ray tracing has gained much momentum and first approaches pursue ray tracing of particle-based liquid surfaces [MWE16, BSS\*18, WTYH18]. The liquid's volume, however, has so far been neglected. A generalized hierarchical error analysis that estimates sampling errors for rays from arbitrary directions would allow to devise a direct adaptive volume rendering approach that supports refractions at liquid surfaces. In combination with the proposed implicit surface definition, a highly efficient direct surface and volume ray tracing of particle-based fluids could be achieved.

Adaptive on-the-fly ray tracing

While the proposed visualization of advective-diffusive flows has been demonstrated to give valuable insight into complex phenomena, the metaphor of stream feathers causes occlusions and the seeding also had to be steered manually. Even though there are methods to automatically seed streamlines in advective flows [ELM\*12], new seeding criteria have to be investigated when

Automatic stream feather seeding

Advection-diffusive  
LIC

diffusion is involved and visual clutter of stream feathers has to be addressed. An interesting alternative visualization approach that allows more dense representations of advection-diffusion scenarios is to interactively seed stream surfaces [BFTW09, MSE14] that trace the total flux and use line integral convolution to superpose the advective and diffusive contributions.

# Bibliography

- [AAOT13] AKINCI G., AKINCI N., OSWALD E., TESCHNER M.: Adaptive surface reconstruction for sph using 3-level uniform grids. In *Proceedings of the International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)* (2013), pp. 195–204. 20
- [AAT13] AKINCI N., AKINCI G., TESCHNER M.: Versatile surface tension and adhesion for SPH fluids. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 182:1–182:8. 25, 32, 44, 103
- [AIA\*12] AKINCI N., IHMSEN M., AKINCI G., SOLENTHALER B., TESCHNER M.: Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics (TOG)* 31, 4 (July 2012), 62:1–62:8. 25, 33, 43
- [AIAT12] AKINCI G., IHMSEN M., AKINCI N., TESCHNER M.: Parallel surface reconstruction for particle-based fluids. *Computer Graphics Forum (CGF)* 31 (2012), 1797–1809. 3, 5, 20, 60, 86
- [ALS09] ADAM G., LÄUGER P., STARK G.: *Physikalische Chemie und Biophysik*. Springer-Lehrbuch. Springer, 2009. 12, 34, 93
- [APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Transactions on Graphics (TOG)* 26 (2007). 35, 36, 62
- [ATO16] ALDUÁN I., TENA A., OTADUY M. A.: DYVERSO: A versatile multi-phase position-based fluids solution for VFX. *Computer Graphics Forum (CGF)* 36, 8 (sep 2016), 32–44. 15, 32, 36, 37
- [ATW13] ANDO R., THÜREY N., WOJTAN C.: Highly adaptive liquid simulations on tetrahedral meshes. *ACM Transactions on Graphics (TOG)* (July 2013). 16
- [BCP\*12] BRAMBILLA A., CARNECKY R., PEIKERT R., VIOLA I., HAUSER H.: Illustrative Flow Visualization: State of the Art, Trends and Challenges. In *Eurographics 2012 - State of the Art Reports* (2012), pp. 75–94. 24

- [BFTW09] BÜRGER K., FERSTL F., THEISEL H., WESTERMANN R.: Interactive Streak Surface Visualization on the GPU. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 15, 6 (nov 2009), 1259–1266. 24, 104
- [BHMF08] BEYER J., HADWIGER M., MÖLLER T., FRITZ L.: Smooth mixed-resolution GPU volume rendering. In *Proceedings of the Eurographics/IEEE VGTC Symposium on Point-Based Graphics* (2008), pp. 163–170. 61
- [BK02] BONET J., KULASEGARAM S.: A simplified approach to enhance the performance of smooth particle hydrodynamics methods. *Journal of Applied Mathematics and Computation* 126, 2-3 (2002), 133–155. 30, 61
- [BK15] BENDER J., KOSCHIER D.: Divergence-free smoothed particle hydrodynamics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2015), pp. 147–155. 15, 25, 34
- [BKKW17] BENDER J., KOSCHIER D., KUGELSTADT T., WEILER M.: A micropolar material model for turbulent SPH fluids. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2017), pp. 1–8. 15
- [BKZ92] BRACKBILL J. U., KOTHE D. B., ZEMACH C.: A continuum method for modeling surface tension. *Journal of Computational Physics* 100, 2 (June 1992), 335–354. 32
- [Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)* 1, 3 (jul 1982), 235–256. 19
- [BOT01] BØRVE S., OMANG M., TRULSEN J.: Regularized smoothed particle hydrodynamics: A new approach to simulating magnetohydrodynamic shocks. *The Astrophysical Journal Supplement Series* 561 (2001), 345–367. 62
- [BPHK13] BREINLINGER T., POLFER P., HASHIBON A., KRAFT T.: Surface tension and wetting effects with smoothed particle hydrodynamics. *Journal of Computational Physics* 243 (jun 2013), 14–27. 32, 50
- [BPR01] BÜRKLE D., PREUSSER T., RUMPF M.: Transport and anisotropic diffusion in time-dependent flow visualization. In *Proceedings of the IEEE Conference on Visualization* (2001), pp. 61–68. 86



