SMOOTHED PARTICLE HYDRODYNAMICS FOR FLUID SIMULATION IN COMPUTER GRAPHICS

YANRUI XU

Cover:
Simulation of two-dimensional two-phase ink-drop scenario based on the proposed Smoothed Particle Hydrodynamics method in this thesis. Each dot denotes a Lagrangian particle of the heavier phase in the simulation.

Smoothed Particle Hydrodynamics for Fluid Simulation
in Computer Graphics

Yanrui Xu
PhD Thesis

The research for this dissertation was conducted at:

Scientific Visualization and Computer Graphics (SVCG) research group, part of the Bernoulli Institute (BI) and the Faculty of Science and Engineering (FSE) at the University of Groningen, the Netherlands

and

Artificial Intelligence and 3D Visualization (AI3D) research group, part of the School of Intelligence Science and Technology at University of Science and Technology Beijing, China.

university of groningen

Smoothed Particle Hydrodynamics for Fluid Simulation in Computer Graphics

PhD thesis

to obtain the degree of PhD at the
University of Groningen
on the authority of the
Rector Magnificus Prof. J.M.A. Scherpen
and in accordance with
the decision by the College of Deans.

This thesis will be defended in public on

Tuesday 21 January 2025 at 16.15 hours

by

Yanrui Xu

born on November 11th, 1995
in Beijing, China

Supervisors
Prof. J. Kosinka
Prof. A.C. Telea

Co-supervisors
Dr. S.D. Frey
Prof. X. Ban

Assessment committee
Prof. J.B.T.M. Roerdink
Prof. C.A. Bertoglio
Prof. F. Sadlo

The highest goodness is like water. Water is beneficial to all things and does not compete with them.

— Lao Tzu
Tao Te Ching

ABSTRACT

Physics-based fluid simulation has emerged as a pivotal element in the field of computer graphics, facilitating the generation of intricate and authentic visual effects via advanced computational techniques. Among various Lagrangian simulation methods, Smoothed Particle Hydrodynamics (SPH) stands out due to its adaptability and remarkable ability to capture complex dynamic phenomena accurately. However, the simulation task often involves navigating the challenging trilemma of balancing (1) efficiency, (2) accuracy, and (3) versatility. This balancing act is increasingly tested by the growing demand for more refined, accurate effects alongside the need for computational efficiency. Our research aims to tackle this trilemma by focusing on three key areas: refining the discretization scheme (balancing efficiency and accuracy), pioneering advancements in the multiphase model (balancing accuracy and cersatility), and enhancing the visualization pipeline (balancing efficiency and accuracy). In doing so, each area specifically targets two out of the three challenges, aiming for an optimal balance. We propose innovative improvements to the SPH framework that address these critical issues, as detailed in our subsequent contributions, on top of which we carried out a thorough survey of recent advancements around these topics, providing a robust foundation for our innovations.

The first area of our work focuses on efficiently enhancing numerical accuracy in fluid simulations, achieving stunning visuals while minimizing computational demands. We introduce an adaptive SPH method that leverages a boundary-distance approach. By utilizing a signed-distance field relative to the fluid's boundaries, our method dynamically adjusts particle resolutions based on their proximity to these boundaries. This results in higher resolutions near the boundary, which gradually transition to lower resolutions with increasing distance, up to a predefined limit. Particle sizes are finely tuned through a process of splitting and merging, ensuring optimal resolution distribution. Furthermore, we incorporate a novel wake flow preservation strategy, maintaining enhanced resolution temporarily for particles passing close to boundary objects, thereby safeguarding against the loss of intricate flow details.

The second area of our research addresses the challenge of accurately simulating the intricate interactions between multiphase fluids. We develop an implicit mixture model tailored for SPH. Our model eschews the conventional reliance on an explicit, unified mixture field for dynamic computations and inter-particle phase transitions. Instead, it calculates phase momentum sources within the mixture model to generate explicit, continuous velocity fields for each phase. The mixture field itself is derived implicitly through a novel phase-mixture momentum-

mapping technique, ensuring the rigorous conservation of incompressibility, mass, and momentum across the system. Furthermore, our model introduces a refined mixture viscosity framework, which moderates the viscous interactions between the combined fluid and its constituent phases, thereby mitigating instabilities in scenarios characterized by high inertial forces.

The third focus of our research is on efficiently visualizing particle-based fluid simulations for real-time applications. We introduce a screen space rendering approach that applies anisotropic transformations to point sprites, elongating them along optimized axes to achieve smoother fluid surfaces. This technique leverages a weighted principal component analysis of particle distributions to determine optimal stretching directions. Further enhancing visual fidelity, we integrate this anisotropic point sprite treatment with advanced screen space filters, such as curvature flow and narrow-range filters, to refine depth information processing. This fusion of techniques significantly improves the clarity and stability of fluid boundaries, bringing us closer to high-quality, real-time fluid visualization.

SAMENVATTING

Fysica-gebaseerde vloeistofsimulatie is uitgegroeid tot een cruciaal element in het veld van computergrafiek, waarbij het genereren van complexe en authentieke visuele effecten via geavanceerde computationele technieken wordt gefaciliteerd. Onder de verschillende Lagrangiaanse simulatiemethoden valt Smoothed Particle Hydrodynamics (SPH) op door zijn aanpasbaarheid en opmerkelijke vermogen om complexe dynamische fenomenen nauwkeurig vast te leggen. Echter, de simulatietaak omvat vaak het navigeren door het uitdagende drieluik van het balanceren van (1) efficiëntie, (2) nauwkeurigheid en (3) veelzijdigheid. Deze evenwichtsoefening wordt steeds meer op de proef gesteld door de groeiende vraag naar meer verfijnde, nauwkeurige effecten naast de behoefte aan computationele efficiëntie. Ons onderzoek richt zich op het aanpakken van dit drieluik door te focussen op drie sleutelgebieden: het verfijnen van het discretisatieschema (evenwicht tussen efficiëntie en nauwkeurigheid), het pionieren van vooruitgang in het multiphasemodel (evenwicht tussen nauwkeurigheid en veelzijdigheid), en het verbeteren van de visualisatiepijplijn (evenwicht tussen efficiëntie en veelzijdigheid). Daarbij richt elk gebied zich specifiek op twee van de drie uitdagingen, met als doel een optimale balans. We stellen innovatieve verbeteringen voor van het SPH framework die deze kritieke kwesties aanpakken, zoals gedetailleerd in onze volgende bijdragen, waarbovenop we een grondige overzichtsstudie hebben uitgevoerd over recente ontwikkelingen rond deze onderwerpen, wat een robuuste basis biedt voor onze innovaties.

Het eerste gebied van ons werk richt zich op het efficiënt verbeteren van numerieke nauwkeurigheid in vloeistofsimulaties, waarbij verbluffende visuele effecten worden bereikt terwijl de computationele eisen worden geminimaliseerd. We introduceren een adaptieve SPH methode die gebruik maakt van een boundary-distance benadering. Door gebruik te maken van een signed-distance veld ten opzichte van de grenzen van de vloeistof, past onze methode de resoluties van deeltjes dynamisch aan op basis van hun nabijheid tot deze grenzen. Dit resulteert in hogere resoluties nabij de grens, die geleidelijk overgaan in lagere resoluties met toenemende afstand, tot een vooraf gedefinieerde limiet. Deeltjesgroottes worden fijn afgesteld door een proces van splitsen en samenvoegen, om een optimale resolutieverdeling te waarborgen. Bovendien incorporeren we een nieuwe strategie voor het behoud van kielzog-stromingen, waarbij tijdelijk een verbeterde resolutie wordt onderhouden voor deeltjes die dichtbij grensobjecten passeren, waardoor verlies van ingewikkelde stroomdetails wordt voorkomen.

Het tweede onderzoeksgebied richt zich op de uitdaging van het nauwkeurig simuleren van de ingewikkelde interacties tussen multifase vloeistoffen. We ontwikkelen een impliciet mengmodel op maat voor SPH. Ons model ziet af van de conventionele afhankelijkheid van een expliciet, verenigd mengveld voor dynamische berekeningen en interdeeltjesfase-overgangen. In plaats daarvan berekent het fase-momentumbronnen binnen het mengmodel om expliciete, continue snelheidsvelden voor elke fase te genereren. Het mengveld zelf wordt impliciet afgeleid via een nieuwe fase-mengmomentum-kaarttechniek, wat zorgt voor de rigoureuze instandhouding van incompressibiliteit, massa en moment door het systeem heen. Bovendien introduceert ons model een verfijnd mengviscositeitskader, dat de viskeuze interacties tussen de gecombineerde vloeistof en zijn samenstellende fasen matigt, waardoor instabiliteiten in scenario's met hoge traagheidskrachten worden verlicht.

Het derde focusgebied van ons onderzoek is gericht op het efficiënt visualiseren van op deeltjes gebaseerde vloeistofsimulaties voor real-time toepassingen. We introduceren een schermruimte-renderingbenadering die anisotrope transformaties toepast op puntsprites, die verlengd worden langs geoptimaliseerde assen om vloeiendere vloeistofoppervlakken te bereiken. Deze techniek maakt gebruik van een gewogen hoofdcomponentenanalyse van deeltjesdistributies om optimale rekkingsrichtingen te bepalen. Om de visuele getrouwheid verder te verhogen, integreren we deze anisotrope punt-sprite-behandeling met geavanceerde schermruimtefilters, zoals kromte stroming en smalbereikfilters, om de diepte-informatieverwerking te verfijnen. Deze fusie van technieken verbetert de duidelijkheid en stabiliteit van vloeistofgrenzen aanzienlijk, waardoor we dichter bij hoogwaardige, real-time vloeistofvisualisatie komen.

PUBLICATIONS

This thesis is the result of the following publications:

- X. Wang*, **Y. Xu***, S. Liu, B. Ren, J. Kosinka, A. C. Telea, J. Wang, C. Song, J. Chang, C. Li, J. J. Zhang, and X. Ban. Physics-based fluid simulation in Computer Graphics: Survey, research trends, and challenges. *Computational Visual Media*. Springer, 2024. doi: 10.1007/s41095-023-0368-y

- **Y. Xu**, C. Song, X. Wang, X. Ban, J. Wang, Y. Zhang, and J. Chang. Spatial adaptivity with boundary refinement for smoothed particle hydrodynamics fluid simulation. *Computer Animation and Virtual Worlds*, Volume 34, Issue 5. Wiley, 2023. doi: 10.1002/cav.2136

- **Y. Xu**, X. Wang, J. Wang, C. Song, T. Wang, Y. Zhang, J. Chang, J. J. Zhang, J. Kosinka, A. C. Telea, and X. Ban. An Implicitly Stable Mixture Model for Dynamic Multi-fluid Simulations. In *SIGGRAPH Asia 2023 Conference Paper*. ACM, 2023. doi: 10.1145/3610548.3618215

- **Y. Xu**, Y. Xu, Y. Xiong, D. Yin, X. Ban, X. Wang, J. Chang, and J. J. Zhang. Anisotropic screen space rendering for particle-based fluid simulation. *Computers & Graphics*, Volume 110, 118-124. Elsevier, 2023. doi: 10.1016/j.cag.2022.12.007

* Equal contribution

CONTENTS

NOTATIONS OF PHYSICAL QUANTITIES xvii

1 INTRODUCTION 1
- 1.1 Challenges within the Trilemma 2
- 1.2 Contents and Structure 4

2 SURVEY ON FLUID SIMULATION 7
- 2.1 Fluid Simulation Overview 8
  - 2.1.1 Fluid Mechanics 8
  - 2.1.2 Navier–Stokes Equations 9
  - 2.1.3 Simulation Strategies 10
- 2.2 Advancements in Fluid Simulation 15
  - 2.2.1 Survey Structure 15
  - 2.2.2 Classification of Topics 16
- 2.3 Advanced Computational Approaches 19
  - 2.3.1 Adaptive Solutions 20
  - 2.3.2 Parallelization 27
  - 2.3.3 Data-driven Approaches 29
- 2.4 Multi-material Fluid Coupling 32
  - 2.4.1 Meshless Methods 32
  - 2.4.2 Mesh-based Methods 35
  - 2.4.3 Coupling with Complex Boundaries 37
- 2.5 Multiphase Liquids 40
  - 2.5.1 Non-mixing Fluids 40
  - 2.5.2 Mixing Fluids 42
- 2.6 Gas–liquid Interfaces 45
  - 2.6.1 Free Surface Fluids 46
  - 2.6.2 Bubbles, Foam, and Glugging 48
  - 2.6.3 Spray and Splashing 50
- 2.7 Fine Detail Enhancement 51
  - 2.7.1 Reduced-dimensional Simulation on the Fluid Surface Only 51
  - 2.7.2 Dynamical Methods for Reducing Numerical Dissipation 55
  - 2.7.3 Data-driven Methods for Detail Enhancement 58
- 2.8 Fluid Control 59
  - 2.8.1 Scenario Editing 59
  - 2.8.2 Artificial Effects 61
  - 2.8.3 Media-directed Formation 62
- 2.9 Special Fluids 63

2.9.1 Highly Viscous Fluids 63
2.9.2 Ferrofluids 66
2.9.3 Thin Films 69
2.10 Conclusion and Discussion 72

3 SPATIAL ADAPTIVITY WITH BOUNDARY REFINEMENT FOR SPH FLUID SIMULATION 75
3.1 Introduction 76
3.2 Preliminaries 78
3.2.1 Theory of SPH 78
3.2.2 SPH Approximation 79
3.2.3 Explicit Form of Solving Navier-Stokes Equations with SPH 81
3.3 Split-merge-redistribute Mechanism 82
3.4 Semi-analytic Boundary Handling 85
3.5 Adaptive Boundary Coupling 86
3.5.1 Adaptive Boundary Mechanism 86
3.5.2 Wake Flow Preservation by Delaying Merge 87
3.6 Results 88
3.6.1 Efficiency Comparison 88
3.6.2 Evaluation of Complex Coupling Effects 91
3.7 Discussion and Conclusion 94
3.7.1 Summary 94
3.7.2 Limitations 94

4 IMPLICITLY STABLE MIXTURE MODEL FOR DYNAMIC MULTI-FLUID SIMULATIONS 99
4.1 Introduction 100
4.2 Preliminaries 101
4.2.1 Volume Fraction Scheme 101
4.2.2 Governing Equations of the Mixture Model 103
4.3 Volume Flux Free SPH 104
4.4 Implicit Mixture Model for Multiphase Interactions 108
4.4.1 Interphase Momentum 109
4.4.2 Phase-mixture Momentum Mapping 111
4.4.3 Mixture Viscosity Model 113
4.4.4 Implementation 113
4.5 Results 115
4.5.1 Performance Analysis 115
4.5.2 Comparisons of Effects 122
4.5.3 Effectiveness under Complex Scenarios 122
4.6 Discussion and Conclusion 124
4.6.1 Summary 124

4.6.2 Limitations 124

5 ANISOTROPIC SCREEN SPACE RENDERING FOR PARTICLE-BASED FLUID SIMULATION 127
5.1 Introduction 128
5.2 Real-time Screen Space Fluid Rendering 129
5.2.1 OpenGL Transformation Pipeline 131
5.2.2 Screen Space Rendering 132
5.3 Anisotropic Transformation of Point Sprites for Fluid Particles 134
5.3.1 Tracing Surface Using Smoothing Kernels 135
5.3.2 Deriving the Anisotropy Matrix 135
5.3.3 Transforming Sphere Particles onto the Screen Space 138
5.4 Results 140
5.4.1 Anisotropic Processing Results 140
5.4.2 Combination with Popular Smoothing Filters 141
5.4.3 Performance Analysis of the Anisotropic Transformation 144
5.5 Discussion and Conclusion 145
5.5.1 Summary 145
5.5.2 Limitations 145

6 CONCLUSION 147
6.1 Contributions of This Thesis 147
6.1.1 A Survey on Fluid Simulation 147
6.1.2 An Adaptivity Mechanism with Boundary Refinement for SPH 148
6.1.3 An Implicitly Stable Mixture Model for Dynamic Multi-fluid Simulations 148
6.1.4 An Anisotropic Screen Space Rendering Scheme 149
6.2 Addressing Our Research Questions 149
6.3 Future Work 152
6.3.1 Integration of Adaptive Mechanisms and Multiphase Simulations 152
6.3.2 Enhancing Screen Space Rendering for Multiphase Fluids 152
6.3.3 Unified Framework Development 153

BIBLIOGRAPHY 155

ACKNOWLEDGMENTS 185

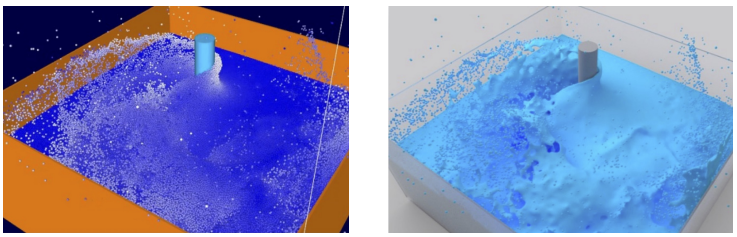
SHORT RÉSUMÉ 187

Variable	Type	Meaning	Unit
A	$*$	any physical property	$[*]$
t	scalar	time	$[s]$
m	scalar	mass	$[kg]$
V	scalar	volume	$[m^3]$
p	scalar	pressure	$[Pa]$
$\mathcal{E}$	scalar	energy	$[J]$
$\dot{\epsilon}$	scalar	shear rate (Herschel-Bulkley model)	$[s^{-1}]$
ρ	scalar	density	$[kg \cdot m^{-3}]$
ψ	scalar	stream function	$[m^2 \cdot s^{-1}]$
μ	scalar	dynamic viscosity coefficient	$[Pa \cdot s]$
γ	scalar	surface tension coefficient	$[N \cdot m^{-1}]$
$\mathbf{x}$	vector	position	$[m]$
$\mathbf{u}$	vector	velocity	$[m \cdot s^{-1}]$
$\boldsymbol{\omega}$	vector	angular velocity	$[rad \cdot s^{-1}]$
$\mathbf{f}$	vector	force field	$[N \cdot m^{-3}]$
$\mathbf{g}$	vector	gravitational acceleration	$[m \cdot s^{-2}]$
$\mathbf{s}$	vector	momentum source	$[N \cdot m^{-3}]$
$\mathbf{q}$	vector	diffusion flux	$[* \cdot m^{-2} \cdot s^{-1}]$
$\mathbf{F}$	vector	force	$[N]$
$\mathbf{B}$	vector	magnetic flux density	$[T]$
$\mathbf{H}$	vector	magnetic field intensity	$[A \cdot m^{-1}]$
$\mathbf{J}$	vector	free current density	$[A \cdot m^{-2}]$
$\mathbf{D}$	vector	electric displacement field	$[C \cdot m^{-2}]$
$\mathbf{K}$	vector	magnetization field	$[A \cdot m^{-1}]$
$\boldsymbol{\sigma}$	matrix	Cauchy stress tensor	$[N \cdot m^{-2}]$
$\boldsymbol{\tau}$	matrix	viscous stress tensor	$[N \cdot m^{-2}]$
$\mathbf{T}$	matrix	diffusion tensor	$[N \cdot m^{-2}]$
$\mathbf{E}$	matrix	strain rate tensor	$[s^{-1}]$

1 INTRODUCTION

With a pivotal role in the domain of computer graphics, physics-based simulation techniques serve as the cornerstone for infusing realism into the digital realm. Simulating fluid dynamics is particularly challenging due to the complex and unpredictable nature of fluids, as shown in Figure 1.1, which makes them one of the most difficult elements to replicate accurately in simulations. The overarching goal of fluid simulation research is to find an optimal balance among three critical aspects: efficiency, accuracy, and versatility. As shown in Figure 1.2, these elements form a trilemma, often illustrated by the metaphor of navigating all three sides of a Penrose triangle simultaneously – an inherently impossible task.

Given these challenges, this thesis strategically focuses on optimizing two of these factors at any given time. This targeted approach allows for the creation of simulations that are either more realistic and computationally efficient, or versatile while maintaining a high degree of accuracy. By adopting this method, the research advances the development of sophisticated and practical fluid simulation techniques, thereby expanding the potential of digital environments.



(a) Particle state (b) Rendered result

Figure 1.1: Animation with physics-based fluid simulation using Smoothed Particle Hydrodynamics. Particles are individually tracked and interact based on local dynamic properties to compute accelerations for each particle at every time step (a). The surface of the fluid is traced and reconstructed from particle data, and the surface mesh is then rendered using ray-tracing techniques to produce a realistic visual representation of the fluid dynamics (b).

From a classical physics standpoint, the behavior of fluids is governed by the Navier–Stokes equations. These fundamental equations explain that the movement of tiny fluid parcels is influenced by factors such as pressure, viscosity, and external forces, leading to a nonlinear system of

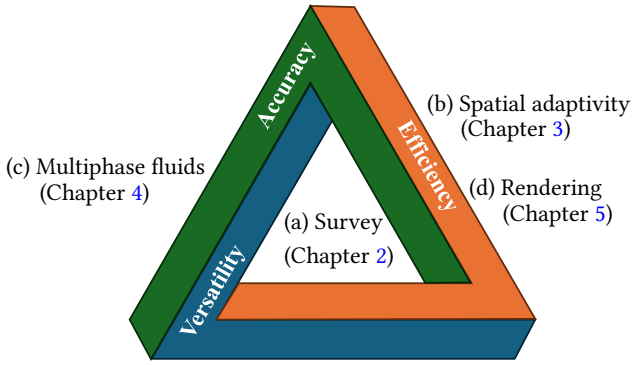


Figure 1.2: The Penrose triangle in this diagram symbolizes the fluid simulation trilemma—achieving optimal efficiency, accuracy, and versatility is inherently challenging, as one cannot address all three simultaneously. Each side of the triangle represents one of the three main areas of focus in this thesis: (b) spatial adaptivity on the Efficiency-Accuracy edge, (c) multiphase fluids on the Versatility-Accuracy edge, and (d) rendering on the Efficiency-Accuracy edge. Additionally, (a) a comprehensive survey of advancements in fluid simulation over the past decade is presented in Chapter 2, providing foundational insights that support the innovations in each area.

behavior. To manage this complexity, computational methods involve breaking down both space and time into discrete units. This process reduces the complex governing equations into linear problems that are easier to handle. The motion of these discretized fluid elements is then reconstructed by calculating the acceleration due to these forces and integrating this over time, which allows for the detailed simulation of fluid dynamics. To translate these simulated quantities, like position and density, into visually engaging animations that mimic real-world fluid behavior, an advanced rendering pipeline is utilized to map numerical data into its graphical counterpart. However, this multi-step process introduces several challenges, particularly trade-offs between efficiency and accuracy, and between accuracy and the complexity of the simulations, each vying for priority in the quest for realistic fluid simulations.

1.1 CHALLENGES WITHIN THE TRILEMMA

To enhance the quality of the simulation, there is a preference to discretize the space as finely as possible. However, in a 3D environment, the computational demand for a single iteration increases cubically with the fineness of the spatial discretization. Furthermore, the time step required for accurate simulations decreases linearly with finer spatial discretization, leading to an overall quartic increase in computational cost for 3D fluid simulations. This trade-off highlights the inherent difficulty

in balancing the desire for high-quality simulations with the practical limitations of computational resources. Moreover, the increase in spatial discretization also demands a corresponding cubic increase in working memory, exacerbating the challenge by introducing more pronounced cache-hit issues, particularly for Lagrangian simulation methods, further complicating the balance between simulation quality and computational feasibility. Consequently, we face the challenge of balancing efficiency and accuracy for simulation. In light of these considerations, we aim to investigate:

RQ1: *How to enhance simulation accuracy without incurring prohibitive computational cost?*

As the pursuit of increasingly complex and natural effects grows, the straightforward application of the Navier–Stokes equations falls short. The incorporation of additional governing equations to manage surface behaviors, interactions with other objects, and the dynamic interplay between different fluids has become essential, complicating the discretization framework and pushing the limits of existing simulation capabilities. In scenarios where multiple fluids coexist, the traditional Navier–Stokes framework must be expanded to a more comprehensive mixture model. This model includes considerations for volume fractions and accounts for varying physical properties, such as the densities of each fluid phase and interaction-specific attributes like diffusivity. These enhancements, while enriching the simulation's complexity, necessitate greater approximations within the numerical model to manage the computational load. Unfortunately, this can compromise the accuracy of the simulations, leading to less realistic outcomes and a more fragile solver, thereby constraining the range of phenomena that can be effectively simulated. This complexity leads us to our second research question:

RQ2: *How to accommodate complex effects in simulations without significantly compromising accuracy and stability?*

Visualizing the results of simulations is a crucial step in computer animation, essential for transforming numerical data into high-quality visual content. This typically involves reconstructing explicit meshes from the numerical flow fields, which are first converted into distance fields and then processed using algorithms like marching cubes to create detailed meshes. Calculating the normals of these meshes is a subsequent step necessary for lighting and rendering processes. However, this visualization pipeline can become particularly unsuited in real-time, large-scale applications where the simulation detail is high. For particle-based Lagrangian simulations, additional complexities arise due to the need for background grids to track particle movement and map their values, which can be computationally intensive. Screen-space rendering has emerged as a popular alternative for its efficiency, bypassing

the need for explicit mesh generation and enabling real-time visualization even in extensive scenarios. However, this method comes with its own set of trade-offs, notably in the quality of the visual output. Issues such as bumps and gaps near surfaces and irregular artifacts at sharp edges are common due to the discrete nature of the particles. To address these challenges, ongoing efforts are dedicated to enhancing visual quality and reducing artifacts, aiming to strike a balance between efficiency and fidelity in the visualization of particle-based fluid simulations. Addressing these visualization challenges while maintaining efficiency leads us to our final inquiry:

RQ3: *How to refine visualization techniques for enhanced accuracy in an efficient manner?*

1.2 CONTENTS AND STRUCTURE

This thesis is organized as follows to address the research questions posed:

Chapter 2 provides a comprehensive survey of fluid simulation in computer graphics. This survey includes a brief history of fluid simulation development in the computer graphics field, a review of the physical principles underlying these effects, and an analysis of various simulation schemes. It offers an in-depth examination of recent trends and emerging challenges over the past decade.

In Chapter 3, we focus on the balance between simulation efficiency and accuracy in detail by introducing a spatially adaptive scheme for Smoothed Particle Hydrodynamics (SPH). This chapter begins with a further detailed introduction to the numerical computation of the SPH approach to ensure readers have a solid foundation for understanding subsequent discussions. It then explores a split-merge scheme for SPH particles and presents our novel approach to applying this scheme to enhance simulation details, particularly those involving fluid-solid interactions, thus achieving effective and accurate simulation results.

Chapter 4 proposes an implicit mixture model for SPH to simulate the coexistence of multiple fluid phases and their interactions while maintaining the accuracy and stability of SPH solvers. We start with the theoretical background on volume fraction schemes that transform the single-phase Navier-Stokes equations into a mixture model. This is followed by the introduction of a volume flux-free scheme that enables the implicit SPH solver to handle fluid couplings with high-density ratios. The chapter concludes with the application of this scheme to construct the implicit mixture model.

In Chapter 5, we address the visualization aspect of the simulation-visualization pipeline. The focus is on enhancing the quality of the highly efficient screen space rendering technique. We propose the integration of an anisotropic smoothing kernel computation, commonly

used in the marching cubes algorithm, with smoothing filters to minimize artifacts such as glitching edges and bumpy surfaces.

The thesis concludes with Chapter 6, where we discuss the contributions of this research and potential improvements that could further enhance performance. We also outline future work that can be pursued to address the quest for computationally efficient realism in fluid simulations based on our contributions.

2 SURVEY ON FLUID SIMULATION

Physics-based fluid simulation has played an increasingly important role in the computer graphics community. Recent methods in this area have greatly improved the generation of complex visual effects and their computational efficiency. Novel techniques have emerged to deal with complex boundaries, multiphase fluids, gas-liquid interfaces, and fine details. The joint use of machine learning, image processing, and fluid control technologies has brought many interesting and novel research perspectives. In this survey, we provide an introduction to theoretical concepts underpinning physics-based fluid simulation and their practical implementation, with the aim for it to serve as a guide for readers of this thesis, regardless of newcomers and seasoned researchers, to explore the field of physics-based fluid simulation, with a focus on developments in the last decade. Driven by the distribution of recent publications in the field, we structure our survey to cover physical background; discretization approaches; computational methods that address scalability; fluid interactions with other materials and interfaces; and methods for expressive aspects of surface detail and control[1].

In this chapter, we embark on a journey through the evolution of fluid simulation within the realm of computer graphics. Starting with an exploration of the foundational physical principles and simulation methodologies in Section 2.1, we aim to establish a comprehensive knowledge base that will not only facilitate the understanding of subsequent sections but also enrich the context for the contributions made in this thesis. Progressing to Section 2.2, we conduct a thorough analysis of the advancements in fluid simulation technology that have unfolded over the last decade. We spotlight the key methods and important discoveries that have contributed to advancing the field, setting the stage for an in-depth discussion of these pioneering topics from Section 2.3 through to Section 2.9. We wrap up this chapter in Section 2.10 by synthesizing these developments and engaging in a forward-looking discussion on the potential challenges and opportunities that lie on the horizon for fluid simulation in computer graphics.

1 Parts of this chapter were published in: X. Wang, Y. Xu, S. Liu, B. Ren, J. Kosinka, A. C. Telea, J. Wang, C. Song, J. Chang, C. Li, J. J. Zhang, and X. Ban. Physics-based fluid simulation in Computer Graphics: Survey, research trends, and challenges. *Computational Visual Media*. Springer, 2024. doi: 10.1007/s41095-023-0368-y

2.1 FLUID SIMULATION OVERVIEW

The development of fluid simulation in computer graphics is deeply rooted in the history of physics. From the 17th to the early 19th centuries, scientists such as Sir Isaac Newton and Claude-Louis Navier contributed significantly to the understanding of fluid mechanics, paving the way for the Navier–Stokes equations. These equations, which govern fluid motion, form the foundation of modern fluid simulation algorithms.

This section offers a basic introduction to fluid simulation and provides background knowledge for the remainder of the survey. For a more comprehensive understanding of fluid simulation, we refer to Bridson's book [44]. For more specific knowledge about Lagrangian-based smoothed particle hydrodynamics and material point methods, we refer to the surveys of Koschier *et al.* [166] and Jiang *et al.* [154], respectively. Readers less familiar with this field are highly encouraged to review this section before proceeding. We first introduce relevant physical principles behind fluid simulation, such as the continuum hypothesis (Section 2.1.1) and Navier–Stokes equations (Section 2.1.2). We next present the early development of this area (Section 2.1.3), including a brief overview of the ideas behind different discretization strategies.

2.1.1 *Fluid Mechanics*

Matter in nature is built up of atoms and molecules that are discrete and separated by space. Simulating fluid at the microscopic level to describe macroscopic phenomena is only possible on supercomputers with weeks, if not months, of computing time. The study on computer graphics pursues a balance between efficiency and fidelity. For this, fluid mechanics based on a continuum hypothesis is the level on which physical properties are modeled.

Fluid mechanics models an object with matter continuously distributed over its body, an approximation called the *continuum hypothesis*. This means that any infinitely small volume element in the fluid is seen as a continuous medium, also called a *fluid parcel*. As Landau and Lifshitz stated [169], a fluid parcel is "very small compared with the volume of the body under consideration, but large compared with the distances between molecules."

In fluid mechanics, the *continuity equation* describes the transportation of physical properties in space and time as follows:

$$\frac{\partial A\left(\mathbf{x}, t\right)}{\partial t} + \nabla \cdot \left(A\left(\mathbf{x}, t\right) \mathbf{u}\left(\mathbf{x}, t\right)\right) = s\left(\mathbf{x}, t\right), \tag{2.1}$$

where A can be an arbitrary scalar, vector, or tensor physical property, $\mathbf{u}$ is the velocity, and s is the source term for A, all described at time t and location $\mathbf{x}$. Equation (2.1) states that the change rate $\frac{\partial A}{\partial t}$ of any

physical property at a fixed position depends on the variation brought by the flux of $A\mathbf{u}$ and source term s.

Lagrangian and Eulerian viewpoints. Considering the physical attribute A in Equation (2.1), a flow field can be described from a Lagrangian or Eulerian viewpoint as follows.

The *Eulerian viewpoint* studies the physical field using fixed positions typically located at the vertices of a grid. The change rate of the physical value A at a given position $\mathbf{x}$ is the $\partial A(\mathbf{x}, t)/\partial t$ term in Equation (2.1), which comes from both the flux and source terms. While intuitive, this does not explicitly express the motion of the fluid parcel in the continuum hypothesis, as parcels constantly travel through fixed locations at all times.

In contrast, the *Lagrangian viewpoint* studies the change rate of physical attributes with respect to the fluid parcel by recasting Equation (2.1) as

$$\underbrace{\frac{\partial A(\mathbf{x}, t)}{\partial t} + (\mathbf{u}(\mathbf{x}, t) \cdot \nabla) A(\mathbf{x}, t)}_{\frac{DA(\mathbf{x},t)}{Dt}} + A(\mathbf{x}, t) \nabla \cdot (\mathbf{u}(\mathbf{x}, t)) = s(\mathbf{x}, t)$$
$$\frac{DA_i(t)}{Dt} + A_i(t) \nabla \cdot \mathbf{u}_i(t) = s_i(t), \tag{2.2}$$

where $D(\cdot)/Dt$, the so-called *material derivative*, is the change rate of A within a fluid parcel. In Equation (2.2), $\mathbf{u}$ and s are the velocity and source term of a specific fluid parcel, respectively. Hence, all positions $\mathbf{x}$ can be substituted with the parcel identifier i. For brevity, hereafter, we omit the explicit mention of $(\mathbf{x}, t)$, (t), and i unless required by the context.

2.1.2 *Navier–Stokes Equations*

Numerous methods for calculating fluid motion have been developed, spanning from Lagrangian to Eulerian perspectives. However, the underlying physical principles for almost all of these approaches are rooted in the *Navier–Stokes equations*, which govern the dynamics of fluid flow and serve as a fundamental foundation for fluid simulations. Thus, we describe these briefly next.

Mass conservation. In a closed system, fluid mass is conserved over time. This principle is represented by the continuity equation (Equation (2.1)). Letting A be the fluid density ρ and setting $s \equiv 0$, Equation (2.1) can be rewritten as

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{u} = 0. \tag{2.3}$$

For the case of *incompressible flow*, the density within the flow is conserved, *i.e.*, $D\rho/Dt = 0$. This condition further implies a divergence-free velocity field, as expressed, by

$$\nabla \cdot \mathbf{u} = 0. \tag{2.4}$$

Navier–Stokes momentum Equation. To further describe the motion of incompressible fluid flow, one can analyze the momentum of each fluid parcel. By introducing a momentum term $\rho\mathbf{u}$ in Equation (2.1) and next using Equation (2.3), we obtain

$$\frac{\partial \rho\mathbf{u}}{\partial t} + \nabla \cdot (\rho\mathbf{u} \otimes \mathbf{u}) = \rho\frac{D\mathbf{u}}{Dt} = \mathbf{s}, \tag{2.5}$$

where $\mathbf{s}$ is the momentum source altering the speed of each fluid parcel, and $\otimes$ represents the outer product operation. Following this, a basic form of the Navier–Stokes momentum equation for viscous compressible flow further specifies $\mathbf{s}$ into three separate terms as

$$\rho\frac{D\mathbf{u}}{Dt} = -\nabla p + \mu\nabla^2\mathbf{u} + \rho\mathbf{g}, \tag{2.6}$$

where p is pressure, $\mathbf{g}$ is gravitational acceleration, and μ is the dynamic viscosity coefficient describing how viscous a fluid is. Equation (2.6) states that the velocity change rate for a fluid parcel is affected by three force terms: pressure ($-\nabla p$), viscosity ($\mu\nabla^2\mathbf{u}$), and gravity ($\rho\mathbf{g}$).

2.1.3 *Simulation Strategies*

2.1.3.1 *Early developments*

As computer technology advanced in the 20th century, numerical methods became popular for solving partial differential equations, including the Navier–Stokes equations. With the advent of powerful computer hardware and software, computer graphics began to incorporate these physics-based algorithms, enabling increasingly realistic fluid simulations.

Dating back to the 1970s, William T. Reeves, a member of Lucasfilm's Computer Division, Computer Graphics Group, pioneered the development of particle systems [236, 261]. These systems enabled the realistic depiction of elements such as smoke and fire in films, as seen in "Star Trek II: The Wrath of Khan." This breakthrough laid the foundation for early fluid simulation techniques in computer graphics. In the 1990s, physics-based fluid simulation began to gain traction. Wejchert and Haumann [307] used a simplified version of the Navier–Stokes equations to animate irrotational, incompressible linearized fluid flow, providing a physics-based foundation for their fluid animations. Subsequently, Stam and Fiume [268] incorporated the complete Navier–Stokes equations to create turbulent wind effects.

On the *Lagrangian* side, Desbrun and Cani [76] introduced *Smoothed Particle Hydrodynamics (SPH)* to the computer graphics field for simulating highly deformable bodies. On the *Eulerian* side, Foster and Metaxas [94] used the Navier–Stokes equations on fixed grids to simulate fluid motion. The study of fluid simulation reached a significant milestone at the end of the 20th century with Stam's Stable Fluids method [267]. This finally made stable, three-dimensional, physics-based fluid simulation an attainable goal, producing realistic fluid effects. This was the first unconditionally stable method for fluid simulation and introduced the concept of *semi-Lagrangian advection*. Additionally, this was one of the earliest works to apply the idea of *hybrid* simulation in the field.

Hybrid methods in fluid simulation merge the strengths of both Lagrangian and Eulerian approaches, yielding more versatile and robust systems. Two foundational principles that underpin hybrid fluid simulation are Harlow's [117] *Particle-in-cell (PIC)* method and the refined *Fluid Implicit Particle (FLIP)* method of Brackbill and Ruppel [41]. These techniques have contributed significantly to the widespread success and adoption of hybrid fluid simulation in the 21st century. The field also saw a major advancement when Zhu and Bridson [355] applied the FLIP method to incompressible flow simulation. This moved hybrid fluid simulation to new heights as it enabled the exploration of complex fluid dynamics with enhanced precision and stability. The continuous evolution of hybrid fluid simulation techniques has had a profound impact on computer graphics, facilitating the creation of realistic and visually stunning effects.

2.1.3.2 *Discretization strategies*

As fluid simulations in computer graphics have evolved since the early development in the 20th century, the field has branched out in three distinct directions: Eulerian, Lagrangian, and hybrid schemes. Each of these approaches offers unique advantages and challenges, contributing to the comprehensive understanding of fluid dynamics in computer graphics.

Eulerian schemes. These simulation methods use the Eulerian viewpoint introduced in Section 2.1.1, *i.e.*, compute property values at fixed points in the simulation domain. For this, the domain is typically divided into evenly-distributed cells. In a traditional *collocated grid* structure (Figure 2.1a), all physical values are evaluated at the center of each cell. To derive a continuous flow field with values at arbitrary positions, *e.g.*, the gray dot in Figure 2.1c, one can use a weighted interpolation of neighboring cell values. The *staggered grid* (Figure 2.1b) stores physical values at cell edges and centers separately. Compared with collocated grids, staggered grids are currently more popular for simulating

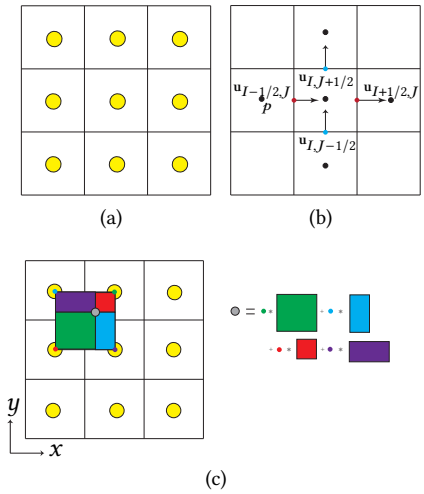


Figure 2.1: Schematic diagram of Eulerian grids. (a) A Collocated grid where physical quantities are stored in the cell centers (yellow points). (b) A Staggered grid where different variables are stored at different locations; in this example, pressure is stored at the cell centers (black points), while velocity is split into its two Cartesian components and stored at the centers of the vertical and horizontal cell edges (red and blue points). Subscripts I and J denote spatial indices. (c) Using bilinear interpolation to obtain the value of a physical quantity at any position.

incompressible fluids given their higher stability. It is noteworthy that staggered grids are related to the *Marker-and-cell (MAC)* method [119], which was used in the early days of computational fluid dynamics to solve incompressible flow problems.

Lagrangian schemes. In the Lagrangian framework, domain discretization is based on a set of particles moving with the fluid flow, each approximating the physical values of a fluid parcel. Hence, Lagrangian schemes conserve mass by construction. Since particle locations can be more flexibly distributed over the computational domain compared to a fixed grid, Lagrangian schemes are particularly effective at modeling complex free surface details.

Currently, SPH is one of the most popular Lagrangian methods for fluid simulation, with origins in the works by Lucy [192] and Gingold and Monaghan [105]. SPH has evolved significantly over time, with various advancements and improvements.

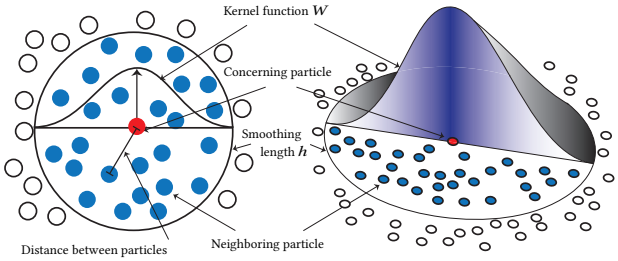


Figure 2.2: Schematic diagram of Lagrangian-based smoothed particle hydrodynamics.

Figure 2.2 shows how SPH performs interpolation, where the physical value A at location $\mathbf{x}_i$ of particle i is computed using a *smoothing kernel* W as follows:

$$A(\mathbf{x}_i) = \sum_j V_j A_j W\left(\left\|\mathbf{x}_i - \mathbf{x}_j\right\|, h\right), \tag{2.7}$$

where h is called the smoothing length, V is the volume of (the parcel of) each particle, and j indicates all particles closer to i than the distance h. The selection of an appropriate smoothing kernel is crucial and must meet several key criteria, including:

- **Compact Support**: The kernel function is zero beyond a specified distance from the particle, ensuring that each particle interacts only with its immediate neighbors. This property is essential for maintaining computational efficiency and feasibility.
- **Normalization**: The kernel must be normalized such that its integral over all space equals one. This normalization is crucial for conserving mass, momentum, and energy during interpolation processes.
- **Smoothness**: To accurately compute gradients and Laplacians, the kernel should be sufficiently smooth. This smoothness is necessary for minimizing numerical errors and ensuring the stability of the computations.

To compute higher-order quantities, *e.g.*, pressure gradients, one can simply replace the kernel function W in Equation (2.7) with its higher-order counterpart. Detailed numerical implementations for SPH are presented in Section 3.2.

Initially, the *Weakly-compressible SPH (WCSPH)* [26] approach was introduced, where pressure computation was performed explicitly. Later, the *Predictive-corrective Incompressible SPH (PCISPH)* [264] method was proposed, which introduced a prediction-correction scheme for implicit pressure computation. This technique improved the stability and accuracy of fluid simulations by enforcing incompressibility more effectively.

Further developments led to the introduction of *Implicit Incompressible SPH (IISPH)* [139], which provided a more strictly incompressible simulation with increased computational efficiency. Most recently, the *Divergence-free SPH (DFSPH)* [28] method has been developed, which further enforces the divergence-free condition within a simulation.

Position-based Dynamics (PBD) is a versatile and efficient simulation method for handling various physical phenomena, including fluids, deformable solids, and cloth. PBD was first introduced by Müller *et al.* [206] as an alternative to traditional force-based dynamics, focusing on the direct manipulation of object positions instead of computing forces and accelerations. In the context of fluid simulation, the *Position-based Fluids (PBF)* method was proposed by Macklin and Müller [194], building upon the principles of PBF and enforcing incompressibility by iteratively adjusting particle positions.

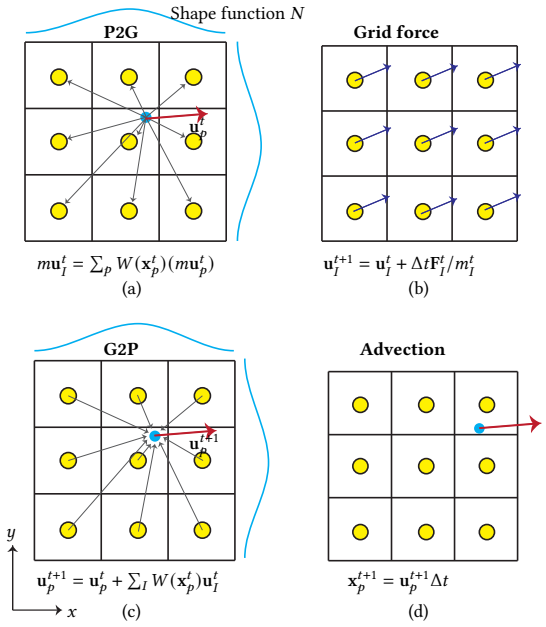


Figure 2.3: Hybrid scheme of the Particle-In-Cell method. This example uses a cell-centered grid, where information is stored at the yellow points. During simulation, momentum and weight (which can be used to obtain the velocity $\mathbf{u}_I^t$ on the grid) are transferred from particles to the cell centers (a). In the next step, forces are applied to grid nodes to compute the new velocity $\mathbf{u}_I^{t+1}$ (b). Finally, the velocity is re-transferred from the grid to particles (c), causing the movement of the particles (d). When particles get their new velocities, the new positions can be easily found by forward Euler integration.

Hybrid schemes. These schemes combine the advantages of Lagrangian and Eulerian schemes by representing the motion of the fluid flow with Lagrangian particles while computing dynamics (forces) on an Eulerian grid.

As Figure 2.3 shows, to combine particles and grids, physical values must be separately mapped from particles to grids (P2G) and from grids to particles (G2P) before and after the dynamic simulation. A so-called shape function, similar to the kernel function W for the SPH method, performs these mapping procedures.

In the original PIC [118], only the momentum term is transferred between P and G. The later proposed FLIP [41] transfers the differential of momentum to obtain better dynamic effects at the cost of stability. The *Material Point Method (MPM)* introduced by Sulsky *et al.* [273] is another extension of the original PIC. It adds a new dimension to fluid simulation by considering the deformation gradient information along with the momentum term, making it suitable for simulating a wide range of materials, including fluids, granular materials, and deformable solids.

Throughout the development of the PIC, FLIP, and MPM methods, these techniques have evolved and merged to form more advanced approaches. The *Affine Particle-in-cell (APIC)* method [153] extends the MPM framework by incorporating affine velocity fields, which reduces numerical dissipation and offers improved stability compared to both PIC and FLIP. The *Polynomial Particle-in-cell (PolyPIC)* [95] method takes the MPM framework one step further by incorporating higher-order polynomial velocity fields, building upon the advancements made by the APIC method. Finally, the *Moving Least Squares Material Point Method(MLS-MPM)* [129] utilizes moving least squares for grid interpolation and differentiation in MPM simulations, further enhancing the accuracy and robustness of the approach.

2.2 ADVANCEMENTS IN FLUID SIMULATION

Fluid simulation is a dynamic field characterized by numerous active research areas. It has seen a continual influx of innovative proposals and solutions. To highlight significant milestones and notable contributions, we conducted a comprehensive survey of the advancements in fluid simulation within the realm of computer graphics over the past decades.

2.2.1 *Survey Structure*

Physics-based fluid simulation is a research area that has been active for many decades, with input from a diverse range of fields, including engineering, physics, mathematics, and computer science. A fluid simulation survey covering the scope of *all* these fields would be too exten-

sive for one thesis. Furthermore, we believe that the interests of typical researchers and practitioners in *computer graphics* focus on a subset of the above aspects, and we structure our survey accordingly as follows.

As main information sources, we selected articles published in *ACM Transactions on Graphics (TOG)*, *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, and *Computer Graphics Forum (CGF)*, which are arguably the three most influential and representative computer graphics journals. As our survey aims to cover recent tendencies, we included all relevant papers from these journals published in the last decade (2012–2022). We further included some papers presented at key graphics conferences like the *ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)*, *ACM SIGGRAPH (SIGGRAPH)*, and *ACM SIGGRAPH Asia (SIGGRAPH Asia)*. We also included earlier papers that have significantly impacted recent research. In total, we collected and further analyzed 327 papers meeting the above criteria.

The origins of physics-based fluid simulation in computer graphics can be traced back to the 1970s with the development of "particle systems" [236]. However, a fully developed, reasonably stable physics-based system for fluid animation was not achieved until the end of the last century [267]. In the decade following, various simulation strategies continued to evolve, focusing on improving stability, accuracy, and efficiency in fluid simulations. As we enter the 2010s, which is the primary focus of our survey, the main research interests in fluid simulation shifted towards addressing *specific effects* that are challenging to achieve using conventional fluid simulation methods. Alongside this shift, advances in machine learning technologies have opened up new ways to integrate neural networks with simulation algorithms, pushing the boundaries of what can be accomplished in fluid simulations. As our survey aims to discuss the current advancements in this area comprehensively, we *classify* the collected papers into seven relevant topics that span the past decade based on our detailed analysis of these papers. We then select a subset of representative papers within each topic and discuss these in greater detail.

2.2.2 *Classification of Topics*

Figure 2.4 shows the seven identified topics at the first level of a hierarchy depicting our survey's structure. Further levels refine these into sub-topics. The seven main topics are as follows:

- **Advanced computational approaches:** (Section 2.3) Methods that aim to enhance the efficiency, accuracy, and scalability of simulations by optimizing resource utilization and reducing computational complexity.

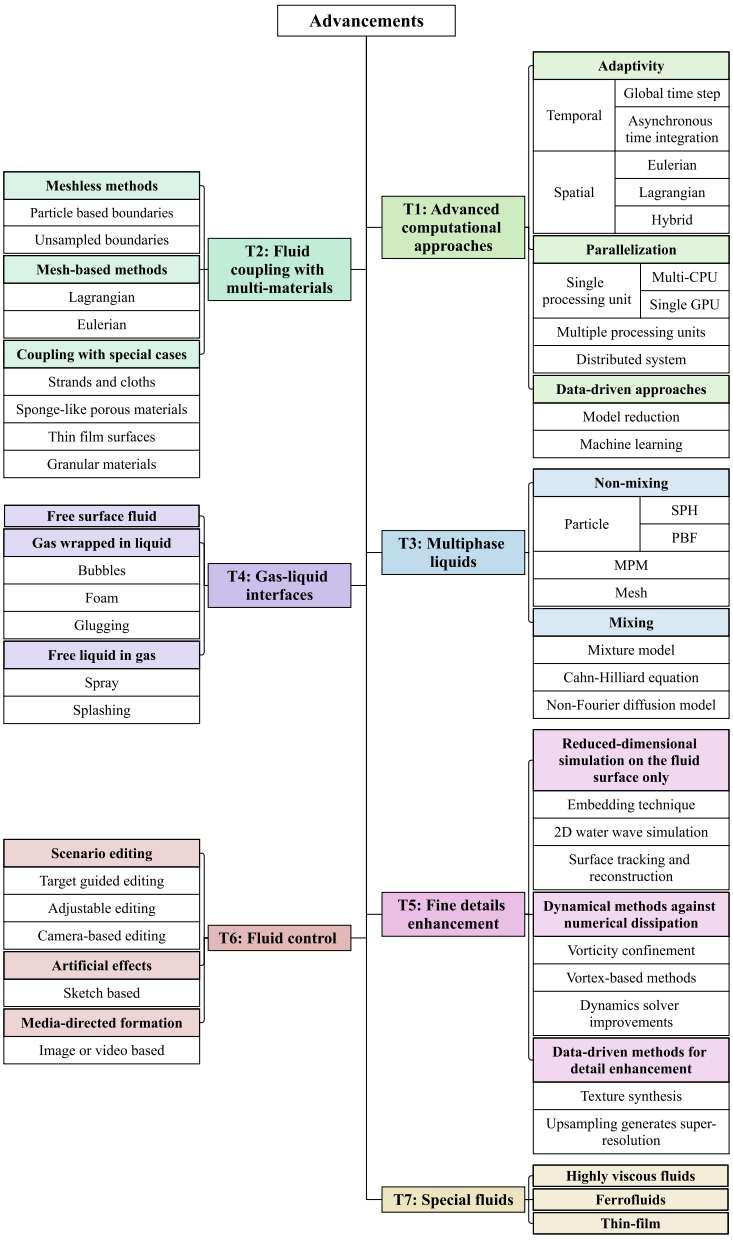


Figure 2.4: Classification of advancements based on the topics.

- **Fluid coupling with multi-materials:** (Section 2.4) Methods that model the interaction between fluid and solid objects of various shapes and textures.
- **Multiphase liquids:** (Section 2.5) Methods for the simulation of liquid–liquid interaction effects of various phases.
- **Gas–liquid interfaces:** (Section 2.6) Methods dealing with scenarios where forces on gas–liquid interfaces dominate the fluid motion.
- **Fine detail enhancement:** (Section 2.7) Methods that concentrate on preserving/enhancing fluid motion on a detailed level.
- **Fluid control:** (Section 2.8) Methods that allow visual designers to control the appearance and style of fluid simulations.
- **Special fluids:** (Section 2.9) Methods that simulate non-conventional fluids, *e.g.*, highly viscous/thin, sensitive to magnetic fields, or targeting materials that are, strictly speaking, not fluids but behave like fluids.

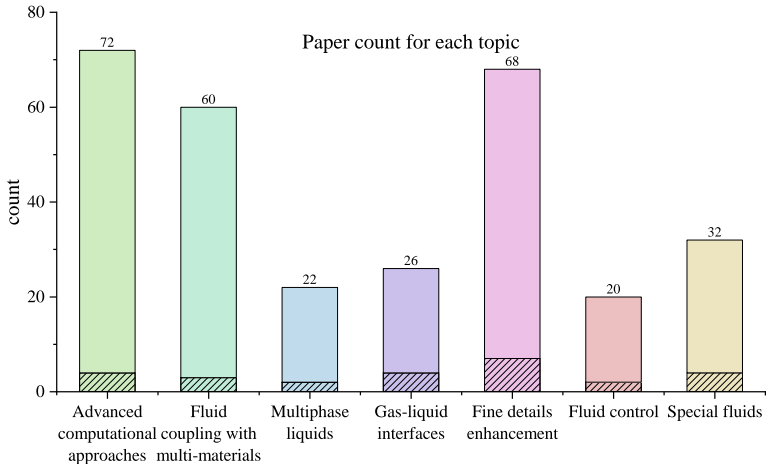


Figure 2.5: Number of studied papers per identified topic. For each topic, the top (pure color) bar segment represents papers published in TOG, CGF, and TVCG. The bottom (shadowed) bar segment represents other key papers considered in our survey.

Our seven-topic classification aims first and foremost to identify salient *trends* in the past decade. As such, the emergence of these topics is based on a significant portion of the found papers that can be grouped within each topic; in other words, the topics reflect a "data-driven organization" of how research on fluids proceeded in the past decade. This is

in contrast with other surveys that group works based on a *predefined, model-driven* taxonomy proposed by their authors.

It is insightful to analyze the distribution of topics in the found papers and how these topics evolve over time. Figure 2.5 shows the numbers of papers found for each of the seven identified topics. We see that while variations exist (some topics being more popular than others), each topic has a significant number of papers, with 4–8 top-tier papers per year on average, thereby supporting the claim that our identified topics are a good way to organize the research.

Figure 2.6 refines the above insight, showing the number of articles per topic and per year. We see that while some topics show an increase in publications (*e.g.,* advanced computational approaches or red curve in Figure 2.6), all identified topics have been "alive" over the past decade – another indication that they are suitable for the organization of our survey.

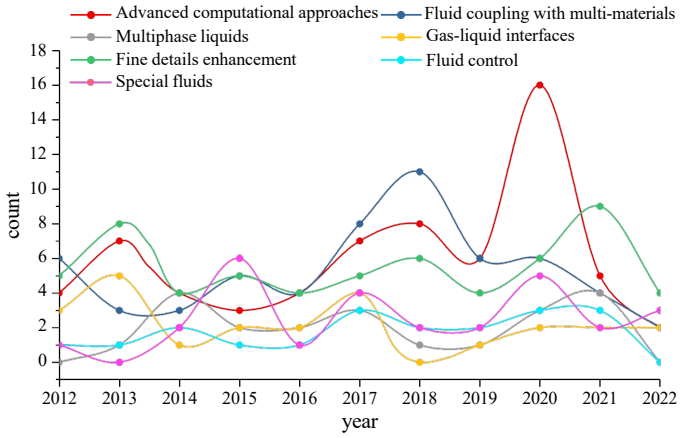


Figure 2.6: Trends in number of papers published per topic over the past decade.

2.3 ADVANCED COMPUTATIONAL APPROACHES

Fluid simulation requires a high discretization resolution to reach high visual quality. However, more discrete particles or denser grids demand more computing resources. This section surveys recent approaches for improving computational efficiency. We organize these into approaches that use adaptive time and/or space sampling (Section 2.3.1), GPU or CPU parallelization (Section 2.3.2), and the more recent data-driven approaches (Section 2.3.3). For a more extensive survey of this area, we refer to the work of Manteaux *et al.* [197].

2.3.1 *Adaptive Solutions*

A stable, sufficiently accurate, and detailed simulation requires adequate temporal and spatial resolution. Time steps must be short enough to ensure stability, and high-resolution grids or dense particles are needed to capture fine details. However, computational cost increases with both spatial and temporal resolution, and an overall high resolution is not always needed. For example, time steps must be small for fast motion but can be longer when the overall movement is slow; high spatial resolution is needed to capture delicate splashes and sprays, but it is less important deep inside the fluid, where such detail is not visible.

As such, adaptivity uses high resolution only at necessary time and space instances and uses low resolution elsewhere to reduce computing costs. Figure 2.7 illustrates this strategy with particles as an example. Adaptive methods can be categorized into temporal and spatial adaptivity. *Temporal* adaptivity dynamically changes the time step, either globally or locally, for different parts of the fluid. *Spatial* adaptivity adjusts the resolution for different fluid regions or changes the method of discretization for a similar effect. These two approaches are described next.

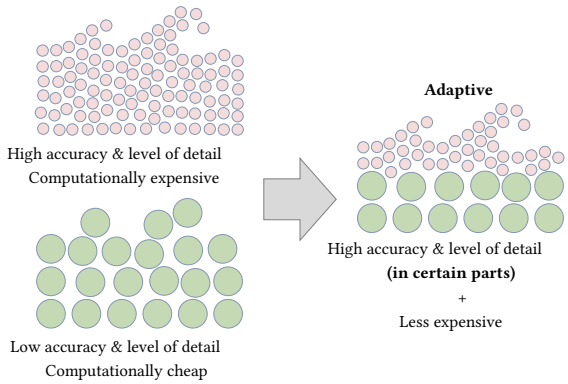


Figure 2.7: Schematic diagram of particle-based adaptivity. Particle-based adaptivity adjusts particle size dynamically to reduce cost and preserve detail simultaneously.

2.3.1.1 *Temporal adaptivity*

Temporal adaptivity adjusts the time step length dynamically. A straightforward strategy is to adapt the time step globally, *i.e.*, use the same time step for the entire simulation domain. The time step size is determined dynamically at each time step under a restriction. For further performance gains, different time steps can be used for different

spatial domain zones, thereby reducing the total number of integration steps needed.

Global time step. The *Courant–Friedrichs–Lewy (CFL)* condition [66] is a well-known method for determining the time step size. Most current simulation methods compute a global time step according to the CFL condition at each time step. Generally, the CFL condition takes the form

$$C \equiv \frac{\|\mathbf{u}_c\|\Delta t}{\Delta x} \leq C_{\max}, \tag{2.8}$$

where $\|\mathbf{u}_c\|$ is the speed of information propagation, Δx is the grid-cell size for Eulerian and hybrid simulations or smoothing length for Lagrangian ones, $C_{\max}$ is a constant based on the size of discrete operators, and C is the CFL or Courant number. In practice, $\|\mathbf{u}_c\|$ typically represents the speed of sound in the material or maximum velocity in the simulation. The time step length Δt is usually chosen so that C is in the range of $[0, 1]$. The choice of the maximum Courant number $C_{\max}$ is generally dictated by the type of simulation algorithm being used, but it should not exceed 1. Methods such as PIC or MPM tend to offer greater flexibility in choosing $C_{\max}$ compared to SPH. Using the same method with an implicit time integration scheme allows for larger $C_{\max}$ values while maintaining simulation stability.

Determining an optimal value for $C_{\max}$ often involves an extensive trial-and-error process tailored to a specific scenario. Sun *et al.* [275] addressed this issue by considering metrics related to the stability of MPM simulations, such as the deformation gradient. By using these metrics, they were able to more effectively identify the performance limits and improve the overall stability of simulations.

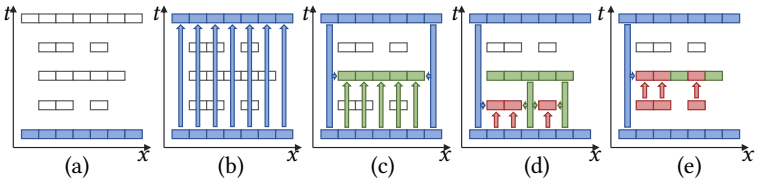
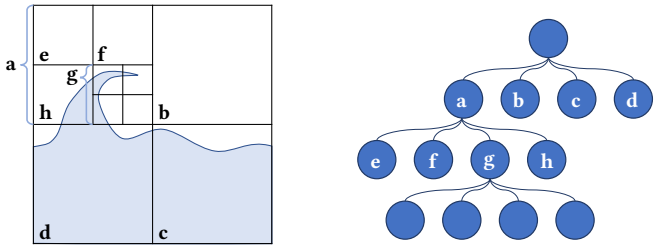


Figure 2.8: Schematic diagram of the asynchronous time integration scheme in Koike *et al.* [164]. Boxes refer to discretized quantities, such as velocity. Color refers to the level of regions (red: 0, green: 1, blue: 2). The entire region (a) is first advanced with the largest time step (b). Next, smaller time steps are used (c, d). If values needed for computation are calculated using a larger time step, they are interpolated to match the current step size. Once the smallest time step is reached, the small time step is applied again and overwrites variables of level 1 (e). This procedure is applied recursively to update all quantities.

Asynchronous time integration. When dealing with scenarios involving both intense waves and calm regions, implementing a global time step restriction can be inefficient and wasteful. To address this, the concept of *regional time stepping* was initially introduced to the SPH method by Goswami *et al.* [111]. This approach subdivides the simulation space into smaller regions, allowing each region to have its own independent time step. Recognizing the grid-based nature of the subdivided regions, Fang *et al.* [82] extended this idea to the MPM method. In their technique, a scheduler determines the order in which blocks are updated, while a buffer block is employed to handle boundaries between blocks with different time steps. This resulted in significant performance improvements, achieving speed-ups of 9.8 times compared to traditional synchronous MPM implementations. Inspired by Fang *et al.* [82], Koike *et al.* [164] proposed an asynchronous time integrator for Eulerian liquid simulation, with an interpolation strategy to deal with boundaries between different-time-step zones and an advection scheme to prevent seams at the boundaries, as is shown in Figure 2.8. While the abovementioned methods effectively enable the time step to be asynchronized for separate regions, they still necessitate synchronization for all regions at simulation time barriers. Reinhardt *et al.* [237] presented a fully asynchronous time integration model for SPH fluid animation, where each particle has an individual time step and is processed using a priority queue.

2.3.1.2 *Spatial adaptivity*

These methods change the spatial resolution or discretization method in different spatial regions to keep fine detail in some regions but use coarser (thus faster to compute) detail in less important regions. Spatial adaptivity methods are heavily dependent on the underlying discretization. We next detail different spatial adaptive approaches for the Eulerian, Lagrangian, and hybrid approaches.



(a) Example of an octree (b) Corresponding data structure

Figure 2.9: Schematic diagram of a quadtree represented in 2D, where each cell has four children. 3D octrees have eight children per cell.

Eulerian grids. Grid-based Eulerian methods use adaptive grid structures to achieve dynamic spatial resolution. However, compared to uniform grids, it is challenging to design a stencil, which refers to the local pattern used to approximate differential operators like the Laplacian, on an adaptive grid for pressure solving. This difficulty arises when trying to attain high-order accuracy and form a symmetric positive-definite linear system that can be efficiently solved on non-symmetric adaptive grids.

The *octree data structure* is one grid adaptivity approach that allows the resolution of axis-aligned structured grids to be changed. As shown in Figure 2.9, each cell is divided into four equal children by cutting it in half along each axis. Octrees have the advantage of regularity, supporting fast discretization, and simple implementation. However, on the transition between different grid levels, octrees have T-junctions, which cause challenging numerical issues.

Losasso *et al.* [190] proposed the first octree-based liquid solver using a set of symmetric differential operators, which enables the Poisson equation to be solved on unrestricted octree grids. In the octree, velocity is stored on cell faces, while pressure is stored at cell centers. The velocity divergence $\nabla \cdot \mathbf{u}$ at cell centers is computed considering all cell faces f as

$$\nabla \cdot \mathbf{u} = \frac{1}{V_c} \sum_f (\mathbf{u}_f \cdot \mathbf{n}_f) S_f, \tag{2.9}$$

where V_c is the cell volume, and $\mathbf{n}_f$, $\mathbf{u}_f$, and S_f are the outward-pointing normal, velocity, and area of face f, respectively. The pressure gradient on each face is computed from the pressure of the two adjacent cells using

$$\frac{\partial p}{\partial x} = \frac{p_2 - p_1}{(\Delta x_1 + \Delta x_2)/2}, \tag{2.10}$$

where Δx denotes the cell size, and subscripts 1 and 2 denote the adjacent cells to that face.

Dynamically adjusted octree grids also present challenges in terms of modifying and accessing data. Implementing such grids is inherently complex as it requires efficiently subdividing or merging cells based on the data's properties while ensuring these operations are error-free and optimized for performance. Frequent adjustments to the grid in response to rapidly changing data can also lead to substantial memory fragmentation. As the size of the dataset increases, the scalability of the octree grid must be addressed. Scalability not only involves handling larger volumes of data but also maintaining performance efficiency and managing increased memory and processing demands.

Setaluri *et al.* [255] proposed a *Sparse Paged Grid (SPGrid)* data structure that constructs the octree as a hierarchy of sparsely populated regular grids instead of a standard pointer-based tree. Goldade *et al.* [110]

recognized the limitations of the first-order accuracy of the velocity field for octrees and applied a variational finite difference discretization method to it, enabling a more efficient viscous simulation. Ando and Batty [9] focused on using octree grids to enhance surface detail exclusively. This approach further reduces implementation complexity while retaining the benefits of octrees. While the particular attention to maintaining data order for efficient computation is advantageous, it also presents a challenge in system design. Shao *et al.* [258] noted this issue and identified an underutilized potential within the regular Cartesian grid structure. They ingeniously integrated the single instruction, multiple data (SIMD) approach with a multigrid structure, aiming to streamline and minimize the required number of multiplications. Their method showed significant speed-ups of 2.0 to 14.6 times compared to contemporary adaptive octree solvers found in commercial software for large-scale simulations.

Several approaches have been inspired by and extended from the octree grid concept. These works aim to improve efficiency and accuracy in various ways. Ferstl *et al.* [90] proposed a hexahedral finite element discretization multigrid solver on adaptive octree grids. By specially treating boundary conditions on the free surface, they achieved second-order accuracy on the surface. Aanjaneya *et al.* [2] focused on enhancing pressure projection on octrees. They used a finite volume power diagram to accurately recover irregular embedded boundaries that cross grids, satisfying both second-order accurate and symmetric positive definite (SPD) conditions. Xiao *et al.* [320] introduced an adaptive staggered-tilted (AST) grid for conducting adaptive fluid simulations on a regular discretization. By adding a tilted grid to an octree structure, they avoided T-junctions and further improved the adaptivity of the simulation.

Some fluid simulation methods employ multiple grids with different resolutions or structures to simulate various parts of the fluid, later compositing these elements together. This approach contrasts with using a single adaptive grid for the entire fluid domain. Gao *et al.* [104] devised a technique that divides the domain into nested partitions with different resolutions, effectively handling multi-resolution fluid behavior. English *et al.* [80] used overlapping Cartesian grids with varying scales and rotations to represent the fluid domain, constructing a local Voronoi diagram for managing pressure projection near grid interfaces. Li *et al.* [177] introduced an adaptive relaxation method for kinetic approaches, enabling fluid sampling at arbitrary overlapping resolutions and providing an efficient representation of fluid behavior across a wide range of scales.

While many adaptive methods that use single or multiple grid structures can disrupt the uniform data structure of the original Cartesian grid, some works have found a balance between maintaining uniformity and introducing adaptivity. Zhu *et al.* [353] used a uniform grid

within a cubic region of interest, extending the grid into the far-field by stretching cells along an axis. This approach retains the benefits of a uniform grid while providing adaptivity in specific areas. Ibayashi *et al.* [138] proposed a technique for dynamically warping uniform grids, combining the advantages of both unstructured and structured grids.

Lagrangian methods. Particle-based Lagrangian approaches, such as SPH, achieve spatial adaptivity by defining a desired resolution for each particle with a sizing function. By adjusting particle sampling through local merging or splitting of particles (as shown in Figure 2.7), these methods are able to dynamically change the resolution, offering more efficient and accurate simulations while focusing on areas of interest.

The early study of adaptive SPH can be traced back to the work of Adams *et al.* [3]. They introduced a sizing function based on geometric local feature size that allows computational resources to be focused on geometrically complex regions. However, adaptive particles yield density errors due to different resolution scales, which can lead to instabilities. To address this issue, Orthmann and Kolb [220] proposed a temporal blending technique to limit the rate of temporal resolution change, thereby significantly reducing the error. With the advent of more strictly incompressible implicit SPH approaches, the size difference between neighboring particles must be minimized to avoid instability. Winchenbach *et al.* [313] achieved this by forming a continuous transition of particle resolution by tuning the splitting and merging pattern and introducing mass redistribution between particles. A simplified version of temporal blending was also incorporated. In their method, splitting supports arbitrary $1 : n$ patterns. Merging uses an $(n + 1) : n$ pattern, where one particle is merged into the others. Mass redistribution divides the excessive mass m_{ex} of one particle i equally among n particles. The physical attributes A of the mass-receiving particles j are updated to A_j^* by

$$A_j^* = \frac{\frac{m_{\text{ex}}}{n} A_i + m_j A_j}{\frac{m_{\text{ex}}}{n} + m_j}. \tag{2.11}$$

In this thesis, we extend this scheme to optimize resolution at the coupling boundaries between fluid and solid particles, detailed in Section 3.3. Zhai *et al.* [345] took inspiration from this method to propose an adaptive scheme for Power Particles [73]. Winchenbach *et al.* [311] addressed the challenge of coupling fluid particles of varying sizes with particle-based boundary representations. They proposed a semi-analytic approach to handle boundaries in solving this problem. Detailed information on this implementation is provided in Section 3.4. Recently, Winchenbach and Kolb [314] introduced optimized refinement for splitting patterns with a discretized objective function that models the error, thereby significantly improving stability. Neighbor search for

adaptive particles also needs to be specifically optimized. In adaptive simulations, a particle can have a widely varying number of neighbors depending on its size. Smaller particles, when neighbored by larger ones, tend to have fewer neighbors than those in the opposite scenario. To solve this, Winchenbach and Kolb [312] proposed constrained neighbor lists to determine the neighbors in a user-specified range. To further accelerate the neighbor search process, Winchenbach *et al.* [310] introduced a sparse data structure for efficient neighbor search and ray tracing for adaptive SPH based on hash-maps.

Among vortex methods, ways to solve the Poisson problem efficiently with adaptive data structures have also been studied. The Poisson problem is an N-body problem, where the interaction between each object and the remaining objects is considered. Naively solving this problem requires $O(N^2)$ computations, so adaptive methods are used to reduce this complexity. The Fast Multipole Method (FMM) [113] uses an octree to approximately solve the N-body problem in $O(N \log^{\eta} N)$, where $\eta \in \{0, 1\}$ by approximating interactions between far-away bodies by the body centers instead of computing all pairwise interactions. Zhang and Bridson [346] proposed a novel Particle–Particle Particle–Mesh method, which is easier to implement and parallelize on GPUs, and applied it to a vortex segment solver. Angelidis [13] used FMM with added support for non-uniform particle sampling to simulate incompressible smoke with vortices.

Hybrid methods. Hybrid methods offer greater flexibility in implementing adaptive schemes due to their intrinsic combination of Lagrangian and Eulerian representations for fluid simulation. Ando *et al.* [11] applied the particle splitting–collapsing scheme, similar to traditional Lagrangian adaptive mechanisms, to adjust the granularity of fluid representation in relation to the distance to the fluid surface for FLIP. They used the finest particles to represent splashes and sheets. Ando *et al.* [12] later introduced an adaptive liquid solver on tetrahedral meshes, which combined a variant of FEM with FLIP advection, adapting the sizes of both the particles and tetrahedral meshes coherently for more efficient simulation. For highly stable situations, Yue *et al.* [341] explored the possibility of simulating the interior area as soft continuum materials to reduce computational costs at the solver level.

To further save computational costs, more recent hybrid approaches aimed to "hollow out" the inner area of the simulated fluid by using Eulerian simulation only, with particles applied near the surface. Chentanez *et al.* [56] proposed coupling pure Lagrangian and Eulerian methods to simulate single fluid bulks, addressing the issue of fluid representation transition and coupling stability between two different fluid solvers. However, coupling two different solvers can still be prone to instability, so Ferstl *et al.* [89] later returned to adaptive FLIP simulation using FLIP particles within a narrow band of the fluid surface. Nakan-

ishi *et al.* [210] focused on constructing a proper data structure to adaptively merge and split octree grids for FLIP, adapting the size of the background grid only near the fluid surface.

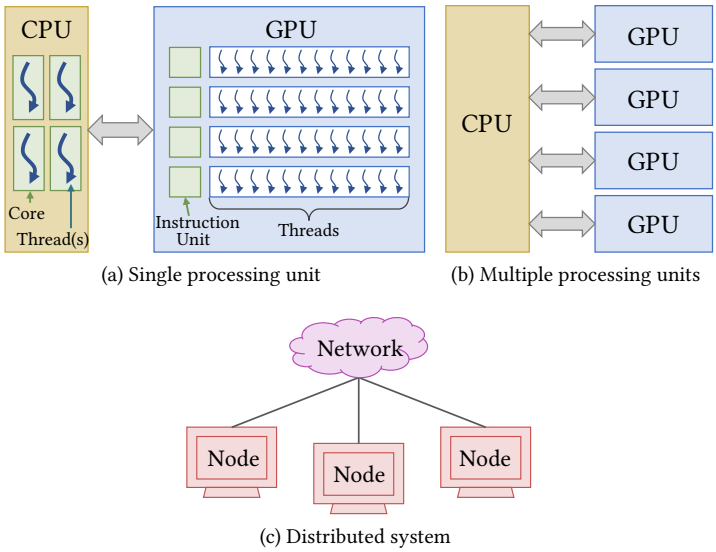
Sato *et al.* [250] took adaptivity to the extreme by extending Ferstl *et al.*'s work [89] to replace some of the fluid surfaces using level set surfaces via an introduced transition function. This idea represents a breakthrough in the field, enabling a seamless merging of physical simulation details and large-scale representations. To address the setback of inefficient, highly-dissipative wave propagation during the transition of surface representation, Huang *et al.* [134] employed hybridization of volumetric and surface-based advection-projection discretizations, with the boundary element method (BEM) applied for long-lasting waves.

2.3.2 *Parallelization*

Parallelization of simulation algorithms offers a promising pathway to augment fluid simulation speed by capitalizing on the existing parallel computation capacities of modern GPUs and CPUs. We divide such methods into three classes: parallelization on a single processing unit (multi-core CPU or a GPU); parallel techniques with multiple processing units, especially multiple GPUs; and using distributed systems (see also Figure 2.10).

2.3.2.1 *Single processing unit*

Parallelization can be most easily achieved on a single processing unit, such as a multicore CPU or GPU. Pure Eulerian and Lagrangian simulation techniques can be readily parallelized under these conditions. However, hybrid methods necessitate meticulous measures due to their composite nature of particles and grids. The method of using a volumetric, dynamic grid that shares several characteristics with B+trees (VDB) pioneered by Museth *et al.* [207] offers a robust framework for manipulating sparse volumetric data. Wu *et al.* [316] successfully integrated this approach into the FLIP method by resolving the parallel particle-to-grid rasterization problem inherent in hybrid methods. Gao *et al.* [101] accomplished a similar integration for MPM based on the SPGrid structure [255], wherein the particle-to-grid transfer mechanism was a crucial aspect to be solved. Furthermore, Chu *et al.* [59] applied and optimized the Schur complement theory for FLIP simulations. They divided the simulation domain into multiple subdomains with face edges and cross-points to establish a parallel-friendly data structure. Aiming to streamline the development of parallel programs, Hu *et al.* [130] proposed a new data-oriented programming language — Taichi. This language facilitates efficient authorship, access, and maintenance of sparse data structures and is accompanied by a compiler designed to automatically optimize and parallelize code on CPU or GPU platforms. Subse-



(a) Single processing unit (b) Multiple processing units

(c) Distributed system

Figure 2.10: Schematic diagrams of different parallelization types. A single processing unit can be a multicore CPU or GPU. A CPU has few but high-processing-power cores. A GPU has weaker but many more cores, which can result in better performance for repetitive operations.

quently, Hu *et al.* [131] enhanced Taichi to include bit-level memory control over numerical data types.

2.3.2.2 *Multiple processing units*

Multi-GPU techniques are the approach of choice for scaling up the simulation size. In this case, multiple GPUs cooperate with the CPU(s), together forming a heterogeneous computing structure. The most challenging aspect of this structure is to reduce both the frequency and amount of data exchange between CPU–GPU and GPU–GPU, which is very expensive. Liu *et al.* [185] initially introduced the Schur complement method to the graphics community as a strategy to tackle this issue. As previously outlined, the Schur complement method has the significant benefit of segmenting the entire simulation domain into several regions. Each region can be efficiently computed using a single GPU, with the interaction boundary between these regions elegantly managed by the CPU(s). This setup eliminates the need for data transmission from a GPU to a CPU and subsequently to another GPU. Meanwhile, Wang *et al.* [304] took a different approach by directly adapting the MPM algorithm structure to fit a multi-GPU framework. They accomplished this by developing a particle data structure to encour-

age coalesced memory access and circumvent atomic operations during particle-to-grid data writing. Additionally, they proposed a kernel fusion strategy to reduce the number of GPU kernel launches and global memory requirements. Chen *et al.* [52] further optimized kinetic methods for scenarios containing complex solids. They introduced a multi-kernel launch methodology for parallelism enhancement and a parametric cost model to improve performance optimization. This exploration of fluid simulation parallelization demonstrates the richness and versatility of strategies within this domain.

2.3.2.3 *Distributed systems*

In the pursuit of scalability in fluid simulation, the potential of distributed platforms is harnessed to delegate tasks across multiple computing nodes. These distributed simulations frequently utilize automatic task allocation to ensure efficient processing. Biddiscombe *et al.* [33] laid a crucial groundwork by providing a GUI-based interface and analysis system for high-performance computing (HPC). Their innovative approach involved substituting the I/O layer in the Hierarchical Data Format Version 5 (HDF5) with a parallel data transfer driver, thereby enabling parallel simulation, analysis, and GUI operation concurrently on one or multiple devices. Mashayekhi *et al.* [198] developed a system that automatically distributes tasks across many multi-core cloud computing nodes to dynamically manage fluid partitions. Shah *et al.* [256] proposed a load-balancing scheme for sparse fluid simulations. Qu *et al.* [233] outlined a simple yet effective solution to accelerate distributed fluid simulations, which uses micro-partitioning to greatly improve load balance and communication performance.

2.3.3 *Data-driven Approaches*

Although great progress has been made in space–time adaptive and parallel computing in recent years, the simulation of fluid by traditional physical methods still requires high computational resources. Strictly limited time step sizes are needed to ensure simulation stability when solving governing equations in a discretized space. Implementing efficient and accurate end-to-end fluid simulation pipelines is also technically highly challenging. *Data-driven* approaches provide an alternative solution for real-time interactive fluid simulation, which we outline next.

2.3.3.1 *Model reduction*

Model reduction is achieved by precomputing a set of simulation sequences to obtain a low-dimensional representation of fluid motion, allowing for efficient and fast re-simulation. Treuille *et al.* [290] first in-

troduced model reduction to fluid simulation via Galerkin projection. They constructed vector field basis functions for fluid dynamics based on principal component analysis (PCA) to generate real-time fluids. For this, they computed the Galerkin projection of the following differential equation $\mathcal{F}$ onto a reduced-dimensional space:

$$\mathcal{F}(\mathbf{r}) = P \circ \mathcal{F}(\mathbf{v}) \circ P^{-1}. \tag{2.12}$$

The high-dimensional vector $\mathbf{v} \in \mathbb{R}^n$ and low-dimensional vector $\mathbf{r} \in \mathbb{R}^m$ are transformed by the projection operator $P(\mathbf{v}) = \mathbf{r}$ and its inverse $P^{-1}(\mathbf{r}) = \mathbf{v}$.

Later, improvements, such as the extension of the Galerkin projection to non-polynomial systems [269] and a multidimensional cubature method supporting semi-Lagrangian advection [162], were proposed. De Witt *et al.* [74] used Laplace eigenfunctions instead of PCA eigenvectors. The method performs a Galerkin projection of the vorticity form of the Navier–Stokes equations and is therefore not data-driven but physically driven. Liu *et al.* [183] further stabilized the method using a variational integrator, providing structure coefficients without artifacts. Zhai *et al.* [344] proposed a model reduction method based on empirical modal decomposition (EMD), which can decompose the flow field into various frequency components as the basis vectors for model reduction. This method can extract the characteristic parameters of the original fluid to achieve inverse modeling. To reduce the memory requirement to store basis functions, Cui *et al.* [68] generalized the dynamics to Neumann boundary conditions using analytic eigenfunctions and the Fast Fourier Transform (FFT). This approach allows the use of thousands of basis functions to produce more convincing and fine-grained fluid dynamics. Using this, they proposed an analytical extension of the Laplace eigenfunction method [67]. This spiral–spectral fluid simulation method is capable of producing realistic turbulent effects over a variety of radial domains, both surface and bulk. Mercier and Nowrouzezahrai [200] constructed an anisotropic vector field basis function that can accommodate curved boundaries and coupling with dynamic obstacles. The method sacrifices a physically accurate solution for a visually plausible simulation. A reduced model for fluids based on incompressible polynomial vector fields was proposed by Panuelos *et al.* [225] to reduce the computational cost of highly viscous fluids.

2.3.3.2 *Machine learning*

Machine learning methods have brought about a revolution in physics-based fluid simulation. In particular, deep learning techniques have proven the possibilities of data-driven approaches. Ladicky *et al.* [167] expressed physics-based fluid simulation as a regression problem and used a regression forest algorithm to approximate the dynamic behavior

of fluid particles. This method strongly generalizes to simulating large-scale scenes in real time. Raveendran *et al.* [235] used interpolation on existing fluid simulations to rapidly generate a large number of new simulation results. Thuerey [289] improved this method using a signed distance function to fully automate the matching process. Recently, Oh and Lee [216] proposed a temporal interpolation network based on optical flow and forward advection that can derive high-frame-rate smoke simulations from low-frame-rate simulations.

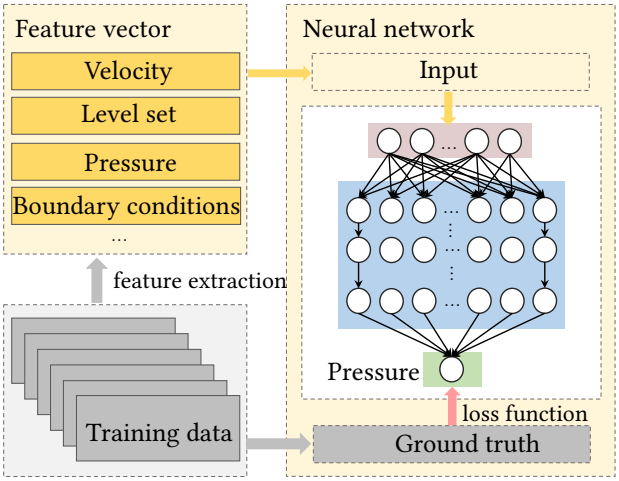


Figure 2.11: Using Artificial Neural Networks to solve the pressure projection [103]. The network inputs feature vectors extracted from training data to output a pressure that is as close as possible to the ground truth.

With the development of deep learning, other types of Artificial Neural Networks (ANNs) [103, 319] have been introduced to solve pressure calculations in fluid simulations and accelerate the pressure projection step (Figure 2.11). Wiewel *et al.* [308] proposed an Long Short-Term Memory (LSTM) architecture to predict the evolution of fluids over time. They used Convolutional Neural Networks (CNNs) to map the 3D fluid simulation to a low-dimensional latent space, greatly speeding up the simulation. They further improved the method using latent space subdivision [309], allowing for more stable predictions of complex long-term series. Takahashi and Lin [283] proposed a framework capable of extracting physical parameters from real fluid videos and applying them to new scenarios to generate the user's ideal fluid behavior. Eckert *et al.* [78] created ScalarFlow, the first large-scale volumetric dataset for real smoke reconstruction using computer graphics and machine learning. ScalarFlow also makes an important contribution by providing reliable benchmark data and evaluation criteria for reconstruction accuracy.

2.4 MULTI-MATERIAL FLUID COUPLING

A key topic frequently mentioned in fluid simulation is the *coupling* of fluids with their environment, which is composed of different materials. Indeed, in computer graphics, fluids are attractive mostly due to the way they interact with their surroundings, as the attention of the spectator is arguably attracted by the interfaces, or boundaries, between fluids and the rest of the virtual world. Moreover, the behavior of fluid is strongly affected by how these surrounding factors themselves evolve. In this section, we explore this topic with a focus on recent works that aim to accurately and efficiently model coupling with multiple complex materials. We split the discussion into three separate subtopics: meshless methods (Section 2.4.1), mesh-based methods for handling fluid boundary conditions in the case of solid boundaries (Section 2.4.2), and solutions designed to model more complex couplings with multiple boundaries (Section 2.4.3).

2.4.1 *Meshless Methods*

Particle-based boundaries. For most Lagrangian simulation approaches, the solid boundary sampled by so-called boundary particles is the main enabler of inter-particle interactions between fluid and other objects (Figure 2.12). To couple fluid with solid boundaries, early methods used various approaches such as collision detection and ghost particles. Becker *et al.* [27] computed the contact point between the fluid and solid particles and controlled the normal and tangential velocities to impose boundary conditions. Yang *et al.* [329] facilitated the interaction between an SPH fluid and nonlinear FEM deformable solid by sampling proxy particles across the boundary, and they handled fluid–solid coupling with momentum-conserving collisions. Schechter and Bridson [251] used the ghost particle method to generate a thin layer of ghost particles in the nearby solid and air, reducing numerical errors caused by non-uniform particle distributions near boundaries. He *et al.* [121] generated staggered particles between neighboring particles to enhance data structures. This method specifically improves the representation of simulation quantities in areas near boundaries, thereby facilitating the support of various slip boundary conditions.