- [BR86] BRACKBILL J. U., RUPPEL H. M.: Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics* 65, 2 (Aug. 1986), 314–343. 16
- [Bri07] BRIDSON R.: Fast poisson disk sampling in arbitrary dimensions. In *SIGGRAPH sketches* (2007), p. 22. 45
- [Bri08] BRIDSON R.: *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press, 2008. 9, 11, 14, 15, 30
- [BSL07] BIRD R., STEWART W., LIGHTFOOT E.: *Transport Phenomena*. Wiley, 2007. 9, 12, 46, 85, 91
- [BSS\*18] BIEDERT T., SOHNS J.-T., SCHRÖDER S., AMSTUTZ J., WALD I., GARTH C.: Direct Raytracing of Particle-based Fluid Surfaces Using Anisotropic Kernels. In *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)* (2018), pp. 1–11. 20, 103
- [BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2007), pp. 209–217. 15, 25, 32, 34, 103
- [BTT09] BECKER M., TESSENDORF H., TESCHNER M.: Direct forcing for lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 15, 3 (2009), 493–503. 25, 33
- [BZBP09] BUNGARTZ H.-J., ZIMMER S., BUCHHOLZ M., PFLÜGER D.: *Modellbildung und Simulation: Eine anwendungsorientierte Einführung*. Springer-Verlag, Berlin, Mar. 2009. 18, 19
- [CCB\*08] CHATELAIN P., CURIONI A., BERGDORF M., ROSSINELLI D., ANDREONI W., KOUMOUTSAKOS P.: Billion vortex particle direct numerical simulations of aircraft wakes. *Computer Methods in Applied Mechanics and Engineering* 197, 13-16 (feb 2008), 1296–1304. 3, 4, 16
- [CIPT14] CORNELIS J., IHMSEN M., PEER A., TESCHNER M.: IISPH-FLIP for incompressible fluids. *Computer Graphics Forum (CGF)* 33, 2 (may 2014), 255–262. 15
- [CL93] CABRAL B., LEEDOM L. C.: Imaging vector fields using line integral convolution. In *Proceedings of SIGGRAPH* (1993), pp. 263–270. 23, 84, 86

- [CM99] CLEARY P. W., MONAGHAN J. J.: Conduction modelling using smoothed particle hydrodynamics. *Journal of Computational Physics* 148 (1999), 227–264. 18, 34, 35, 45, 60
- [CM14] CHENTANEZ N., MÜLLER M.: Mass-conserving eulerian liquid simulation. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 20, 1 (jan 2014), 17–29. 14
- [CMK15] CHENTANEZ N., MÜLLER M., KIM T.-Y.: Coupling 3d eulerian, heightfield and particle methods for interactive simulation of large scale liquid phenomena. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 21, 10 (oct 2015), 1116–1128. 3, 4, 16
- [COJ15] CHANDLER J., OBERMAIER H., JOY K. I.: Interpolation-based path-line tracing in particle-based flow visualization. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 21, 1 (jan 2015), 68–80. 87
- [CPK09] CUNTZ N., PRITZKAU A., KOLB A.: Time-adaptive lines for the interactive visualization of unsteady flow data sets. *Computer Graphics Forum (CGF)* 28, 8 (2009), 2165–2175. 24
- [Cus09] CUSSLER E.: *Diffusion: Mass Transfer in Fluid Systems*. Cambridge Series in Chemical Engineering. Cambridge University Press, 2009. 89
- [DC96] DESBRUN M., CANI M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (1996), pp. 61–76. 15, 25, 27
- [DC99] DESBRUN M., CANI M.-P.: *Space-Time Adaptive Simulation of Highly Deformable Substances*. Tech. Rep. 3829, INRIA, 1999. 35
- [DCGG13] DOMÍNGUEZ J. M., CRESPO A. J. C., GÓMEZ-GESTEIRA M.: Optimization strategies for {cpu} and {gpu} implementations of a smoothed particle hydrodynamics method. *Computer Physics Communications* 184, 3 (2013), 617–627. 36
- [Des04] DESERNO M.: The shape of a straight fluid meniscus, 2004. 50
- [DH92] DANSKIN J., HANRAHAN P.: Fast algorithms for volume ray tracing. In *Proceedings of the Workshop on Volume Visualization (VVS)* (1992), vol. I, pp. 91–98. 61