To further reduce computation time and numerical errors, Akinci *et al.* [7] proposed a versatile and efficient method for SPH boundary handling without the need for collision detection or generating extra particles. The method resolved to directly handle the problem of uneven boundary sampling by evaluating the relative contribution of each boundary particle using Shepard interpolation, given by

$$\rho_{f_i} = m_{f_i} \sum_j W_{ij} + \sum_k \Theta_{s_k} \left(\rho_{0_i}\right) W_{ik}, \tag{2.13}$$

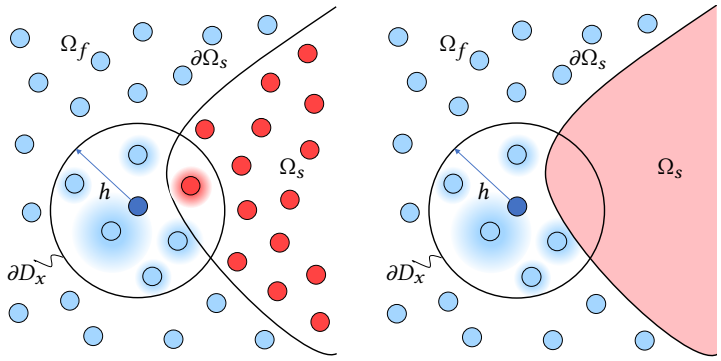


Figure 2.12: Schematic diagram of fluid–solid coupling using meshless methods. Left: Discretization of the SPH approach in coupling with materials. Fluid and solid particles both contribute to the boundary handling. Ω denotes the domain, f denotes fluid, s denotes solid, and ∂D_x denotes the surrounding spherical neighborhood. Right: Solid domain in the neighborhood of a fluid particle representing the contribution of the boundary density value at the particle position.

$$\Theta_{s_i}(\rho_0) = \rho_0 V_{s_i} \tag{2.14}$$

to compute the correct density without depending on boundary sampling. The subscripts f and s represent fluid and solid particles, respectively; ρ_0 denotes the fluid rest density; Θ_s computes the contributions of a boundary particle according to its volume. Compared to the volume of fluid particles with the same size V_f, the volume of solid particles V_s varies based on the local solid sampling distribution. Denser sampled regions have smaller solid volumes to maintain interaction stability. At the mere cost of a one-time evaluation of the above procedure, thin boundary geometries with only one layer of particles and non-manifold geometries can be supported.

Macklin *et al.* [195] applied the method of Akinci *et al.* [7] to unify various physical behaviors using a single-particle system for real-time applications built on the PBD. Cornelis *et al.* [64] extended this approach [7] to couple high-resolution FLIP with a low-resolution implicit SPH method. Peer *et al.* [228] built an implicit formulation for the simulation of incompressible linearly elastic solids embedded in the ISPH pressure solver, which further enables a pressure-based boundary treatment using the method of Akinci *et al.* [7]. Takahashi *et al.* [280, 281] integrated this approach [7] into their multilevel particle-based solver, in which they adaptively assigned various roles to the particles to guarantee the solvability of the linear system in a unified manner regardless of the arrangements of the particles.

Although Akinci *et al.* [7] eliminated a variety of artifacts of particle-based fluid–rigid coupling, numerical issues, such as the lack of higher-order accuracy of pressure due to the mirroring scheme of physical values from fluid to boundary particles, still limit the time step size and stability under drastic scenarios. Shao *et al.* [259] treated surface and inner boundary particles differently to prevent particle deficiency and penetration issues. Instead of using the state of density compression to denote the magnitude of the pressure force, Band *et al.* [23] introduced the notion of "volume compression" into coupling problems, where the rest volumes rather than the mass of boundary particles are applied to derive a continuous pressure force

$$\mathbf{F}^{\mathrm{p}}_{f_i} = -V_{f_i} \sum_j V_j \left(p_{f_i} + p_j\right) \nabla W_{f_i j}, \tag{2.15}$$

which considers fluid samples f_i with all fluid and boundary neighbors j of the fluid sample f_i in the same way. The results of their experiments showed a significant improvement in stability and a wider range of possible time step sizes. Gissler *et al.* [106] resolved the stability issue in the work of Akinci *et al.* [7] by interlinking an artificial pressure solver for boundary particles with the fluid pressure, achieving a fully dynamic two-way coupling. Truong *et al.* [292] prevented penetration by treating particle collisions with particle merging and splitting.

Unsampled boundaries. The influence of a solid model on the surrounding fluid particles can also be handled using a mesh-based (Eulerian) solid representation, which takes into account the local mesh geometry for stable and effective coupling. Vines *et al.* [297] coupled Lagrangian vortex particles to mesh-based solids by generating vorticity at the solid boundary. Fujisawa and Miura [96] considered the influence of triangle mesh boundaries on the integration of a kernel function for SPH without the need for boundary particles. However, the method cannot handle solid boundaries with complex geometry and is computationally expensive compared to the method of Akinci *et al.* [7]. Chang *et al.* [48] extended the work of Fujisawa and Miura [96] to support arbitrarily shaped solid boundaries by converting the volume integral inside the solid boundary to a surface integral. Koschier and Bender [165] presented the "density maps" method, precomputing a continuous boundary density field to efficiently handle arbitrary boundary geometries. Bender *et al.* [31] also targeted the expensive renormalization process in the work of Fujisawa and Miura [96] by storing the volume contribution from the boundary on a spatial grid that can be efficiently queried at runtime.

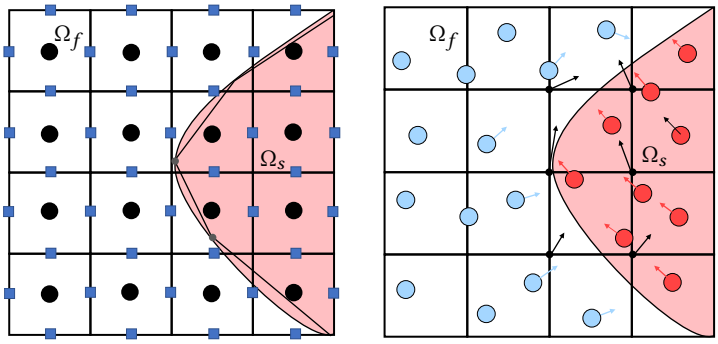


Figure 2.13: Schematic diagram of fluid–solid coupling using mesh-based methods. Left: A sample MAC grid used in the fluid–solid coupling. The lines connecting the gray dots denotes the mesh surface by the solid. Fluid pressure values are stored at cell centers. Fluid velocity components are stored on cell faces. Right: A coupled solid–fluid system with the MPM method, which allows the system with particles to be discretized on a Cartesian grid and the pressure to be updated based on DOFs on the grid.

2.4.2 *Mesh-based Methods*

Another mainstream approach for fluid simulation is to use Eulerian and Lagrangian meshes, as shown in Figure 2.13. However, loss of mass and difficulties in handling extreme deformations often limit the practicability of these approaches. We organize mesh-based methods into four classes as follows.

Lagrangian/Eulerian meshes. Early on, Clausen *et al.* [61] used fully Lagrangian tetrahedral meshes to significantly reduce numerical viscosity in simulations with relatively low resolutions and long time steps. Azevedo and Oliveira [20] proposed a semi-Lagrangian method that introduced curvilinear grids and achieved more accurate boundary conditions in simulations with moderate resolutions. Besides these Lagrangian-meshed methods, Teng *et al.* [287] later incorporated previous work into an Eulerian fluid solver and resolved complex contact scenarios between multiple solids and fluids. Recently, Takahashi and Lin [282] formulated implicit viscosity integration as a minimization problem in which the volume fractions are consistently evaluated to handle sub-grid details.

Focusing on the poor boundary conditions for irregular boundaries defined under coarse grids, the cut-cell approach became a major trend to improve convergence for Neumann boundary conditions. Fluid grid cells are clipped against the solid boundary represented by a triangle mesh, forming several distinct polyhedral sub-cells at each time step, allowing small details to be handled without refining or rotating the grid.

Based on the multigrid scheme proposed by Chentanez and Mueller-Fischer [54] with a variational discretization compatible on all levels, Weber *et al.* [305] presented a cut-cell-based multigrid scheme on staggered grids that is second-order accurate for Neumann boundaries. To better capture the flow across thin solids and gaps, Azevedo *et al.* [19] further proposed a topology-preserving pressure projection scheme on cut-cell meshes. Following this, Zarifi and Batty [342] used cut-cell discretization for two-way fluid-deformable interaction, enforcing the free-slip boundary condition at the actual interface. Their method computes the pressure based on the MAC grid of three dimensions x,y, and z as

$$\int_{\partial\Omega_f} \nabla p d\mathbf{n} \approx \sum_{\beta\in\{x,y,z\}} \left(S\left(\Omega_\beta\right) \frac{\partial p}{\partial \beta}\left(\Omega_\beta\right) + S\left(\Omega_{-\beta}\right) \frac{\partial p}{\partial \beta}\left(\Omega_{-\beta}\right) \right), \quad (2.16)$$

where Ω contributes to the domain region and $S(\Omega)$ is the area of the domain region, respectively. The cut-cell was further extended by Chen *et al.* [51] to represent sub-grid structures on the free surface of the liquid, with a new iso-surface Poisson solver with desirable properties, such as second-order accuracy and symmetric positive definiteness. Tao *et al.* [286] introduced an algorithm based on VEM for simulating fluid flow on cut-cell meshes, which effectively handles complex geometries and accurately captures intricate features, including thin tubes and extremely thin walls.

Material Point Method. Since its introduction to computer graphics, the MPM has garnered significant attention. By integrating features of Lagrangian particle representation and Eulerian grid representation, MPM offers a powerful technique for coupling fluid and solid simulations. However, despite their physical realism and geometric convenience, conventional MPM solvers have drawbacks, such as computational inefficiency and limited capability to handle self-contact collisions. To improve upon these, Gao *et al.* [100] presented an adaptive Generalized Interpolation Material Point (GIMP) method with extensively optimized particle–grid transfer memory efficiency and parallelism. Hu *et al.* [129] proposed Compatible Particle-in-cell (CPIC), which enables the handling of discontinuous material points and infinitely thin boundaries by leveraging the relative positions of grid nodes and particles. They also embedded the Moving Least Squares (MLS) method into MPM to double the computation speed. However, such MPM approaches do not address the inconsistent tangential velocities at the interface between multiple materials, leading to visually unpleasant artificial stickiness. To alleviate this, Fang *et al.* [83] presented a ghost matrix operator-splitting scheme for monolithic coupling between incompressible fluids and elastic solids and designed a novel interface quadrature cut-cell MPM formulation for free-slip boundary conditions. Subsequently, Cao *et al.* [47] extended some ideas of Fang *et al.* [83] from incompressible to compressible flow.

Monolithic schemes. Monolithic solvers simulate various materials and their interactions within a unified system that includes boundary conditions. These schemes naturally ensure a more robust interface of large density ratios and enable large time steps. They are not only used in SPH methods [106] but also in mesh-based (*e.g.*, MPM) methods [47, 83]. Aanjaneya [1] proposed a monolithic solver for efficiently simulating the interaction between rigid bodies and incompressible fluids. The solver remains robust even in poorly conditioned scenarios with large density ratios between the solid objects and fluid. Lai *et al.* [168] introduced a V-cycle of the Full Approximation Scheme (FAS) multigrid method to solve the linear complementary problems to achieve better scalability and efficiency compared to previous methods. Takahashi and Batty [277] proposed a monolithic pressure-viscosity-contact solver to simulate the complex interactions between rigid bodies and liquids, efficiently managing incompressibility and offering the option for implicit viscosity integration in liquids. The method also addresses contact resolution for rigid bodies and handles mutual coupling.

Partitioned schemes. Compared to the monolithic scheme, partitioned schemes can deal more flexibly with multiple co-existing solvers by alternating between the solid and fluid while applying suitable boundary conditions. Akbay *et al.* [5] employed fluid and solid solvers as independent components with restricted interfaces, promoting modularity and facilitating code reusability. Lee *et al.* [173] used a partitioned methodology to connect a coarse-background Eulerian grid with a fine arbitrary Lagrangian-Eulerian mesh in their simulation framework for character– and hair–water interactions.

Recently, several coupling approaches for the special treatment of fluid have emerged. Brandt *et al.* [42] modeled fluids and deformable objects as incompressible media, avoiding expensive operations such as interface tracking and boundary condition handling. Ruan *et al.* [243] used a three-way coupling method, employing a thin liquid membrane to model contact between solid objects and fluid driven by strong surface tension.

2.4.3 *Coupling with Complex Boundaries*

Besides interacting with the dynamic or static boundaries of surroundings composed of rigid or elastoplastic materials, fluids often interact with other solid matter with complex and diverse physical attributes. Such cases require intricate boundary models and corresponding solvers. Next, we discuss recent advances in fluid coupling with thin film surfaces and/or porous materials, including hair, cloth, sponges, and sand.

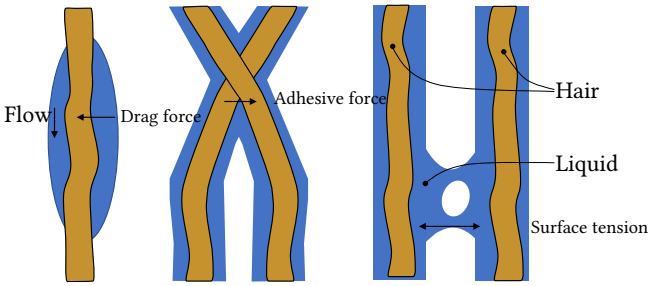


Figure 2.14: Schematic diagram of wet hairs. Left: Wet hairs with a thin liquid layer flowing over their surface; their motion is influenced by the surrounding liquid flow. Middle: Proximity and collision between wet hairs cause adhesive and contact forces between hairs. Right: Cohesion of wet hair. Surface tension creates liquid bridges between closely positioned wet hairs.

Strands and cloths. Coupling between hairs and fluid is complex due to the wetting of hairs. Such phenomena are shown in Figure 2.14. Rungjiratananon *et al.* [244] used an Eulerian approach to capture hair porosity and wetting effects and a Lagrangian approach to simulate individual hair strands and their interactions, resulting in a detailed and dynamic hair simulation. Chen *et al.* [53] proposed a real-time painting system that aimed to generate realistic paintings by simulating the interactions among the brush bristles, paint, and canvas. Fei *et al.* [87] proposed a multi-component framework to model wet hair. PIC and Kirchhoff Rods were applied to model fluid and hair separately. A height field was introduced to represent the liquid volume around each hair strand, considering the wet condition. Fei *et al.* [86] next extended the fluid attributes to compressible, shear-dependent liquids. A modified second-order Coulomb cone model was also designed to capture cohesion and friction during strand collisions. Lee *et al.* [172] used a tetrahedral volume mesh to embed hair, enabling the hairs to adhere to their embedded positions and facilitating simulations with millions of hairs during water–hair interactions.

For the interaction between cloths and fluid, Huber *et al.* [136] proposed an efficient method for two-way interaction between particle-based fluid and thin triangular meshes, enabling cloth–fluid coupling even under large time steps. Jiang *et al.* [152] created an anisotropic hyperelastic model that distinguishes the response to manifold strain, shearing, and compression in orthogonal directions. This model facilitates the coupling of various materials, such as elastic surfaces, curves, fluids, and granular materials. Fei *et al.* [85] introduced a method for simulating the intricate dynamics of woven or knitted fabrics, both partially and fully saturated, interacting with liquids, using the method of Jiang *et al.* [152] to deal with contact and collisions. To simulate stain

formation and evolution on cloths, Wang *et al.* [301] developed a pigmented solution by utilizing a homogenization process that combines inhomogeneous and/or anisotropic properties with bulk anisotropic diffusion tensors. Zheng *et al.* [351] formalized the spreading of stains in woven fabric as in-yarn diffusion and cross-yarn diffusion and introduced a triple-layer model to manage wetting and wicking calculations.

Sponge-like porous materials. Patkar and Chaudhuri [227] simulated liquid flow within a porous object with a deforming unstructured mesh and modeled liquid diffusion based on saturation, as well as allowing the liquid to be absorbed by, or leave, the solid.

Thin film surfaces. Real-life situations often involve surface flow phenomena, such as rainwater cascading down a tree trunk or the gradual progression of water drops in a shower room. For such flows, Vantzos *et al.* [295] proposed a triangle mesh model to simulate the motion of a thin viscous fluid film on a curved surface. The model includes discretization for curvature and advection operators to ensure accurate simulation results. Ren *et al.* [241] expanded the standard shallow-water flow model to accommodate general triangle meshes. They introduced a feature-based bottom friction model, allowing non-viscous flow motion to be captured along edges and creases on detailed 3D meshes.

Granular materials. Such materials, *e.g.*, sand, can themselves exhibit flow-like behavior. In addition to their porous behavior, simulating the dissolution of granular materials has also attracted much attention. Yan *et al.* [328] combined a hypoplastic model with SPH to simulate granular materials diffusing into fluid. Yang *et al.* [333] integrated the phase-field method to simulate liquids and multiple types of solids, with dissolution achieved by evolving the granular particle's concentration and phase. Tampubolon *et al.* [284] used continuum mixture theory to simulate water–sand coupling in MPM, where different phases are coupled by momentum exchange. Gao *et al.* [99] modeled the motion of solid sediment particles inside fluids with MPM. Sediment was modeled by Drucker–Prager elastoplasticity, achieving two-way coupling between sediment and fluid. He *et al.* [122] proposed position-based constraints for granular flows, using cohesion and friction models that vary across space, with cohesion affected by water saturation. Takahashi and Batty [278] simulated two-way coupling between rigid bodies and continuum granular materials or liquids with a monolithic solver that combines pressure, friction, and contact interactions. Gao *et al.* [102] used a hybrid scheme to accurately simulate the behavior of discontinuous fluid-like substances. This approach integrated an affine particle-in-cell solver with density fields, enabling transformations across granular particles, dust clouds, powders, and their mixtures within a unified framework.

2.5 MULTIPHASE LIQUIDS

The real world is replete with complex fluid phenomena, including dissolution, dispersion, and Rayleigh-Taylor instabilities, all of which are closely related to multiphase environments. The study of multiphase fluid simulation, a distinct subject within Computational Fluid Dynamics (CFD), has garnered significant attention in the realm of computer graphics. We next give an overview of this topic. We group various phenomena and their corresponding simulations into non-mixing (Section 2.5.1) and mixing fluids (Section 2.5.2) based on whether a clear immiscible interface can be formed between two different phases. For non-mixing fluids, we classify their simulations according to the discretization methods used. We also survey recent methods by grouping them into two categories: mixture models where phase velocities are separately calculated and non-trivial diffusion-based models where phase velocities are considered jointly.

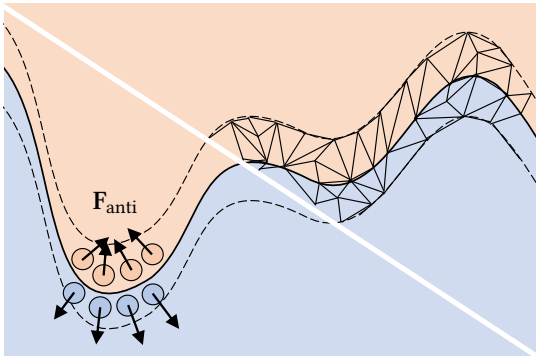


Figure 2.15: Schematic diagram of non-mixing fluids. The lower-left part shows the anti-penetration force ($\mathbf{F}_{\text{anti}}$), which is applied to particles in opposite directions along the surface normal. The embedding 3D fluid mesh can be re-tessellated to accommodate vertex displacement and produce changes in the interface topology.

2.5.1 *Non-mixing Fluids*

Recent methods for simulating non-mixing fluids include particle-based methods, *e.g.*, SPH [263], PBF [8], and MPM [327], and mesh-based methods [69, 181, 201, 331]. Phases in non-mixing fluids typically do not merge together. Non-mixing fluid phases inherently resist fusion, creating challenges in (a) tracking the interfaces between different phases and (b) accurately calculating force interactions between phases at liquid–liquid or liquid–solid interfaces. The unique properties of gas–liquid interfaces are discussed separately in Section 2.6.

SPH simulations of multiple fluids, particularly those with high density ratios, can produce erroneous interface tension and non-physical separation between the fluids. In response to this issue, Solenthaler and Pajarola [263] modified the standard SPH equations to account for the density discontinuity across the interface. They used an SPH density interpolation formulation, $\rho_i = m_i \sum_j W(\mathbf{x}_i - \mathbf{x}_j, h)$, instead of the standard one, $\rho_i = \sum_j m_j W(\mathbf{x}_i - \mathbf{x}_j, h)$. Particularly in explicit pressure solvers like WCSPH, this modification allows the pressure to align more closely with the actual compression state of the local fluid, even at the interface between two different fluid phases with different rest densities. However, integrating these modifications directly into implicit solvers can lead to convergence issues, which are discussed further in Section 4.3.

Alduán *et al.* [8] proposed a versatile simulation framework based on PBF methods designed to address VFX production demands. On the phase interface, they used an SPH density interpolation formulation, which is similar to the method in the work of Solenthaler and Pajarola [263]. Subsequent PBF calculations were also modified using this uniform-density formulation. High viscosity was handled by breaking the XSPH calculation into multiple lower-viscosity stable iterations. The surface tension effect was bounded for stable artistic controls.

Yan *et al.* [327] extended the MPM method to cope with interacting solid–fluid simulations. Upon the detection of a solid–fluid collision along the phase boundary, an anti-penetration force is applied to both the fluid and solid particles in opposite directions along the surface normal. The same strategy was used to simulate non-mixing fluids, where the anti-penetration force is calculated between each distinct phase and all other phases collectively (Figure 2.15).

Mesh-based methodologies offer the advantage of explicit interface tracking using high-resolution geometric structures. Da *et al.* [69] considered a special emphasis on topological change handling in 2D surface tracking for non-mixing fluid simulations (Figure 2.15). They constructed highly distorted interfaces featuring thin sheets and tiny regions. Meanwhile, Misztal *et al.* [201] aimed to prevent mismatches between phase occupancy regions and the simulation quantity storage grid; they achieved this by discretizing each phase region using unstructured 3D tetrahedral grids and tracking the deformation of the tetrahedra. They managed topology changes and mesh quality enhancements using a modified 3D deformable simplicial complex method. In contrast, Li *et al.* [181] avoided complex remeshing operations by combining mesh-based tracking and surface reconstruction from distance fields. They reconstructed the surface meshes between each phase using an unsigned distance function and indicator function. The mesh was stored as a later interpolation reference in a semi-Lagrangian update of the two functions in the next time step. Yang *et al.* [331] developed an advanced method for surface tracking across multi-fluid inter-

faces, specifically designed to manage the complexities involved when dealing with more than three fluid phases. This method emphasizes the prevention of mesh penetrations and ensures a consistent relationship between the meshes and the regional level-set functions at the interfaces. An extended triangulation template strategy was also proposed to handle triple junctions, where standard marching cubes would result in volumetric holes.

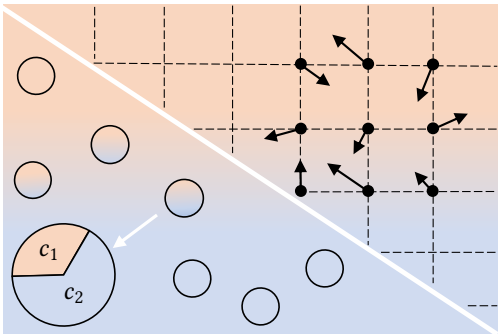


Figure 2.16: Schematic diagram of mixing fluids. The lower-left part shows the fractions c_i (volume fraction, mass fraction, or concentration) of different phases in each particle during the diffusion process. The upper-right part shows the 3D fluid grid; black arrows represent grid forces due to diffusion.

2.5.2 *Mixing Fluids*

Another category of multiple-fluid flows involves miscible or dispersed fluid mixtures, for which interfaces can be challenging to track continuously or may not exist at all. In contrast to the non-mixing case (Section 2.5.1), different phases now always co-exist at the same spatial position (Figure 2.16). A key problem is to calculate how the local volume fractions c_k^V of each phase k change during the simulation, *i.e.*, solving the multiphase continuity equations. Other challenges include simulating diffusive behavior, incompressibility enhancements, and stability improvements. We summarize several typical methods that can be integrated into the SPH [156, 239], PBF [333, 334], and MPM [50] solvers.

Mixture model. Ren *et al.* [239] used a multiphase mixture model for complex multiphase mixing and unmixing effects. Their SPH-based approach (called WCSPH) solves the mixture continuity and momentum equations for each particle as follows:

$$\frac{\partial}{\partial t}\rho_m + \nabla \cdot (\rho_m \mathbf{u}_m) = 0, \tag{2.17}$$

$$\frac{\partial}{\partial t}(\rho_m \mathbf{u}_m) + \nabla \cdot (\rho_m \mathbf{u}_m \otimes \mathbf{u}_m) = -\nabla p_m + \rho_m \mathbf{g} + \nabla \cdot (\boldsymbol{\tau}_m + \mathbf{T}_m), \quad (2.18)$$

where $\rho_m = \sum_k c_k^V \rho_k$ is the mixture density, $\mathbf{u}_m = \frac{1}{\rho_m} \sum_k c_k^V \rho_k \mathbf{u}_k$ is the mixture velocity, p_m is the mixture pressure, and $\boldsymbol{\tau}_m$, $\mathbf{T}_m$ are the mixture viscous stress and diffusion tensors, respectively. A detailed derivation of this mixture model is provided in Section 4.2, where we also discuss its correspondence with the standard Navier-Stokes model. Phase velocities are assumed to be different from each other. For each phase k, a drift velocity $\mathbf{u}_{mk} = \mathbf{u}_k - \mathbf{u}_m$ is analytically computed at the start of each time step. These phase-wise drift velocities are used to calculate the phase volume fraction change Dc_k^V/Dt as well as the mixture diffusion tensor $\mathbf{T}_m$. Following this, the aggregate particle motion, individual phase velocities, and phase volume fraction changes on the particles during the simulation are solved. Yan *et al.* [328] extended this mixture model to cope with solid phases. Ren *et al.* [240] further introduced a virtual phase concept for multiphase simulations containing porous solids, considering the absorbed and non-absorbed parts of a single phase as two virtual phases that can be universally handled by the mixture model. The result was a unified algorithm framework for multiphase flows inside and outside porous solids. To alleviate the incompressibility issue of the WCSPH framework [239], Jiang *et al.* [156] used volume-weighted mixture velocities $\mathbf{u}_m = \sum_k c_k^V \mathbf{u}_k$ to ensure a divergence-free mixture velocity field solvable by an iterative incompressible SPH solver for the single-fluid case. To capture multiphase fluids with highly dynamic relative motions, Jiang and Lan [155] presented a dynamic mixture model that abandoned the local equilibrium condition. This method also allowed for fluid control in the multiphase environment by solving the Navier–Stokes equations for each phase flexibly. In contrast, Ren *et al.* [238] used the deformation gradient to construct a set of linear equations that match the local volume change resulting from the momentum-equation-solved velocities, which resulted from the continuity-equation-solved fraction changes, and solved these equations for enhanced incompressibility. In this thesis, we contribute to advancing the application of the mixture model in simulating mixing fluids, as detailed in Section 4.4.

Non-trivial diffusion. Traditionally, phase-mixing effects in fluid simulation have been modeled using the diffusion equation $\frac{Dc}{Dt} = C_d \nabla^2 c$, where C_d is the diffusion coefficient and c is the concentration, which assumes uniform phase velocities and movement in accordance with the aggregate motion. This approach has been employed in various works, such as Im's [141] diffusive dissolved gas transfer model for calculating bubble distribution in freezing ice blocks and He *et al.*'s [124] two-

phase diffusive model for simulating diffusive appearances with varying sharpness in materials like ink and bubbles.

Other researchers have advanced this field with more sophisticated models. Yang *et al.* [334] integrated the Cahn–Hilliard equation into multiphase simulation using an energy-based model to capture complex multiphase effects, such as unmixing and extraction. They computed the change of the mass fraction c_k^m of each phase k as

$$\frac{Dc_k^m}{Dt} = \nabla \cdot (\mathcal{M} \nabla \phi_k), \tag{2.19}$$

where $\mathcal{M}$ is a degenerate mobility, which can be constant or may vary according to the value of c, and ϕ_k is the k-th phase's chemical potential relying on the derivative of a case-specified Helmholtz free energy function at the current concentration composition. This model is able to capture complex multiphase effects, such as unmixing and extraction. Yang *et al.* [333] extended this model using a unified Helmholtz free energy form to handle both solid and liquid phases, thereby expanding the capacity of the PBF multiphase solver. Chen *et al.* [50] proposed a moving least square reproducing kernel particle method for better precision and stability of particle-based simulations. Using an advanced interpolation scheme, they integrated the Cahn–Hilliard equations into MPM solvers and achieved good mass conservation, stability, and sub-grid details in multiphase fluids.

Xue *et al.* [325] modeled anisotropic diffusive effects using non-Fourier diffusion, which was integrated into a phase field formulation using an MPM solver. The resulting constitutive model is given by

$$\begin{aligned} \mathbf{q} &= \mathbf{q}_C + \mathbf{q}_F, \\ \mathbf{q}_C + \tau \dot{\mathbf{q}}_C &= -(1 - G_T) C_d \nabla A, \\ \mathbf{q}_F &= -G_T C_d \nabla A, \end{aligned} \tag{2.20}$$

where $\mathbf{q}$ is the associated diffusion flux, $\mathbf{q}_C$ and $\mathbf{q}_F$ represent Cattaneo and Fourier diffusion, respectively, τ is the relaxation time with respect to the flux, C_d is the diffusion coefficient, and A is the quantity being diffused. G_T is a dimensionless parameter that represents the weight between Cattaneo-type and Fourier-type diffusion. Their method reproduced complex folding effects of poroelastic materials during wetting and also directional diffusive transportation effects. Su *et al.* [272] adopted the anisotropic diffusive model [325] for temperature transport in an extended MPM phase change solver, allowing the simulation of richer phenomena. They also introduced an integration scheme that provides second-order accuracy with only first-order algorithmic overhead.

In recent years, additional works have studied other mixing-related phenomena. Stomakhin *et al.* [270] proposed an MPM approach to solve heat-induced phase change of various materials. A carefully designed

projection solver allowed them to simulate nearly incompressible phase-changing materials in MPM. Hochstetter and Kolb [126] presented an SPH method to simulate the evaporation and condensation of liquids. Their technique utilized particles to signify the liquid phase, while the grid primarily served as a medium for simulating the gas phase and facilitating water vapor transport. This method used Fourier's law as a basis for heat transfer between grid cells and particles, thereby advancing the understanding of multiphase heat and mass transfer phenomena.

2.6 GAS–LIQUID INTERFACES

In fluid simulations, the influence of gas is often ignored. However, numerous real-world fluid phenomena, including the formation of water droplets and bubbles, cannot be accurately represented without considering the role of gases. The phenomena formed by gas–fluid interactions are complex and diverse. In this section, we discuss gas–liquid interface phenomena by grouping them into three categories: free surface fluids (Section 2.6.1), bubbles, foam, and glugging (Section 2.6.2), and spray and splashing (Section 2.6.3).

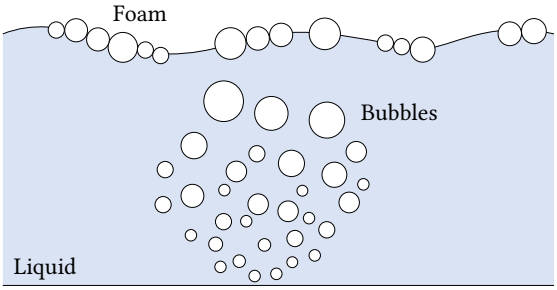


Figure 2.17: Schematic diagram of bubbles and foam. Gas accumulates underwater and forms bubbles. As bubbles rise to the surface, their volume increases due to a pressure decrease. When reaching the surface, bubbles may form foam.

In free surface fluid simulation, the emphasis is typically placed on calculating the fluid surface and accounting for surface tension, rather than explicitly modeling the presence of air or gases. Here, we introduce some typical methods, such as contact angle, surface tracking, and continuous surface force. We then discuss bubbles, foam, and glugging together because of their similar characteristics, *i.e.*, they are the result of a small amount of gas being wrapped by the enclosing fluid. The bubbles we are discussing here are those formed by gas gathering in water, while the foam is formed by bubbles rising to the surface of the fluid (Figure 2.17). Glugging occurs, *e.g.*, when a liquid is rapidly poured

from a bottle with a narrow opening and is a multiphase phenomenon where bubbles are generated automatically. Finally, spray and splashing are formed by free liquids in the gas. These are usually produced by violent collisions of fluids and require more accurate simulation methods.

2.6.1 *Free Surface Fluids*

Physically correct or at least plausible gas–liquid interface modeling is challenging. This is largely due to the fact that while scalar fields such as pressure can be well approximated using particles or grids at macroscopic scales, surface tension (and similar) effects are the result of microscopic inter-molecular forces (Figure 2.18). This makes introducing surface tension effects into standard Lagrangian and Eulerian solvers (see Section 2.1.3) non-trivial.

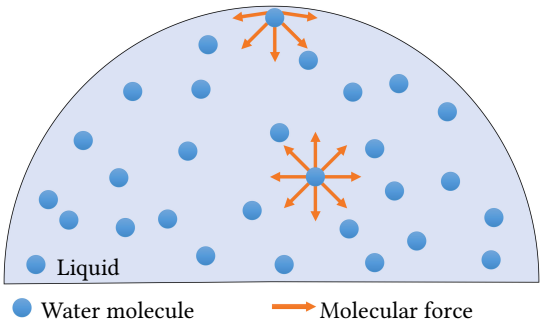


Figure 2.18: Schematic diagram of surface tension. Liquids have forces between the same-kind molecules (cohesion) and different-kind molecules (adhesion). The molecular force on the liquid surface is unbalanced, resulting in surface tension effects.

Wang *et al.* [298] introduced the *contact angle* to calculate surface tension. This angle exists at the junction of solid, liquid, and gas, which indicates the hydrophilicity and hydrophobicity of solid materials (Figure 2.19). They used signed distance fields to represent such surfaces and constructed a virtual surface below the solid one to replace the real solid–fluid interface. The distance field can be modified by the virtual surface. Following this, the stable contact angle θ_s can be obtained to estimate the surface tension from

$$\gamma_{sa} - (\gamma_{la} \cos\theta_s + \gamma_{ls}) = 0, \tag{2.21}$$

where γ_{sa}, γ_{la}, and γ_{ls} are the interfacial tension coefficients for the solid–air, liquid–air, and liquid–solid surfaces, respectively. However, this method uses a grid to represent the volume of the fluid, which requires significant memory and computation time.

Water drop animation was the main focus of Zhang *et al.* [349]. The crucial part of their Lagrangian system that allows efficient simulations

of water drop motions is the reduction of volumetric fluid dynamics over the whole liquid volume to a deformable surface model. While also using the contact angle method like the work of Wang *et al.* [298], their model focuses only on the surface, and as such, it is more computationally efficient.

Da *et al.* [71] proposed a surface-only model that avoids dealing with degrees of freedom inside liquids and (often) far away from their surface. This is the first such model for 3D liquids with the first advection–projection scheme for surface-based liquids, albeit partly limited to bodies dominated by surface tension and inertia, although still capable of modeling effects such as crown splashing.

Akinci *et al.* [6] employed SPH to model surface tension and adhesion forces. Theirs is the first method that correctly handles large surface tension (and adhesion) without the need for ghost particles or artificial pressure forces. The cohesion force is described as

$$\mathbf{F}_{i\leftarrow j}^{coh} = -\gamma m_i m_j B\left(\left\|\mathbf{x}_i - \mathbf{x}_j\right\|\right) \frac{\mathbf{x}_i - \mathbf{x}_j}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|}, \tag{2.22}$$

where γ represents the surface tension coefficient and B denotes a spline function. The method is simple to integrate with existing SPH solvers and can simulate effects such as water crown formation and rolling water droplets.

By assigning each particle a value corresponding to an estimate of its surface area, leading to an implicit definition of the free surface of the fluid, Orthmann *et al.* [219] achieved conservative transport within and between surfaces, including correct handling of thin sheets and other singularities. This allows for effective simulations of detergents, cleansing, and coating.

Yang *et al.* [335] used a pairwise-force model called PF-SPH, which relies on larger support radii than traditional SPH. The method improves the accuracy of the surface tension calculation by using anisotropic filtering to scale neighboring particle interaction forces.

Energy-based methods have also been used to simulate free surface fluids. He *et al.* [123] modified earlier surface tension and air pressure formulations for SPH-based free surface flows, building on the diffuse interface model. They computed surface tension energy $\mathcal{E}^s$ based on Helmholtz free energy as

$$\mathcal{E}^s = \int_V \frac{k}{2} \|\nabla l\|^2 dV, \tag{2.23}$$

where V represents the volume of the liquid, k is a coefficient associated with squared gradient energy, and l denotes the condensation field, assigned a value of 1 inside and 0 outside the volume. The surface tension energy $\mathcal{E}^s$ is directly related to the surface area of a fluid interface. Its gradient can be computed to determine the surface tension force acting

on the interface. This improves the robustness of the model *vs* particle sparsity and in turn leads to increased stability. The model can simulate delicate surface tension effects, such as water/milk crowning.

Classical methods often struggle with relatively high coefficient or parameter values, such as those controlling surface energy. To address this, Hyde *et al.* [137] developed an implicit Lagrangian formulation that specifically targets liquids with significant surface energy, such as liquid metals. By treating discrete forces as gradients of the potential energy that are proportional to the surface area of the liquid, this approach enables more accurate and stable simulations. Chen *et al.* [49] proposed an MPM approach that generalizes the work of Hyde *et al.* [137] by improving resampling via new types of temporary "balance" particles, achieving the perfect conservation of grid linear and angular momenta.

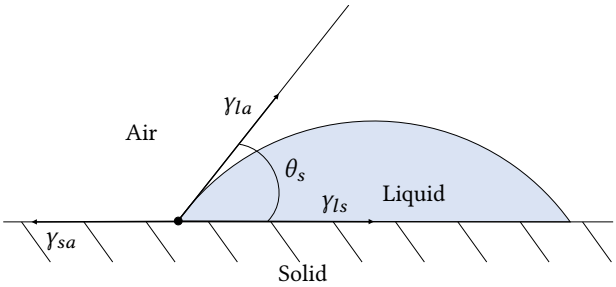


Figure 2.19: Schematic diagram of stable contact angle θ_s. γ_{sa}, γ_{la}, and γ_{ls} are the interfacial tension coefficients for solid–air, liquid–air, and liquid–solid surfaces, respectively. When a drop of liquid rests on a solid surface in equilibrium, the angle between the solid–liquid interface and gas–liquid interface is called the stable contact angle.

2.6.2 *Bubbles, Foam, and Glugging*

The influence of gas on fluid simulation extends beyond just the free surface of the fluid. It encompasses the behavior of the fluid interior and involves more intricate interaction processes. Single-phase liquid simulations typically struggle to capture phenomena like bubbles, foam, and glugging effects, which necessitates the modeling of gas and liquid as two-phase flows.

Patkar *et al.* [226] presented a hybrid Lagrangian–Eulerian scheme for converting between small (*i.e.*, sub-grid and under-resolved) Lagrangian bubbles and larger well-resolved bubbles modeled with an Eulerian approach based on level sets. Their framework includes a bubble seeding mechanism to realistically simulate fluid structure interaction with complex (moving) objects. Cho and Ko [57] combined the volume of fluid (VOF) with sub-grid refinement of the level set method to simulate moving interfaces in two-phase flows.

Goldade *et al.* [108] developed a model for immersed bubble simulation, which avoids advection and projection inside bubbles. The method is based on constraint-based incompressible bubbles (with zero density) and affine fluid regions (to account for non-zero density coefficients). The simulation region is divided into a fluid region Ω_f, solid region Ω_s, and air region Ω_a. Any enclosed and continuous region filled with air is treated as a bubble. Linear velocity constraints are imposed on each bubble via

$$\iint_{\Omega_f \cap \partial\Omega_{a_i}} \mathbf{u}_f \cdot \mathbf{n}_f \, da + \iint_{\Omega_s \cap \partial\Omega_{a_i}} \mathbf{u}_s \cdot \mathbf{n}_s \, da = 0, \tag{2.24}$$

where Ω_{a_i} is the continuous region of bubble i; $\mathbf{n}_f$ and $\mathbf{n}_s$ are the fluid's normal and solid boundary's normal, respectively.

Additional special bubble simulations have been conducted. Paddilla *et al.* [221] modeled bubble rings via vortex filaments of various thickness, assuming that advective inertial forces are small compared to viscous forces. Filaments are expressed as a configuration manifold on which the equations of motion are geodesic. Langlois *et al.* [170] introduced a set of techniques aimed at generating sound representations for intricate two-phase liquid animations. They extended the open-source Gerris solver [231] (a finite-volume-based multigrid solver) to achieve audio–visual fluid (and bubble) simulations.

Although some of the above methods can handle foam to some extent, specialized methods exist for this. Busaryev *et al.* [45] animated bubble interactions in liquid foams by treating (small) bubble particles as sites of Voronoi cells in a weighted diagram. Their framework handles bubble–bubble, bubble–liquid, and also bubble–solid interactions, giving rise to foam simulations with bursting and coalescing. Kim *et al.* [161] modeled foam waves using the FLIP solver. Foam particles projected to 2D give rise to depth and acceleration maps, making the method efficient. The method provides the option to art-direct the foam effects using sketches and level-of-detail controls.

Recently, Wretborn *et al.* [315] presented a realistic model for white-water simulation. Their method enhances simulations with (tiny) bubble and foam detail by a stable coupling scheme between bubbles and water, a novel bubble emission scheme, and manifold advection for accurate foam tracking.

For the simulation of glugging, Boyd and Bridson [39] proposed the MultiFLIP method, which extended the FLIP method to two-phase flows. They treated not only liquid but also air as incompressible phases, both modeled via particles. This (re)produces, among other effects, the glugging effect.

Ando *et al.* [10] introduced a stream function-based solver as a FLIP variant. In this approach, the stream function ψ is used to determine the divergence-free velocity field $\mathbf{u}$, given by $\mathbf{u} = \nabla \times \psi$. Interestingly, their work shows that solvers based on stream functions are just as viable

as regular pressure solvers. The method is able to simulate glugging without modeling the second phase (air) explicitly.

2.6.3 *Spray and Splashing*

Spray and splashing are very common phenomena in fluid scenes (Figure 2.20). For scenes with intense collisions like turbulence, the final visual effect largely depends on the fidelity of the spray and splash simulation.

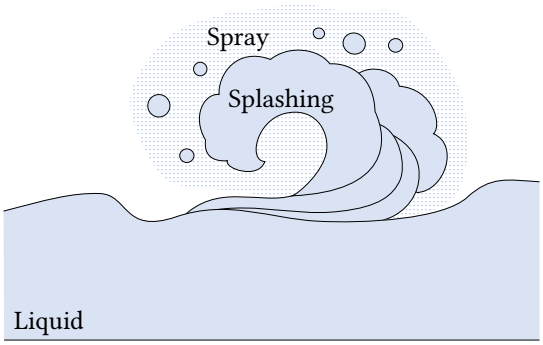


Figure 2.20: Schematic diagram of spray and splashing. Spray and splashing are different in simulation scales. Compared with splashing, spray is composed of finer droplets.

Nielsen and Østerby [214] modeled spray as two-way coupled two-continua with different volume fractions to achieve realistic spray motion. However, a grid-based density field cannot capture the motion of a single droplet. In contrast, Jones and Southern [157] focused on efficient physics-based droplet interaction. They introduced coalescence, separating, and fragmenting collision outcomes into a novel particle interaction model to simulate droplets. This provides a ballistic particle system for liquid droplets and spray.

Yang *et al.* [330] focused on spray simulation, such as that arising from high-speed/violent liquid streams. Similar to the work of Patkar *et al.* [226], they also used a hybrid Lagrangian–Eulerian model (with FLIP components in their case) to model mixture phenomena with high fidelity. Their efficient CUDA implementation facilitated the integration of a FLIP solver tailored for liquid behaviors with an additional solver dedicated to droplet and spray effects. This combination allows for the realistic rendering of dynamic water features such as waterfalls and fountains in simulations.

Guo *et al.* [116] addressed the stability challenges encountered in the two-phase lattice Boltzmann model (TP-LBM) by introducing a novel density-aware sub-grid-scale model. Their approach can uniformly sim-

ulate different gas–liquid phenomena, allowing for realistic and visually compelling representations of gas–liquid flow dynamics.

Li *et al.* [179] proposed a multiphase flow method to simulate complex effects, such as bubbling, glugging, wetting, and splashing. A single model captures all these effects by building on the kinetic-based Lattice Boltzmann Phase-Field (LBM-PF) method. The interface motion is governed by the conservative phase-field equation

$$\frac{\partial c_\Phi}{\partial t} + \nabla \cdot (c_\Phi \mathbf{u}) = \nabla \cdot \left[\mathcal{M} \left(\nabla c_\Phi - \frac{4}{\xi} c_\Phi (1 - c_\Phi) \mathbf{n}_{\text{inter}} \right) \right], \tag{2.25}$$

where the phase field c_Φ represents the percentage of the flow phase, the mobility $\mathcal{M}$ controls the degree of interface splitting, ξ denotes the interface width, and $\mathbf{n}_{\text{inter}}$ corresponds to the interface normal. This method is highly general and versatile and can produce results comparable to industry-standard CFD simulations like the method from Fakhari *et al.* [81].

In a similar vein, Li *et al.* [180] introduced a kinetic approach to multiphase fluids. Their scheme incorporates an accurate collision model and is able to robustly capture intricate and visually appealing behaviors, such as the injection of gas into the liquid.

2.7 FINE DETAIL ENHANCEMENT

Capturing high-frequency details of fluid surfaces, such as vortices, waves, and turbulence, is key to enhancing the realism or artistry of fluid simulations. The use of higher-order advection numerical methods or finer discretization can alleviate the numerical dissipation problems inherent in fluid simulation. However, this creates very high memory and computation time costs. To cope with this, in the last decade, several methods have been designed to specifically add fine details to a coarse fluid simulation. We group these into three classes: reduced-dimensional simulation on the fluid surface only (Section 2.7.1), dynamic methods to combat numerical dissipation (Section 2.7.2), and data-driven methods (Section 2.7.3).

2.7.1 *Reduced-dimensional Simulation on the Fluid Surface Only*

These techniques decouple the surface simulation from the volume simulation, allowing a secondary model with high-resolution surface features to be added to a coarse (thus fast to compute) volume fluid model. We further split methods in this class into embedding techniques, 2D water wave simulation, and surface tracking and reconstruction as follows.

Embedding techniques. The Closest Point Method (CPM) [246] is a numerical method for solving PDEs on surfaces. Unlike 2D surface

parametrization, CPM typically uses a 3D Cartesian grid to discretize narrow spatial bands around the surface, which allows it to scale according to the complexity of the surface rather than the volume.

Auer *et al.* [16] used CPM to simulate fluid effects on static surfaces in real time. Auer and Westermann [15] followed up with a semi-Lagrangian CPM that alleviated some technical limitations of previous applications of CPM to deformed surfaces. Their method is unconditionally stable for surface deformation. Kim *et al.* [163] used CPM to explicitly perform high-resolution wave simulations on the liquid surface. They used the iWave algorithm [288] to produce more realistic water waves than the traditional wave equation. The iWave equation is given by

$$\frac{\partial^2 H}{\partial t^2} = -g\sqrt{-\nabla^2}H, \tag{2.26}$$

where H is the fluid height, g is the gravity constant, and $\sqrt{-\nabla^2}$ is a fractional Laplacian operator.

Mercier *et al.* [199] added a sub-grid wave model to particle-based liquid simulations to enhance such simulations with additional turbulence. Goldade *et al.* [109] worked to eliminate sub-grid errors in underlying surfaces and reduced artifacts in narrow bands around surfaces. Morgenroth *et al.* [204] used CPM to efficiently compute high-resolution 2D simulations on rough surfaces. Their method is similar to the work of Auer and Westermann [15] but adds mass and momentum conservation and can produce interesting effects, such as oil films on water surfaces and thermal convection on a hemisphere.

2D water wave simulation. To reduce computational complexity while retaining surface detail, some researchers have investigated the simulation of water waves on fluid surfaces. These water wave simulations operate independently of the physical simulation system's degrees of freedom, allowing for the creation of high-frequency visual detail without the need to increase the overall simulation resolution.

In response to the inability of the shallow water equations (SWE) to capture motion details such as wakes, Pan *et al.* [222] used a 2D discrete vortex method to capture the wake behind a moving rigid body. Their method requires only a small number of wake particles and is sufficiently fast for real-time applications. However, this method fails to handle the complex wake patterns caused by vortex stretching and tiling in 3D flow. Azencot *et al.* [18] used a scalar vorticity function on 2D domains to describe the vortex behavior of fluid surfaces, greatly simplifying the analysis and simulation of fluids. Later, Azencot *et al.* [17] simulated the complex behavior between multiple waves by solving the EPDiff [127] on arbitrary triangle meshes. EPDiff is the zero-dispersion limit of the Camassa–Holm (CH) equation for shallow water waves in

one spatial dimension. They achieved simulating a series of phenomena including annihilation, recreation, splitting, and merging.

A key challenge in wave simulation is handling the coupling between waves and obstacles. Canabal *et al.* [46] generated rich water waves using a dispersion kernel as the spatially variant filter and further simulated interactions between waves and static or moving obstacles by modulating this dispersion kernel. The dispersion relation under the Airy wave theory [4] defines the propagation speed of each wave u_c as

$$u_c = \sqrt{\left(\frac{g}{w} + \frac{\gamma}{\rho} w\right) \tanh(wH)}, \tag{2.27}$$

where w is the wave number, g is the gravitational constant, ρ and γ are the density and surface tension of the fluid, respectively, and H is the fluid height. Jeschke and Wojtan [150] simulated the movement and interaction of a large amount of waves by a wavefront tracking algorithm with multivalued function interpolation. Their method can model the dispersion, refraction, reflection, and diffraction of waves well, but it only handles scenes with static obstacles. Later, they introduced the concept of wave packets [151], which can handle the interaction of water waves and moving objects. They used an improved Lagrangian particle method to simulate the diffusion of water waves to add more visual detail. However, this method cannot be extended to moving 3D fluid simulations of surfaces. The same problem was addressed by Skrivan *et al.* [262], who decoupled the wave resolution from the simulated resolution using Lagrangian wave packets. This method significantly increases the visible detail on the fluid surface as a post-processing step.

Creating large open-water animations and adding wave detail is a common requirement for a variety of interactive and offline applications. Implementing this requires visual quality *vs* computational resources to be carefully balanced. Nielsen *et al.* [215] proposed a wave synthesis technique based on the Fourier transform to enhance the details of wave animation. However, wave–obstacle interactions are difficult to incorporate into the spectral solver. Keeler and Bridson [158] proposed an efficient surface-only simulation of deep ocean waves and used a new indirect boundary integral equation to deal with wave–solid boundary interactions. The method of fundamental solutions (MFS) was also used to generate realistic waves behind moving obstacles. Schreck *et al.* [253] proposed a novel discretization for MFS using wavelets and achieved naturally-looking wave interactions with complex boundaries. Their method achieved impressive results on a large-scale ocean scene. Jeschke *et al.* successively developed two interactive systems for the simulation of large ocean scenes that can handle detailed wave features [149] and coupled interactions with complex terrain [148], respectively. Recently, Schreck and Wojtan [254] proposed a coupled method of 3D liquid simulation and 2D wave propagation

to simulate infinitely large bodies of water and fine surface wave detail. An empirically-driven error compensation method was also used to remove coupling errors from the simulation to achieve a seamless transition between 2D and 3D.

Surface tracking and reconstruction. Tracking and reconstructing fluid surfaces is important for generating realistic animation effects. This is difficult to achieve due to the complex shapes and frequent topological changes of fluids.

For Eulerian fluid simulations, robust handling of surface triangle mesh splitting and merging can remove visual artifacts to preserve important surface features. Bojsen-Hansen *et al.* [36] proposed a method for tracking the topological evolution of surfaces that can solve the wave equations on lower-resolution fluid surfaces to synthesize high-frequency details. Later, Bojsen-Hansen and Wojtan [37] presented a novel physics-based surface fairing method that solved the physical and topological artifacts arising from coupling high-resolution surface trackers with low-resolution fluid simulations by introducing an error metric and surface correction force. Edwards and Bridson [79] presented a new approach to adaptive fluid simulation. They tracked explicit triangulated mesh surfaces and used the p-adaptive Discontinuous Galerkin (DG) method [14] within detailed cut-cells near the surface. By using coarse-grid fluid simulations, the treatment of dynamics is guaranteed to be physically consistent while reducing computational costs. Chentanez *et al.* [55] devised a grid-free surface tracking method that deals with topological changes by removing overlapping triangles and performing effective triangulation of the generated holes. This method can be used in both mesh-based and particle-based simulations.

Inaccurate detection of free surface particles in particle-based fluid simulations can lead to unrealistic artifacts. Also, irregularly distributed particles can make the reconstructed surface bumpy. To address the fact that particles do not keep connectivity information, Yu *et al.* [339] proposed periodically projecting surface meshes to match implicit surfaces defined by fluid particles. This method allows the simulation of high-resolution surface waves without the limitation of particle resolution. Later, Yu and Turk [337] proposed a method for reconstructing surfaces in particle-based fluid simulations. They utilized a stretched anisotropic smooth kernel to represent each simulated particle, resulting in a greatly improved surface quality (Figure 2.21). In this thesis, we have incorporated the anisotropic smoothing kernel approach into a highly efficient screen space rendering method. Further details on this implementation are provided in Section 5.3. Sandim *et al.* [247] proposed a fast free surface detection method that only requires the positions of particles to identify surface particles without using kernel functions or normal vectors. This method is applicable to cases with non-

uniform particle distributions and complex free surface deformations. Dagenais *et al.* [72] used an explicit mesh projection method based on signed distance fields to preserve surface detail and introduced a new topology matching operation to maintain consistency between explicit surface and particle behavior.

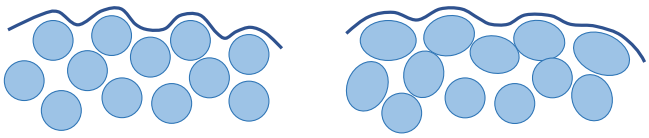


Figure 2.21: Schematic diagram of isotropic and anisotropic particles. In contrast to isotropic particle fluids (left), anisotropic particle fluids (right) have a smoother surface.

2.7.2 *Dynamical Methods for Reducing Numerical Dissipation*

The advection-projection method [58] leads to *numerical dissipation*, which results in kinetic energy decay and suppression of motion, such as vortices and turbulence. Bulk enhancement methods aim to improve the whole fluid volume rather than only its free surface. We further group such methods that aim to improve system energy conservation and detail preservation by reducing numerical dissipation in vorticity confinement, vortex-based methods, and various variants of dynamics solvers, as described next.

Vorticity confinement methods. Such methods are based on the principle of vorticity conservation, which adds a vorticity control term to restrain the diffusion of vortices, thus simulating fluid dynamics problems, such as turbulence and vortex streets, without dissipation. Fedkiw *et al.* [84] first applied the vorticity confinement method to smoke simulations by adding an additional force field to maintain the airflow vorticity. Second-order vorticity confinement (VC2) [120] further improved this method by ensuring momentum conservation. The confinement term of VC2, $\mathbf{f}_{\text{conf}}$, is given by

$$\mathbf{f}_{\text{conf}} = -\Delta x \nabla \times (\alpha \boldsymbol{\omega} - \zeta \mathbf{m}), \tag{2.28}$$

where Δx is the grid size, α and ζ are the positive and negative diffusion coefficients, respectively, $\boldsymbol{\omega}$ is the angular velocity, and $\mathbf{m}$ is the harmonic mean of the local vorticity stencil. Lentine *et al.* [174] improved the vorticity confinement model by allocating global momentum to ensure momentum conservation. Jang *et al.* [147] applied the multi-level vorticity confinement method to simulate water turbulence to capture large- and small-scale vortices and complex flow details. He and Lau [120] proposed adaptive adjustment of the positive diffusion

term to balance constraints, which broadened the stability conditions of the VC2 method and made it capable of generating highly turbulent flows. However, the vorticity constraint method can only enhance existing vortices or turbulence and may not be effective for other types of flows, such as laminar flow. Moreover, computational costs can increase significantly due to additional constraints, especially for large-scale complex fluid scenarios.

Vortex-based methods. The potential vorticity field can be effectively modeled using vortex-based methods. These methods simulate the vorticity of the velocity field rather than the velocity field itself, so this automatically guarantees a divergence-free velocity field and removes numerical dissipation. Most such methods are Lagrangian and model the vorticity form $\boldsymbol{\omega}_v$ of the Navier–Stokes equations as

$$\frac{\partial \boldsymbol{\omega}_v}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega}_v = (\boldsymbol{\omega}_v \cdot \nabla)\mathbf{u} + \mu \nabla^2 \boldsymbol{\omega}_v, \tag{2.29}$$

$$\nabla \cdot \boldsymbol{\omega}_v = 0. \tag{2.30}$$

This formulation represents the vorticity distribution as a superposition of singularities.

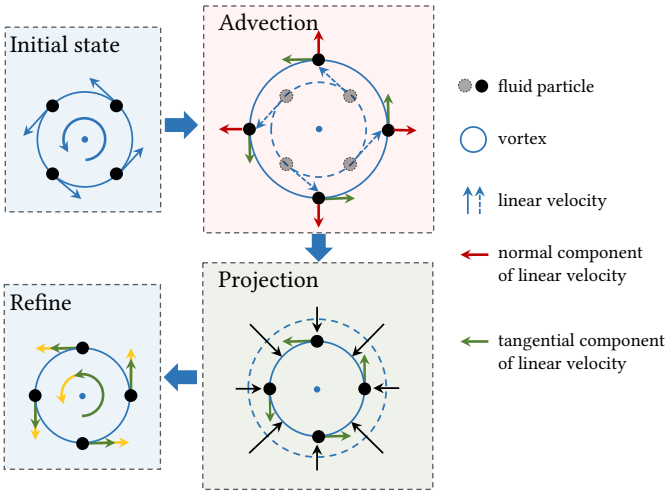


Figure 2.22: Schematic diagram of the causes of vorticity dissipation and methods of refinement [187].

Zhang *et al.* [347] proposed a new scheme named IVOCK, which aims to solve the errors and energy loss caused by the self-advection step through compensating for vorticity error. However, this method is only applicable to fluid simulation on uniform grids. Liu *et al.* [187] extended

this idea to particle-based turbulent detail simulation (Figure 2.22). Recently, Xiong *et al.* [322] proposed a vortex segment method to simulate flows with strong anisotropic vortical features. Compared with existing Lagrangian vortex particle methods, this method can more vividly model complex phenomena, such as the splitting and reconnection of two vortex tubes or vortex shedding near a solid boundary.

The main challenge of vortex methods is the handling of fluid–solid coupling and creating vortices at this coupling boundary. For this, Golas *et al.* [107] proposed a combination of Eulerian simulation and vortex singularity bases. By using Lagrangian vortex elements inside the fluid and enforcing boundary conditions in the Eulerian mesh, robust interaction of free surfaces and non-rigid obstacles can be achieved. Zhang *et al.* [348] used a FLIP approach to solve the Navier–Stokes equation using viscous particle strength exchange, handling the momentum transformation at the solid boundary effectively. Liao *et al.* [182] proposed a new wall-bounded turbulent smoke simulation method, which introduced particle–particle interactions to traditional vortex filament mesh calculations to accurately capture the vortices and thin turbulence generated by smoke–obstacle interactions.

Variants of dynamics solvers. Additional methods improve on current advection–projection solvers or extend classical dynamical methods to achieve detail enhancement. In the advection–projection step, detail, and energy preservation are greatly improved by the introduction of detail capture and shape correction techniques [317], the use of energy-preserving reflection operators [343], and feature mapping with convectors [208].

Yang *et al.* [332] first introduced the Clebsch wave function [62] as a system scaling variable to evolve Eulerian flow fields, which significantly improved the ability to generate and sustain vorticity in simulations of various gases and liquids. Later, to solve the numerical instability of Clebsch's method near dynamic interfaces, Xiong *et al.* [323] proposed a new wave function correction scheme and extrapolation algorithm, which achieved detailed simulation of various vortex structures on free surfaces. Recently, Feng *et al.* [88] proposed a numerical method for solving the Navier–Stokes equations based on the pulse gauge transformation, which can generate rich vortical details by treating the fluid pulse as an auxiliary variable.

Liu *et al.* [188] and Li *et al.* [178] created two turbulence simulation methods using an adaptive multi-relaxation scheme and statistical mechanics, respectively. Later, Lyu *et al.* [193] further improved the boundary treatment of dynamic solids, enabling the simulation of fluid–solid coupling between thin structures and turbulent fluids.

The micropolar fluid model is an extension of the classical Navier–Stokes equation, which takes into account not only the linear but also angular velocities of the fluid particles, thus enhancing the eddy and

turbulence details of the fluid. Bender *et al.* [30] first used the micropolar fluid model to simulate the turbulence phenomenon of non-viscous fluids. Subsequently, they post-processed the foam phenomenon on this basis to significantly improve the realism of the visual effect.

2.7.3 *Data-driven Methods for Detail Enhancement*

Texture synthesis. Geared toward the production of artistic effects based on fluid elements, texture synthesis is the technique of choice for adding detailed features, such as higher-resolution images, to surfaces. As a post-processing method, texture synthesis achieves detailed surface features through patch or style transfer based on deep learning.

Patch-based texture synthesis maps image textures or simulated features to the target flow field, improving the appearance of the source simulation by matching the target dataset features. Jamriška *et al.* [146] used per-pixel best-fit search to achieve rich visual effects through 2D input image appearance transfer. However, this method is limited to image space synthesis and is difficult to extend to 3D fluid surfaces. Gagnon *et al.* [98] proposed a temporally coherent patch-based texture synthesis method to handle scenes with significant deformation and topological changes. This approach aims to maintain a Poisson disk distribution of patches on a free surface to find the optimal parameter values and locations of time-varying patches. For the ghosting problem of overlapping patches in the work of Gagnon *et al.* [98], Gagnon *et al.* [97] followed up with a solution scheme based on patch erosion. They used feature-aware erosion to remove patch distortion textures to ensure realism of the mapping.

In contrast to surface texture synthesis, *Deep Neural Networks (DNN)* perform various stylization tasks on volumetric data. Sato *et al.* [248] proposed a style transfer method that migrates high-resolution turbulent details to low-resolution flow fields, which speeds up the fine surface detail simulation almost 30-fold. In addition, they used an optimized texture synthesis method to solve the problem of discontinuity at the patch boundary. Kim *et al.* [159] first proposed a transport-based neural style transfer algorithm that enables automatic conversion of the semantic structure of 2D images into 3D smoke simulations. The method achieves complex artistic effects by optimizing the transport of smoke to the desired stylized velocity field at each time step. However, this method cannot handle the transfer of color information. Therefore, the authors further redefined the method in a Lagrangian setting to ensure better temporal consistency and support for color stylization [160]. Unlike stylization methods that focus on fluid simulation shapes, Guo *et al.* [115] proposed a Stylizing Kernel Prediction Network (SKPN) aimed at stylizing physical color appearances. The method can easily generate the user's desired color appearance without complex parameter tuning.

Upsampling methods for super-resolution. Generating super-resolution simulations from low-resolution inputs is challenging. With the popularity of machine learning techniques, recovering fine details of fluids through super-resolution reconstruction techniques or upsampling methods has received increasing attention.

Zhang and Ma [350] proposed a spatio-temporal extrapolation technique that enables high-resolution flow features on coarse grids. Chu and Thuerey [60] enhanced the turbulence detail of the smoke simulation on the coarse grid by using local patch descriptors. Um *et al.* [293] proposed a deep neural network to capture small-scale splashed droplet details from low-resolution liquid simulations. Xie *et al.* [321] used a conditional generative adversarial network with a temporal discriminator to directly generate advected quantities with highly detailed and temporally coherent features for smoke simulation. CNNs have been used to create matching models to correct the shape of low-resolution smoke simulations [318] and estimate physical parameters to guide the reconstruction of high-resolution velocity fields [176]. Bai *et al.* [21] used a dictionary-based neural network for fluid upsampling. However, the choice of their training set was somewhat limited, and the spatial and temporal consistency of the results could not be guaranteed. They next significantly improved the prediction quality of the network by adding filtering to the training process [22]. Roy *et al.* [242] proposed a DNN-based method for improving the resolution of coarse particle liquid simulations.

2.8 FLUID CONTROL

The wealth of methods for fluid simulation surveyed so far leads to a key question: How do we *control* such simulations? Although a method can technically produce highly accurate results, a critical objective is to enable its users to steer the method toward the desired results. We next group and discuss fluid control methods in three classes using different control perspectives: scenario editing (Section 2.8.1), artificial effects (Section 2.8.2), and media-directed formation (Section 2.8.3).

2.8.1 *Scenario Editing*

Scenario editing steers the fluid by generating new simulation scenarios based on existing simulation results without losing the characteristics of the original simulations (Figure 2.23). On the positive side, such control is technically the closest to how a simulation works, so it can steer the fluid most directly. On the negative side, this control requires the end users' advanced skills and understanding of the underlying simulation and overall fluid flow technicalities. We further divide scenario edit-

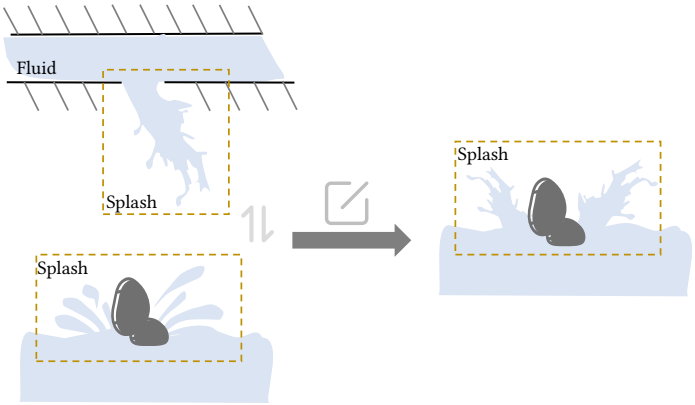


Figure 2.23: Schematic diagram of scenario editing based on changing an existing simulation to achieve desired effects.

ing into three sub-types based on the implementation approach: target-guided editing, adjustable editing, and camera-based editing.

Target-guided editing. Such methods modify an existing fluid simulation to match a given target, *e.g.*, a higher resolution. Gregson *et al.* [114] connected low-resolution smoke capture with its velocity field. They treated the pressure projection as a proximal operator and tracked the fluid by estimating its velocity. Through advection, their method obtained a high-resolution re-simulated smoke. Forootaninia and Narain [93] successfully guided high-resolution smoke flow by replacing its low-frequency component with a given guiding field. Generally, the guiding task is seen as an optimization problem that minimizes errors. This optimization problem was solved efficiently by using a fast primal-dual method [142]. To achieve a more desirable artistic effect, it is essential to guide smoke animation in such a way that it aligns with one or multiple target density keyframes provided by the artist. To address this control problem, Pan and Manocha [224] formulated it as a space-time optimization. They employed an alternating direction method of multipliers (ADMM) optimizer [40] to derive a dense sequence that forces the smoke to meet the desired target shape. Tang *et al.* [285] proposed an advanced method that effectively addresses both the issues of unconstrained optimization and high-dimensionality of the parameter space simultaneously.

Adjustable editing. These methods aim to control, edit, or resize fluid simulation results. Raveendran *et al.* [234] focused on control with an emphasis on the liquid surface and proposed a method that provides the animator with multiple control levels for creating high-quality fluid animations. Later, Raveendran *et al.* [235] proposed a smooth blending

method to interpolate between two or more existing pre-computed liquid simulations. With their method, one can generate hundreds of different plausible results at interactive rates with potential applications in games and virtual reality. Sato *et al.* [249] proposed a smoke blending method to help animators synthesize the desired fluid animation. Velocities at the boundaries are interpolated by minimizing an energy function. This approach significantly reduces computational costs by reusing existing flow data instead of creating realistic fluid animations by numerical simulation. *Fluid carving* is another way to edit fluid simulations. By utilizing seam carving, efficient and effective resizing of 4D fluid simulation data can be achieved [91]. Flynn *et al.* [92] proposed a lattice-guided seam computation method that can overcome the limitation of rectangular boundary and that reduces calculation time.

Camera-based editing. In the context of large-scale scene simulation, due to computational cost considerations, it is often necessary to perform coarse-grained simulations of the entire world and subsequently integrate finer-grained details into the scene. To integrate two simulated fluid scenes seamlessly, Bojsen-Hansen and Wojtan [38] presented a fluid modification approach with "non-reflecting" boundary handling. They extended the simple Perfectly-Matched Layers (PMLs) [32] method to handle coupling inflow/outflow boundaries with varying spatial and temporal conditions. The boundary can be modified easily during the simulation, and it handles the multi-resolution combination. Stomanuykhin and Selle [271] introduced the fluxed animated boundary (FAB) method, in which the boundary can have a custom shape and vary over time, with materials outside the boundary dynamically removed using volume flux.

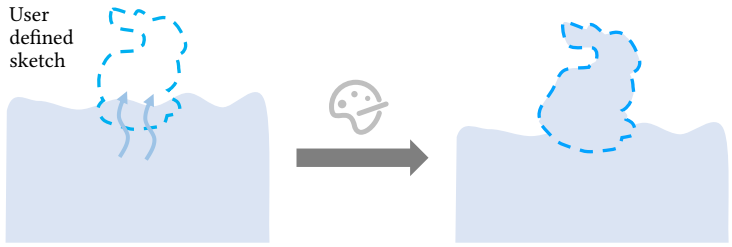


Figure 2.24: Schematic diagram of artificial effects: Artist-directed control to achieve non-physical effects.

2.8.2 *Artificial Effects*

One often needs to artificially edit and control fluids to achieve specific artistic effects. Different from scenario editing, *artificial effects* add new characteristics to a simulation by artificially guiding the formation of

fluid shapes or movements during the process (Figure 2.24). Due to the complex motion of fluids, keyframe animation, which involves controlling the flow of fluids to match keyframes, is a commonly used method for fluid control to reduce unrealistic effects in simulations. However, manually designed frames often lack volume preservation and exhibit excessive smoothness, resulting in the loss of simulation details.

Pan *et al.* [223] proposed a local control method instead of globally manipulating the entire fluid, allowing users to edit and control fluid shapes in specific regions using a brush-like tool. However, controlling the simulation process between keyframes is challenging. To address this issue, Lu *et al.* [191] drew inspiration from skeletal animation techniques. They introduced a method that controls fluid motion by manipulating a point cloud with rigid body motion and incompressible deformations, subsequently performing skinning operations on the point cloud. Similar to the work of Lu *et al.* [191], Yan *et al.* [326] applied conditional generative adversarial networks to generate fluid splashes based on simple user-defined sketch input. Control particles with attractive forces provide an efficient way to reproduce complex motion using pre-computed templates [252]. In this work, a set of shape-constraint particles were seeded and a repulsion force field was computed to control the shape of the final result.

2.8.3 *Media-directed Formation*

Pure physics simulations often suffer from significant computational time requirements, making them impractical for real-world production applications. *Media-directed formation* aims to set up simulation scenarios based on real-life videos or images of the fluid, reproducing real-world scenes (Figure 2.25). These methods estimate fluid properties such as volume, density, motion, and style from visual data.

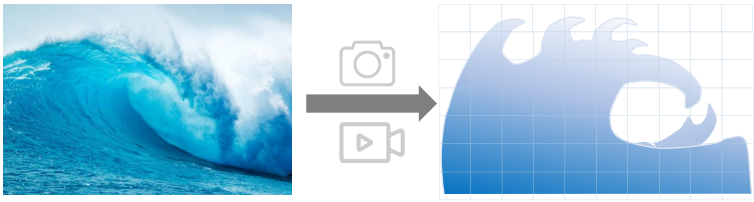


Figure 2.25: Schematic diagram of media-directed formation; images or videos are used to reproduce real-world scenes.

Okabe *et al.* [217] focused on reconstructing a detailed 3D model of a fluid volume, such as smoke or liquid, from sparse multi-view images. This involved sparse reconstruction and appearance transfer to capture the underlying structure and enhance the visual fidelity of the reconstructed fluid volume. Unlike Okabe *et al.* [217], Eckert *et al.* [77] es-

timated both the density and motion of a fluid from a single view or sequence of images without the need for multiple views. Nie *et al.* [213] proposed a fluid reconstruction and editing model to generate particle-based simulations based on monocular videos. Using the SPH method with external forces, they could obtain a simulated fluid volume under the guidance of a pre-processed water surface.

2.9 SPECIAL FLUIDS

2.9.1 *Highly Viscous Fluids*

Viscosity is an attribute that measures the ability of a fluid to resist deformation at a given rate. With viscosity, moving fluids will generate internal stress responding to the deformation, which causes energy dissipation of the fluids and affects their behavior. For low-viscosity fluids like water, inertial forces are dominant. For high-viscosity fluids like molasses and chocolate sauce, viscosity leads to special phenomena like bulking and coiling. Moreover, for different kinds of fluids, the viscous characteristics can vary significantly with their shear rate (see Figure 2.26). The simulation of high-viscosity fluids has attracted recent interest in computer graphics.

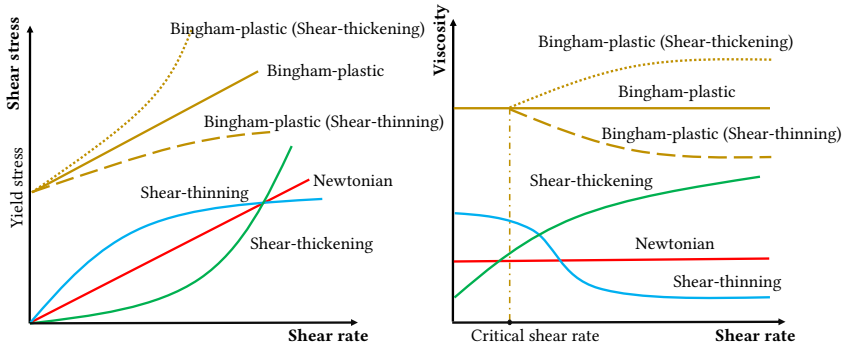


Figure 2.26: Diagrams illustrating the relationship between shear rate and shear stress, as well as shear rate and viscosity, for various highly viscous fluids, with design inspiration taken from Zhu *et al.* [352]. (a) Plots of shear stress versus shear rate for different highly viscous fluids. (b) Plots of viscosity versus shear rate for different highly viscous fluids.

Newtonian fluids. Following Newton's viscosity law, the viscosity of a Newtonian fluid (incompressible and isotropic) can be expressed by

a material parameter μ called dynamic viscosity coefficient. Using this, the inner viscous stress tensor $\boldsymbol{\tau}$ is computed as

$$\boldsymbol{\tau} = 2\mu\mathbf{E}, \tag{2.31}$$

where $\mathbf{E}$ is a symmetric strain rate tensor that describes the shear strain rate. This equation indicates that the viscous stresses of Newtonian fluids are linearly correlated to the local strain rate at every point. Using the spatial derivatives of the velocity field, the strain rate tensor $\mathbf{E}$ can be defined as

$$\mathbf{E} = \frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T). \tag{2.32}$$

Substituting Equation (2.31) and Equation (2.32) into the viscosity force field formulation $\mathbf{f}_{\text{vis}} = \nabla \cdot \boldsymbol{\tau}$ and adding the incompressible condition $\nabla \cdot \mathbf{u} = 0$, the viscosity force field $\mathbf{f}_{\text{vis}}$ can be computed as

$$\mathbf{f}_{\text{vis}} = \mu\nabla^2\mathbf{u}. \tag{2.33}$$

This gives the viscosity term in the momentum equation, corresponding to the viscosity term in the form of kinematic viscosity in Equation (2.6).

Discretizing the viscosity term is a challenging problem for SPH-based methods. Takahashi *et al.* [279] proposed an implicit Euler integration to solve the viscosity term separately, using two SPH first derivatives to discretize the strain tensor and the divergence of the stress tensor, respectively. This method supports a larger range of viscosity and time step values, but a second-ring neighbor computation is required, which impacts efficiency. Peer and Teschner [229] decomposed the velocity gradient into three tensors: spin rate, expansion rate, and shear rate. A user-defined viscosity parameter modifies the shear component, which describes the dissipation due to viscosity. This leads to a target velocity gradient that is used to reconstruct the final velocity field with a first-order Taylor approximation. Because the velocity gradient field is decomposed, the linear system for solving the velocity field can be handled separately. However, shear viscosity does physically affect the rotation component in the velocity gradient because of the tangential component in rotation. Peer *et al.* [230] extended this method using a vorticity diffusion scheme. The spin rate tensor in the target velocity gradient is modified by solving a vorticity diffusion process, which uses the viscosity parameter in the work of Peer and Teschner [229] such that vorticity damping is introduced to achieve more realistic effects. Instead of using the strain rate, Weiler *et al.* [306] introduced an implicit viscosity solver based on the Laplacian of the velocity field in Equation (2.33). With a symmetric form of the approximation discretization of the viscosity term [202], an implicit linear system for a new velocity field can be obtained.

The above-mentioned solvers separate the solving of pressure and viscosity, which reduces accuracy and cannot generate free surface details. Larinov *et al.* [171] introduced a unified pressure–viscosity solver

based on implicit variational unsteady Stokes flow problems for grid-based methods, where the inertial force is considered to improve accuracy and achieve a wider viscosity range. Combining the semi-implicit equation of correlation pressure (SIMPLE) method with SPH, Liu *et al.* [186] used the result of the pressure Poisson equation to improve the pressure in the viscosity solver in an iterative process, which converges to a globally optimal solution.

Even with a stable solver, mimicking the viscosity coefficient of the target fluid is essential to realistically simulate highly viscous fluids. Takahashi and Lin [283] proposed a framework to find the required viscosity parameter from real videos of highly viscous fluids by minimizing an objective function that evaluates the difference between the silhouettes extracted from video frames and those obtained from the simulation.

Non-Newtonian fluids. Such fluids do not follow Newton's viscosity law but rather show a non-linear relation between shear stress and strain rate. For example, the viscosity of a shear-thickening or dilatant fluid (*e.g.*, starch paste) increases when the shear rate increases; the opposite happens for a shear-thinning or pseudoplastic fluid (*e.g.*, ketchup). Some non-Newtonian fluids have properties of solids, such as Bingham plastic fluids like toothpaste. Hence, it is impossible to use a constant viscosity for non-Newtonian fluids, and appropriate constitutive models are required to simulate such fluids.

The Carreau–Yasuda model is a well-known method to simulate non-Newtonian fluids by defining a shear-rate-related viscosity μ as

$$\mu = \mu_\infty + (\mu_0 - \mu_\infty)[1 + (R\dot{\epsilon})^a]^{(n-1)/a}, \tag{2.34}$$

where μ_0 is the zero-shear viscosity, μ_∞ is the infinite viscosity, $\dot{\epsilon}$ is the shear rate, R is the relaxation time that scales the shear rate, n is a power law index, and a models the transition smoothness between the Newtonian plateau and power law regime. When $n = 1$, this model becomes a Newtonian fluid with dynamic viscosity μ_0. For shear-thinning fluids ($n < 1$), as the strain rate increases, viscosity will vary from μ_0 to μ_∞. For shear-thickening fluids ($n > 1$), the viscosity increases as the shear rate increases.

Bingham plastic fluids are another kind of typical non-Newtonian fluid that behaves as a rigid body at low stress but flows as a Newtonian viscous fluid once the yield stress is exceeded. For many viscoplastic fluids, the stress curve of the flowing part is nonlinear when the shear rate exceeds a critical value. To capture shear thickening/thinning, μ is modeled by the Herschel–Bulkley model given by

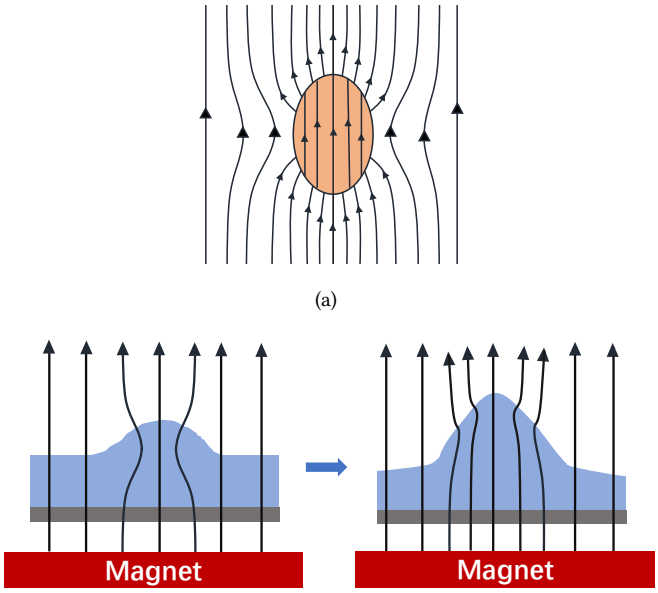
$$\mu(\dot{\epsilon}) = \begin{cases} k\dot{\epsilon}^{n-1} + \tau_0\dot{\epsilon}^{-1}, & \dot{\epsilon} > \dot{\epsilon}_0 \\ \mu_0, & \dot{\epsilon} \le \dot{\epsilon}_0 \end{cases} \tag{2.35}$$

where k is the consistency coefficient, τ_0 is the yield stress, and $\dot{\epsilon}_0$ is the critical shear rate. Similar to Equation (2.34), when $n = 1$, this model describes the ideal Bingham plastic, $n > 1$ models shear thickening Bingham plastic, and $n < 1$ models the shear thinning Bingham plastic fluid.

Recent work used the above two models to simulate non-Newtonian fluids. Zhu *et al.* [352] simulated various co-dimensional features of different non-Newtonian fluids, *e.g.*, shear thinning and thickening for Bingham plastics, and elastoplastics. The Carreau–Yasuda model for non-Newtonian fluids was used on a multi-level-set model; semi-implicit methods were used for elasticity and variable viscosity. On the rims of thin fluid sheets, viscosity had an improved treatment to yield a twisting motion. Yue *et al.* [340] used the non-Newtonian Herschel–Bulkley model to simulate dense foams composed of microscopic bubbles using MPM. They also proposed a particle resampling method for MPM and a tearing model to simulate tearing/connectivity recovery by explicitly handling the weakening regions detected from space. Mixtures of non-Newtonian fluids were studied by Nagasawa *et al.* [209]. Using the Herschel–Bulkley model, a nonlinear blending model that satisfies the five blending laws [245], along with mass conservation, was proposed to capture non-Newtonian fluid mixture behavior. For viscoelastoplastic materials, a constraint-based method [24] extended position-based dynamics to simulate elastoplastic and highly viscous fluids by recasting a constitutive model of viscoelasticity, which defined governing equations for a conforming tensor.

2.9.2 *Ferrofluids*

The dynamic interactions of ferrofluids — liquid media responsive to external magnetic fields — have emerged as an area of considerable interest within the computer graphics research community. These magnetically active fluids, originally conceived by NASA to facilitate fuel transfer in spacecraft under microgravity conditions, derive their magnetic properties from the incorporation of nanoscale magnetic particles. Upon exposure to an external magnetic field, these dispersed particles within the ferrofluid polarize, thereby generating a distinct internal magnetic field. This induced magnetic field, working synergistically with the external one, is pivotal to the magnetization process of the ferrofluid, as shown in Figure 2.27. The computer graphics simulation of these captivating fluids, however, has not received significant attention until the very recent pioneering work of Michels *et al.*, which directed attention towards this specialized field [132]. We recall several key insights from their contributions in the subsequent discussion.



(a)

(b)

Figure 2.27: Schematic representation of ferrofluids in interaction with a magnetic field. (a) Superposition of a uniform vertical magnetic field and the magnetic field originating from an ellipsoid magnetization. There is a pronounced discontinuity in the magnetic field on the surface, most significantly at the extremities where there is a sharp escalation in the field strength. (b) A surface perturbation in the magnetized ferrofluid, resulting in a localized concentration of magnetic induction lines. The ferrofluid is drawn towards this bump due to a heightened field strength, which consequently enhances the gradient, amplifying the perturbation and leading to the formation of a spike.

The interrelationships of these spatial magnetic fields, both internally produced and externally imposed, conform to the well-known Maxwell equations

$$\begin{aligned} \nabla \cdot \mathbf{B} &= 0, \\ \nabla \times \mathbf{H} &= \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}, \end{aligned} \tag{2.36}$$

where $\mathbf{B}$ is the magnetic flux density describing the spatial magnetic field, $\mathbf{H}$ is the total magnetic field intensity, $\mathbf{J}$ is the free current density, and $\mathbf{D}$ is the electric displacement field. The ferrofluids further discussed in this section comply with the following model

$$\begin{aligned} \mathbf{B} &= H_0(\mathbf{H} + \mathbf{K}), \\ \mathbf{H} &= \mathbf{H}_{\text{ext}} + \mathbf{H}_{\text{int}}, \end{aligned} \tag{2.37}$$

where H_0 is the vacuum permeability, $\mathbf{K}$ is the magnetization field describing the density of the magnetic moment induced by the total magnetic field, $\mathbf{H}_{\text{ext}}$ is the external magnetic field, and $\mathbf{H}_{\text{int}}$ is the internal magnetic field generated by the ferrofluid. The magnetization field is a function of the known external magnetic field and internal magnetic field, and the current internal magnetic field can be obtained by solving Poisson's equation with the assumptions of $\nabla \times \mathbf{H}_{\text{ext}} = \mathbf{J}$ and $\frac{\partial \mathbf{D}}{\partial t} = 0$ in Equation (2.36), as the free current density $\mathbf{J}$ is not influenced by $\mathbf{H}_{\text{int}}$, and the electric displacement produced by the flow of ferrofluid is not strong enough to influence the system.

Under the effect of magnetic force, fluid particles gather on tiny bumps near the surface where the synthesized magnetic field is stronger, pulling the fluid to form spikes with visually attractive features. The spike shapes are also influenced by gravity and surface tension.

The past few years have witnessed the proposal of various methods for simulating ferrofluids within the field of computer graphics. Huang *et al.* [132] presented the smooth magnets method, which utilizes Lagrangian particles embedded with magnetic nanoparticles to discretize fluids. This method was the first in computer graphics to address the macroscopic simulation of ferrofluids based on first principles. Their proposed magnetization model, along with the magnetic field's Poisson equation, can be discretized using a smooth kernel function akin to that used in SPH.

Approaching from a Lagrangian perspective, the volumetric Kelvin force model is utilized to characterize the magnetic force interactions between particles. Standard SPH computations employ particles, enabling the enforcement of incompressibility and surface tension within ferrofluids. However, the implementation of a Kelvin force model engenders an unanticipated outward-directed force on the surface, prompting particles to exhibit levitation near this surface. To address this phenomenon, Shao *et al.* [257] introduced a modification, replacing the Kelvin force model with a current loop model, thereby creating an inward force. This alteration permitted the integration of the magnetic model into Implicit SPH models. Consequently, it enhanced system stability and facilitated the use of larger time-step increments.