- [DH93] DELMARCELLE T., HESSELINK L.: Visualizing second-order tensor fields with hyperstreamlines. *IEEE Computer Graphics and Applications* 13, 4 (July 1993), 25–33. 86
- [DPR00] DIEWALD U., PREUSSER T., RUMPF M.: Anisotropic diffusion in vector field visualization on euclidean domains and surfaces. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 6, 2 (Apr. 2000), 139–149. 86
- [EH57] EVANS M. W., HARLOW F. H.: *The Particle-in-Cell Method for Hydrodynamic Calculations*. Tech. Rep. LA-2139, Los Alamos Scientific Lab., N. Mex., 1957. 16
- [EHK\*06] ENGEL K., HADWIGER M., KNISS J. M., REZK-SALAMA C., WEISKOPF D.: *Real-time Volume Graphics*. A. K. Peters, Ltd., Natick, MA, USA, 2006. 9, 21, 22, 60, 61, 84
- [Ein05] EINSTEIN A.: Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen. *Annalen der Physik* 322, 8 (1905), 549–560. 87, 89
- [EJGP09] EL HAJJAR J. F., JOLIVET V., GHAZANFARPOUR D., PUEYO X.: A model for real-time on-surface flows. *The Visual Computer* 25, 2 (2009), 87–100. 43
- [EKE01] ENGEL K., KRAUS M., ERTL T.: High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of the ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware* (2001), pp. 9–16. 22
- [ELM\*12] EDMUNDS M., LARAMEE R., MALKI R., MASTERS I., CROFT T., CHEN G., ZHANG E.: Automatic stream surface seeding: A feature centered approach. *Computer Graphics Forum (CGF)* 31, 3pt2 (2012), 1095–1104. 24, 103
- [EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Transactions on Graphics (TOG)* 21, 3 (July 2002), 736–744. 14
- [FAB\*11] FELL D., AUERNHAMMER G. K., BONACCURSO E., LIU C., SOKULER R., BUTT H.-J.: Influence of surfactant concentration and background salt on forced dynamic wetting and dewetting. *Langmuir* 27, 6 (2011), 2112. 33, 84, 103

- [FAW10] FRAEDRICH R., AUER S., WESTERMANN R.: Efficient high-quality volume rendering of SPH data. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 16, 6 (2010), 1533–1540. 6, 22, 37, 60, 62, 63, 74
- [FAW\*16] FERSTL F., ANDO R., WOJTAN C., WESTERMANN R., THUEREY N.: Narrow Band FLIP for Liquid Simulations. *Computer Graphics Forum (CGF)* 35, 2 (may 2016), 225–232. 16
- [FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proceedings of SIGGRAPH* (2001), pp. 23–30. 14
- [FGE10] FALK M., GROTTTEL S., ERTL T.: Interactive image-space volume visualization for dynamic particle simulations. In *Proceedings of SIGRAD* (2010). 37, 60
- [FGG\*17] FU C., GUO Q., GAST T., JIANG C., TERAN J.: A polynomial particle-in-cell method. *ACM Transactions on Graphics (TOG)* 36, 6 (nov 2017), 1–12. 16
- [FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graphical Models and Image Processing* 58, 5 (Sept. 1996), 471–483. 14
- [FM97] FOSTER N., METAXAS D.: Modeling the motion of a hot, turbulent gas. In *Proceedings of SIGGRAPH* (1997), pp. 181–188. 14
- [FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *Proceedings of SIGGRAPH* (2001), pp. 15–22. 3, 14, 15, 41, 44
- [FSW09] FRAEDRICH R., SCHNEIDER J., WESTERMANN R.: Exploring the millennium run – scalable rendering of large-scale cosmological datasets. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 15, 6 (Nov. 2009), 1251–1258. 3, 22, 62
- [FW08] FALK M., WEISKOPF D.: Output-sensitive 3d line integral convolution. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 14, 4 (jul 2008), 820–834. 24
- [FW12] FRAEDRICH R., WESTERMANN R.: Motion visualization of large particle simulations. In *Proceedings of IS&T/SPIE Electronic Imaging 2012, Conference on Visualization and Data Analysis* (2012), pp. 82940Q–1 – 12. 6, 86
- [GB14] GOSWAMI P., BATTY C.: Regional time stepping for SPH. In *Eurographics 2014 - Short Papers* (2014), pp. 1–5. 35