From an Eulerian perspective, Ni *et al.* [212] presented a level-set method for simulating various magnetic bodies, including ferrofluids. The interplay between the magnetic field and mechanical system was addressed as an interfacial issue, and a weighted average of the internal and external magnetic fields was calculated to manage discontinuities. The resulting magnetic force, coupled with surface tension, was integrated into the Navier–Stokes equations to direct the dynamics of the ferrofluids.

Advancing the research further, Huang and Michels [133] introduced the concept of surface-only ferrofluids, a singular study to date, tested successfully against real ferrofluids. Unlike previous approaches that

incorporated magnetic force as an additional term in the momentum equation, their method infuses the discontinuity of magnetic pressure into the Dirichlet boundary condition within the pressure-projection of the Galerkin Boundary Element Method (BEM)-based surface-only liquid solver [71], specifically at the fluid–air interface. This pioneering surface scheme enhances the Helmholtz decomposition step in surface-only fluid solvers via a more precise analytic integration process. Additionally, it augments the accuracy of the pressure projection step within surface-only fluid solvers through a Galerkin BEM.

From the hybrid discretization perspective, Sun *et al.* [274] utilized the MPM structure to further simulate nonlinear magnetic substances in pursuit of a more general magnetic description. They used the physically-realistic Langevin's nonlinear magnetization model to bound the magnetic force between magnetic micro elements without additional numerical approximation. Following the concept of MPM, this method uses particles to carry microscopic magnetic quantities and solves Poisson's equation of the magnetic fields and Kelvin force on Cartesian grids. Without integrating the surface tension, this method cannot form stable spikes when simulating ferrofluids. However, thanks to the versatility of MPM, it can achieve a unified simulation and coupling of different magnetic materials.

2.9.3 *Thin Films*

Thin films and bubbles are fascinating phenomena that have received special attention. A common example is a soap bubble floating in air. Bubbles produced from pure water are usually few, small, and disappear quickly due to gravity, pressure, and strong surface tension. To produce more, larger, and longer-lasting bubbles, surfactants are added to the water, *e.g.*, fatty acids common in soaps. With surfactants interspersed among water molecules, surface tension is reduced so that larger bubbles appear. The tensile deformation of the film will recover due to the difference in surface tension working like elasticity — the so-called Marangoni effect (Figure 2.28). While the Marangoni effect models the resilience given by inconsistent surface tension, the Young–Laplace equation describes the capillary pressure difference Δp caused by surface tension between air and the fluid as

$$\Delta p = -\gamma \nabla \cdot \mathbf{n}, \tag{2.38}$$

where γ is the surface tension, and $\mathbf{n}$ is the surface outward-pointing normal. Equation (2.38) relates the pressure difference to film shape.

Batty *et al.* [25] developed discrete viscous sheets by building on (Lagrangian) elastic thin shells. This reduced-dimensional technique describes the sheets using triangular meshes with local thickness and uses the area-based surface tension derived from the mid-surface of the (thin) shell. Wang *et al.* [299] enhanced this to capture the surface tension flow

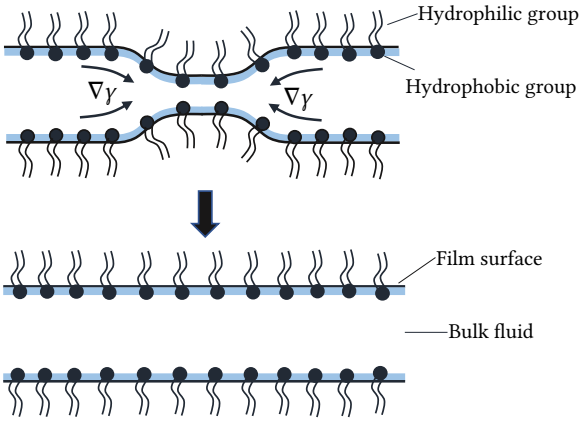


Figure 2.28: Schematic diagram of the Marangoni effect. Surfactants gather on both sides of the thin film. The tensile deformation yields an inconsistent surfactant concentration, which leads to the surface tension gradient shown in the figure. The film recovers its shape due to this gradient.

using moving-least-squares particles. The mixed Lagrangian–Eulerian approach models not only volumetric phenomena (3D) but also those arising from thin shells (2D), filaments (1D), and even individual points (0D). Surface tension (and other forces) are handled in a unified way across all (co)dimensions, including codimensional transitions. This enables complex scenarios requiring careful (surface) tension treatment, such as two water jets colliding and forming a thin sheet, to be simulated. Similarly to the work of Wang *et al.* [299], the codimensional surface tension flow of Zhu *et al.* [354] also relies on simplicial complexes and transitions between elements of different (co)dimensions, covering thin fluid sheets, filaments, and surface tension effects.

Wang *et al.* [300] extended the work of Wang *et al.* [299] to yield a thin-film SPH fluid model. Films are modeled as (surface) particles of codimension one with local thickness estimates. This particle setup interacts with a Lagrangian flow simulation using a thin film shape description. This physically couples aggressive surface deformations and strong tangential flows. A process of transformation from codimension one to codimension zero is used to simulate rupture.

Focusing on viscous thin films, Vantzos *et al.* [296] proposed a numerical scheme to simulate the thin film equation (modeled as a height function) on a planar domain, including gravity and other forces. They added a novel quadratic term to the governing equation to stabilize flow while maintaining visual fidelity. Their scheme is fully local; thus, it allows an efficient GPU implementation that leads to real-time simulations, such as that of honey flowing through honeycombs.

Da *et al.* [70] studied soap films and foams whose dynamics are captured by a Lagrangian vortex sheet model with an emphasis on circulation. Surfaces are represented using multi-material triangular meshes supporting topological changes, and their tension forces lead to a circulation update rule based on mean curvature.

By Plateau's laws, a steady-state film consists of constant mean curvature parts and minimal surfaces (vanishing mean curvature). Ishida *et al.* [144] used this to model evolving foams via hyperbolic geometric flow, a type of mean curvature flow.

Most existing soap film models [70, 144] assume that a film is infinitesimally thin and has no influence on its evolution. Ishida *et al.* [143] extended such methods to use film thickness, modeled on non-manifold meshes, as a reduced degree of freedom in the Navier–Stokes equations and to derive the motion equations. This provides an incompressible fluid solver for 2.5D films.

In addition to dynamic effects, bubbles also produce rich color effects due to the light interference caused by uneven film thickness. Besides soap bubbles, other fluid dynamics simulations are also possible on a thin film for additional visual effects.

Two further methods focused on *spherical bubbles* and the fluid around them. Hill and Henderson [125] efficiently simulated fluids on a spherical surface. They handled poles/singularities of spherical coordinates, which would otherwise render the motion equations complex if used naively on the sphere. Their method also enables vector and scalar controls for art-directed spherical flows. Huang *et al.* [135] focused on the chemical–mechanical simulation of soap film flows on spherical bubbles using lubrication theory. Considering the Marangoni effect and the capillary pressure difference, the stress condition at the film surface is given by

$$\boldsymbol{\sigma} \cdot \mathbf{n} = (-\gamma \nabla \cdot \mathbf{n} - p_a)\mathbf{n} + \nabla_s \gamma, \tag{2.39}$$

where $\boldsymbol{\sigma}$ is Cauchy stress tensor, $\mathbf{n}$ is the outward unit normal vector at the surface, p_a is the air pressure, and ∇_s is the 2D gradient operator. The surface tension γ is defined by a linear model

$$\gamma(\Gamma) = \gamma_0 - \gamma_r \Gamma, \tag{2.40}$$

where γ_0 is the surface tension of pure water, Γ is the surfactant concentration, and γ_r is a constant that describes the Marangoni elasticity of the film. The surfactant concentration Γ is advected by an advection-diffusion equation. Huang *et al.*'s advection scheme used spherical coordinates. Changes of coordinate orientation in curvilinear coordinate systems can be implicitly handled when transforming from global to local coordinates and does not need to be solved separately [125]. This last method also proposed a physically accurate shader for real-time rendering under environmental lighting.

Recently, Deng *et al.* [75] introduced MELP (Moving Eulerian–Lagrange Particles), a novel mesh-free method for incompressible fluids on moving foams and thin films. Their approach, including multi-MELP for interfacial flow, is able to model both large-scale surface deformations and detailed flows.

2.10 CONCLUSION AND DISCUSSION

Physics-based fluid simulation has been successful in games, film, and animation. Current advances enable high levels of control in production and have shown increasing acceptance and potential in virtual and augmented reality and other real-time graphics-intensive applications. Recent advances in physics-based fluid simulation methods rely on a complex mix of computational efficiency, realism, controllability, and ability to simulate diverse scenarios. This survey presented an in-depth overview of these methods over the last decade. We discussed the different goals in this field, techniques proposed to address these goals, and challenges of these techniques.

Our survey found seven major themes present in approximately 300 fluid simulation papers from the computer graphics community in the last decade: advanced computational approaches, interaction with materials, multiphase simulations, gas–liquid interfaces, enhancing fine details, simulation control, and special fluids. These themes formed the structure of our survey to outline important developments in this period and community.

As outlined in our survey, the field of physics-based fluid simulation faces numerous challenges, spanning from computational approaches to the detailed interaction of different fluid types. Our exploration has uncovered several ongoing challenges. In the subsequent chapters of this thesis, we address some of these challenges from three distinct perspectives, proposing new solutions and further advancements.

Advanced computational approaches. Adaptive solutions and GPU parallelization are ubiquitous in computer graphics, but there is much room for developing such new methods for physics-based fluids. For instance, adaptivity usually makes an implementation complex and difficult to apply to GPU hardware. Spatial and temporal resolution are often related aspects. In space discretization, interactions between grids or particles at different scales may cause instability and fidelity loss, which can also create energy diffusion. Hence, in addition to improving computational efficiency and reducing overhead, proposing methods to reduce such unwanted effects on a wide range of resolution scales is important for future research. Using neural networks to learn fluid dynamic behavior will be a hot research topic in the future, as it has strong prospects for real-time simulation and industrial control. However, only summarizing physical laws from a large amount of training data lacks

underlying logical support. As such, more attention is likely to be paid next to how to inject physical prior knowledge into deep learning models.

In Chapter 3, we tackle the optimization of spatial adaptivity mechanisms through the lens of fluid-solid dynamics. We introduce a boundary-distance-based adaptive method for Smoothed Particle Hydrodynamics simulations, utilizing a signed-distance field relative to fluid-solid boundaries to adjust particle resolution. This ensures the highest resolution near boundaries, which gradually decreases to a preset threshold with increasing distance.

Fluid coupling with multi-materials. The key challenge for such simulations is to keep accuracy, stability, and efficiency when coupling multiple materials concurrently in one scenario. Using different solvers *vs* materials limited the diversity of fluid animation until recent years. Current research has moved from merging multiple solvers to developing monolithic ones. Monolithic solvers can simulate different materials and their interaction in a single framework, which can eliminate stability issues. However, such solvers currently demand high computational resources. Thus, an open challenge is to mix hardware and algorithm designs to better support such solvers.

Multiphase liquids. Future work can explore many interesting aspects. One challenge is that current methods for incompressible fluid simulation do not handle high-density ratios well. Linear systems become ill-conditioned under high-density contrasts, and Jacobi-like solvers fail to converge. Currently, many parameters of mixed fluid are manually adjusted. How to control parameters more intuitively to achieve the desired visual results is worth further study. Modeling temperature, chemical reactions, elasticity blending, and optical blending are equally important open aspects in this area.

In Chapter 4, we address the problems associated with ill-conditioned linear systems and the convergence challenges in implicit fluid solvers. We have integrated a multiphase fluid solver into existing frameworks for implicit linear systems. This integration not only aims to improve the robustness and versatility of fluid simulations but also ensures more stable and realistic representations in complex scenarios involving multiple fluid phases.

Gas–liquid interaction. Challenges for liquid–gas interaction include simulating the gas–liquid phase transition and modeling its effect on surface tension, supporting non-manifold thin film structures, handling the transition between different codimensionalities, producing realistic surface colors for bubbles, reducing the simulation complexity while preserving accuracy for liquid–air coupling, and adding fine splash details that are not limited by particle size. Achieving richer gas–

liquid interaction phenomena while ensuring stability and efficiency is a subject of ongoing and future long-term research.

Fine detail enhancement. Many existing detail enhancement techniques can achieve detailed fluid surfaces without using high-resolution discretization. Realistic and fine-grained appearance representations are the primary pursuit in this direction. Energy conservation and detail preservation in accordance with physical laws are also important goals. How to improve the efficiency and scalability of detail enhancement is an open challenge for the end goal of providing real-time, interactive, and generalizable tools for artists. Future work will likely use deep learning techniques with innovative explorations in style transfer, high-resolution reconstruction, and detail generation.

In Chapter 5, we extend these concepts by introducing an innovative real-time fluid rendering technique that integrates the anisotropy matrix into the screen space rendering pipeline. Our approach adapts point sprites within particle-based simulations, stretching them along feature vectors through anisotropic transformations. This method not only achieves smoother fluid surfaces but also leverages a weighted principal component analysis to optimize particle distribution.

Fluid control. Issues still exist in fluid control: high memory and computation costs, sensitive parameters, and manual feature labeling. A key difficulty of control methods is to achieve precise control. Achieving precise control, along with high accuracy and efficient computation, is — regarding the other topics studied in this survey — one of the grand challenges of fluid simulation research.

Special fluids. The challenge of simulating fluids with special forms lies in unifying governing equations and other physical characteristics in a consolidated system, which means integrating different solvers together. The targeted formulation of a monolithic-style solver is required to perform high-performance simulation results.

We hope this survey helps to introduce the theoretical concepts underpinning physics-based fluid simulation and their practical implementation to serve as a guide for researchers and practitioners as well as facilitate future works to exploit on the basis of recent developments.

3

SPATIAL ADAPTIVITY WITH BOUNDARY REFINEMENT FOR SPH FLUID SIMULATION

As we already remarked in Section 2.2, fluid simulation is renowned for its visually stunning outputs, yet it is also known for its high computational demands. Spatial adaptivity can significantly reduce these demands by varying the resolution of the simulation space. Current adaptive methods primarily enhance the refinement of fluid surfaces to produce more detailed splashes and wave effects. However, these methods often fail to optimize performance in scenarios where large areas of the fluid surface remain tranquil, and they struggle to capture the dynamic flows beneath the surface when the interior fluid is simulated with a coarse discretization. This work introduces a novel boundary-distance-based adaptive method for Smoothed Particle Hydrodynamics fluid simulations. We employ a signed-distance field, constructed relative to the fluid-solid boundaries, to modulate particle resolution across different spatial positions. The resolution is highest near the boundary and decreases smoothly with increasing distance until reaching a preset threshold. Particle sizes are dynamically adjusted through a split and merge process. Additionally, we implement a wake flow preservation mechanism that maintains high particle resolution temporarily after a particle interacts with the boundary object, thus preserving detailed flow dynamics. Experimental evaluations show that our method not only refines fluid-solid interaction details more effectively than traditional global fine-resolution approaches but also surpasses surface-only adaptivity schemes in capturing dynamic effects beneath the surface[1].

This chapter begins with an overview of basic concepts related to spatial adaptivity in Section 3.1, highlighting significant innovations in this field. The background and mathematical principles of the SPH approach, including its application to solving Navier-Stokes equations, are discussed in Section 3.2. This section serves as the foundation not only for the subsequent sections of this chapter but also informs the content of Chapters 4 and 5. Detailed descriptions of the particle split and merge processes, along with the special treatments required for boundary coupling in particles of varying sizes, are provided in Sections 3.3 and 3.4, respectively. Our boundary-optimized adaptive method, including the wake flow preservation scheme, is proposed in Section 3.5. The perfor-

1 Parts of this chapter were published in: Y. Xu, C. Song, X. Wang, X. Ban, J. Wang, Y. Zhang, and J. Chang. Spatial adaptivity with boundary refinement for smoothed particle hydrodynamics fluid simulation. *Computer Animation and Virtual Worlds*, Volume 34, Issue 5. Wiley, 2023. doi: 10.1002/cav.2136

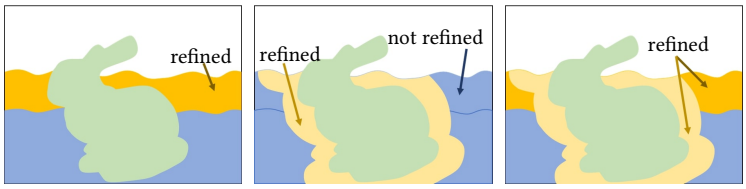
mance of this method is tested and discussed in Section 3.6. The chapter concludes with a summary of the findings in Section 3.7.

3.1 INTRODUCTION

Fluid simulation plays an essential role in visual effects, for the physical-based result can produce magnificent eye-catching results to maximize the sense of reality. Nevertheless, higher demand for simulation details usually means the rapid growth of computational cost due to the need for the finer discretization of multidimensional space and time. As we discussed in Section 2.3.1, although the choice of time step (*temporal adaptivity*) can be readily adapted using Courant-Friedrichs-Lewy (CFL) condition, the mechanism of *spatial adaptivity* can still be further exploited considering variable conditions for better efficiency. The spatial adaptivity mechanism enables achieving exquisite fluid behaviors with more affordable expenses by optimizing the interval of sampling points to refine local areas with pertinence.

Methods for spatial adaptivity in fluid simulation depend heavily on the fluid simulation approaches on which they are based. We discussed related work in spatial adaptivity in Section 2.3.1.2. Here we refine that discussion with additional references which bring more insight into the context of our work discussed in this chapter.

For Eulerian approaches that simulate the fluid with grids, a standard scheme uses octrees to split some grids into finer grids [9]. Meanwhile, tilted grids can also be used [320] to avoid numerical issues. For particle-based Lagrangian approaches, the sizes of particles are adaptively adjusted [313]. For hybrid approaches that inherit the traits of both Eulerian and Lagrangian methods, one can improve the efficiency by limiting the regions where particles are used [250] or combining octrees with an adjustable number of particles based on regional characteristics [210].



(a) Surface-based refinement (b) Our boundary refinement scheme (c) Ours + surface-based

Figure 3.1: Contrary to the previous surface-based refinement method of Winchenbach *et al.* [313] (a), our method (b) refines fluid particles near boundaries of objects to allow the fine-scale fluid-boundary coupling to enhance boundary coupling details. Our method can also be combined with the surface-based method to refine near the boundary and the surface (c).

For adaptive SPH, Adams *et al.* [3] resampled particles with different radii based on the size of local geometric features. Solenthaler and Gross [266] introduced a two-scale mechanism that coupled a low-resolution simulation with a high-resolution one, and particles were directly inserted or deleted at the boundary between the two resolutions, which broke the mass-preserving condition. Horvath and Solenthaler [128] extended the previous work [266] to support multiple levels of resolution and conserve mass. Orthmann and Kolb [220] increased the spatial resolution by splitting particles using a $1 : 2$ pattern and applied a temporal blending technique to achieve a smooth particle splitting process by maintaining continuous physical fields. Vacondio *et al.* [294] used the variational principle to search for an optimized splitting pattern that minimized density error. They also introduced a coalescing scheme to realize the dynamic reduction of resolution. Winchenbach *et al.* [313] achieved adaptivity in incompressible SPH by introducing a new split-merge mechanism that produced an approximately continuous resolution based on the distance from the free surface of the fluid. Winchenbach and Kolb [314] later proposed a method to optimize the density error from the splitting process into an arbitrary number of particles by optimizing the position and mass of the split particles to increase the stability of the previous method [313].

Nowadays, research on spatial adaptivity for Lagrangian fluid simulation mainly targets surface-based optimization, aiming at refining surface details like splashes, waves, and thin films to enrich visual effects. Horvath and Solenthaler [128] use a higher resolution for particles within a certain distance to the free surface of the fluid; Winchenbach *et al.* [313] make the desired size of particles increase smoothly with the distance to the surface.

However, the surface-based strategy has the disadvantage of an inflexible intervention mechanism, which causes low computational efficiency and the failure to enhance specific dynamic areas. On the wide fluid surface, the area with the most high-frequency details is often the part where the fluid is coupled with other objects, such as a boat sailing across the water surface, arousing the waves around and behind the body. Meanwhile, the boat's rotating propeller below the surface can constantly transmit kinetic energy into the water, further producing splashes to the surface. When such a scenario is simulated using a coarse discretization with surface-based refinement, the vast and calm surface brings a large computational burden, causing computational resources to be wasted in these unimportant areas. Also, the energy dissipation from the propeller area cannot be restrained effectively. Although the boundary handling for adaptive SPH has been well-studied as discussed in Section 2.3.1.2, more attention needs to be paid to designing the adaptive mechanism dedicated to the fluid-solid boundary. So in our work, we present a spatial adaptivity method with a bound-

ary refinement mechanism for SPH fluid simulation to capture detailed interaction effects.

To conduct spatial adaptivity with more flexibility and efficiency, we propose a novel adaptivity method with boundary refinement for SPH fluid simulation, where particle resolution increases as the distance to the object boundary of interest decreases (see Figure 3.1). The sizes of the particles are adjusted towards the desired resolution via splitting and merging. Moreover, a wake flow preservation mechanism is introduced to retain the refined resolution for a specific period after a particle flows through the boundary object to prevent loss of flow details. Our results show that our method can enrich fluid details more economically and produce more accurate visual effects in contrast to the surface-based strategy.

3.2 PRELIMINARIES

3.2.1 *Theory of SPH*

SPH is a Lagrangian discretization method that converts continuous space into a series of discrete particles. This approach is deeply rooted in the classical physics concept of a *fluid parcel*, the smallest unit of fluid indistinguishable in the Navier-Stokes equations, as discussed in Sections 2.1.1 and 2.1.2. When the discretization of these Lagrangian particles matches the scale of fluid parcels, the physical behavior described by the Navier-Stokes model (Equation (2.6)) is closely approximated.

The actual size of each fluid parcel is in the same way that the *Dirac function* $\delta(x)$ has a finite integral but an infinitely small support, a fluid parcel has an infinitely small volume but describes a finite mass of the fluid. This function $\delta(x)$ can be expressed as

$$\delta(x) = \begin{cases} +\infty & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases}. \tag{3.1}$$

This function is normalized such that $\int_{-\infty}^{\infty} \delta(x)dx = 1$, embodying the ideal, infinitely narrow spike at zero with total integral one over all space.

Mass for an object in space is usually calculated by integrating the density over the occupied domain Ω, as $m = \int_{\Omega} \rho(\mathbf{x})d\mathbf{x}$, with $\mathbf{x} \in \mathbb{R}^3$. However, for a fluid parcel, represented as a point with infinitely small volume, its mass cannot be directly integrated in this manner. Utilizing the Dirac delta function, which can be seen as an extreme case of a Gaussian distribution with $\sigma^2 \to 0$, with the property of

$$A(\mathbf{x}) = \int_{\Omega} A(\mathbf{x}')\delta(\mathbf{x} - \mathbf{x}')d\mathbf{x}', \tag{3.2}$$

we can define the density at a point $\mathbf{x}$ in 3-dimensional space as

$$\rho(\mathbf{x}) = \int_{\Omega} \rho(\mathbf{x}')\delta(\mathbf{x} - \mathbf{x}')d\mathbf{x}'. \tag{3.3}$$

To render the aforementioned continuous expressions computationally feasible, SPH approximates the Dirac delta function using a *smoothing kernel* W which meets the requirements discussed in Section 2.1.3.2. This kernel, which extends over a finite non-zero region, effectively transforms Equation (3.2) into a solvable form as presented in Equation (2.7).

3.2.2 *SPH Approximation*

In addition to approximating values at specific positions for a physical field, the ability to compute higher-order derivatives such as gradient, divergence, curl, and Laplacian is also crucial in the study of fluid dynamics. SPH is well-equipped to handle these computations as well.

First order derivatives. For computing first order derivatives of a physical field A, the computation directly involves the smoothing kernel W as described in Equation (2.7)

$$\begin{aligned} \nabla A_i &= \sum_j V_j A_j \nabla W_{ij}, \\ \nabla \times A_i &= \sum_j V_j A_j \nabla \times W_{ij}, \\ \nabla \cdot A_i &= \sum_j V_j A_j \nabla \cdot W_{ij}, \end{aligned} \tag{3.4}$$

where A_i is a shorthand notation for $A(\mathbf{x}_i)$, and W_{ij} denotes $W(\|\mathbf{x}_i - \mathbf{x}_j\|, h)$. However, the accuracy of SPH's basic form is highly dependent on the sampling density of particles and their distribution [166]. To maintain zero-order accuracy, a spatially constant field $c_1(\mathbf{x})$ with a value of 1 everywhere can be utilized. This field satisfies $\nabla c_1 = \mathbf{c}_0$, with each entry equaling zero. Using Equation (3.4), we have

$$\nabla c_1 = \sum_j V_j \nabla W_{ij} = \mathbf{0}, \tag{3.5}$$

which may not hold true if neighboring particles are insufficient or if they are irregularly distributed. To correct this, we subtract ∇c_1 from Equation (3.4) as

$$\begin{aligned} \nabla A_i - A_i \nabla c_1 &= \sum_j V_j (A_j - A_i) \nabla W_{ij}, \\ \nabla \times A_i - A_i \nabla c_1 &= \sum_j V_j (A_j - A_i) \nabla \times W_{ij}, \\ \nabla \cdot A_i - A_i \nabla c_1 &= \sum_j V_j (A_j - A_i) \nabla \cdot W_{ij}. \end{aligned} \tag{3.6}$$

This formulation is referred to as the *differential form* of derivative computation. Another desired attribute for certain scenarios is the *conservation of values*, which implies that for a pair of particles i and j, values computed from SPH should be symmetrically opposite. This is particularly useful when considering the conservation of momentum, focusing solely on the differences in particle masses. To achieve this, derivatives are computed using the symmetric form as

$$\nabla A_i = \rho_i \left[\frac{A_i}{\rho_i^2} \nabla \rho_i + \nabla \frac{A_i}{\rho_i} \right] = \rho_i \sum_j \rho_j V_j \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2} \right) \nabla W_{ij}, \tag{3.7}$$

where ρ_i denotes $\rho(\mathbf{x}_i)$. This approach is known as the *symmetric form*. For a detailed derivation, we refer to the survey by Price [232].

Laplacian. The calculation of the Laplacian in SPH using a direct second-order difference approach on the smoothing kernel typically results in poor accuracy. To overcome this, an alternative method that approximates the Laplacian using first-order differences is often employed, which is detailed in the work of Koschier *et al.* [166]. It has the form

$$\nabla^2 A_i = 2(d+2) \sum_j V_j \frac{(A_i - A_j) \cdot (\mathbf{x}_i - \mathbf{x}_j)}{\left\| \mathbf{x}_i - \mathbf{x}_j \right\| + 0.01h^2} \nabla W_{ij}, \tag{3.8}$$

where d represents the dimension of the simulation. The regularization term $0.01h^2$ is incorporated to prevent division by zero. This formulation is particularly tailored for applications involving divergence-free velocity fields.

Smoothing kernel. The choice of the smoothing kernel in SPH is critical and varies depending on specific requirements. A commonly used kernel is the cubic spline kernel, defined as

$$W(\|\mathbf{r}\|, h) = \frac{\sigma_d}{h^d} P\left(\frac{\|\mathbf{r}\|}{h} \right), \tag{3.9}$$

where σ_d is a dimensional normalizing factor that varies with the dimension d of the simulation. The values of σ_d are defined as follows:

$\sigma_1 = \frac{4}{3}$, $\sigma_2 = \frac{40}{7\pi}$, and $\sigma_3 = \frac{8}{\pi}$. These coefficients ensure that the kernel's integral over space equals 1, thereby maintaining consistency in different dimensions [203]. P represents a symmetric decaying spline with finite support, given by

$$P(q) = \begin{cases} 6(q^3 - q^2) + 1 & \text{for } 0 \leq q \leq \frac{1}{2} \\ 2(1-q)^3 & \text{for } \frac{1}{2} < q \leq 1 \\ 0 & \text{otherwise} \end{cases}. \tag{3.10}$$

This kernel is used by default throughout this thesis unless specified otherwise.

3.2.3 *Explicit Form of Solving Navier-Stokes Equations with SPH*

To compute the acceleration of all discretized SPH particles, the Navier-Stokes model (Equation (2.6)) is solved using the SPH discretization scheme, as detailed in Equations (2.7), (3.4), (3.6), (3.7), and (3.8). Solving these equations explicitly involves calculating each physical quantity present in the equations, except for known constants and the acceleration that needs to be computed. Here, we outline a basic fluid simulation algorithm by going through the derivation of all relevant physical quantities.

Density. By setting $A = \rho$ in Equation (2.7), and noting that $\rho_j V_j = m_j$ (the mass of particle j), density ρ_i at the location of particle i can be expressed as

$$\rho_i = \sum_j m_j W_{ij}. \tag{3.11}$$

This computation provides a discretized approach to estimating fluid density using the SPH kernel.

Pressure force. Computing the pressure force is a critical aspect of solving the Navier-Stokes equations. For an explicit calculation, a pressure field p must be determined at each time step. Becker and Teschner [26] suggest calculating the pressure field based on the local density compression, as described by

$$p_i = B\left(\left(\frac{\rho_i}{\rho_0}\right)^{\kappa} - 1\right), \tag{3.12}$$

where $B = \rho_0 c_s^2 / \kappa$. Here, ρ_0 is the rest density, representing the density of the fluid in an uncompressed state. c_s is the speed of sound within the fluid, typically set to 100m/s. The stiffness parameter κ is set to $\kappa = 7$ for weakly compressible fluids to allow the simulation of fluids with varying compressibility characteristics.

From Equation (2.6), the relationship between the pressure force $\mathbf{F}_i^p$ and the pressure field for particle i is given by $\mathbf{F}_i^p = V_i \nabla p_i$. To ensure adherence to Newton's third law of motion, the symmetric form of the SPH approximation (Equation (3.7)) is used to compute this force as

$$\begin{aligned}\mathbf{F}_i^p &= V_i \nabla p_i = V_i \rho_i \sum_j \rho_j V_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \\ &= \sum_j m_i m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij}.\end{aligned} \tag{3.13}$$

This method ensures that the pressure forces are symmetrical between interacting particles, thus conserving momentum in accordance with physical laws.

Viscosity. Viscosity in fluid dynamics models the internal friction resulting from the differential movement between layers of the fluid. As indicated by the Navier-Stokes model (Equation (2.6)), the calculation of viscous forces necessitates computing the Laplacian of the velocity field. Utilizing the adapted form of the Laplacian in SPH as given in Equation (3.8), the viscous force $\mathbf{F}_i^{\text{vis}}$ exerted on particle i is formulated as

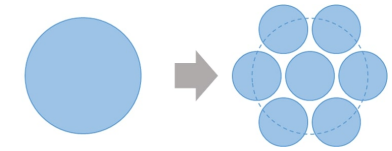
$$\mathbf{F}_i^{\text{vis}} = V_i \mu \nabla^2 \mathbf{u}_i = 2(d+2) V_i \mu \sum_j V_j \frac{(\mathbf{u}_i - \mathbf{u}_j) \cdot (\mathbf{x}_i - \mathbf{x}_j)}{\left\| \mathbf{x}_i - \mathbf{x}_j \right\| + 0.01h^2} \nabla W_{ij}. \tag{3.14}$$

3.3 SPLIT-MERGE-REDISTRIBUTE MECHANISM

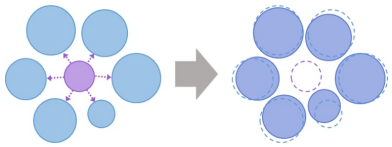
Winchenbach *et al.* [313] achieved SPH adaptivity by employing methods of *splitting*, *merging*, and *redistributing* particle mass, while strictly conserving the total mass of the fluid, as illustrated in Figure 3.2. In this method, the desired size for each particle is computed using a sizing function based on the distance to the fluid's free surface:

$$m_i^{\text{opt}} = m^{\text{base}} \left(\frac{\min(|\phi_i^f|, |\phi_{\max}|)}{|\phi_{\max}|} (1 - \alpha) + \alpha \right), \tag{3.15}$$

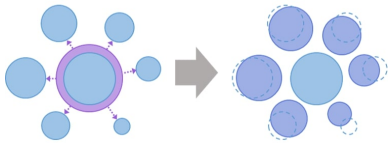
where m^{opt} is the optimal mass that is the desired mass for the particle; α is the adaptivity ratio that denotes the largest mass ratio allowed between particles; m^{base} is the largest allowed particle mass; ϕ_i^f is the distance to the fluid's free surface similar to Horvath and Solenthaler [128] (negative inside the fluid); $\phi_{\max}$ is the max distance to the surface within which particles are refined. Equation (3.15) ensures that the optimal mass decreases linearly (and thus also smoothly) as the distance to the free surface decreases.



(a) Splitting



(b) Merging



(c) Redistribution

Figure 3.2: Schematic diagram of particle splitting, merging, and redistribution. In (a), a particle classified as L (blue particle on the left side) undergoes the splitting process, dividing into several smaller particles (small blue particles on the right side), each with a mass close to the optimal mass m^{opt}. In (b), a particle of class S (purple particle on the left side) is merged into surrounding particles of class S or s (blue particles on the left side), making their masses closer to m^{opt} (violet particles on the right). In (c), a particle classified as l (the particle with blue and purple colors) redistributes the excess mass (purple part) to the neighboring s particles (blue ones on the left turning into purple ones on the right).

In our study, we adjust particle mass towards the optimal mass, m^{opt}, through three distinct operations: splitting, merging, and redistribution. This methodology employs the classification scheme introduced by Winchenbach *et al.* [313], which categorizes particles based on the relative mass ratio, $m_i^{\text{rel}} = \frac{m_i}{m_i^{\text{opt}}}$. The particle categories are organized as follows:

- S: $m_i^{\text{rel}} < 0.5$, indicating particles are too small.
- s: $0.5 \leq m_i^{\text{rel}} \leq 0.9$, indicating particles are slightly small.
- o: $0.9 < m_i^{\text{rel}} < 1.1$, indicating particles are optimally sized.
- l: $1.1 \leq m_i^{\text{rel}} \leq 2$, indicating particles are slightly large.
- L: $m_i^{\text{rel}} > 2$, indicating particles are too large.

Each category corresponds to specific operational interventions:

- **Splitting:** Applied to particles in category L, where each large particle is divided into multiple smaller particles.
- **Merging:** Applied to particles in category S, where each small particle is combined with adjacent particles to increase its mass.
- **Redistribution:** Applied primarily to particles in categories s and l. Particles in l redistribute excess mass to adjacent s particles, while those in s may gain mass from or merge with neighboring particles to reach an optimal size.

Particles of class L are split into n children particles where $n = \lceil m_i/m_i^{\text{opt}} \rceil$. The mass and positions of children particles are determined using precomputed split patterns. They are further optimized online by solving a minimization problem on density error to reduce the error induced by splitting [314]. The other physical attributes of children particles are inherited from their parents. Particles of class S undergo merging, distributing all of their mass to nearby s or S particles, while particles of class l redistribute their excess mass $m_i - m_i^{\text{opt}}$ to nearby s particles. The distance between the mass-distributing particle i and the mass-receiving particle j must be within $h_i/2$. The mass to distribute $m_{\text{ex}} = m_i - m_i^{\text{opt}}$ is equally divided among the mass-receiving particles, while the other physical attributes are weight-averaged between particles i and j as in Equation (2.11).

Techniques increasing stability. Even with optimized splitting patterns and online optimization, splitting and merging can still introduce some density errors [314]. Temporal blending [313] and local viscosity [314] can increase the stability of particle splitting and merging.

Temporal blending is applied to particles that recently participated in splitting, merging, or mass redistribution. The original particle o before splitting or redistributing mass is stored. Then, o is used to modify the density and velocity of its children or particles that received mass from it. The blended density is computed as

$$\rho_i^{\text{blended}} = (1 - \beta)\rho_i + \beta\rho_o, \tag{3.16}$$

where ρ_o is the density of the original particle, computed using Equation (3.11), but ignoring the density contribution from its children. β is a temporal blending factor that is heuristically initialized to 0.5 for particles generated in splitting and 0.2 for particles involved in redistribution or merging. Then, β is decreased by 0.1 every time step until it reaches 0. ρ_i^{blended} is used instead of ρ_i in further computations. The configuration for β follows the method outlined by Winchenbach *et al.* [313]. The velocity of the original particle $\mathbf{v}_o$ is set to the average velocity of all its children particles and is used to update the original particle's position $\mathbf{x}_o$. The blended velocity $\mathbf{v}_i^{\text{blended}} = (1 - \beta)\mathbf{v}_i + \beta\mathbf{v}_o$ is used instead of $\mathbf{v}_i$ to update the particle's position $\mathbf{x}_i$.

In the local viscosity technique, the viscosity coefficient μ in Equation (3.14) is multiplied with a factor $1 + 0.5(\beta_i + \beta_j)/2$ to increase the viscosity of newly split, merged or redistributed particles locally, thus increasing stability.

3.4 SEMI-ANALYTIC BOUNDARY HANDLING

A suitable boundary handling method for adaptive SPH is proposed in Winchenbach *et al.* [311]. This method treats all solid boundaries as locally planar through a locally planar approximation. Consequently, the solid contribution λ_i for a fluid particle i concerning value A_i, derived from Equation (2.7), is given by

$$A_i = \sum_j A_j V_j W_{ij} + A_b \lambda_i, \tag{3.17}$$

where A_b represents the value of A within the solid boundary, and λ_i is the weight contributed by the solid.

Given that the boundary is treated as locally planar and W is isotropic, λ for particle i can be analytically derived based on the distance d from the particle position $\mathbf{x}_i$ to the approximated boundary plane. Assuming the cubic spline kernel for W, with support radius h_i for particle i, the contribution can be expressed as

$$\lambda'(d) = \begin{cases} \frac{1}{30}[81q^6 - 144q^5 + 80q^3 - 42q + 15] & 0 \leq q \leq 0.5 \\ -\frac{8}{15}[2q^6 - 9q^5 + 15q^4 - 10q^3 + 3q - 1] & 0.5 < q \leq 1 \\ 1 - \lambda'(-q) & -1 \leq q < 0 \end{cases}, \tag{3.18}$$

where $q = d/h_i$. For a detailed derivation, see the appendix in Winchenbach *et al.* [311]. An additional penalty term $\beta(d) = 1 - \frac{d}{h_i}$ is introduced to penalize solid penetration. This term is directly applied to $\lambda'(d)$ to obtain the corrected value $\lambda(d) = \beta(d)\lambda'(d)$.

The distance d is determined using a signed distance field (SDF) sdf^b, based on the boundary domain $\partial\Omega_s$ (see Figure 2.12). For any position $\mathbf{x}$, the value of sdf^b is defined as

$$\text{sdf}^b(\mathbf{x}) = s(\mathbf{x}) \inf_{\mathbf{x}^* \in \partial\Omega_s} \|\mathbf{x} - \mathbf{x}^*\|, \quad s(\mathbf{x}) = \begin{cases} -1, & x \in \Omega_s \\ 1, & \text{else} \end{cases}. \tag{3.19}$$

For practical methods to find the appropriate $\mathbf{x}^*$, refer to the work of Winchenbach *et al.* [311].

The density equation (3.11) is then modified to consider the boundary as

$$\rho_i = \sum_j m_j W_{ij} + \sum_b \rho_0 \lambda_i^b, \tag{3.20}$$

Algorithm 3.1: Boundary refinement for SPH. New steps in our method are marked blue.

SPH Computation
- Establish neighbor list
- Compute and blend density
- Calculate advection forces
- Compute fluid pressure
- Apply boundary pressure and friction
- Update and blend velocity
- Update position

Adaptive Mechanism
- Compute ϕ^b from boundary SDF (Equation (3.23))
- Wake flow preservation (Algorithm 3.2)
- Surface-based adaptivity [313]
- Calculate optimal mass (Equation (3.24))
- Splitting, merging, and redistribution

where b denotes the boundary object(s), and λ_i^b denotes $\lambda(\text{sdf}^b(\mathbf{x}_i))$.

The pressure at the boundary can be mirrored [7] or extrapolated [23] from the pressure of nearby fluid particles. In the case of pressure mirroring, the pressure force from the boundary is

$$\mathbf{F}^p_{i\leftarrow b} = -m_i\rho_0\frac{p_i}{\rho_i^2}\nabla\lambda_i^b. \tag{3.21}$$

The friction between the fluid and solid follows the Coulomb model, which is proportional to the boundary's pressure force on the particle as

$$\mathbf{F}^{\text{fric}}_{i\leftarrow b} = \mu||\mathbf{F}^p_{i\leftarrow b}||\mathbf{t}_{ib}, \tag{3.22}$$

where $\mathbf{t}_{ib}$ is a unit vector pointing to the direction of relative tangential velocity between fluid and boundary.

The total force from the boundary objects to a particle is $\mathbf{F}_i^{\text{bound}} = \sum_b(\mathbf{F}^p_{i\leftarrow b} + \mathbf{F}^{\text{fric}}_{i\leftarrow b})$.

3.5 ADAPTIVE BOUNDARY COUPLING

3.5.1 *Adaptive Boundary Mechanism*

The region near the boundary is often of interest in fluid-boundary coupling. Therefore, it is desirable to use the refined particles near the important boundary objects to emphasize the details of boundary coupling. However, refining the entire surface can cause an unnecessary increase in particle number. Previous work [313] refines particles only near the free surface of the fluid, which cannot cover the entire boundary region

Algorithm 3.2: Wake flow preservation.

for each particle i
  for each solid object of interest b^*:
    if $\text{sdf}^{b^*}(\mathbf{x}_i) \leq h_i$:
      set duration $\tau_{i,t} = \max(\tau_{b^*}^{\max}, \tau_{i,t})$
  decrement duration $\tau_{i,t+1} = \tau_{i,t} - \Delta t$
  if $\tau_i > 0$:
    $\phi_{i,t}^b = \max(\phi_{i,t}^b, \phi_{i,t-1}^b)$

and can cause detail loss in boundary coupling. Moreover, in some cases, only the region near the boundary is of interest, such as a ship sailing on a vast water surface.

Given the above problems, we propose a boundary refinement method to refine particles near specific boundary objects. The method can either be used alone to refine only near the boundary or combined with surface-based refinement schemes [313] to refine both the surface and the boundary regions. It is outlined in Algorithm 3.1.

To achieve particle refinement near boundary objects, we modify the sizing function of Equation (3.15) to consider the distance to boundary objects. Instead of optimizing particle mass according to the distance to the free surface ϕ^f, we use the distance to boundary objects to define a value ϕ^b to adjust the particle size:

$$\phi_i^b = \min\left(\max_{b^*}\left(-\text{sdf}^{b^*}(\mathbf{x}_i) + \phi_{b^*}^{\text{fine}}\right), 0\right), \tag{3.23}$$

where b^* denotes the solid objects of interest near which the fluid particles should be refined. $\phi_{b^*}^{\text{fine}}$ is a user-controlled parameter: all particles with distance to b^* smaller than $\phi_{b^*}^{\text{fine}}$ are refined to the smallest scale. We use $\phi_{b^*}^{\text{fine}} = h_i$ unless explicitly mentioned otherwise.

To only refine the region near the boundary, we replace ϕ_i^f with ϕ_i^b in Equation (3.15), namely

$$m_i^{opt} = m^{\text{base}}\left(\frac{\min(|\phi_i^b|, |\phi_{\max}|)}{|\phi_{\max}|}(1-\alpha) + \alpha\right), \tag{3.24}$$

and to refine both near the boundary region and the fluid surface, we substitute ϕ_i^f in Equation (3.15) with $\phi_i = \max(\phi_i^b, \phi_i^f)$.