- [GBP\* 17a] GISSLER C., BAND S., PEER A., IHMSEN M., TESCHNER M.: Approximate air-fluid interactions for SPH. In *Proceedings of the Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)* (2017). 5, 33
- [GBP\* 17b] GISSLER C., BAND S., PEER A., IHMSEN M., TESCHNER M.: Generalized drag force for particle-based simulations. *Computers & Graphics* 69 (dec 2017), 1–11. 5, 33
- [GM77] GINGOLD R., MONAGHAN J.: Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Notices of the Royal Astronomical Society* 181 (1977), 375–389. 3, 25, 27
- [GP11] GOSWAMI P., PAJAROLA R.: Time adaptive approximate SPH. In *Proceedings of the Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)* (2011), pp. 19–28. 35
- [Gre09] GREEN S.: *Particle Simulation using CUDA*. Tech. rep., NVIDIA, 2009. 15, 36, 37, 45
- [GRT13] GÜNTHER T., RÖSSL C., THEISEL H.: Opacity optimization for 3d line fields. *ACM Transactions on Graphics (TOG)* 32, 4 (July 2013), 120:1–120:8. 24
- [GS04] GUTHE S., STRASSER W.: Advanced techniques for high-quality multi-resolution volume rendering. *Computers & Graphics* 28, 1 (Feb. 2004), 51–58. 61
- [GSSP10] GOSWAMI P., SCHLEGEL P., SOLENTHALER B., PAJAROLA R.: Interactive SPH simulation and rendering on the GPU. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2010), pp. 55–64. 20, 36, 37
- [GTG17] GÜNTHER T., THEISEL H., GROSS M.: Decoupled opacity optimization for points, lines and surfaces. *Computer Graphics Forum (CGF)* 36, 2 (may 2017), 153–162. 24
- [Har64] HARLOW F.: The particle-in-cell computing method for fluid dynamics. *Methods in Computational Physics* 3 (1964), 319–343. 14
- [HCCC12] HONGFENG YU, CHAOLI WANG, CHING-KUANG SHENE, CHEN J. H.: Hierarchical streamline bundles. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 18, 8 (aug 2012), 1353–1367. 24

- [HE03] HOPF M., ERTL T.: Hierarchical splatting of scattered data. In *Proceedings of the IEEE Conference on Visualization* (2003), pp. 433–440. 22, 62
- [HEW15] HUBER M., EBERHARDT B., WEISKOPF D.: Boundary handling at cloth-fluid contact. *Computer Graphics Forum (CGF)* 34, 1 (feb 2015), 14–25. 41
- [HHK08] HONG W., HOUSE D. H., KEYSER J.: Adaptive particles for incompressible fluid simulation. *The Visual Computer* 24, 7 (2008), 535–543. 62
- [HK17] HOCHSTETTER H., KOLB A.: Evaporation and condensation of SPH-based fluids. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2017), pp. 3:1–3:9. 8, 39, 46
- [HKK07a] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Sliced data structure for particle-based simulations on GPUs. In *Proceedings of GRAPHITE* (2007), pp. 55–62. 37
- [HKK07b] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics in complex shapes. In *Proceedings of the Spring Conference on Computer Graphics (SCCG)* (2007), pp. 191–197. 5, 41
- [HKK07c] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics on GPUs. In *Proceedings of Computer Graphics International (CGI)* (2007), pp. 63–70. 15, 25, 32, 37
- [HKS06] HADWIGER M., KRATZ A., SIGG C., BÜHLER K.: GPU-accelerated deep shadow maps for direct volume rendering. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Graphics Hardware* (2006), p. 49. 21, 103
- [HLSR08] HADWIGER M., LJUNG P., SALAMA C. R., ROPINSKI T.: Advanced illumination techniques for GPU volume raycasting. In *SIGGRAPH Asia Courses* (2008). 21, 60
- [HLY10] HERNELL F., LJUNG P., YNNERMAN A.: Local Ambient Occlusion in Direct Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics* 16, 4 (jul 2010), 548–559. 21
- [HM08] HUNT W., MARK W. R.: Ray-specialized acceleration structures for ray tracing. In *Proceedings of the IEEE/Eurographics Symposium on Interactive Ray Tracing* (2008), pp. 3–10. 62