3.5.2 *Wake Flow Preservation by Delaying Merge*

For scenarios where the fluid flows past the boundary object or the boundary object moves through the fluid, interesting wake flow effects are generated behind the boundary object, such as the wave behind a

ship moving through water. However, when using ϕ_i^b to determine particle size, the wake flow loses detail because the optimal mass quickly returns to the largest value as the boundary object moves away from the particle. Consequently, the particles in the wake flow merge into larger particles, reducing the detail level.

To preserve wake flow detail, we inhibit the particle merging process if the particle was recently in the vicinity of the boundary, as Algorithm 3.2 shows. For each solid object of interest b^*, the user can set a duration to delay merging $\tau_{b^*}^{\max}$ for particles that have flowed around it, and each particle tracks its remaining duration to delay merging at every time step t, denoted as $\tau_{i,t}$. For each particle i and each solid object of interest b^*, if the particle at position $\mathbf{x}_i$ satisfies $\text{sdf}^{b^*}(\mathbf{x}_i) \leq h_i$, we set its remaining duration $\tau_{i,t} = \max(\tau_{b^*}^{\max}, \tau_{i,t})$. This duration decreases at each time step by the time step length Δt as $\tau_{i,t+1} = \tau_{i,t} - \Delta t$. As long as the particle satisfies $\tau_{i,t} > 0$, ϕ_i^b is not allowed to decrease: if the newly computed value $\phi_{i,t}^b$ for the time step t is lower than the previous value $\phi_{i,t-1}^b$, we set it to $\phi_{i,t}^b = \phi_{i,t-1}^b$. To retain the smoothness of the ϕ^b field, the ϕ_i^b values for $\tau_{i,t} > 0$ are then propagated to surrounding particles using the method from Horvath and Solenthaler [128]. This approach effectively delays particle merging in the wake flow by $\tau_{b^*}^{\max}$.

3.6 RESULTS

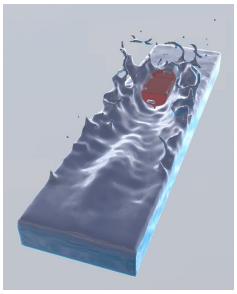
To test the effectiveness and capabilities of our method, in this section, we use multiple scenarios to compare the proposed approach with simulations of uniform particle scale and the surface-based state-of-the-art SPH adaptive scheme [313], which adapts resolution near the free surface.

We use WCSPH [26] for fluid simulation and the semi-analytic boundary handling method [311] for boundary coupling. The time step length is determined using the CFL condition with a scaling parameter of 0.5. Surface reconstruction and rendering are conducted using Houdini and Mantra with Intel Xeon Gold 5218. All simulations are coded using the Taichi programming language [130] and run on an NVIDIA Tesla V100 GPU.

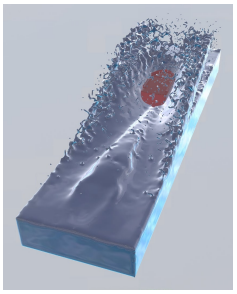
3.6.1 *Efficiency Comparison*

Efficiency is a crucial aspect of adaptive simulation mechanisms. We show that our method can produce a similarly detailed result using fewer particles than the surface-based adaptive mechanism [313] and is more vivid than a low-resolution simulation.

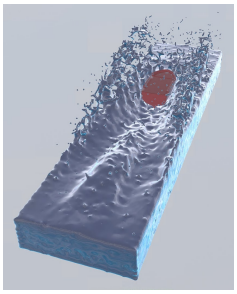
Boat-sailing. In this scenario, a boat sails quickly across a calm fluid surface, stirring up waves and splashes around and behind the body.



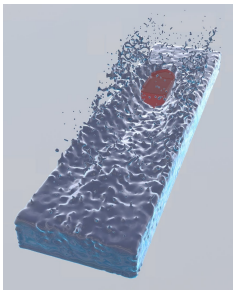
(a) Low-resolution.



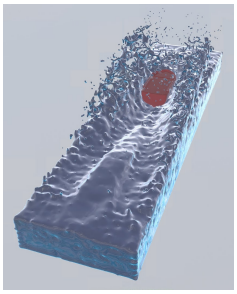
(b) High-resolution.



(c) Surface-based [313].



(d) Ours, $\tau_{\text{boat}}^{\max} = 0$.

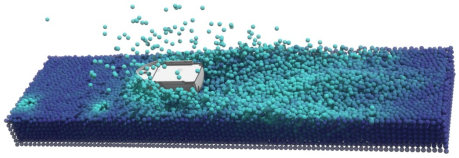


(e) Ours, $\tau_{\text{boat}}^{\max} = 1$.

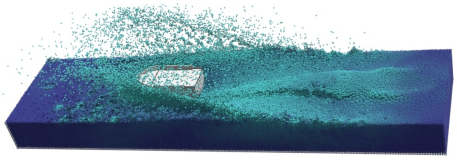
Figure 3.3: Surface reconstruction result of the boat-sailing experiment: (a) and (b) are uniform scale simulations with $r = 0.2$ and $r = 0.063$; (c), (d) and (e) are adaptive simulations using max particle size $r_{\text{base}} = 0.2$ and adaptive ratio $\alpha = 1/32$. Our method in (d) and (e) effectively reduces the average particle count over time (shown in Figure 3.5) when compared to the surface-based method used in (c).

For the adaptive methods, we use the max particle size $r_{\text{base}} = 0.2$ and an adaptive ratio $\alpha = 1/32$ and choose $\phi_{b^*}^{\text{fine}} = 0$ in our method. For uniform size simulations, we perform a low-resolution and a high-resolution simulation using $r = 0.2$ and $r = 0.063$, where the particle size respectively equals the largest and smallest particle size in adaptive simulation. The particles are color-coded according to velocity in Figure 3.4.

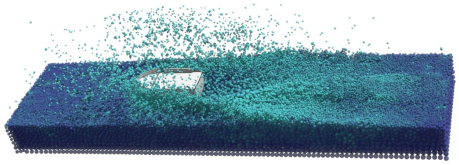
Figure 3.3 and Figure 3.4 show the experimental results rendered with reconstructed fluid surface and particles, respectively. Considering the splashes around the ship, the visual level of detail is similar between the previous surface-based method [313] (Figure 3.3c) and our method with or without wake flow preservation (Figure 3.3e, 3.3d). The three adaptive results are all much more detailed than the uniform low-resolution simulation (Figure 3.3a), though they may lack some detail compared to high-resolution (Figure 3.3b). For the wake flow effect, the result of our method without wake flow preservation (Figure 3.3d) is rela-



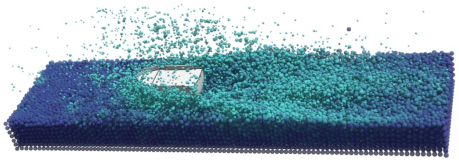
(a) Low-resolution ($r = 0.2$m).



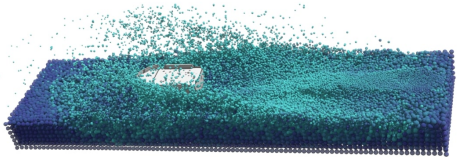
(b) High-resolution ($r = 0.063$m).



(c) Surface-based [313].



(d) Ours, $\tau_{\text{boat}}^{\max} = 0$.



(e) Ours, $\tau_{\text{boat}}^{\max} = 1$.

Figure 3.4: Boat-sailing experiment. Adaptive methods use $r_{\text{base}} = 0.2$ and $\alpha = 1/32$.

tively coarse, but when wake flow preservation is added to our method (Figure 3.3e), the visual detail level is comparable to the surface-based method (Figure 3.3c).

To evaluate the efficiency of the proposed method, the particle count and time consumption using different methods are plotted in Figure 3.5. The solid lines show the particle count, and the dashed lines show the

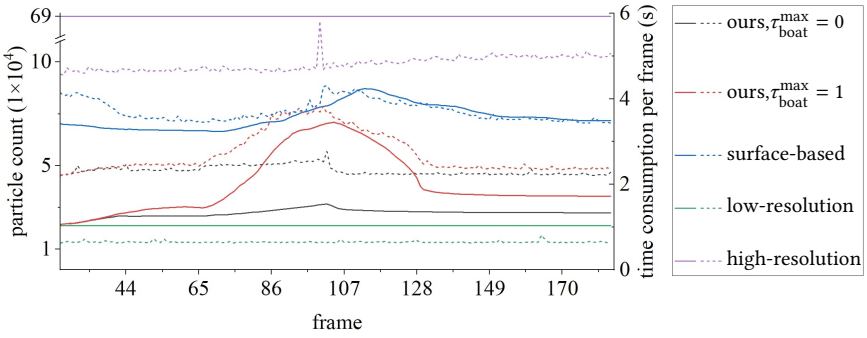
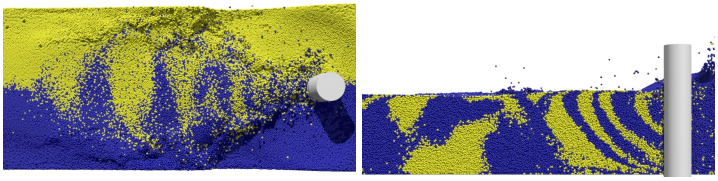


Figure 3.5: Particle count and time consumption in boat-sailing experiment. Solid lines denote particle count; dashed lines denote time consumption. One frame equals 1/24s.

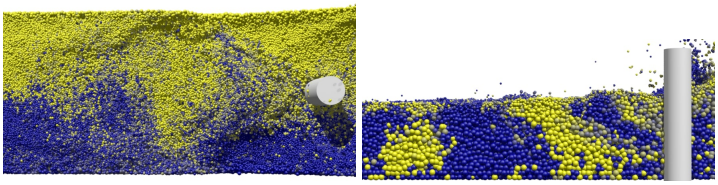
time consumption. We see that the uniform low-resolution simulation uses the least particles, while our adaptive method with $\tau^{\max}_{\text{boat}} = 0$ increases the particle count slightly. Further, our method with $\tau^{\max}_{\text{boat}} = 1$ begins with a low particle count, which increases since frame 65 when the boat starts moving and producing wake flow, and decreases back to a low value after frame 107 when the boat leaves the simulated zone because the duration to delay merging for the wake flow has expired. Under both conditions, our method significantly reduces the average particle count over time compared to the surface-based method [313] and the high-resolution simulation. The time consumption curves follow a similar trend. This experiment shows that our method can reduce the time cost compared to the state-of-the-art surface-based adaptive mechanism [313] with no or negligible negative influence on visual quality.

3.6.2 *Evaluation of Complex Coupling Effects*

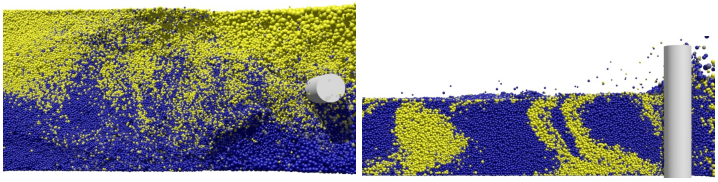
Cylinder moving. Figure 3.6 shows an experiment where a cylinder is moved horizontally through a water tank in a straight line to generate a wave pattern. The particles are colored according to whether their initial positions are on one side of the moving path of the cylinder. A jagged pattern can be seen from the vertical view (first column), and curved patterns can be seen from the side view of a cross-section (second column). Figure 3.6a shows the result of a uniformly scaled high-resolution simulation performed using the smallest allowed particle size in the two adaptive simulations, $r = 0.063$, which should be the most accurate among the three simulations. Figure 3.6c uses our boundary refinement method where particles are adaptively refined near the cylinder. We use $\tau^{\max}_{\text{cylinder}} = 21$, which is longer than the simulation duration, to prevent wake flow merging for the entire simulation to achieve a high level of detail. Figure 3.6b uses surface-based refinement [313].



(a) High-resolution ($r = 0.063$m).



(b) Surface-based [313].



(c) Ours, $\tau^{\max}_{\text{cylinder}} = 21$.

Figure 3.6: Cylinder moving experiment, where the cylinder moves horizontally in the tank. The first column is the vertical view; the second column is a cross-section of the side view. Our method better reproduces the jagged (vertical view) and curved (side view) wave patterns in the high-resolution simulation. Adaptive methods use max particle size $r_{\text{base}} = 0.2$ and $\alpha = 1/32$.

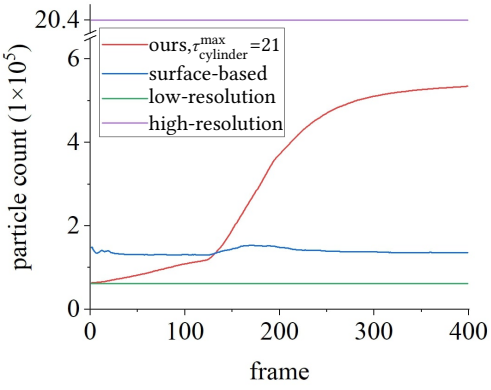
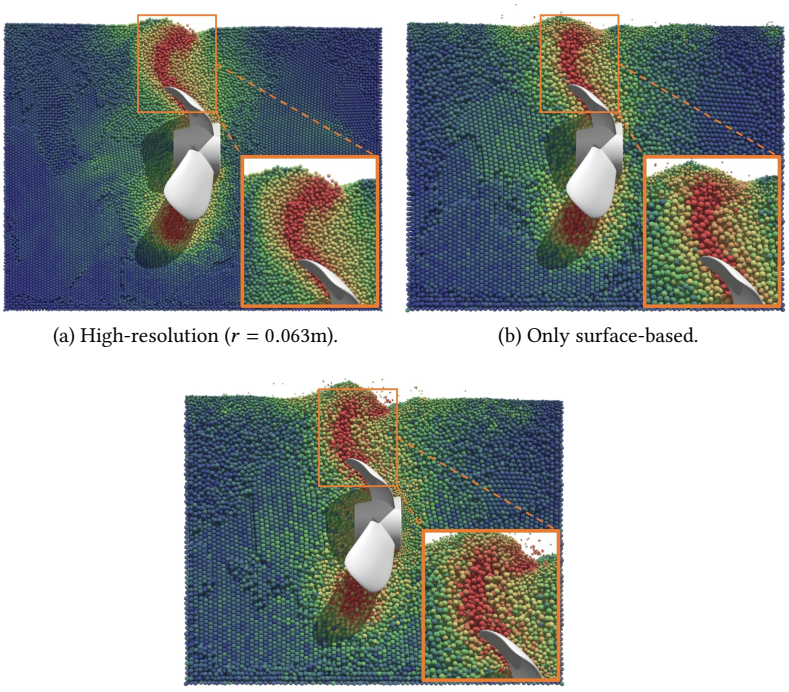


Figure 3.7: Particle count in cylinder moving experiment. Solid lines denote particle count. One frame equals 1/24s.



(a) High-resolution ($r = 0.063$m).

(b) Only surface-based.

(c) Ours + surface-based.

Figure 3.8: A cross-section of the propeller spinning experiment where color denotes velocity. Adaptive methods use $r_{\text{base}} = 0.1$, $\alpha = 1/32$, $\tau^{\text{max}}_{\text{propeller}} = 0$. In the zoomed-in areas, our method (b) generates a more detailed vortex compared to (c), closer to the high-resolution result (a).

The two adaptive simulations use $r_{\text{base}} = 0.2$ and $\alpha = 1/32$. Compared to the surface-based method, our method is more similar to the high-resolution simulation in terms of the size and number of jags from the vertical view and the shape of the curved pattern in the cross-section, indicating our method can achieve a better accuracy compared to surface-based method [313] under certain settings.

The particle count of the cylinder moving experiment is displayed in Figure 3.7. Our method uses a larger number of particles compared to the surface-based method [313] to achieve higher accuracy, but the particle count is still considerably lower than the high-resolution simulation.

Propeller spinning. In this experiment, a propeller is fully submerged in the fluid and spins horizontally to generate turbulence. We

use this scenario to demonstrate the ability of our method to be combined with surface-based refinement [313].

Figure 3.8 shows the particle view of a cross-section of the propeller spinning experiment. From Figure 3.8c, it can be seen that the particle size is refined both near the surface and around the propeller. Compared with only using surface-based refinement as in Figure 3.8b, the combined method can generate more details in a vortex, as shown in the zoomed-in areas, and also has a more similar shape to the vortex in a high-resolution simulation (Figure 3.8a). This experiment shows that adding our method to the surface-based method [313] can enhance dynamic details produced by boundary coupling.

3.7 DISCUSSION AND CONCLUSION

3.7.1 *Summary*

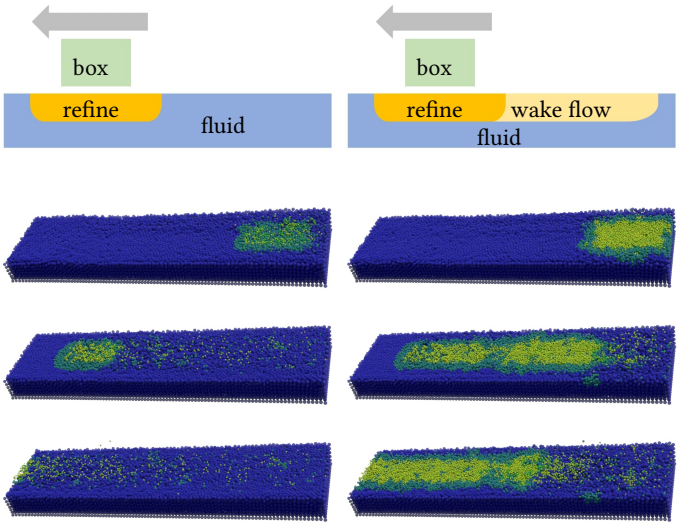
We proposed a new boundary refinement mechanism for adaptive SPH to refine particles near the coupling boundary and retain resolution for the flow generated by solid objects. Experiments show that, compared to the surface-based only refinement mechanism, our method can improve the computation efficiency significantly for enhancing fluid details when simulating fluid-solid coupling scenarios. Further, the proposed method can produce more accurate visual results for underwater coupling scenarios such as propeller spinning. Therefore, our method can be widely applied to most fluid simulation scenarios to improve efficiency and visual quality since many fluid simulations focus on the interaction between fluid and solid.

A limitation of our method is that when a solid particle leaves the refined region due to splashing etc., it lacks merge partners and remains in the unrefined region, causing a slightly higher number of particles than the ideal state. Also, the splitting and merging process of our method may cause a potential violation of energy conservation and generate visual artifacts such as unrealistic splashes.

In the future, we will explore a more stable and accurate particle splitting-merging mechanism for adaptive SPH fluid simulation. The critical issue is balancing the particles' sizes close to each other and creating a progressive merging process to avoid a sudden change of local density field.

3.7.2 *Limitations*

Visual effects. In the surface-based only SPH adaptive mechanism, the sizes of the particles are always transformed smoothly in space according to a continuous and stable distance. However, our boundary-based adaptive method may encounter the difficulties of particle merg-



(a) Our method without wake flow preservation ($\tau_{\text{box}}^{\max} = 0$).

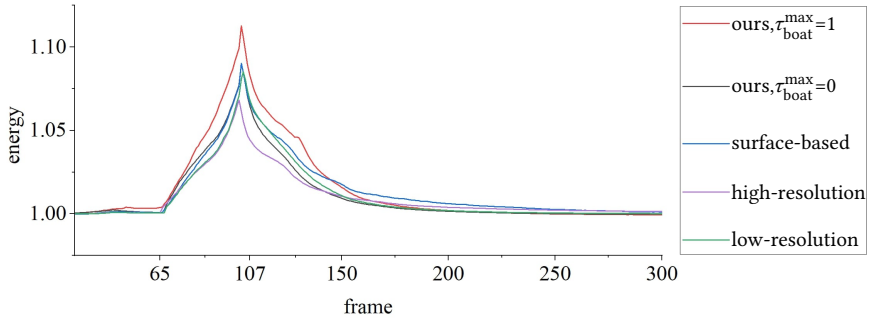
(b) Our method with wake flow preservation ($\tau_{\text{box}}^{\max} = 1$).

Figure 3.9: Moving box experiment. In this experiment, a box-shaped boundary object moves close to the fluid surface but does not touch the fluid, causing particle refinement near it but not disturbing it. The leftmost column shows the experimental setup; the other columns show the result. Color denotes particle size. Our method uses $r_{\text{base}} = 0.2$ and $\alpha = 32$. The distance between the box and the fluid surface is 0.1.

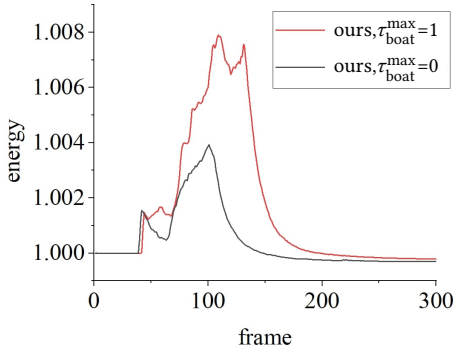
ing. The divided small particles, as a part of the waves stirred up by the solid boundary, may not have suitable s or S neighbor particles to be merged when falling back to the fluid surface.

As shown in Figure 3.9, we make a solid box fly over the fluid surface without substantive contact to demonstrate this issue. Only the sizes of the fluid particle will be affected by the signed distance field from the solid object. We carried out this experiment separately with and without the wake flow preservation mechanism. The sizes of the particles are color-coded with smaller particles having brighter appearances. Here we can see that our wake flow preservation mechanism delays the merging process effectively.

However, we can also observe that when the solid object leaves the fluid, some small particles are left unmerged on the surface. This is because our method only allows the merging procedure to take place between particles of class s or S. When small particles begin to merge, there may be no s or S particles in the range of merging. On the other hand, the surface-based adaptive mechanism naturally guarantees fluid particles near each other obtain similar sizes as it only depends on parti-



(a) Boat-sailing experiment.



(b) Moving box experiment.

Figure 3.10: Statistical analysis of energy change (sum of total kinetic and gravitational potential energy) in the boat-sailing and moving box experiments with a frame rate of 24 fps. The energy change in time is expressed with the ratio of the current time's energy to the first frame's.

cles' distances to the fluid surface, so particles of suitable class are easier to be found.

Energy conservation. The particle splitting and merging techniques applied in our work may also introduce potential energy conservation violations into the simulation. We measured the total energy (including kinetic and gravitational potential energy) for the boat-sailing experiment in Figure 3.4 and the moving box experiment in Figure 3.9 and drew the statistical diagram in Figure 3.10.

In Figure 3.10a, we can see that for the boat-sailing experiment, both our method and the surface enhancement method [313] obtain energy higher than both high-resolution and low-resolution simulation results. This manifests in the random spatter of fluid particles during the simulation, indicating the instability issue of the method. This issue is further

demonstrated in Figure 3.10b, where, as the chart shows, the violation of energy conservation happens during the merging process. The energy cannot remain stable when the solid box starts to cause the fluid to split and merge. Moreover, for the experiment applying wake flow preservation, the period of conserving energy performs worse. Although this instability is relatively small (generally less than 1‰ energy fluctuation), it can generate unexpected artifacts if the simulation scenario is not carefully configured.

4

IMPLICITLY STABLE MIXTURE MODEL FOR DYNAMIC MULTI-FLUID SIMULATIONS

As pointed out in Section 2.5, particle-based simulations have become increasingly popular for generating highly dynamic fluid effects. However, the swift and stable simulation of interactions among distinct fluids continues to pose challenges for current mixture model techniques. When using a single-mixture flow field to represent all fluid phases, numerical discontinuities in phase fields can result in significant losses of dynamic effects and unstable conservation of mass and momentum. To tackle these issues, we present an advanced implicit mixture model for smoothed particle hydrodynamics. Instead of relying on an explicit mixture field for all dynamic computations and phase transfers between particles, our approach calculates phase momentum sources from the mixture model to derive explicit and continuous velocity phase fields. We then implicitly obtain the mixture field using a phase-mixture momentum-mapping mechanism that ensures the conservation of incompressibility, mass, and momentum. In addition, we propose a mixture viscosity model and establish viscous effects between the mixture and individual fluid phases to avoid instability under extreme inertia conditions. Through a series of experiments, we show that, compared to existing mixture models, our method effectively improves dynamic effects while reducing critical instability factors. This makes our approach especially well-suited for long-duration, efficiency-oriented virtual reality scenarios[1].

This chapter begins with an introduction to current methods of simulating multiphase fluids using SPH, discussing their limitations and unresolved challenges in Section 4.1. We proceed by detailing the derivation of governing equations for multiphase fluid simulation through a volume fraction scheme in Section 4.2, laying the groundwork for our physical model. Before delving deeper into our proposed multiphase model, we present enhancements to the SPH simulation solver for implicitly handling fluid interactions with high-density ratios in Section 4.3. Our contributions to improving SPH multiphase simulation are proposed in Section 4.4. A series of experiments in Section 4.5 showcase the efficacy of our method. The chapter concludes with a discussion on the contributions made in this research area in Section 4.6.

1 Parts of this chapter were published in: Y. Xu, X. Wang, J. Wang, C. Song, T. Wang, Y. Zhang, J. Chang, J. J. Zhang, J. Kosinka, A. C. Telea, and X. Ban. An Implicitly Stable Mixture Model for Dynamic Multi-fluid Simulations. In *SIGGRAPH Asia 2023 Conference Paper*. ACM, 2023. doi: 10.1145/3610548.3618215

Figure 4.1: Tea diffuses from a teabag into a cup of water. After the teabag is removed, the mixture is stirred with a (glass) rod.

4.1 INTRODUCTION

Simulating multiphase flows [43] has gained interest in the graphics community due to its ability to create distinct diffusion and stratification effects, such as brewing tea bags (Figure 4.1) and crafting cocktails. Simplified mixture models [196] have been widely adopted to produce stunning visuals [155, 239] at lower computational cost. We discussed related work in multiphase flows in Section 2.5. Here we refine that discussion with additional references which bring more insight into the context of our work discussed in this chapter.

Mixture models deal with the coexistence of multiple phases within a singular discretization unit. A distinctive drift velocity is leveraged to illustrate the relative motion between the mixture and individual phases. This movement stems from the multiphase governing equations, which, in turn, find their roots in the single-phase Navier-Stokes equations. A significant challenge in this domain arises from the fact that many fluid simulation techniques are tailored primarily for single-phase fluids. Designing a suitable method that can integrate additional sub-discretization unit dynamics computation is far from trivial.

A common workaround to this complexity has been the adoption of the *Local Equilibrium Assumption (LEA)* [156, 238, 239]. While the LEA simplifies simulations by forcing drift states back to non-drift states, enabling compatibility with existing simulation models, it is not without limitations. Such an approach can often introduce discontinuities

in phase velocities, compromising the transfer between phases during time integration. Consequently, this diminishes the dynamism stemming from multiphase interactions.

Alternatively, energy-based multiphase models have been explored [284, 334]. Although they can bypass the challenges of LEA and drift velocity computations associated with phase-phase interactions, these models often make concessions in numerical accuracy to align with existing simulation structures.

Numerical instability is another challenge for multiphase flow simulation. The standard SPH strategy [28] struggles to conserve mass and momentum using the volume fraction scheme. Jiang *et al.* [155] abandoned LEA for improved dynamic performance but sacrificed mixture incompressibility and mass conservation.

We propose an implicit mixture model to address dynamic and stability issues in multiphase flow simulation. We construct interphase momentum sources using no-slip and free-slip conditions among phases, calculate the change rate of phase velocity fields, and conserve divergence-free conditions for mixture flow and numerical consistency for phase velocities. To avoid instability caused by excessive drift velocities, we introduce a mixture viscosity model to smoothly limit drift velocity while maintaining conservation. The main contributions of our implicit mixture model are:

- A volumetric flux SPH solver that effectively alleviates numerical instability and improves simulation efficiency for iterative PPE solvers when dealing with interactions of high density ratio fluids;
- An *analytic form* of interphase momentum sources describing how the mixture flow affects the movement of individual fluid phases;
- A phase-mixture *momentum mapping mechanism* that preserves the incompressibility of the mixture and ensures numerical consistency and accuracy of the phases;
- A mixture *viscosity model* to prevent instability arising from overly dynamic interphase movement.

4.2 PRELIMINARIES

4.2.1 *Volume Fraction Scheme*

The volume fraction scheme [196] models the coexistence state of phases within a single fluid parcel (see Figure 4.2). For each fluid parcel, the mixture model divides each particle into several portions according to the number of fluid phases. For each phase in each parcel, it has a

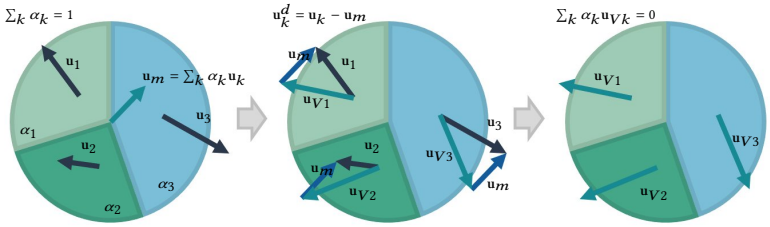


Figure 4.2: An illustration of the concept of the mixture model. The images show a particle with mixed phases: (a) relationship between rest volume fraction α_k, phase velocity $\mathbf{u}_k$, and mixture velocity $\mathbf{u}_m$; (b) relationship between $\mathbf{u}_k$, $\mathbf{u}_m$ and drift velocity $\mathbf{u}_{Vk}$; (c) drift velocity $\mathbf{u}_{Vk}$.

volume fraction α_k (corresponds to c_k^V in Equation (2.17)), denoting the portion it takes in the parcel. k denotes the fluid phase. For a single fluid parcel, all phases should conserve

$$\sum_k \alpha_k = 1. \tag{4.1}$$

Let $\mathbf{u}_k$ denote the velocity field of phase k. The mixture velocity giving the velocity of the volume center of a fluid parcel is

$$\mathbf{u}_m = \sum_k \alpha_k \mathbf{u}_k, \tag{4.2}$$

where $\mathbf{u}_m$ is called *mixture velocity*, or *volume flux* for it describes the motion by volume rather than mass. The mixture velocity conserves $\nabla \cdot \mathbf{u}_m = 0$ as it represents the mixture's incompressibility [156].

The *drift velocity* further depicts the relative motion of a fluid phase with respect to the velocity of the fluid parcel ($\mathbf{u}_m$). The drift velocity of phase k is given by

$$\mathbf{u}_{Vk} = \mathbf{u}_k - \mathbf{u}_m. \tag{4.3}$$

According to Equation (4.2), the drift velocity should further follow

$$\sum_k \alpha_k \mathbf{u}_{Vk} = 0. \tag{4.4}$$

Similarly, the mixture density can be computed using the rest density ρ_k of each phase as

$$\rho_m = \sum_k \alpha_k \rho_k. \tag{4.5}$$

4.2.2 *Governing Equations of the Mixture Model*

The governing equations of the mixture model originate from the Navier-Stokes model (Equation (2.6)), which considers the dynamic computation of a single fluid field.

Conservation of mass. Starting from the continuity Equation (2.1), compared to the conservation of mass for a single fluid phase (2.3), the volume fraction as well as the coexistence states with other phases should be taken into consideration. Taking $\alpha_k \rho_k$ into Equation (2.1) for phase k, we have

$$\frac{\partial}{\partial t}\left(\alpha_k \rho_k\right) + \nabla \cdot \left(\alpha_k \rho_k \mathbf{u}_k\right) = 0. \tag{4.6}$$

Note that we do not consider the conversion of one phase to another, making the right-hand side of Equation (4.6) zero.

Mass transfer. By substituting Equation (4.3) into Equation (4.6), we can transform Equation (4.6) into the form of

$$\begin{aligned}
\frac{\partial}{\partial t}\left(\alpha_k \rho_k\right) + \nabla \cdot \left(\alpha_k \rho_k \mathbf{u}_k\right) &= 0 \\
\rho_k \frac{\partial \alpha_k}{\partial t} + \nabla \cdot \left(\alpha_k \rho_k \left(\mathbf{u}_{Vk} + \mathbf{u}_m\right)\right) &= 0 \\
\rho_k \frac{\partial \alpha_k}{\partial t} + \rho_k \left(\mathbf{u}_m \cdot \nabla\right) \alpha_k &= -\rho_k \nabla \cdot \left(\alpha_k \mathbf{u}_{Vk}\right) \\
\frac{\partial \alpha_k}{\partial t} + \left(\mathbf{u}_m \cdot \nabla\right) \alpha_k &= -\nabla \cdot \left(\alpha_k \mathbf{u}_{Vk}\right) \\
\frac{D}{Dt} \alpha_k &= -\nabla \cdot \left(\alpha_k \mathbf{u}_{Vk}\right),
\end{aligned} \tag{4.7}$$

which captures how the substance of fluid phase transfers from one fluid parcel to another.

Conservation of momentum. Similar to Equation (2.5), the mixture model replaces the single phase momentum term $\rho \mathbf{u}$ with $\alpha_k \rho_k \mathbf{u}_k$, leading to

$$\frac{\partial}{\partial t}\left(\alpha_k \rho_k \mathbf{u}_k\right) + \nabla \cdot \left(\alpha_k \rho_k \mathbf{u}_k \otimes \mathbf{u}_k\right) = -\alpha_k \nabla p_k + \nabla \cdot \left(\alpha_k \boldsymbol{\tau}_k\right) + \alpha_k \rho_k \mathbf{g} + \mathbf{s}_k^I, \tag{4.8}$$

where p_k, $\boldsymbol{\tau}_k$, $\mathbf{g}$, and $\mathbf{s}_k^I$ are the pressure, viscous stress tensor, gravity, and interphase momentum source, respectively. Equation (4.8) ensures momentum conservation for all phases. Despite appending volume fraction to all momentum terms for the right-hand-side of Equation (2.6), an additional interphase momentum source needs to be taken into consideration to analyse how a single fluid phase is affected by the others

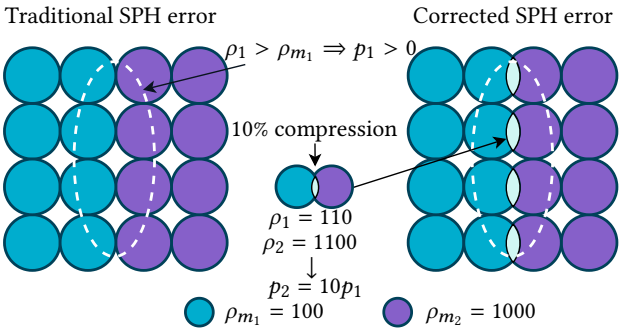


Figure 4.3: The pressure error of traditional SPH and corrected SPH [263] when simulating fluids with different densities.

in the mixture model. This obeys a rigorous force balance model [196] complying with

$$\sum_k \mathbf{s}_k^I = 0. \tag{4.9}$$

4.3 VOLUME FLUX FREE SPH

The state-of-the-art SPH implicit pressure solver, DFSPH [29], prevents volume compression and enforces a divergence-free velocity field. This method is significantly faster and more stable than previous SPH methods [139, 265] for incompressible fluids. DFSPH relates density compression and pressure via

$$p_i = \kappa(\rho_i - \rho_{m_i}), \tag{4.10}$$

where κ is the stiffness coefficient determined by the magnitude of compressibility, and ρ_{m_i} is the rest density of the particle i in the mixture model as computed by Equation (4.5). The pressure gradient can therefore be expressed using Equation (3.4) as

$$\nabla p_i = \kappa_i \nabla \rho_i = \kappa_i \sum_j m_j \nabla W_{ij}. \tag{4.11}$$

However, in the simulation of fluids with non-uniform rest density fields, the rest density of different particles would vary. As a result, DFSPH will have a large error when the densities of adjacent particles are significantly different, as illustrated in Figure 4.3. There are two types of particles, colored cyan and purple, with rest density either $\rho_{m_1} = 100kg/m^3$ or $\rho_{m_2} = 1000kg/m^3$. According to Equation (3.11), although the particles are evenly distributed in space without overlapping, the density near the interface between the two fluids is different from the rest density. This causes one type of particles to move away

Algorithm 4.1: Volume flux free SPH solver

A. Attribute preparation

1. attributes from previous time step: $\mathbf{u}_m$, α_k
2. $\mathbf{u}^*_{m_i} \leftarrow \mathbf{u}_{m_i}$
3. **compute** m_i (4.5), χ_i (4.14), V_i (4.15), β_i (4.28)

B. Projection step: volume flux free solver

4. **while** avg $\left(\frac{DV_i}{Dt}\right) < \eta_1$

4.1 $\frac{DV_i}{Dt} = -V_i \nabla \cdot \mathbf{u}_{m_i}$ (4.24)

4.2 $k_i = \frac{DV_i}{Dt} \Delta t^{-1} \left(V_i^0\right)^{-3} \beta_i$ (4.28)

4.3 $\mathbf{u}^*_{m_i} \leftarrow \mathbf{u}^*_{m_i} - \Delta t \mathbf{F}^p_{i,\text{total}} / m_i$ (4.30)

5. $\mathbf{u}_{m_i} = \mathbf{u}^*_{m_i}$ **(update velocity)**

C. Advection step

6. $\mathbf{u}^*_{m_i} \leftarrow \mathbf{u}_{m_i} + \Delta t \left(\mathbf{F}^{vis}_i / m_i + \mathbf{g}\right)$ (3.14)

D. Projection step: VFSPH constant volume solver

7. **while** avg$(\Delta V_i) < \eta_2$

7.1 $\frac{DV_i}{Dt} = -V_i \nabla \cdot \mathbf{u}_{m_i}$ (4.24)

7.2 $V^*_i = V_i + \Delta t \frac{DV_i}{Dt}$

7.3 $\Delta V_i = V^*_i - V^0_i$

7.4 $k_i = \Delta V_i \Delta t^{-2} \left(V_i^0\right)^{-3} \beta_i$ (4.28)

7.5 $\mathbf{u}^*_{m_i} \leftarrow \mathbf{u}^*_{m_i} - \Delta t \mathbf{F}^p_{i,\text{total}} / m_i$ (4.30)

8. $\mathbf{u}_{m_i} = \mathbf{u}^*_{m_i}$ **(update velocity)**

9. $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{u}_{m_i}$ **(update position)**

from those of the other type, which then leads to an unwanted gap between them.

Solenthaler and Pajarola [263] discussed this phenomenon in detail, and provided a corrected version of the density computation in their work as

$$\rho_i = \sum_j m_i W_{ij}, \tag{4.12}$$

where the density of all neighboring particles j is replaced with the density of particle i itself when computing the density. However, the limitation of this scheme is that it ignores the proportional relationship between the volume and the pressure, resulting in unreasonable pressure, which in turn makes the system more prone to instability.

To resolve the issue mentioned in Figure 4.3, we propose the Volume Flux free SPH method (VFSPH). The concepts of the motion mass point and density compression are replaced with volumetric flux and an abstract notion which we call *compression ratio*.

We first introduce the notion of *rest volume* V^0 representing the volume in the incompressible state, expressed as

$$V_i^0 = \frac{m_i}{\rho_{m_i}}. \tag{4.13}$$

We then introduce the notion of *compression ratio* χ, which is defined for a particle i using the rest volume as

$$\chi_i = \sum_j V_j^0 W_{ij}. \tag{4.14}$$

The volume derived from SPH can be expressed using the compression ratio and the rest volume as

$$V_i = \chi_i V_i^0 = V_i^0 \sum_j V_j^0 W_{ij}. \tag{4.15}$$

Our VFSPH reduces to the traditional SPH method when all particles share the same rest density

$$\rho_i = \chi_i \rho_{m_i}. \tag{4.16}$$

However, the value of density approximation differs between VFSPH and SPH when dealing with a non-uniform rest density field because of the approximation error of the traditional SPH form. Hence, we use Equation (4.16) to model density in the rest of our work.

In VFSPH, the original SPH approximation (2.7) becomes

$$A_i = \sum_j \frac{V_j^0}{\chi_j} A_j W_{ij}, \tag{4.17}$$

with the gradient given by

$$\nabla A_i = \sum_j \frac{V_j^0}{\chi_j} A_j \nabla W_{ij}. \tag{4.18}$$

Following Equation (3.6), another differential form, which can further reduce the approximation error of Equation (4.18), is

$$\nabla \cdot A_i = \frac{1}{\chi_i} \sum_j V_j^0 (A_j - A_i) \cdot \nabla W_{ij}. \tag{4.19}$$

Based on the above model, we now adapt the fluid simulation SPH solver of DFSPH [29] for VFSPH. We start by defining that pressure is proportional to the compression ratio of the volume

$$p_i = k_i (V_i - V_i^0) = k_i (\chi_i - 1) V_i^0, \tag{4.20}$$

where k_i is the stiffness coefficient, which we derive further below. The gradient of the pressure via Equation (4.18) becomes

$$\nabla p_i = k_i V_i^0 \nabla \chi_i = k_i V_i^0 \sum_j V_j^0 \nabla W_{ij}. \tag{4.21}$$

The pressure force is then computed as

$$\mathbf{F}_i^p = -\frac{m_i}{\rho_i}\nabla p_i = -\frac{k_i}{\chi_i}\left(V_i^0\right)^2 \sum_j V_j^0 \nabla W_{ij}. \tag{4.22}$$

For the conservation of momentum, the pressure force has the form

$$\mathbf{F}_{i\to j}^p = \frac{k_i}{\chi_i}\left(V_i^0\right)^2 V_j^0 \nabla W_{ij}. \tag{4.23}$$

The rate of change of the volumetric flux at the location of the particle i can be expressed using divergence according to Equation (4.19) as

$$\frac{DV_i}{Dt} = -V_i \nabla\cdot\mathbf{u}_{m_i} = -V_i^0 \chi_i \nabla\cdot\mathbf{u}_{m_i} = V_i^0 \sum_j V_j^0 (\mathbf{u}_{m_i} - \mathbf{u}_{m_j})\cdot\nabla W_{ij}, \tag{4.24}$$

where $\mathbf{u}_{m_i}$ is the mixture velocity for particle i. In the above, in contrast to the original SPH method which uses $\mathbf{u}$ for the velocity of particles as the velocity of their mass center, we use the volumetric flux $\mathbf{u}_m$ [196] to model the velocity of the volume center of each particle in VFSPH.