- [Hoe16] HOETZLEIN R. K.: GVDB: Raytracing Sparse Voxel Database Structures on the GPU. In *Proceedings of the ACM SIGGRAPH/Eurographics High Performance Graphics (HPG)* (2016), 20
- [HOK16] HOCHSTETTER H., ORTHMANN J., KOLB A.: Adaptive sampling for on-the-fly ray casting of particle-based fluids. In *Proceedings of the ACM SIGGRAPH/Eurographics High Performance Graphics (HPG)* (2016), pp. 129–138. 8, 59
- [HRWE15] HUBER M., REINHARDT S., WEISKOPF D., EBERHARDT B.: Evaluation of surface tension models for SPH-based fluid animations using a benchmark test. *Proceedings of the Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)* (2015), 41–50. 32
- [HS13] HORVATH C. J., SOLENTHALER B.: *Mass Preserving Multi-Scale SPH*. Tech. rep., Pixar, Emeryville, CA, 2013. 35, 103
- [HWK15] HOCHSTETTER H., WURM M., KOLB A.: Vector field visualization of advective-diffusive flows. *Computer Graphics Forum (CGF)* 34, 3 (jun 2015), 481–490. 8, 83
- [HWZ\*14] HE X., WANG H., ZHANG F., WANG G., ZHOU K.: Robust simulation of sparsely sampled thin features in SPH-based free surface flows. *ACM Transactions on Graphics (TOG)* 34, 1 (2014), 1–8. 33
- [IABT11] IHMSEN M., AKINCI N., BECKER M., TESCHNER M.: A parallel SPH implementation on multi-core CPUs. *Computer Graphics Forum (CGF)* 30 (2011), 99–112. 15, 36, 37
- [IAGT10] IHMSEN M., AKINCI N., GISSLER M., TESCHNER M.: Boundary handling and adaptive time-stepping for PCISPH. In *Proceedings of the Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)* (2010), pp. 79–88. 18, 33, 36
- [ICS\*14] IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 20, 3 (2014), 426–435. 4, 15, 25, 34
- [IOS\*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH fluids in computer graphics. In *Eurographics 2014 - State of the Art Reports* (2014), no. 2, pp. 21–42. 5, 15, 19, 25, 26, 27, 32, 33, 34, 60

- [Jak10] JAKOB W.: Mitsuba renderer, 2010. <http://www.mitsuba-renderer.org>. 53
- [JH04] JOHNSON C., HANSEN C.: *Visualization Handbook*. Academic Press, Inc., Orlando, FL, USA, 2004. 9, 23, 88
- [JSS\*15] JIANG C., SCHROEDER C., SELLE A., TERAN J., STOMAKHIN A.: The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 10. 16
- [JSYR14] JÖNSSON D., SUNDÉN E., YNNERMAN A., ROPINSKI T.: A Survey of Volumetric Illumination Techniques for Interactive Volume Rendering. *Computer Graphics Forum (CGF)* 33, 1 (feb 2014), 27–51. 21, 103
- [KAG\*06] KEISER R., ADAMS B., GUIBAS L. J., DUTRÉ P. P., PAULY M.: *Multiresolution Particle-Based Fluids*. Tech. rep., ETH, 2006. 36, 62
- [KC05] KOLB A., CUNTZ N.: Dynamic particle coupling for GPU-based fluid simulation. In *Proceedings of the Symposium on Simulation Technique* (2005), pp. 722–727. 22
- [KFW16] KANZLER M., FERSTL F., WESTERMANN R.: Line density control in screen-space via balanced line hierarchies. *Computers & Graphics* 61 (dec 2016), 29–39. 24
- [KHW\*09] KNOLL A., HIJAZI Y., WESTERTEIGER R., SCHOTT M., HANSEN C., HAGEN H.: Volume ray casting with peak finding and differential sampling. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 15, 6 (2009), 1571–1578. 61
- [Kis93] KISTER S. F.: *Wettability*. Surfactant Science Series. Marcel Dekker, 1993, ch. Hydrodynamics of wetting. 51
- [KSW\*12] KARCH G. K., SADLO F., WEISKOPF D., MUNZ C.-D., ERTL T.: Visualization of advection-diffusion in unsteady fluid flow. *Computer Graphics Forum (CGF)* 31, 3pt2 (2012), 1105–1114. 84, 86
- [KW06] KINDLMANN G., WESTIN C.-F.: Diffusion tensor visualization with glyph packing. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 12, 5 (Sept.-Oct. 2006), 1329–1335. 86
- [KWFY99] KAMHOLZ A. E., WEIGL B. H., FINLAYSON B. A., YAGER P.: Quantitative analysis of molecular interaction in a microfluidic channel: the t-sensor. *Analytical Chemistry* 71, 23 (1999), 5340–5347. 84, 95



- [KWH00] KINDLMANN G., WEINSTEIN D., HART D.: Strategies for direct volume rendering of diffusion tensor fields. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 6, 2 (Apr.–June 2000), 124–138. 86
- [LAF11] LENTINE M., AANJANEYA M., FEDKIW R.: Mass and momentum conservation for fluid simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2011), pp. 91–100. 14
- [LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics* 21, 4 (Aug. 1987), 163–169. 19, 20, 56
- [Lev88] LEVOY M.: Display of surfaces from volume data. *IEEE Computer Graphics and Applications* 8, 3 (May 1988), 29–37. 21
- [LGM\*08] LEDERGERBER C., GUENNEBAUD G., MEYER M. D., BÄCHER M., PFISTER H.: Volume MLS ray casting. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 14, 6 (2008), 1372–1379. 61
- [LJS\*15] LADICKÝ L., JEONG S., SOLENTHALER B., POLLEFEYS M., GROSS M.: Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics (TOG)* 34, 6 (oct 2015), 1–9. 3
- [LL16] LIENHARD IV J., LIENHARD V J.: *A Heat Transfer Textbook*, 4th ed. Phlogiston Press, Cambridge, MA, 2016. Version 2.05. 46
- [LSSF06] LOSASSO F., SHINAR T., SELLE A., FEDKIW R.: Multiple interacting liquids. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 812–819. 39, 41
- [Luc77] LUCY L. B.: A numerical approach to the testing of the fission hypothesis. *Astronomical Journal* 82 (1977), 1013–1024. 3, 25
- [Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 1, 2 (1995), 99–108. 21
- [MC10] MAX N., CHEN M.: Local and global illumination in the volume rendering integral. In *Scientific Visualization: Advanced Concepts* (2010), vol. 1, pp. 259–274. 21, 22

- [MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2003), pp. 154–159. 15, 19, 25, 27, 32, 34
- [MFZ97] MORRIS J. P., FOX P. J., ZHU Y.: Modeling low reynolds number incompressible flows using SPH. *Journal of Computational Physics* 136 (1997), 214–226. 28, 32
- [MKC09] MOORE R. E., KEARFOTT R. B., CLOUD M. J.: *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009. 62, 63
- [MLP\*10] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum (CGF)* 29, 6 (2010), 1807–1829. 24, 84
- [MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Transactions on Graphics (TOG)* 32, 4 (jul 2013), 104:1–104:12. 15, 20, 25, 34
- [MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)* 33, 4 (jul 2014), 1–12. 3, 5, 15, 25, 37, 39
- [Mon92] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Annual review of Astronomy and Astrophysics* 30 (1992), 543–574. 27
- [Mon05] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Reports on Progress in Physics* 68 (2005), 1703–1759. 15, 18, 25, 26, 28, 29, 31, 33, 34, 36, 60
- [MS93] MA K.-L., SMITH P.: Cloud tracing in convection-diffusion systems. In *Proceedings of the IEEE Conference on Visualization* (Oct 1993), pp. 253–260. 86
- [MSD07] MÜLLER M., SCHIRM S., DUTHALER S.: Screen space meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2007), pp. 9–15. 20
- [MSE14] MACHADO G. M., SADLO F., ERTL T.: Image-based streamsurfaces. In *Proceedings of the Conference on Graphics, Patterns and Images (SIBGRAPI)* (2014), pp. 343 – 350. 24, 104

- [MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2005), pp. 237–244. 39, 40
- [MWE16] MORGENROTH D., WEISKOPF D., EBERHARDT B.: Direct raytracing of a closed-form fluid meniscus. *The Visual Computer* 32, 6-8 (jun 2016), 791–800. 5, 20, 41, 50, 51, 103
- [MWN\*16] MANTEAUX P.-L., WOJTAN C., NARAIN R., REDON S., FAURE F., CANI M.-P.: Adaptive physically based models in computer graphics. *Computer Graphics Forum (CGF)* (jun 2016). 36
- [NMM\*06] NEOPHYTOU N., MUELLER K., MCDONNELL K. T., HONG W., GUAN X., QIN H., KAUFMAN A. E.: GPU-accelerated volume splatting with elliptical RBFs. In *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization (EuroVis)* (2006), pp. 13–20. 60
- [OCD11] ONDERIK J., CHLADEK M., DURIKOVIC R.: SPH with small scale details and improved surface reconstruction. In *Proceedings of the Spring Conference on Computer Graphics (SCCG)* (2011). 20, 41, 60
- [OHB\*13] ORTHMANN J., HOCHSTETTER H., BADER J., BAYRAKTAR S., KOLB A.: Consistent surface model for SPH-based fluid transport. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2013), pp. 95–103. 5, 29, 33, 44, 60, 69
- [OK12] ORTHMANN J., KOLB A.: Temporal blending for adaptive SPH. *Computer Graphics Forum (CGF)* 31, 8 (2012), 2436–2449. 5, 35, 37, 44
- [OKK10] ORTHMANN J., KELLER M., KOLB A.: Topology-caching for dynamic particle volume raycasting. In *Proceedings of the Symposium on Vision, Modeling and Visualization (VMV)* (2010), pp. 147–154. 6, 22, 62, 82
- [Ort14] ORTHMANN J.: *Efficient SPH-based simulation and rendering of fluid transport dynamics*. PhD thesis, University of Siegen, 2014. 4, 59, 62, 70, 73, 74, 75, 82
- [PCPW15] PRAKASH M., CLEARY P. W., PYO S. H., WOOLARD E.: A new approach to boiling simulation using a discrete particle based method. *Computers & Graphics* 53 (dec 2015), 118–126. 39, 40