The volumetric flux and velocity share the same meaning if the fluid is incompressible and no phase exchange happens between particles. The velocity after the advection step (caused by forces except pressure in Equation (2.6)) is

$$\mathbf{u}_{m_i}^* \leftarrow \mathbf{u}_{m_i} + \Delta t\left(\frac{\mu_i}{\rho_i}\nabla^2\mathbf{u}_{m_i} + \mathbf{g}\right), \tag{4.25}$$

where Δt is the time step size, and $\mathbf{u}_{m_i}^*$ is the mixture velocity of particle i in the advection state. The pressure force is used to counteract the compressibility introduced during the advection step. According to Equation (4.24), we have that

$$\sum_j V_j^0 (\mathbf{u}_{m_i}^* - \mathbf{u}_{m_j}^*)\cdot\nabla W_{ij} = -\Delta t \sum_j V_j^0 \left(\frac{\mathbf{F}_i^p}{m_i} - \frac{\mathbf{F}_{i\to j}^p}{m_j}\right)\cdot\nabla W_{ij}. \tag{4.26}$$

Combining Equation (4.24) and Equation (4.26) yields

$$\begin{aligned}\frac{DV_i}{Dt} &= \Delta t V_i^0\left(-\sum_j V_j^0 \frac{\mathbf{F}_i^p}{m_i}\cdot\nabla W_{ij} + \sum_j V_j^0 \frac{\mathbf{F}_{i\to j}^p}{m_j}\cdot\nabla W_{ij}\right)\\ &= \frac{\Delta t k_i \left(V_i^0\right)^3}{\chi_i}\left(\frac{1}{m_i}\left(\sum_j V_j^0 \nabla W_{ij}\right)^2 + \sum_j \frac{\left(V_j^0\nabla W_{ij}\right)^2}{m_j}\right).\end{aligned} \tag{4.27}$$

Hence the stiffness coefficient k is

$$k_i = \frac{DV_i}{Dt}\frac{1}{\Delta t}\frac{1}{(V_i^0)^3}\underbrace{\frac{\chi_i}{\frac{\left(\sum_j V_j^0 \nabla W_{ij}\right)^2}{m_i} + \sum_j \frac{\left(V_j^0 \nabla W_{ij}\right)^2}{m_j}}}_{\beta_i}. \tag{4.28}$$

Considering the conservation of momentum, the total pressure force of particle i is

$$\begin{aligned}\mathbf{F}^p_{i,\text{total}} &= \mathbf{F}^p_i + \sum_j \mathbf{F}^p_{j\to i} \\ &= -\frac{k_i}{\chi_i}(V_i^0)^2 \sum_j V_j^0 \nabla W_{ij} - V_i^0 \sum_j \frac{k_j}{\chi_j}(V_j^0)^2 \nabla W_{ij}.\end{aligned} \tag{4.29}$$

Finally, the change of volumetric flux caused by $\mathbf{F}^p_{i,\text{total}}$ is

$$\mathbf{u}^*_{m_i} \leftarrow \mathbf{u}^*_{m_i} - \frac{\Delta t V_i^0}{m_i}\left(\frac{k_i}{\chi_i}V_i^0 \sum_j V_j^0 \nabla W_{ij} + \sum_j \frac{k_j}{\chi_j}(V_j^0)^2 \nabla W_{ij}\right). \tag{4.30}$$

The whole procedure of the VFSPH solver is summarized by Algorithm 4.1. We now proceed to our multiphase model.

4.4 IMPLICIT MIXTURE MODEL FOR MULTIPHASE INTERACTIONS

We now introduce our implicit mixture model (see the bottom part of Figure 4.4). To ease reading, Table 4.1 lists all momentum sources used throughout our exposition. Note that the advection momentum source in Table 4.1 is the advection caused by drift velocity, not the term in the Navier-Stokes equation caused by fluid velocity; the latter is the mixture velocity in our work. We also adopt the notation $p_k = p_m$ for all mixture phases [196, 239].

Table 4.1: Notations of momentum sources (Section 4.4).

Phase momentum sources $\mathbf{s}_k$			**Corresponding mixture momentum sources $\mathbf{s}_m$**	**Corresponding interphase momentum sources $\mathbf{s}^I_k$**
Component	**Notation**	**Value**	**Notation & Values**	**Notation & Values**
pressure	$\mathbf{s}^p_k$	$-\alpha_k \nabla p_k$	$\mathbf{s}^p_m = \sum_k \mathbf{s}^p_k = -\nabla p_m$	$\mathbf{s}^{Ip}_k$
viscosity	$\mathbf{s}^v_k$	$\nabla \cdot (\alpha_k \boldsymbol{\tau}_k)$	$\mathbf{s}^v_m = \sum_k \mathbf{s}^v_k = \nabla \cdot \boldsymbol{\tau}_m$	$\mathbf{s}^{Iv}_k$
gravity	$\mathbf{s}^g_k$	$\alpha_k \rho_k \mathbf{g}$	$\mathbf{s}^g_m = \sum_k \mathbf{s}^g_k = \rho_m \mathbf{g}$	–
advection	$\mathbf{s}^a_k$	$-\alpha_k \rho_k \mathbf{u}_{Vk} \cdot \nabla \mathbf{u}_k$	$\mathbf{s}^a_m = \sum_k \mathbf{s}^a_k$	$\mathbf{s}^{Ia}_k$
drift	$\mathbf{s}^d_k$	$-\alpha_k \rho_k \frac{D}{Dt}\mathbf{u}_{Vk}$	$\mathbf{s}^d_m = \sum_k \mathbf{s}^d_k$	–

4.4.1 *Interphase Momentum*

Considering the conservation of mass terms inherited from Equation (4.6), the left-hand side of Equation (4.8) is subject to a transformation. This revised representation, encompassing the predefined momentum sources, is given by

$$\begin{aligned}
&\frac{\partial}{\partial t}\left(\alpha_k \rho_k \mathbf{u}_k\right) + \nabla \cdot \left(\alpha_k \rho_k \mathbf{u}_k \otimes \mathbf{u}_k\right) \\
&= \underbrace{\mathbf{u}_k \frac{\partial}{\partial t}\left(\alpha_k \rho_k\right) + \mathbf{u}_k \nabla \cdot \left(\alpha_k \rho_k \mathbf{u}_k\right)}_{\text{Equals 0 according to Equation (4.6)}} + \alpha_k \rho_k \frac{\partial}{\partial t}\mathbf{u}_k + \alpha_k \rho_k \mathbf{u}_k \cdot \nabla \mathbf{u}_k \\
&= \alpha_k \rho_k \left(\frac{\partial}{\partial t}\mathbf{u}_k + \mathbf{u}_k \cdot \nabla \mathbf{u}_k\right) \\
&= \alpha_k \rho_k \left(\frac{\partial}{\partial t}\mathbf{u}_k + \left(\mathbf{u}_m + \mathbf{u}_{Vk}\right) \cdot \nabla \mathbf{u}_k\right) \\
&= \alpha_k \rho_k \frac{D}{Dt}\mathbf{u}_k - \mathbf{s}_k^a.
\end{aligned} \tag{4.31}$$

Then, Equation (4.8) can be rewritten as

$$\alpha_k \rho_k \frac{D}{Dt}\mathbf{u}_k = \mathbf{s}_k^p + \mathbf{s}_k^v + \mathbf{s}_k^g + \mathbf{s}_k^I + \mathbf{s}_k^a. \tag{4.32}$$

Here, $\alpha_k \rho_k \frac{D}{Dt}\mathbf{u}_k$ can be further split as $\alpha_k \rho_k \frac{D}{Dt}\mathbf{u}_m - \mathbf{s}_k^d$ according to Equation (4.3). By summing up Equation (4.32) for all phases under the force balance model [196] condition $\sum_k \mathbf{s}_k^I = 0$, the mixture's momentum is

$$\rho_m \frac{D}{Dt}\mathbf{u}_m = \mathbf{s}_m^p + \mathbf{s}_m^v + \mathbf{s}_m^g + \mathbf{s}_m^d + \mathbf{s}_m^a. \tag{4.33}$$

Subtracting Equation (4.33) from Equation (4.32) yields the change rate of drift velocity

$$\frac{D}{Dt}\mathbf{u}_{Vk} = \frac{D}{Dt}\mathbf{u}_k - \frac{D}{Dt}\mathbf{u}_m = \frac{\mathbf{s}_k^{\{p,v,g,a,I\}}}{\alpha_k \rho_k} - \frac{\mathbf{s}_m^{\{p,v,g,d,a\}}}{\rho_m}, \tag{4.34}$$

where $\{\}$ denotes the sum of the specified momentum sources.

Since the standard mixture model does not explicitly define the interphase momentum source $\mathbf{s}_k^I$, various forms of it have been introduced in previous works. Ren *et al.* [239] proposed a comprehensive term to cover both inertia-induced drag and diffusion effects. Jiang *et al.* [156] simplified it to model only the drag effect. However, these two terms make it challenging to maintain *stable* simulation systems. The LEA must ensure that the drift velocity returns to zero at the start of each time step. Jiang *et al.* [155] further presented a more artificial, yet stabler, approach to enable the continuity of drift velocity. In our work, we

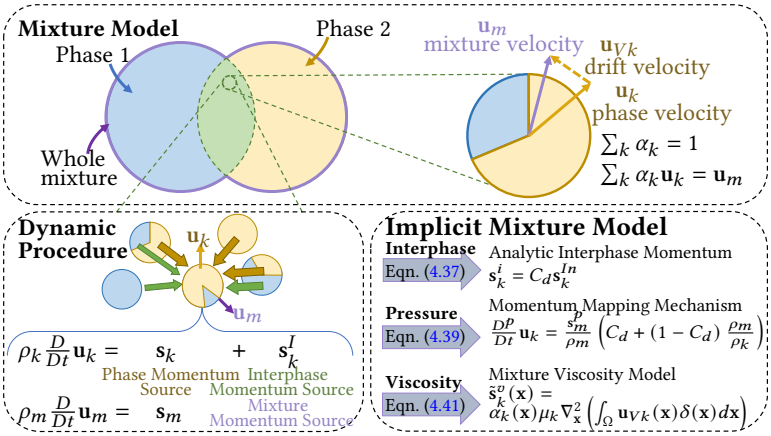


Figure 4.4: Schematic diagram of our proposed implicit mixture model. We use a volume fraction-based mixture model to illustrate the coexistent state of multiphase flows (upper diagram, see Section 4.2). Unlike traditional approaches which use a unified mixture field to compute motion for all phases, we employ a phase-specific interphase momentum mechanism (lower-left diagram, Section 4.4.1) to allow individual phase motion computation. We derive each momentum source for every fluid phase implicitly taking into account both no-slip and free-slip conditions (lower-right diagram, Secs. 4.4.2 and 4.4.3, respectively).

derive the interphase momentum source directly from Equation (4.6) and Equation (4.8), which can overcome all above difficulties without the need for LEA.

To investigate how interphase momentum affects phase-phase interactions, we consider two extreme analysis scenarios.

No-drift scenario. In this scenario, all phases are tightly coupled, indicating strong interphase momentum. This prevents the mixture from separating, resulting in $\frac{D}{Dt}\mathbf{u}_{Vk} = 0$ and $\mathbf{s}_k^d = 0$ being always true. We denote next the interphase momentum as $\mathbf{s}_k^{In}$ and the drift momentum as $\mathbf{s}_k^{d_0}$. Following Equation (4.34), the interphase momentum is given by

$$\begin{aligned}\mathbf{s}_k^{In} &= \frac{\alpha_k \rho_k}{\rho_m}\mathbf{s}_m^{\{p,v,g,d_0,a\}} - \mathbf{s}_k^{\{p,v,g,a\}} \\ &= \alpha_k(\rho_k - \rho_m)\left(\frac{D}{Dt}\mathbf{u}_m - \mathbf{g}\right) + \left(\alpha_k \mathbf{s}_m^{\{p,v,a\}} - \mathbf{s}_k^{\{p,v,a\}}\right).\end{aligned} \tag{4.35}$$

The no-drift interphase momentum can be interpreted as follows. The first term on the second line of Equation (4.35) is the offset created to reduce inertia-related drift, such as sand in water separated by gravity. This term counteracts the acceleration causing sand and water to

stick together. The second term addresses the differences in momentum sources between phase-level and mixture-level, caused by pressure, viscosity, and advection.

Free-drift scenario. In this scenario, all phases move independently without influencing each other, so the interphase momentum is always zero. The free-drift momentum source can be derived from Equation (4.34) as

$$\begin{aligned}\mathbf{s}_k^{d_f} &= \mathbf{s}_k^{\{p,v,g,a\}} - \frac{\alpha_k \rho_k}{\rho_m}\mathbf{s}_m^{\{p,v,g,d_f,a\}} \\ &= \alpha_k\left(\rho_m - \rho_k\right)\left(\frac{D}{Dt}\mathbf{u}_m - \mathbf{g}\right) - \left(\alpha_k \mathbf{s}_m^{\{p,v,a\}} - \mathbf{s}_k^{\{p,v,a\}}\right).\end{aligned} \tag{4.36}$$

Analytic interphase momentum expression. An interesting relation between the no-drift and free-drift scenarios emerges from Equations (4.35) and (4.36) as $\mathbf{s}_k^{d_f} = -\mathbf{s}_k^{I_n}$. This says that the stronger the interphase momentum source, the harder it is to generate drift velocity. Based on this observation, we further assume that $\mathbf{s}_k^I = -\mathbf{s}_k^d$, which perfectly satisfies Equation (4.34). We next analytically express the interphase momentum source using Equation (4.35) as

$$\mathbf{s}_k^I = C_d \mathbf{s}_k^{I_n}, \tag{4.37}$$

where $C_d \in [0, 1]$ models the drift amount: $C_d = 0$ yields free drift and $C_d = 1$ yields no drift. Similar forms of interphase momentum sources have been earlier used to compute the change rate of drift velocity [156, 239]. Yet, while crucial for producing a divergence-free mixture velocity field, a concise expression without *any* approximation during derivation has been absent. In contrast, we provide phase-level operations that show how drift conditions can be altered using an interphase momentum source without violating mixture-level incompressibility.

4.4.2 *Phase-mixture Momentum Mapping*

We next compute each phase's motion changes by each momentum source via Equation (4.37). Four main momentum sources influence $\mathbf{s}_k^I$ according to Equation (4.35), while $\frac{D}{Dt}\mathbf{u}_m$ can be affected by all momentum sources (see Equation (4.33)).

When computing $\mathbf{s}_k^I$, we ignore advection and drift momentum sources. This not only eliminates the computational demand for $\nabla \mathbf{u}_k^a$ and the time derivative of $\mathbf{u}_{Vk}$ but also crucially allows Equation (4.33) to be treated as a single-phase fluid. This strategy enables the use of an implicit pressure solver [29] throughout the simulation, facilitating larger time steps compared to the explicit solver-based method [155], and yielding a speedup of roughly 2–3 times in total. We next show

how viscosity and pressure momentum sources influence each phase's motion.

Since viscosity affects each phase, the change rate of phase velocity generated by viscosity can be derived from Equation (4.32) as

$$\begin{aligned}\frac{D^v}{Dt}\mathbf{u}_k &= \frac{1}{\alpha_k\rho_k}\left(\mathbf{s}_k^v + \mathbf{s}_k^{I_v}\right)\\ &= \frac{C_d\alpha_k\left(\rho_k-\rho_m\right)\frac{D^v}{Dt}\mathbf{u}_m + C_d\alpha_k\mathbf{s}_m^v + \left(1-C_d\right)\mathbf{s}_k^v}{\alpha_k\rho_k}\\ &= C_d\frac{\mathbf{s}_m^v}{\rho_m} + \left(1-C_d\right)\frac{\mathbf{s}_k^v}{\alpha_k\rho_k},\end{aligned} \tag{4.38}$$

where $\mathbf{s}_k^{I_v} = C_d\alpha_k\left(\rho_k-\rho_m\right)\frac{D^v}{Dt}\mathbf{u}_m + C_d\alpha_k\mathbf{s}_m^v$ is the viscosity-related part of the interphase momentum source according to Equation (4.37), and $\frac{D^v}{Dt}\mathbf{u}_m = \mathbf{s}_m^v/\rho_m$ is the viscosity-induced change rate of mixture velocity from Equation (4.33).

Similarly, the change rate of phase velocity produced by pressure is

$$\frac{D^p}{Dt}\mathbf{u}_k = C_d\frac{\mathbf{s}_m^p}{\rho_m} + \left(1-C_d\right)\frac{\mathbf{s}_k^p}{\alpha_k\rho_k} = \frac{\mathbf{s}_m^p}{\rho_m}\left(C_d + \left(1-C_d\right)\frac{\rho_m}{\rho_k}\right), \tag{4.39}$$

where $\mathbf{s}_k^p = -\alpha_k\nabla p_k$. Again, we follow $p_k = p_m$ for all phases [196]. Equation (4.39) is our proposed phase-mixture momentum mapping mechanism which calculates the movement of phase flow using mixture momentum sources.

Previous mixture models [156, 238, 239] focus on the solvable momentum equation, such as Equation (9) in Ren *et al.* [239], considering interphase momentum as a consequence. This creates a *causal* relation between momentum, mixture velocity change rate, and drift velocity (Equations (4.37) and (4.35)). The change rate of mixture velocity, $\frac{D}{Dt}\mathbf{u}_m$, must occur first in a time step to 'generate' the interphase momentum source that next alters drift velocity. This approach violates the relation between phase velocity and mixture velocity in Equation (4.2), making it impossible to obtain a stable mixture-level fluid description without LEA. Although Jiang *et al.* [155] proposed to compute mixture-level fluid motion from phase-level to abandon LEA, the introduced artificial momentum source term could result in weak compressibility for the mixture flow; also, the volume fraction of fluid particles needs to be re-normalized at each time step.

In contrast, our method *simultaneously* solves the velocity change rate for all fluid phases and the mixture flow. According to Equation (4.39), the phase velocity change rate can be implicitly expressed through the pressure component of the mixture momentum source. The term $-\frac{1}{\rho_m}\nabla p_m$ is solvable using any standard fluid simulation approach. This substitution in Equations (4.38) and (4.39) unifies phase, drift, and mixture velocities, ensuring natural mass conservation and consistent volume fraction summation for all particles.

4.4.3 *Mixture Viscosity Model*

The standard mixture model does not specify how phases interact with each other. Since the mixture model uses the LEA, viscosity is assumed to bind phases together, reverting to the mixture velocity. Jiang *et al.* [155] designed their artificial interphase momentum source with this idea in mind. In contrast, we modify the viscous momentum source from a single-phase viscosity computation procedure to a viscosity computation between the phase and the mixture. For the entire computational domain Ω, this is given by

$$\tilde{\mathbf{s}}_k^v(\mathbf{x}) = \alpha_k(\mathbf{x})\,\mu_k \nabla_{\mathbf{x}}^2 \left(\mathbf{u}_m + \int_\Omega \mathbf{u}_{Vk}(\mathbf{x})\,\delta(\mathbf{x})\,d\mathbf{x} \right), \tag{4.40}$$

where μ is the dynamic viscosity coefficient and δ denotes the Dirac delta function. This derivation is consistent with the standard mixture model: a single fluid phase interacts equally with all other phases. Moreover, this form conserves momentum for each fluid parcel while maintaining $\sum_k \tilde{\mathbf{s}}_k^v = \mathbf{s}_m^v$ if the dynamic viscosities of all phases are the same.

4.4.4 *Implementation*

We integrate our implicit mixture model (Section 4.4) into the proposed VFSPH as detailed in Algorithm 4.2. Following implicit SPH, we divide a simulation time-step into two stages: advection and projection, as follows.

Advection. During advection, gravity and viscosity forces are initially used to alter the velocity of each fluid particle, yielding a compressible fluid field. Our advection for each phase (Algorithm 4.2, step C) computes our refined viscosity momentum source (Equation (4.40)) using an artificial SPH Laplacian approximation (Equation (3.8)) as

$$\tilde{\mathbf{s}}_k^v(\mathbf{x}_i) = 2(d+2)\alpha_{k_i}\mu_k \sum_j V_j^0 \frac{\left(\mathbf{u}_{k_i} - \mathbf{u}_{m_j}\right)\cdot(\mathbf{x}_i - \mathbf{x}_j)}{\|\mathbf{x}_i - \mathbf{x}_j\|^2} \nabla W_{ij}, \tag{4.41}$$

where d denotes the dimension ($d = 2$ or $d = 3$) of the scenario.

Projection. During projection, pressure forces are implicitly derived to restore the fluid to an incompressible state. Following VFSPH, we do two projection steps, before and after advection (Algorithm 4.2, steps B and D). Step B uses pressure forces to make the velocity field divergence-free. We denote the induced mixture-level momentum source and change rate of phase velocity as $\sum_k \mathbf{s}_k^{p_{\text{div}}}$ and $\frac{D^{p_{\text{div}}}}{Dt}\mathbf{u}_k$, respectively. Step D predicts the position of the fluid particle at the next time step to make the fluid incompressible. We denote the momentum and

Algorithm 4.2: Mixture model with VFSPH

A. Preparation

1. Calculate mixture velocity $\mathbf{u}_m$ (4.2)
2. Calculate drift velocity $\mathbf{u}_{Vk}$ (4.3)
3. Update particle rest density ρ_m (4.5)

B. VFSPH divergence-free solver (Inherit Step B of Algo. 4.1)

4. Calculate $\mathbf{s}_m^{p_{\text{div}}}$ using VFSPH (4.29)
5. Update phase velocity (from $\mathbf{s}_k^p$)
   $\mathbf{u}_k$ += $\Delta t \frac{D^{p_{\text{div}}}}{Dt}\mathbf{u}_k$ (4.39)
6. Repeat Step A.1 to update $\mathbf{u}_m$

C. Advection

7. Update phase velocity (from $\mathbf{s}_k^v$)
   $\mathbf{u}_k$+=$\Delta t \frac{D^v}{Dt}\mathbf{u}_k$ with $\mathbf{s}_k^v$ as $\tilde{\mathbf{s}}_k^v$ (4.38), (4.40)
8. Update phase velocity (from $\mathbf{s}_k^g$) $\mathbf{u}_k$+=$\Delta t\mathbf{g}$
9. Repeat Step A.1 to update $\mathbf{u}_m$

D. VFSPH incompressible solver (Inherit Step D of Algo. 4.1)

10. Calculate $\mathbf{s}_m^{p_{\text{inc}}}$ using VFSPH
11. Update phase velocity (from $\mathbf{s}_k^p$)
    $\mathbf{u}_k$ += $\Delta t \frac{D^{p_{\text{inc}}}}{Dt}\mathbf{u}_k$ (4.39)
12. Repeat Steps A.1 and A.2 to update $\mathbf{u}_m$ and $\mathbf{u}_{Vk}$
13. Update particle position $\mathbf{x}$ += $\Delta t\mathbf{u}_m$

E. Phase transfer

14. Update volume fraction
    α_k += $\Delta t \frac{D}{Dt}\alpha_k + \Delta t\nabla^2 (D_m\alpha_k)$ (4.7), (3.6)

change rate of phase velocity in step D as $\sum_k \mathbf{s}_k^{p_{\text{inc}}}$ and $\frac{D^{p_{\text{inc}}}}{Dt}\mathbf{u}_k$, respectively.

Phase Transfer. Two factors cause changes in the particles' volume fraction (Algorithm 4.2, step E). The first one is drift velocity. We use the SPH approximation [156] to compute Equation (4.7) as

$$\frac{D\alpha_{k_i}}{Dt} = -\sum_j V_j^0 \left(\alpha_{k_i}\mathbf{u}_{Vk_i} + \alpha_{k_j}\mathbf{u}_{Vk_j}\right)\cdot\nabla W_{ij}. \tag{4.42}$$

The second factor is the diffusion term. Similar to the previous divergence-free mixture model [156], we use a diffusion coefficient D_m to control the diffusion effect as

$$\nabla^2\left(D_m\alpha_{k_i}\right) = D_m\sum_j V_j^0\left(\alpha_{k_i}-\alpha_{k_j}\right)\frac{(\mathbf{x}_i-\mathbf{x}_j)\cdot\nabla W_{ij}}{\|\mathbf{x}_i-\mathbf{x}_j\|^2+0.01h^2}. \tag{4.43}$$

4.5 RESULTS

We evaluate our method under different scenarios both analytically and visually (see Table 4.2) and compare it with a state-of-the-art dynamic mixture model [155]. We implemented our physics simulation framework using Taichi [130] and rendered results with Blender's Cycles engine [34]. We ran our experiments on an NVIDIA 3090 Tensor Core GPU for both simulation and rendering. Our implementation is available as open source for replicability [324].

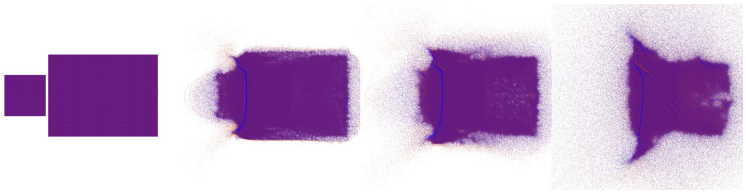
Table 4.2: Experiment configurations.

Scene	**Figure(s)**	**Particle count**	**Particle size** $[m]$	**Drag coefficient**	**Diffusion coefficient**
Tea	Fig. 4.1	499K	0.03	0.5	0.05
Collide	Fig. 4.5	474K	0.0055	C_d: 0, 0.3, 0.7, 1 k_d: 0, 3, 7, 10	0
Hourglass	Fig. 4.10	350K	0.03	C_d: 0.29; k_d: 5	0.05
Rotate	Fig. 4.6	473K	0.0075	C_d: 0, 0.3, 0.7, 1 k_d: 0, 3, 7, 10	0
Ink drop	Fig. 4.9	509K	0.05	C_d: 1, 0.6; k_d: 10	0
Phase separation	Fig. 4.8, 4.7	49K	0.045	C_d: 0.61, 0.29, 0 k_d: 10, 5, 2.53	0
Cocktail 2	Fig. 4.13	1.09M	0.05	0.94	0.6
Propeller	Fig. 4.11	555K	0.035	C_d: 0.95	0.5
Cocktail 1	Fig. 4.12	1.09M	0.05	0.6	0.1

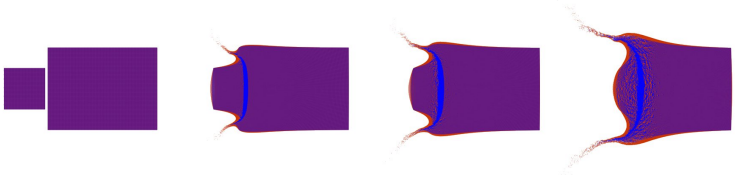
4.5.1 *Performance Analysis*

4.5.1.1 *Momentum conservation*

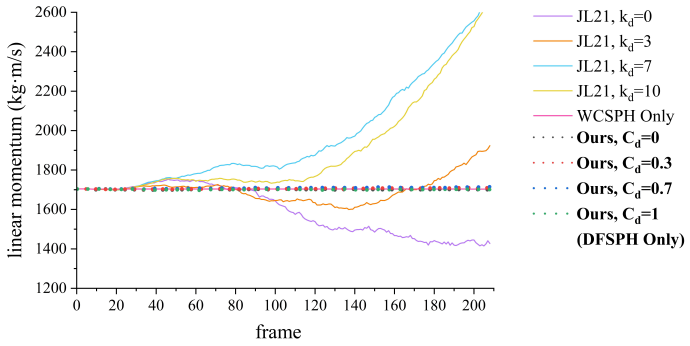
We execute two experiments to evaluate the conservation of linear and angular momentum. In each case, fluid particles are initialized with an equal mixture of two fluid phases, depicted in red and blue, resulting in a purple mixed color. These particles maintained a density ratio of $1:2$ (see Figs. 4.5 and 4.6). We exclude any external momentum sources, *e.g.*, gravity. Our method is tested using drift coefficients $C_d = 0.0$ (free-drift), 0.3, 0.7, and 1.0 (no drift, VFSPH-only). We also assess the mixture model proposed by Jiang *et al.* using drift coefficients $k_d = 0.0$, 3.0, 7.0, and 10.0. Although [155] does not mention free-drift, we use $k_d = 0.0$ to denote the most pronounced effects, which taper off with increasing values, in tandem with WCSPH-only. We exclude viscosity in these tests, barring two sets in the Collide experiment.



(a) Jiang *et al.* [155] with $k_d = 10$. Frame (from left to right) 1, 40, 60, 100



(b) Our method with $C_d = 0.0$. Frame (from left to right) 1, 40, 60, 100



(c) Comparison of linear momentum in the two blocks collision experiment between our method and the dynamics mixture model using various C_d and k_d coefficient values.

Figure 4.5: A collision experiment shows the pressure-induced phase separation abilities of Jiang *et al.*'s method [155] (a) and our method (b). Our approach exhibits superior stability, especially in the absence of gravity and viscosity, by consistently maintaining linear momentum over time (c).

Collision. The first experiment aims to test the conservation of linear momentum. A smaller fluid cube, initially having a velocity of $1m/s$ moving to the right, is set to collide with a larger, stationary fluid cube (Figure 4.5). As observed during the collision, the denser blue phase begins to separate from the contact surface, a result of its heightened inertia. In contrast, the lighter red phase, with its heightened sensitivity to the collision dynamics, quickly congregates in the spherical region of the fluid cubes. Over time, the phase separation between the red and blue fluids becomes increasingly distinct.

The chart in Figure 4.5(c) shows that, in the absence of a multiphase solver (negating drift effects), both VFSPH and WCSPH adeptly conserve momentum. When our implicit mixture model is activated at $C_d < 1$, the phase separation intensifies, yet linear momentum remains largely unaltered over time. In contrast, Jiang *et al.*'s model introduces momentum discrepancies, as most trials using this approach exhibited deviations from the benchmark, particularly over extended simulation duration.

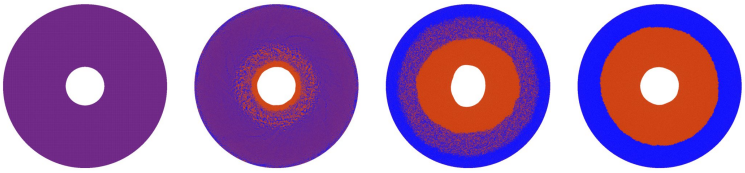
Rotation. Our second experiment focused on the conservation of angular momentum and is set in a round pool containing a swirling fluid ring (Figure 4.6). The fluid ring initially displays an irrotational counterclockwise velocity field. Here, each particle's velocity is tangential to the center, with its magnitude being inversely proportional to its distance to the center. As the experiment progresses, the centrifugal effect manifests itself. The denser blue fluid phase gradually settled in the peripheral regions, while the center became dominated by the pure red phase.

Figure 4.6(c) shows that VFSPH and our method both experience similar angular momentum reduction over time, reflecting the simulation's inherent dissipative nature without external energy. WCSPH dissipates faster than VFSPH and could destabilize without viscosity. The dynamic mixture model of Jiang *et al.* [155] intensifies this, hindering optimal conservation. In our model, as C_d nears zero, angular momentum attenuation slightly surpasses the VFSPH-only case. We believe this is due to updating the volume fraction at step E of Algorithm 4.2, causing minor errors during phase exchanges among particles, especially with notable phase separation. However, this does not adversely impact visuals or stability.

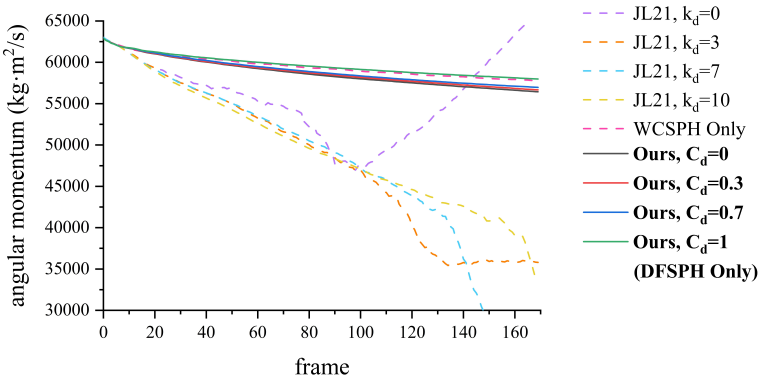
Table 4.3: Time Statistics for the Collide Experiment

Method	**Procedure**	**Time per time step [s]**	**Time per second of animation [s]**
Jiang *et al.*	WCSPH	1.41×10^{-3}	51.41
	Neighbor search	2.18×10^{-3}	79.12
	Multiphase	0.249×10^{-3}	905.33
Our method	VFSPH	0.175×10^{-3}	63.65
	Neighbor search	2.67×10^{-3}	9.70
	Multiphase	0.298×10^{-3}	108.51

(a) Jiang *et al.*'s method [155] with $k_d = 10$. Frame (from left to right) 1, 10, 20, 40



(b) Our method with $C_d = 0.0$. Frame (from left to right) 1, 40, 850, 1800



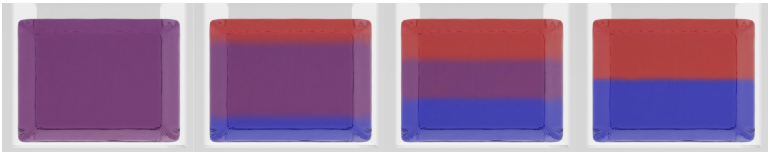
(c) Comparison of angular momentum in the fluid rotation experiment between our method and the dynamics mixture model using various C_d and k_d coefficient values.

Figure 4.6: Under the centrifugal effect, the rotated fluid separates into two distinct phases. Jiang *et al.*'s method [155] struggles to maintain stability without the constraints of gravity and viscosity, leading to unreasonable phase velocities (a). Conversely, our method (b) ensures a stable conservation of angular momentum throughout the simulation (c).

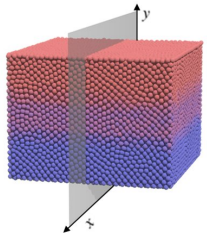
4.5.1.2 *Simulation efficiency*

We assessed the efficiency of our method against the existing approach using performance statistics from the collide experiment in Figure 4.5. For this experiment, time consumption was measured from two vantage points: The average time consumed per segment of a single simulation loop (referred to as the "time step"), and the average time needed to generate 1 second of simulation animation (which involves summing the time cost of each time step up to a second). This distinction is pivotal,

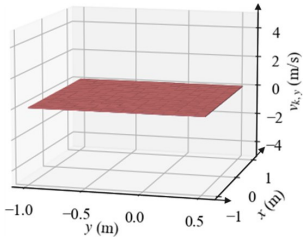
given that VFSPH, an implicit fluid solver, permits a considerably larger time step than the explicit WCSPH solver. As a result, VFSPH requires significantly fewer simulation loops than WCSPH to produce an equivalent duration of simulation. Notably, although each VFSPH loop might individually take longer than a WCSPH loop, fewer loops overall lead to time savings. Table 4.3 shows the average time taken to generate 1 loop/second of animation for our method and Jiang *et al.*'s model. Our approach consistently required significantly less time. This efficiency arises because VFSPH supports a considerably larger time step per simulation loop compared to WCSPH. With a particle size of $0.0055m$, our method's maximum time step is approximately $2.75 \times 10^{-4}s$ and about $2.75 \times 10^{-5}s$ for WCSPH. Our multiphase solver requires a marginal 2% more time per loop than Jiang *et al.*'s model but our substantial speedup due to the time step difference makes this extra cost negligible.



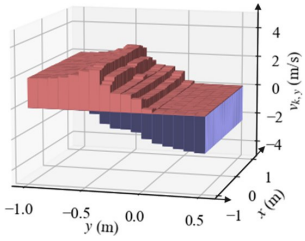
(a) Experiment setup



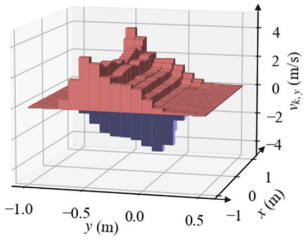
(b) A cross-section



(c) Our method with LEA

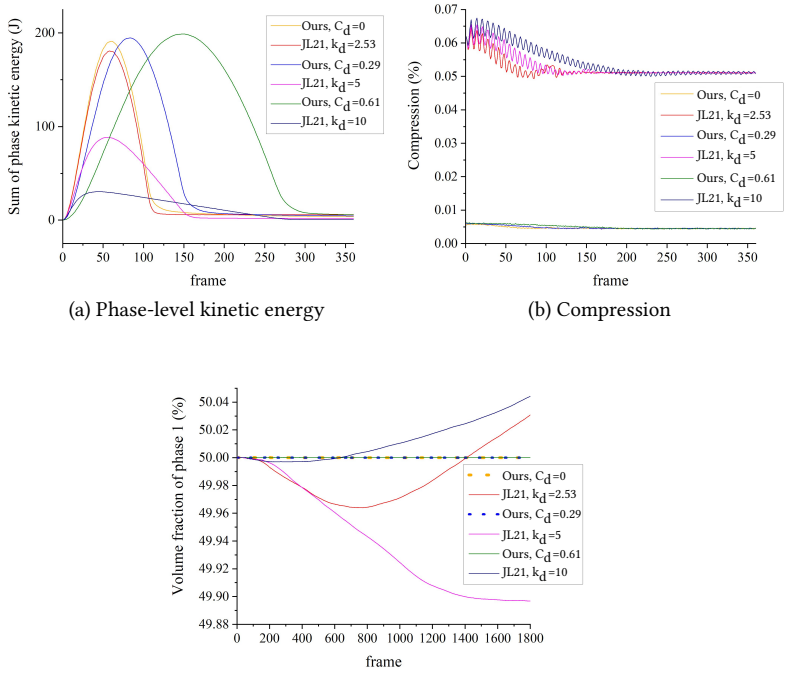


(d) Dynamic mixture model [155]



(e) Our method

Figure 4.7: Our two-phase separation experiment. (a) shows the experiment setup, (b) shows a visualization of phase velocity (c)–(e) using different methods.



(a) Phase-level kinetic energy

(b) Compression

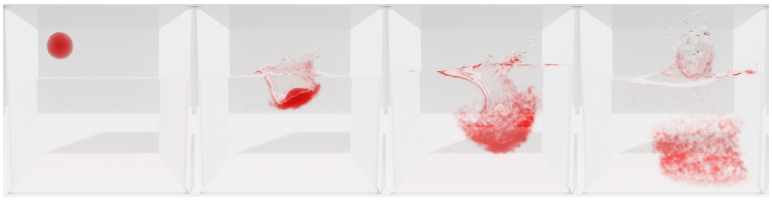
(c) Volume fraction

Figure 4.8: Comparison of phase kinetic energy, compression, and volume fraction in the two-phase separation experiment (Figure 4.7) between our method and the dynamic mixture model of Jiang *et al.* [155] using various C_d and k_d coefficient values.

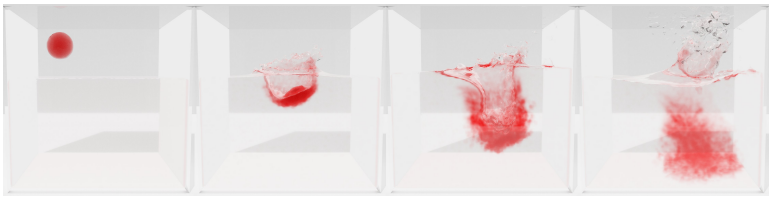
4.5.1.3 *Mass conservation*

Two phase separation. In this experiment, a fluid bulk with two thoroughly mixed phases is left to separate under gravity (Figure 4.7(a)). The volume fraction of the two phases is $1:1$ and the density ratio is $1:2$ (red : blue). This experiment tests the numerical performance of our method, the implicit mixture model [155], and ours with LEA for two-phase separation under gravity. The experiment also maps the relation between the drag k_d and C_d.

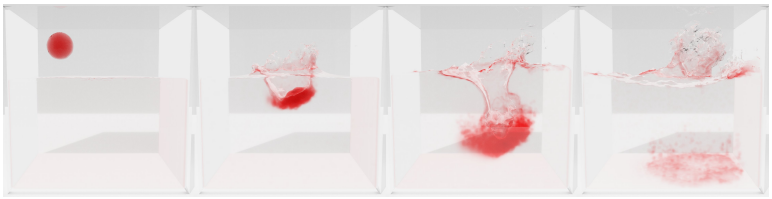
All three methods are capable of simulating this scenario. Yet, our method with LEA separates very slowly. Only a thin separation layer is noticeable after 30 seconds, suggesting that the mixture model sub-optimally handles this scenario. Our method and Jiang *et al.*'s method separate at similar speeds, with $C_d = 0, k_d = 2.53$ being the quickest; followed by $C_d = 0.29, k_d = 5$; and $C_d = 0.61, k_d = 10$ being the slowest. We compare these three coefficient sets because $k_d = 10$ is recommended



(a) Our model with $C_d = 1$



(b) Our model with $C_d = 0.6$



(c) Jiang *et al.* [155] with $k_d = 10$

Figure 4.9: Red ink drop falls into water along a parabolic trajectory.

in the dynamic mixture model [155] and $C_d = 0$ models free drift in our method.

Figure 4.7(c)–(e) shows the phase velocity on the y-axis ($v_{k,y}$) with $C_d = 0, k_d = 2.53$ under the status of Figure 4.7(b). The method proposed by Jiang *et al.* retains some phase velocity at the far ends of the y-axis, where the two phases are separated and no phase velocity should occur. Our method avoids this issue and has the *largest* phase velocity near the separated and mixed regions, where the gradient of volume fraction is the greatest.

Figure 4.8 shows the variation of the incompressible state of the mixture (Figure 4.8(a)), volume fraction of each phase (Figure 4.8(b)), and phase-level kinetic energy measured through the phase velocity (Figure 4.8(c)) over time. The dynamic mixture model [155] conserves neither incompressibility nor mass of each phase as rigorously as our method since they use a weakly-compressible pressure solver and regularize the volume fraction at each time step. Moreover, the kinetic energy of our method always manages to reach the same peak regardless of the speed of the separation process. This means the transformation

of energy from gravitational potential to kinetic is unconditionally conserved, which is not the case for the method of Jiang *et al.* [155].

4.5.2 *Comparisons of Effects*

Ink drop. A drop of red ink (70% ink phase, 30% water phase) follows a parabolic trajectory into the water (Figure 4.9). The diffusion coefficient is set to 0. Results show that our method's drag coefficient can produce more substantial and consistent turbulence effects due to phase interactions. In contrast, in the dynamic mixture model [155], the drag force reduces this effect.

Hourglass. A blue phase (10 times heavier than the transparent one) flows through a narrow neck under the influence of both diffusion and interphase momenta. As Figure 4.10 shows, our method better captures the convection vortex. Also, the momentum does not dissipate during convection, as shown by the lively bubbles in the third column of Figure 4.10.

(a) Dynamic mixture model [155] with $K_d = 5$

(b) Our method with $C_d = 0.29$

Figure 4.10: The blue phase in the hourglass is convected with the transparent one causing vortexes and bubbling effects.

4.5.3 *Effectiveness under Complex Scenarios*

Propeller. We simulate the foam generated by a propeller spinning underwater (Figure 4.11). The foam phase has half the density of the water. Moreover, a strong interphase momentum is applied to make the

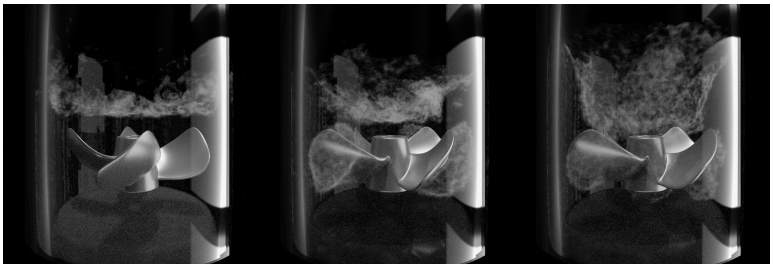


Figure 4.11: Propeller spinning in water. Foams are formed, especially at higher propeller speeds.

two phases challenging to separate. As Figure 4.11 shows, the foam becomes more robust and more agitated as the propeller spins faster. The centrifugal effect can also be seen with the foam being dragged by the blades, demonstrating our method's versatility.

Figure 4.12: A fully mixed cocktail with phase separation due to the lack of interphase momentum.

Phase separation. Five phases of a cocktail with a density ratio of $1:2:4:8:16$ are initially evenly mixed in a glass (Figure 4.12). Then, the drag coefficient is applied to ease the separation of the five phases. The significant density ratio makes the separation quite dramatic. Our method can handle this process with ease.

Drag dominant. Building upon the previous settings, we show in Figure 4.13 an extra strong coupling force and diffusion effect between phases. Upon stirring, the mixture starts to separate as the significant density variation creates a high drag force causing the phases to separate from one another (see also the accompanying video for detailed differences between phase separation and drag-dominant experiments).

Diffusion dominant. In contrast to stirring-induced separation, Figure 4.1 presents an experiment of brewing tea using a tea bag. The

Figure 4.13: A tightly coupled cocktail gets separated after a stir.

tea diffusion into water is obtained by assigning volume fractions to the solid and enabling diffusion between solid and liquid. After stirring with a rod, tea and water become evenly mixed. The density ratio is small enough (1.05) to make the diffusion effect more dominant than the separation.