- [PGSS07] POPOV S., GÜNTHER J., SEIDEL H.-P., SLUSALLEK P.: Stackless KD-tree traversal for high performance GPU ray tracing. *Computer Graphics Forum (CGF)* 26, 3 (2007), 415–424. 62
- [PICT15] PEER A., IHMSEN M., CORNELIS J., TESCHNER M.: An implicit viscosity formulation for SPH fluids. *ACM Transactions on Graphics (TOG)* 34, 4 (jul 2015), 114:1–114:10. 5, 32
- [PTC\*10] PFAFF T., THUREY N., COHEN J., TARIQ S., GROSS M.: Scalable fluid simulation using anisotropic turbulence particles. *ACM Transactions on Graphics (TOG)* 29, 6 (Dec. 2010), 174:1–174:8. 14, 15
- [RCSW14] REICHL F., CHAJDAS M. G., SCHNEIDER J., WESTERMANN R.: Interactive rendering of giga-particle fluid simulations. In *Proceedings of the ACM SIGGRAPH/Eurographics High Performance Graphics (HPG)* (2014), pp. 105–116. 20, 60
- [RHEW17] REINHARDT S., HUBER M., EBERHARDT B., WEISKOPF D.: Fully asynchronous SPH simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2017), pp. 1–10. 35
- [RLY\*14] REN B., LI C., YAN X., LIN M. C., BONET J., HU S.-M.: Multiple-Fluid SPH Simulation Using a Mixture Model. *ACM Transactions on Graphics (TOG)* 33, 5 (sep 2014), 1–11. 25, 40
- [RTW13] REICHL F., TREIB M., WESTERMANN R.: Visualization of big SPH simulations via compressed octree grids. In *Proceedings of the IEEE International Conference on Big Data* (2013), pp. 71–78. 22, 62
- [RWT11] RAVEENDRAN K., WOJTAN C., TURK G.: Hybrid smoothed particle hydrodynamics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2011), pp. 33–42. 15
- [RY96] ROGERS R. R., YAU M. K.: *A short course in cloud physics*, 3rd ed. Butterworth-Heinemann, 1996. 42
- [SB12] SCHECHTER H., BRIDSON R.: Ghost SPH for animating water. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 61:1–61:8. 5, 33
- [Sch96] SCHNOOR J.: *Environmental modeling: fate and transport of pollutants in water, air, and soil*. Environmental science and technology. J. Wiley, 1996. 87

- [SFBP09] SCHINDLER B., FUCHS R., BIDDISCOMBE J., PEIKERT R.: Predictor-corrector schemes for visualization of smoothed particle hydrodynamics data. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 15, 6 (2009), 1243–1250. 86, 87
- [SFK\*08] SELLE A., FEDKIW R., KIM B., LIU Y., ROSSIGNAC J.: An unconditionally stable MacCormack method. *Journal of Scientific Computing* 35 (June 2008), 350–371. 14
- [SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *ACM Transactions on Graphics (TOG)* 30 (2011), 81:1–81:8. 5, 35, 103
- [Sha14] SHAH M. M.: Methods for calculation of evaporation from swimming pools and other water surfaces. *ASHRAE Transactions* 120, 2 (2014), 3–17. 42
- [She68] SHEPARD D.: A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the ACM National Conference* (1968), pp. 517–524. 30
- [SHG09] SATISH N., HARRIS M., GARLAND M.: Designing efficient sorting algorithms for manycore GPUs. In *Proceedings of the IEEE International Symposium on Parallel Distributed Processing* (2009), pp. 1–10. 37
- [SHZO07] SENGUPTA S., HARRIS M., ZHANG Y., OWENS J. D.: Scan primitives for GPU computing. In *Proceedings of the ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware* (2007), pp. 97–106. 37
- [SKE14] SADLO F., KARCH G. K., ERTL T.: Topological features in time-dependent advection-diffusion flow. In *Topological Methods in Data Analysis and Visualization III* (Cham, 2014), Bremer P.-T., Hotz I., Pascucci V., Peikert R., (Eds.), Springer International Publishing, pp. 217–231. 84, 86
- [SLJ94] SMITH C. C., LÖF G., JONES R.: Measurement and analysis of evaporation from an inactive outdoor swimming pool. *Solar Energy* 53, 1 (1994), 3–7. 42
- [SP08] SOLENTHALER B., PAJAROLA R.: Density contrast SPH interfaces. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2008), pp. 211–218. 5, 25, 29, 103

- [SP09a] SCHLEGEL P., PAJAROLA R.: Layered volume splatting. In *Proceedings of the International Symposium on Visual Computing (ISVC)* (2009), vol. 5876, pp. 1–12. 60
- [SP09b] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Transactions on Graphics (TOG)* 28 (2009), 40:1–40:6. 15, 25, 34, 43
- [SRF05] SELLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics (TOG)* 24, 3 (July 2005), 910–914. 16
- [SSP07] SOLENTHALER B., SCHLÄFLI J., PAJAROLA R.: A unified particle model for fluid solid interactions: Research articles. *Computer Animation and Virtual Worlds* 18, 1 (2007), 69–82. 5, 20, 33, 40, 41, 60
- [Sta99] STAM J.: Stable fluids. In *Proceedings of SIGGRAPH* (1999), pp. 121–128. 3, 14, 44
- [TB15] TILLMANN S.-T., BOHN C.-A.: Simulation of water condensation based on a thermodynamic approach. In *Proceedings of the Symposium on Vision, Modeling and Visualization (VMV)* (2015). 39, 41
- [THM\*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANETS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. *Proceedings of the Symposium on Vision, Modeling and Visualization (VMV)* (2003), 8. 36
- [TM05] TARTAKOVSKY A., MEAKIN P.: Modeling of surface tension and contact angles with smoothed particle hydrodynamics. *Physical Review E* 72 (2005). 32
- [UHT17] UM K., HU X., THUREY N.: Perceptual evaluation of liquid simulation methods. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 143:1–143:12. 17
- [vdLGS09] VAN DER LAAN W. J., GREEN S., SAINZ M.: Screen space fluid rendering with curvature flow. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)* (2009), pp. 91–98. 3, 20, 37, 60
- [VTT\*18] VILLA SALAZAR S., TICONA J. A., TORCHELSEN R., NEDEL L., MACIEL A.: Heat-based bidirectional phase shifting simulation using

- position-based dynamics. *Computers & Graphics* 76 (nov 2018), 107–116. 39, 40
- [Wes90] WESTOVER L.: Footprint evaluation for volume rendering. *Computer Graphics* 24, 4 (Sept. 1990), 367–376. 60
- [WHK16] WINCHENBACH R., HOCHSTETTER H., KOLB A.: Constrained neighbor lists for SPH-based fluid simulations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2016), pp. 49–56. 36, 37
- [WHK17] WINCHENBACH R., HOCHSTETTER H., KOLB A.: Infinite continuous adaptivity for incompressible SPH. *ACM Transactions on Graphics (TOG)* 36, 4 (jul 2017), 102:1–102:10. 5, 35, 44, 47, 49, 50, 57, 102
- [WJP14] WALD I., JOHNSON G. P., PAPKA M. E.: CPU ray tracing large particle data with balanced p-k-d trees. In *Proceedings of the IEEE Conference on Visualization* (2014), pp. 57–64. 20
- [WKL99] WEINSTEIN D., KINDLMANN G., LUNDBERG E.: Tensorlines: Advection-diffusion based propagation through diffusion tensor fields. In *Proceedings of the IEEE Conference on Visualization* (1999), pp. 249–253. 86
- [WMT05] WANG H., MUCHA P. J., TURK G.: Water drops on surfaces. *ACM Transactions on Graphics (TOG)* 24, 3 (jul 2005), 921–929. 41, 45, 48
- [WMT07] WANG H., MILLER G., TURK G.: Solving general shallow wave equations on surfaces. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2007), Eurographics Association, pp. 229–238. 43
- [WTYH18] WU K., TRUONG N., YUKSEL C., HOETZLEIN R.: Fast Fluid Simulations with Sparse Volumes on the GPU. *Computer Graphics Forum (CGF)* 37, 2 (may 2018), 157–167. 20, 36, 103
- [YCL\*17] YANG T., CHANG J., LIN M. C., MARTIN R. R., ZHANG J. J., HU S.-M.: A unified particle system framework for multi-phase, multi-material visual simulations. *ACM Transactions on Graphics (TOG)* 36, 6 (nov 2017), 224:1—224:13. 25, 40
- [YML\*17] YANG T., MARTIN R. R., LIN M. C., CHANG J., HU S.-M.: Pairwise Force SPH Model for Real-Time Multi-Interaction Applications. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 23, 10 (oct 2017), 2235–2247. 32

- [YT10] YU J., TURK G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2010), pp. 217–225. 5, 41
- [YT13] YU J., TURK G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Transactions on Graphics (TOG)* 32, 1 (Feb. 2013), 5:1–5:12. 20, 60
- [ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 965–972. 16, 19, 33, 41
- [ZD15] ZIRR T., DACHSBACHER C.: Memory-efficient on-the-fly voxelization of particle data. In *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)* (2015). 20, 60
- [ZD17] ZIRR T., DACHSBACHER C.: Memory-efficient on-the-fly voxelization and rendering of particle data. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 99 (2017). 20, 60
- [ZM13] ZHANG Y., MA K.-L.: Fast global illumination for interactive volume visualization. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)* (2013), pp. 55–62. 21
- [ZSP08] ZHANG Y., SOLENTHALER B., PAJAROLA R.: Adaptive sampling and rendering of fluids on the GPU. In *Proceedings of the Eurographics/IEEE VGTC Symposium on Point-Based Graphics* (2008), pp. 137–146. 37, 62