4.6 DISCUSSION AND CONCLUSION

4.6.1 *Summary*

This work introduces an implicit mixture model for incompressible fluid simulation that seamlessly integrates phase-level and mixture-level descriptions through a combination of interphase momentum sourcing and a phase-momentum mapping mechanism, coupled with a mixture viscosity model. This approach ensures both mass conservation for individual phases and overall volume conservation.

The dynamic of phase transfer introduces additional momentum transfer between fluid parcels, yielding visually appealing results under significant drift velocities. In the future, we plan to explore this phenomenon and refine its integration within our model, aiming for more precise and visually accurate simulations.

4.6.2 *Limitations*

Phase transfer. While the proposed method builds on phase transfer mechanisms from previous works [155, 156, 239], shown in Equation (4.42), it effectively ensures mass conservation across all fluid phases. However, momentum conservation across time steps is not guaranteed.

Our implicit mixture model maintains momentum conservation at each time step by balancing the momentum sources between individual phases and the mixture field as per Equations (4.32) and (4.33). The

experimental setup of colliding fluid cubes (Figure 4.5) demonstrates this feature effectively.

Nevertheless, during phase transfers, as a fluid phase k from particle i with velocity $\mathbf{u}_{k_i}$ merges into particle j with a different phase velocity $\mathbf{u}_{k_j}$, the transferred momentum from particle i adapts to $\mathbf{u}_{k_j}$, leading to a loss of the original momentum. This becomes pronounced with significant drift velocities or larger time steps.

Addressing this via additional momentum computations during phase transfer has introduced stability challenges to the SPH implicit simulation solver in our tests, particularly under high drift velocities. Future work will focus on refining the solver design to fully conserve momentum.

Inclusion of diffusion. Our model currently employs a classical Laplacian diffusion approach, applied directly to the volume fraction of each phase at each fluid particle, independent of drift velocity. This method, typical for simulations with Linear Elasticity Approximation constraints, complicates integration with our implicit mixture model and is treated as a post-processing step.

This limitation hampers our model's ability to effectively represent complex multiphase interaction effects, such as solubility and insolubility, which are characterized by distinct phase separation and surface tension-driven interfacial curvatures in nature.

Future developments will aim to incorporate advanced simulation models for computing momentum sources to enhance the portrayal of diverse interaction effects.

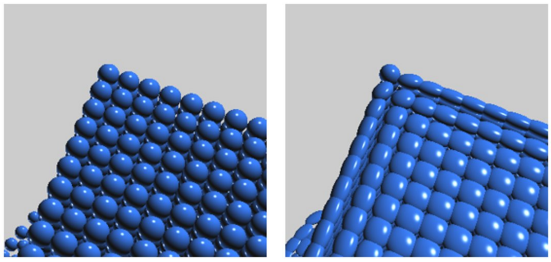
5

ANISOTROPIC SCREEN SPACE RENDERING FOR PARTICLE-BASED FLUID SIMULATION

As we discussed in Section 2.7, the challenge of balancing efficiency and realism in particle-based fluid simulation is significant, especially in scenarios demanding real-time rendering. Traditional explicit mesh tracing methods, while capable of depicting precise fluid motion, demand extensive computational resources, making them impractical for real-time applications in large, complex environments. Conversely, screen space rendering offers enhanced efficiency by bypassing the need for mesh construction and utilizing particle data directly for fluid rendering. However, this approach often results in visual artifacts, such as glitching edges and bumpy surfaces. To address these shortcomings, we introduce a novel real-time fluid rendering technique that builds on the screen space rendering framework for particle-based simulations. Our method implements anisotropic transformations on point sprites, stretching them along the direction of feature vectors to achieve smoother fluid surfaces. This process leverages a weighted principal component analysis of the particle distribution to guide the transformations. Additionally, we enhance the resulting visual fidelity by integrating our transformed point sprites with advanced screen space filters, including curvature flow and narrow-range depth processing. Our experimental results demonstrate that this innovative approach not only effectively mitigates the issues of jagged edges and surface unevenness prevalent in previous methods but also retains sharp, high-frequency details. This balance of efficiency and visual quality makes it a promising solution for real-time fluid rendering in complex simulations[1].

This chapter begins with an overview of screen space rendering, discussing its advantages and limitations in Section 5.1. We then detail the pipeline of screen space rendering and the application of smoothing filters in Section 5.2. Our contribution, which involves the use of an anisotropic smoothing kernel in conjunction with smoothing filters, is detailed in Section 5.3. The results of our approach are demonstrated in Section 5.4, and the chapter concludes with a summary of our findings in Section 5.5.

1 Parts of this chapter were published in: Y. Xu, Y. Xu, Y. Xiong, D. Yin, X. Ban, X. Wang, J. Chang, and J. J. Zhang. Anisotropic screen space rendering for particle-based fluid simulation. *Computers & Graphics*, Volume 110, 118-124. Elsevier, 2023. doi: 10.1016/j.cag.2022.12.007



(a) isotropic (b) anisotropic

Figure 5.1: Comparison of isotropic and anisotropic fluid particles.

5.1 INTRODUCTION

Though *particle-based* approaches, such as SPH, are particularly favored for their algorithmic efficiency and flexible application scenarios [140], a major challenge in particle-based simulations lies in the visualization of simulation results. Specifically, tracing surfaces for particle-represented fluids often becomes a computational bottleneck, making efficient and high-fidelity visualization a crucial area of research.

Traditionally, to visualize the results of Lagrangian simulations, fluid surfaces are first extracted as polygon meshes using the marching cubes algorithm [189, 302, 336]. These meshes are subsequently rendered to represent the fluid visually. However, the major drawback of this two-step process is that mesh generation requires considerable computational resources and can be exceedingly time-consuming, especially when aiming for detailed results. Although some researchers have refined the marching cubes algorithm to facilitate real-time mapping with a minimal number of particles, the quality of the output often remains below expectations [303].

Instead of the traditional method of tracing particle surfaces and constructing polygon meshes, the *screen space rendering* approach simplifies the process by generating surfaces based on the camera's perspective, using depth and thickness data. In this method, particles are treated as *point sprites*—2D textures that face the camera [145], typically circular. Edges between these textures are seamlessly blended to create the appearance of a continuous fluid surface on screen [205]. This technique bypasses the resource-intensive mesh construction process, enabling real-time fluid simulations suitable for gaming and cutting-edge virtual reality applications [112].

Müller *et al.* [205] pioneered the application of this method for fluid rendering, where the choice of the smoothing filter plays a crucial role in the quality of the rendering. As depicted in Figure 5.1, the raw depth information derived from fluid particles is often jagged and requires smoothing to achieve a realistic fluid surface. While a simple Gaussian

filter can smooth surfaces, it tends to introduce unwanted blurring effects and is generally suboptimal [205]. The bilateral Gaussian filter, although it preserves sharp boundaries, can excessively flatten areas of discontinuity and is not separable, leading to potential visual artifacts when approximations are made [112, 211]. Truong and Yuksek [291] developed a narrow-range filtering technique that enhances surface quality by selectively smoothing using a limited depth range, effectively maintaining boundary clarity near discontinuities. Recently, Oliveira and Paiva [218] addressed the issue of particle deficiency identified by Truong and Yuksek [291] through a particle classification mechanism, further refining this approach. Additionally, Liu *et al.* [184] introduced a cost-effective, differentiable screen-space rendering algorithm that is particularly well suited for augmented reality applications, highlighting the ongoing evolution and application of these techniques.

However, despite its advantages, several inherent artifacts of screen space rendering limit its broader application. Common issues include uneven surfaces derived from particles, which can appear excessively blurred, obscuring clear distinctions between foreground and background elements. Inspired by the techniques proposed by Yu and Turk [338], we propose an anisotropic transformation scheme for point textures. This transformation is applied prior to filtering the depth image, facilitating the creation of smooth surfaces in areas of discontinuity while simultaneously preserving detailed, vivid textures.

5.2 REAL-TIME SCREEN SPACE FLUID RENDERING

Fluid rendering in screen space circumvents the traditional requirement of generating surface meshes by directly projecting 3D fluid dynamics onto a 2D screen, a process intimately connected with image processing techniques [112]. These steps involved in this rendering approach are illustrated in the schematic diagram presented in Figure 5.2.

To perform screen space rendering, two attributes must be determined for each pixel on the screen: depth and thickness. The *depth image* captures the distance from the camera to the closest fluid particle at each pixel, providing crucial information about the fluid's surface geometry. Conversely, the *thickness image* records the cumulative depth of fluid along the line of sight, reflecting the volumetric density of the fluid encountered by the light ray. This dual-image approach enables a comprehensive depiction of fluid dynamics directly in screen space.

Due to the discrete nature of particle-based simulations, the depth image often contains noise and discontinuities, which can lead to artifacts in the rendered image. To mitigate these issues, *smoothing filters*, such as a Gaussian filter [175], are applied. These filters can effectively smooth the depth data, reducing noise and improving the visual quality of the fluid surface. It operates by averaging the pixel values within a localized kernel, weighted by a Gaussian distribution, thus preserving im-

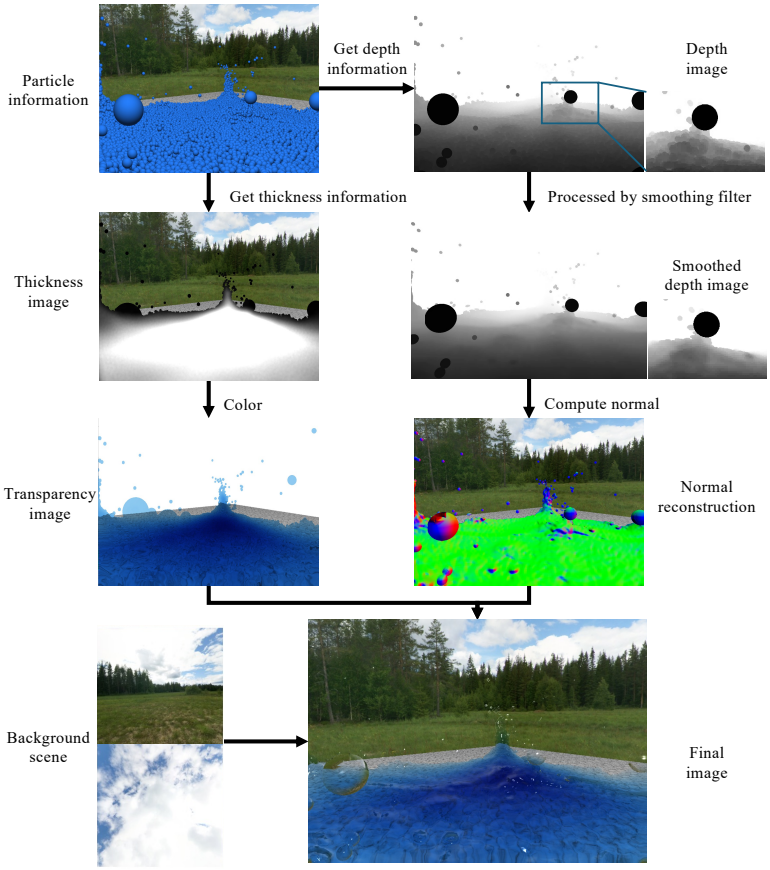


Figure 5.2: Schematic diagram of our screen space rendering pipeline for fluid rendering: First, transform fluid particles to screen space to capture depth and thickness information. Second, post-process the depth image to reconstruct surface normals and render the fluid surface using the combined data of normals and thickness.

portant edges while smoothing out irregularities. Once the depth image has been smoothed, the next step involves reconstructing the normals to the fluid surface. Surface normals are vectors that are perpendicular to the tangent plane at each point on the surface, and they are crucial for lighting calculations such as in the Blinn–Phong reflection model [35].

With the smoothed depth and reconstructed normals, the rendering engine can simulate realistic lighting effects based on the properties of the fluid material. This approach, integrating advanced image processing techniques into fluid rendering, results in a highly realistic depiction of fluid dynamics directly in screen space. Below we detail the rendering pipeline.

5.2.1 *OpenGL Transformation Pipeline*

In this section, we discuss the rendering pipeline using OpenGL-specific terminology, focusing on how fluid particle positions are transformed through various coordinate spaces. In the context of fluid simulations, the position of each particle is defined in *object space*. Each position vector is represented as a homogeneous vector $\mathbf{x}_o = [x_o.x, x_o.y, x_o.z, x_o.w]^T$. The fourth component $x_o.w$ is set to 1 to facilitate transformations involving translation.

The transformation of each particle's position to *camera space* is achieved through the combined Model-View matrix $\mathbf{M}$. This matrix integrates both modeling and viewing transformations:

$$\mathbf{x}_{\text{cam}} = \mathbf{M} \cdot \mathbf{x}_o. \tag{5.1}$$

Here, the Model-View matrix $\mathbf{M}$ accounts for both the position and orientation of the camera relative to the scene's geometry, setting the stage for a viewpoint-centric rendering.

Subsequent to camera space transformation, the coordinates are projected into *clip space* via the projection matrix $\mathbf{P}$, which is responsible for perspective distortion typical in 3D rendering:

$$\mathbf{x}_c = \mathbf{P} \cdot \mathbf{x}_{\text{cam}}. \tag{5.2}$$

In clip space, coordinates remain in a homogeneous form where the actual position and depth of the particle are encoded relative to the viewing frustum. Following this, vertices undergo a perspective division by the w-coordinate, transforming them into normalized device coordinates (NDC). This division scales the x, y, and z components by $1/w$, normalizing them within the bi-unit cube. Below we treat all positions in clip-space as in NDC unless otherwise specified.

Finally, the NDC coordinates are mapped to screen space, which corresponds to the actual pixel locations on the rendering viewport. This transformation is critical for the precise rendering of visuals on the screen.

During screen space rendering, it is crucial to determine the clip space coordinates for each pixel, denoted by $\mathbf{y}_c = [y_c.x, y_c.y, y_c.z, y_c.w]^T$, to distinguish them from the particle positions $\mathbf{x}$. These pixel positions are processed through the fragment shader within OpenGL. The accurate handling of clip space coordinates $\mathbf{y}$ is essential, as they govern how fragments are transformed into screen space, directly influencing the visual outcome of the fluid simulation.

Vertex shader process. In OpenGL, the vertex shader first allocates a point sprite by receiving the vertex coordinate $\mathbf{x}_o$, which acts as the center of the point sprite. The particle's size in object space, r_o, is also

passed to the vertex shader to set the size of the point sprite. To display the point sprite on the screen, the vertex shader transforms $\mathbf{x}_o$ and r_o into clip space. The NDC process is automatically handled by OpenGL, resulting in $\mathbf{x}_c$ and r_c. A square point sprite of size r_c is then initialized. The fragment shader further processes $\mathbf{y}_c$ for each pixel within the allocated point sprite, ensuring that the rendering accurately reflects the fluid particle's appearance and dynamics as per the simulation data.

Fragment shader process. In the fragment shader stage of OpenGL, each pixel within a point sprite, centered at $\mathbf{x}_c$, is processed. The screen space coordinates of each pixel, accessed via the `gl_FragCoord` variable, are used to reconstruct the clip space position $\mathbf{y}_c$. The fragment shader calculates the Euclidean distance between the center of the point sprite $\mathbf{x}_c$ and the pixel's position $\mathbf{y}_c$. If this distance $\|\mathbf{y}_c - \mathbf{x}_c\|$ exceeds the radius $r_c/2$, with

$$\|\mathbf{y}_c - \mathbf{x}_c\| = \left((y_c.x - x_c.x)^2 + (y_c.y - x_c.y)^2 + (y_c.z - x_c.z)^2\right)^{1/2}, \quad (5.3)$$

the pixel is discarded. Note that the w component is not involved in the $\|\cdot\|$ operation in this work. This ensures that the sprite appears circular rather than square by trimming off pixels outside the circular boundary. This process is essential for maintaining the intended visual characteristics of the particles, ensuring that they appear as rounded entities, which is critical for the realistic representation of fluid dynamics in the simulation.

5.2.2 *Screen Space Rendering*

With the above-mentioned OpenGL pipeline for handling point sprites, we now detail its integration into the computation of the depth image, the smoothing process, the thickness image, as well as the surface shading.

Depth image. To obtain the depth image, it is crucial to compute the depth value $y_c.z$ in normalized device coordinates (NDC) for each pixel. This value quantifies the distance from each particle to the camera, viewed from the camera's perspective [65]. Importantly, the depth information for each particle is initially known as $x_c.z$, and each particle is modeled as a sphere with diameter r_c. Thus, the actual depth for each pixel can be calculated based on these parameters.

In scenarios where particles overlap from the camera's view, it is essential to render only the closest, non-occluded particles for each pixel. OpenGL effectively resolves this occlusion challenge using *z-buffer* techniques. By employing these techniques, OpenGL can selectively render each pixel based on the calculated depth $y_c.z$ in the fragment shader, ensuring that occluded particles are not visualized. This

capability is critical for preserving the visual integrity of depth perception in the rendered image.

Smoothing depth image. The use of depth images where each particle is treated as a sphere can often lead to an uneven appearance of the fluid surface. To mitigate this rugged effect, various filters are applied to the depth image to smooth out the abrupt particle contours by averaging the $y_{c.z}$ values across local pixel groups. A commonly used smoothing filter is the Gaussian filter, which considers the influence of neighboring pixels using the Gaussian function $g(\mathbf{y}_{c_i}, \mathbf{y}_{c_j})$ as

$$y^G_{c_i}.z = \frac{\sum_j g(\mathbf{y}_{c_i}, \mathbf{y}_{c_j}) y_{c_j}.z}{\sum_j g(\mathbf{y}_{c_i}, \mathbf{y}_{c_j})}, \tag{5.4}$$

where j represents neighboring pixels to i. The Gaussian function $g(\mathbf{y}_{c_i}, \mathbf{y}_{c_j})$ is defined as

$$g(\mathbf{y}_{c_i}, \mathbf{y}_{c_j}) = \frac{\zeta_i}{\pi} e^{-\| y_{c_i}.xy - y_{c_j}.xy \|^2 \zeta_i}, \tag{5.5}$$

where ζ_i denotes the precision of the distribution, influenced by $y_{c_i}.z$ [175, 291]. While Gaussian convolution effectively smooths particle depth data, it can also lead to over-smoothing, which may blur critical discontinuities. For example, particles that are close in 2D screen space but have significant depth differences should maintain sharp boundaries. However, Gaussian convolution fails to detect these boundaries and tends to smooth them out. Consequently, variants such as the bilateral Gaussian filter are frequently adopted to balance detail preservation with smoothing.

Thickness image. To get the thickness of the fluid being penetrated from the direction of each pixel, we can reuse the process of deriving the depth image by disabling the z-buffer techniques and accumulating thickness r_c in fragment shader. To achieve this, we need to set the blend function as `glBlendFunc(GL_ONE, GL_ONE)` for OpenGL, indicating adding the source value from fragment shader and the destination value that is already in the frame buffer together for each pixel.

Normal reconstruction and surface shading. Once the depth image is derived and smoothed, the next crucial step involves reconstructing the surface normals from the depth image. These normals are essential for accurately simulating light interactions such as reflection and refraction on the fluid surface. In the fragment shader, normals are computed by first linearizing the depth values retrieved from the depth texture. These linearized values are then converted into camera space coordinates. To estimate gradients, differences in positions between adjacent pixels are computed. Normals are subsequently derived by taking

the cross product of these gradients. The resulting normal vector is then normalized and validated to ensure it does not contain invalid values, with a fallback to a predefined normal if necessary.

After reconstructing the normals, the shading of the fluid surface is addressed by calculating refraction effects as described in the work of Conde [63], and employing Blinn-Phong illumination techniques [35] to realistically model light interaction with the fluid surface. The Blinn-Phong model is particularly effective in enhancing visual richness by simulating the specular highlights and diffuse reflections typical of fluid appearances. Additionally, the transmittance of the fluid, which significantly affects how light passes through and emerges from the fluid, is determined using the Beer-Lambert law [276]. These steps not only enhance the visual quality but also contribute significantly to the realism of the rendered scene by allowing the simulation to mimic the dynamic interplay of light with the fluid's surface.

5.3 ANISOTROPIC TRANSFORMATION OF POINT SPRITES FOR FLUID PARTICLES

In Lagrangian fluid simulation, achieving a finer-grained spatial discretization often involves increasing the simulation resolution by using smaller particles. This approach tends to produce smoother edges when rendering the fluid. However, increasing the resolution for 3D fluid simulations can severely impact performance, especially in real-time applications. To strike a balance between efficiency and quality, we employ a technique based on the *Weighted Principal Components Analysis (WPCA)* of SPH.

Our method involves calculating anisotropic kernel functions derived from the distribution of fluid particles. By using the eigenvalues and eigenvectors obtained from the WPCA, we can deform the fluid particles in an anisotropic manner. This deformation is guided by the magnitudes of the eigenvalues along their corresponding eigenvectors, allowing for efficient smoothing while preserving crucial details of the fluid's dynamic properties. The transformative effects of this anisotropic adjustment on the particles are visually represented and can be seen in the screen space rendering comparisons illustrated in Figure 5.1.

This anisotropic transformation method enhances the visual quality of fluid simulations by more accurately modeling the fluid's behavior and interactions within its environment, leading to more realistic and visually appealing results. By adjusting particle shapes based on their spatial distribution, our approach significantly reduces the need for high-resolution meshes while still capturing the essential dynamics and surface characteristics of the fluid.

5.3.1 *Tracing Surface Using Smoothing Kernels*

For explicit surface reconstruction using the marching cubes algorithm, a scalar field $\phi(\mathbf{x})$ is initially determined across the simulation domain. This field is computed using the smoothing kernel W defined in Equation (3.9) as

$$\phi(\mathbf{x}) = \sum_j V_j W(\|\mathbf{x} - \mathbf{x}_j\|, h). \tag{5.6}$$

The computed color field $\phi(\mathbf{x})$ typically takes values in the range $[0, 1]$. For surface reconstruction, the marching cubes algorithm is then applied to this field with a chosen positive threshold value.

However, as W is isotropic, the reconstructed surface often exhibits spherical behavior, which may not accurately capture smooth surfaces and sharp features. To mitigate this limitation, Yu and Turk [338] suggest employing an anisotropic transformation to the smoothing kernel. This approach involves replacing the isotropic support radius h with an anisotropic transformation matrix $\mathbf{G}$, as

$$W(\mathbf{r}, \mathbf{G}) = \lambda \det(\mathbf{G}) P(\|\mathbf{r}\mathbf{G}\|). \tag{5.7}$$

The anisotropic form of the smoothing kernel retains the same functional form as the cubic spline kernel used in Equation (3.9). When the transformation matrix $\mathbf{G}$ is set to $\frac{1}{h}\mathbf{I}_d$, where $\mathbf{I}_d$ is the identity matrix of dimension d, the kernel behaves isotropically. This capability allows the kernel to adapt effectively to local particle distribution variations, enhancing surface reconstruction fidelity, particularly in regions with sharp geometrical changes.

The anisotropy matrix $\mathbf{G}$ modifies the kernel by incorporating the directional characteristics of $\mathbf{r}$, the vector between particles. This adjustment improves the simulation's capacity to capture detailed flow dynamics and directional features within the fluid. As illustrated in Figure 5.1(a), under standard isotropic conditions, particles exhibit spherical behavior. In contrast, Figure 5.1(b) demonstrates how anisotropic processing refines particle shapes, leading to more elongated forms and smoother transitions at the fluid's edges. These visual comparisons highlight the significant benefits of employing an anisotropic kernel, particularly in enhancing edge definition and overall surface smoothness in fluid simulations.

5.3.2 *Deriving the Anisotropy Matrix*

To enhance the smoothing kernel's adaptability to the fluid dynamics, particularly to extend the influence of the kernel along the tangential direction more than the normal direction to the fluid surface, WPCA is utilized. This process involves calculating eigenvalues and eigenvectors by

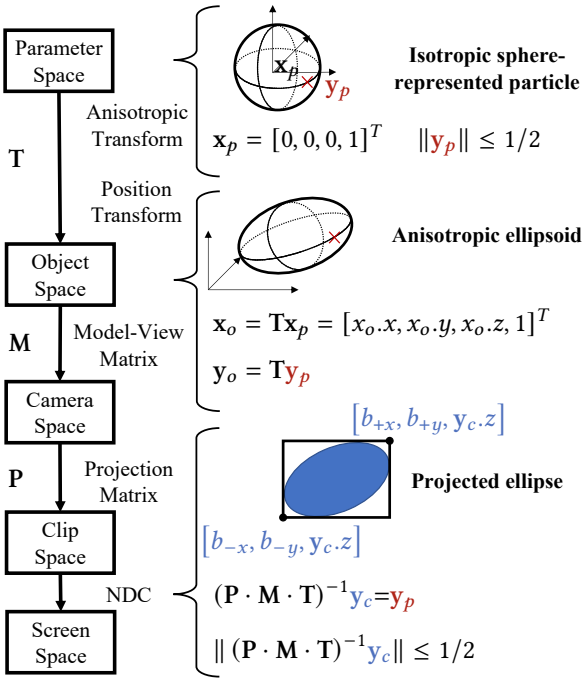


Figure 5.3: The pipeline of rasterizing fluid particles onto screen space with anisotropic transformation. We introduce an additional parameter space into the rendering pipeline, where particles are initiated as unit spheres centered at the coordinate origin. The transformation matrix **T**, constructed based on the anisotropic transformation matrix **G**, converts particles into object space. In NDC, pixel coordinates are inverted back to parameter space to determine whether they fall inside the projected ellipses.

performing eigendecomposition on the covariance matrix, which is formulated based on the positions of the particles. This matrix effectively captures the spatial distribution and orientation of particles within the fluid.

Initially, a weight value ω_{ij} is assigned to each pair of fluid particles. Using these weights, a weighted covariance matrix is constructed, followed by performing Singular Value Decomposition (SVD) on this matrix. The eigenvectors and eigenvalues derived from the SVD are then used to construct the anisotropy matrix **G**. The weighted average position of the fluid particles is given by

$$\bar{\mathbf{x}}_i = \sum_j \omega_{ij}\mathbf{x}_j, \tag{5.8}$$

where ω_{ij} represents the weight assigned to the neighbor particle j relative to particle i. In our work, ω_{ij} is calculated using

$$\omega_{ij} = \frac{W\left(\|\mathbf{x}_i - \mathbf{x}_j\|, h\right)}{\sum_j W\left(\|\mathbf{x}_i - \mathbf{x}_j\|, h\right)}. \tag{5.9}$$

This formulation ensures that a greater normalized weight is assigned to closer particles. The displacement of $\bar{\mathbf{x}}_i$ from $\mathbf{x}_i$ can indicate the distribution trend of the fluid surface. Based on this trend, a covariance matrix $\mathbf{C}_i$ for the fluid particle i is constructed as

$$\mathbf{C}_i = \sum_j \omega_{ij}(\mathbf{x}_j - \bar{\mathbf{x}}_i)(\mathbf{x}_j - \bar{\mathbf{x}}_i)^T. \tag{5.10}$$

An SVD of the covariance matrix $\mathbf{C}_i$ is employed to obtain the decomposition $\mathbf{C}_i = \mathbf{W}\Sigma\mathbf{W}^T$, where $\Sigma = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d)$ is the diagonal matrix containing the eigenvalues. These eigenvalues are ordered from the largest to the smallest, representing the principal directions and scales of the data spread.

To ensure stability in the anisotropic deformation and prevent extreme elongations or compressions, a limit is imposed on the minimum eigenvalue relative to the largest eigenvalue. Specifically, eigenvalues smaller than λ_1/k are adjusted upwards, resetting to λ_1/k and denoted as λ^r. This adjustment ensures that the ratio between the largest and smallest eigenvalues does not exceed a predefined threshold, maintaining the structural integrity of the deformation. In our experiments, this threshold, k, is set to 4.

Moreover, if the number of neighboring particles is less than or equal to 20, no anisotropic kernel function is applied. This decision is based on the reliability of statistical measures in small samples. Additionally, a scale control factor s is introduced to adjust the magnitude of the covariance matrix such that $s\det(\mathbf{C}) \approx 1$, preventing significant changes in particle volume. In our experiments, we use $s = 1400$. The regularized covariance matrix, accommodating these adjustments, is represented as

$$\tilde{\mathbf{C}} = s\mathbf{W}\Sigma^r\mathbf{W}^T, \tag{5.11}$$

where Σ^r is the diagonal matrix of the adjusted eigenvalues. The final transformation matrix $\mathbf{G}$ is the inverse of $\tilde{\mathbf{C}}$ regularized by h as

$$\mathbf{G} = \frac{\tilde{\mathbf{C}}^{-1}}{h}. \tag{5.12}$$

For implementation in screen space rendering, the denominator in Equation (5.12) is removed to be incorporated with the coordinate transformation.

5.3.3 *Transforming Sphere Particles onto the Screen Space*

As depicted in Figure 5.3, a critical implementation step in our work involves the anisotropic transformation of spherical particles into ellipsoids. This transformation presents unique challenges for rendering point sprites as ellipses in screen space, as discussed in Section 5.2. The primary challenge is determining which pixels within the allocated square area of each point sprite fall inside the ellipses. Unlike spherical particles where inclusion within a radius $r_c/2$ can be easily checked with $\|\mathbf{x}_c - \mathbf{y}_c\| \leq r_c/2$, ellipsoids require a more complex approach due to their anisotropic projection onto the screen, which typically results in elliptical shapes rather than circles. While theoretically, ellipsoids under perspective projection can produce parabolic or hyperbolic shapes, these occurrences are exceedingly rare in our settings. Therefore, our discussion will focus exclusively on elliptical projections.

The first issue to resolve is the precise identification of pixels that lie within these ellipses. Since the ellipsoids are transformed based on their orientation and scale in 3D space, their 2D projections—ellipses—do not conform to simple radial checks. Instead, the inclusion test must account for the orientation and aspect ratio of the ellipse, necessitating a transformation of the coordinate check into the space defined by the ellipse's major and minor axes.

A second challenge involves initializing the point sprites with appropriate sizes to ensure all pixels representing the ellipse are captured without incorporating excessive redundant pixels. Directly using r_c as the point sprite size may not suffice, as it does not account for the elongation and orientation of the ellipse in screen space. A possible solution involves calculating the bounding rectangle of the ellipse in screen space, ensuring the size of the point sprite encompasses this rectangle. This approach guarantees that the sprite is large enough to cover the ellipse but not excessively large to waste computational resources.

To address the challenges associated with rendering ellipsoids in screen space, we integrate techniques proposed by Sigg *et al.* [260] that utilize an additional *parameter space* prior to object space transformation, as depicted in Figure 5.3. In this parameter space, each fluid particle is centered at the coordinate origin $\mathbf{x}_p = [0, 0, 0, 1]^T$ with a standardized size $r_p = 1$. Therefore, for pixels to be considered inside the spherical boundary in the fragment shader, they must satisfy the condition $\|\mathbf{y}_p\| \leq 1/2$.

To transition coordinates from parameter space to object space, a transformation matrix $\mathbf{T}$ is utilized:

$$\mathbf{T} = \begin{bmatrix} \begin{bmatrix} & & \\ & r_o\mathbf{G} & \\ & & \end{bmatrix} & \begin{matrix} x_o.x \\ x_o.y \\ x_o.z \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}. \tag{5.13}$$

In this matrix, $\mathbf{G}$, the anisotropic transformation matrix, modifies the spheres in parameter space into ellipsoids, reflecting their orientation and scale. The scalar r_o adjusts the particle size, while $\mathbf{x}_o$ positions the particle appropriately within object space. Notably, when the anisotropic transformation is not applied, $\mathbf{G}$ defaults to $\mathbf{I}$, the identity matrix, which is consistent with the isotropic transformation process previously described.

Size of the point sprite. To accurately determine the dimensions of the projected ellipse for an anisotropic point sprite in screen space, we compute the extreme positions $[b_{-x}, b_{-y}]$ and $[b_{+x}, b_{+y}]$. This is achieved by inversely transforming the clip space positions back to the parameter space. For a point $\mathbf{y}_c$ within the point sprite, the inclusion condition is defined by the plane equation $\mathbf{n}_c^T \mathbf{y}_c \leq 0$, where $\mathbf{n}_c = [1, 0, 0, b_x]^T$ represents the boundary plane in clip space.

To trace the boundary values b_x back to the parameter space, we apply the transformation:

$$\mathbf{n}_p = (\mathbf{P} \cdot \mathbf{M} \cdot \mathbf{T})^T \mathbf{n}_c. \tag{5.14}$$

Since all particles are standardized as a unit sphere in parameter space, the spherical quadratic function for any vector $\mathbf{n}_p$ must satisfy:

$$n_p.x^2 + n_p.y^2 + n_p.z^2 = n_p.w^2, \tag{5.15}$$

which is summarized in the bilinear form:

$$\mathbf{n}_p^T \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{bmatrix} \mathbf{n}_p = 0. \tag{5.16}$$

Using these relationships, the dimensions $[b_{-x}, b_{-y}]$ and $[b_{+x}, b_{+y}]$ of the ellipse can be derived from Equations (5.14) and (5.15).

Determine ellipse in fragment shader. To ascertain whether each pixel resides within the projected ellipse in clip space, we first transform the position of each pixel $\mathbf{y}_c$ back to the parameter space. This is achieved through the inverse transformation of the coordinate system transformations applied to the original point sprites as

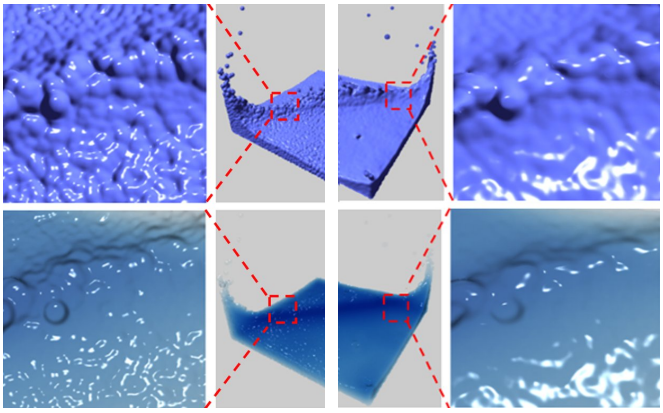
$$\mathbf{y}_p = (\mathbf{P} \cdot \mathbf{M} \cdot \mathbf{T})^{-1} \mathbf{y}_c. \tag{5.17}$$

In the parameter space, the problem of determining whether the pixel lies within the ellipse simplifies to checking whether it lies inside a unit sphere. This conversion effectively reduces the complex geometrical problem in clip space to a simpler geometric test in parameter

space, where the inclusion within a sphere is determined by the condition $\|\mathbf{y}_p\| \leq 1/2$. If this condition is satisfied, the pixel is considered to be inside the projected ellipse, ensuring accurate rendering of the anisotropic particle.

5.4 RESULTS

The experiments below are performed using an AMD Ryzen 7 5800H @3.20 GHz, 32GB memory, and NVIDIA RTX 3060. The 3D graphics API OpenGL is used for particle rendering, and C++ is used for the coding of all additional techniques. In addition, GLSL is used to calculate the fluid optical effects on the GPU. The frame rate exceeds 25 frames per second (FPS) in all tests when the particle count is below 500k.



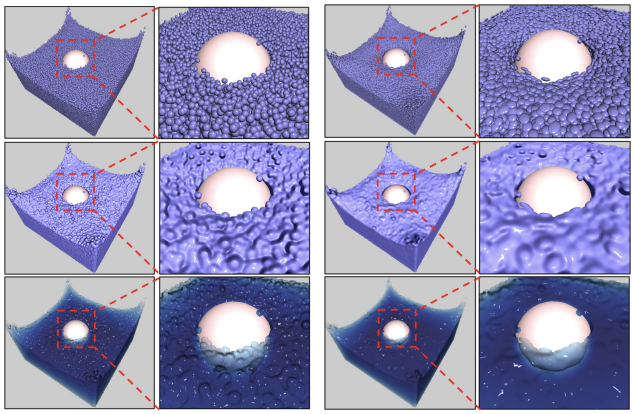
(a) isotropic (b) anisotropic

Figure 5.4: Experimental comparison of isotropy and anisotropy in the dam break scenario.

5.4.1 *Anisotropic Processing Results*

In this subsection, we adopt an anisotropic algorithm to process fluid particles to reduce the roughness of the fluid depth map, which can improve the final rendering effect of the fluid surface. As shown in Figure 5.4, (a) and (b) respectively represent the surface rendering results when two water blocks collide under isotropic and anisotropic conditions. It can be observed that the overall surface rendering results processed by the anisotropic algorithm are smoother, and the illumination shading is more realistic with fine highlights.

In addition, experimental verification is conducted for a fluid-solid coupling scenario, as shown in Figure 5.5. In this scenario, real-time fluid rendering based on the anisotropic algorithm shows better surface



(a) isotropic (b) anisotropic

Figure 5.5: Experimental comparison of isotropy and anisotropy in the fluid-soild coupling scenario.

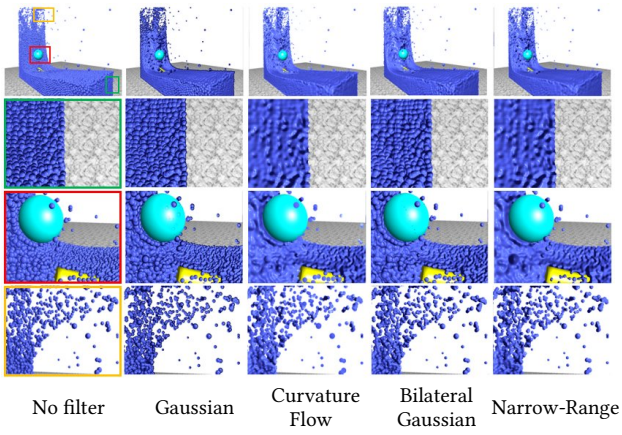
results with smoother surfaces, especially at the interface between the rigid body and liquid.

5.4.2 *Combination with Popular Smoothing Filters*

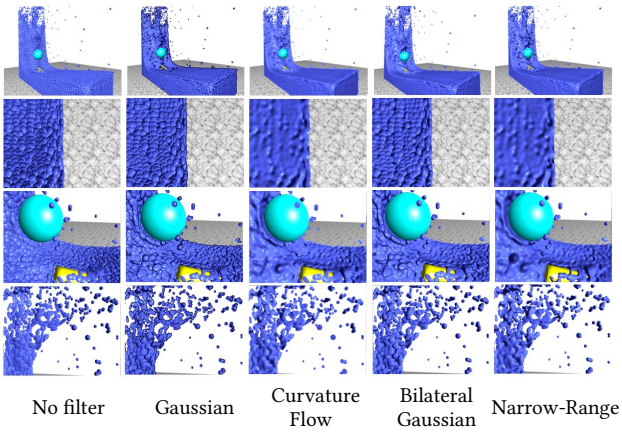
Here we combine our anisotropic algorithm with various popular smoothing filters to verify the effectiveness and practicability of the proposed scheme under actual application scenarios. The Gaussian filter, the bilateral Gaussian filter, and the narrow-range filter use the same number of iterations ($iter = 2$). The curvature flow filter uses more iterations ($iter = 80$) to obtain flat surface results at the cost of performance loss.

Figure 5.6 shows a dam-break fluid colliding with multiple differently shaped solid objects. Complex boundary geometries can be observed in this case. We conducted this experiment using various kinds of smoothing filters considering both isotropic and anisotropic conditions. Figure 5.6a and Figure 5.6b show the surfaces constructed using isotropic and anisotropic point sprites respectively. It can be seen that the proposed anisotropic transformation scheme can enhance surface performance with almost every state-of-the-art smoothing filter. The second to the third row of Figure 5.6 exhibit how anisotropic transformation reduces the unevenness of particle distribution on the free surface and coupling boundaries. The fourth row shows that our method can also help to gather the sparsely distributed fluid particles representing splashes into more well-organized structures.

Figure 5.7 presents an experiment comparing the anisotropic surface generation effects of fluid under different filtering methods in a com-



(a) isotropic point sprites representation



(b) anisotropic point sprites representation

Figure 5.6: Experimental comparison of isotropy and anisotropy in the dam break scenario. Part (a) of this figure demonstrates the surface constructed without anisotropic transformation; part (b) of this figure shows how anisotropic transformation works with and without different smoothing filters.

plex fluid-rigid coupling boundary scenario. This comparison focuses on three main aspects: preservation of details on the wave crest (second row), smoothness of the fluid surface (third row), and the visual effects on thin films with sharp edges (fourth row). To evaluate these filters, we conducted a detailed analysis focusing on critical areas of fluid dynamics that are challenging to render accurately. The wave crest preservation is essential for capturing the dynamic behavior and realism of fluid surfaces, especially in high-energy interactions. Smoothness of the fluid surface ensures a visually appealing and physically accurate represen-

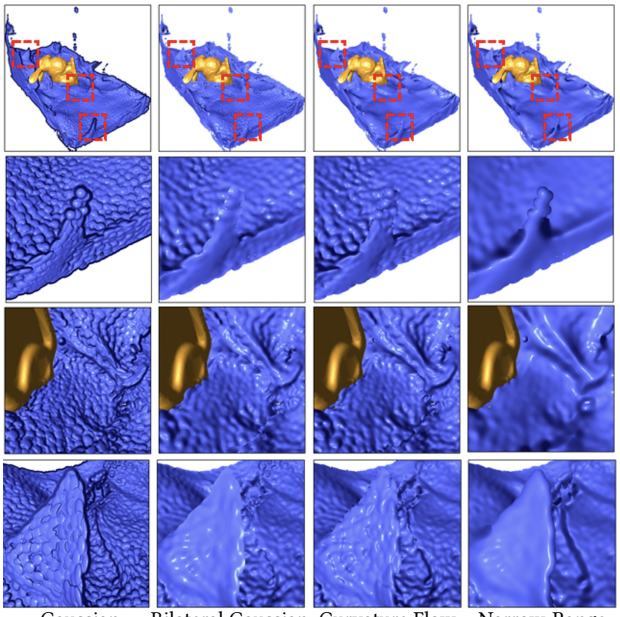


Figure 5.7: From left to right are the Gaussian filter, the bilateral Gaussian filter, the curvature flow filter, and the narrow-range filter.

tation of fluid flow, avoiding artifacts that could detract from realism. The appearance of thin films with sharp edges is crucial in scenarios involving surface tension effects and small-scale fluid details, which are often difficult to reproduce accurately in simulations.

The Gaussian filter, despite its computational efficiency, results in significant surface noise, deviating from the desired smoothness and detail preservation. The bilateral Gaussian filter addresses boundary issues in particle-deficient areas and reduces surface noise. However, it introduces an over-flattening effect on discontinuous surface details, leading to the loss of edge information between particles. The curvature flow filter, after 80 iterations, achieves good surface effects in certain regions. Nevertheless, this method is both unstable and time-consuming, making it less practical for real-time applications. On the other hand, the narrow-range filter effectively mitigates these issues, providing superior surface effects and better time efficiency, making it a more viable solution for rendering fluid surfaces with complex interactions.

Figure 5.8 shows the influence of different filtering algorithms with transparent fluids. It can be seen that compared with other algorithms, the narrow-range filter algorithm can produce a smoother fluid effect. Moreover, the overall highlight can form an obvious bright area, which is more consistent with the water highlight effect in the real world.

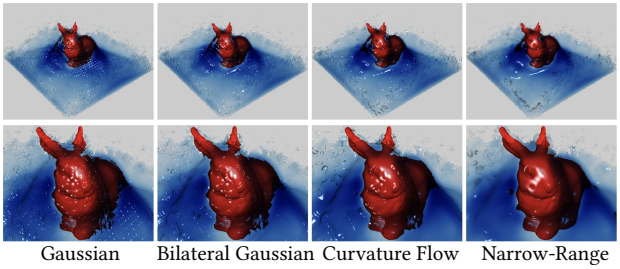


Figure 5.8: From left to right are the Gaussian filter, the bilateral Gaussian filter, the curvature flow filter, and the narrow-range filter.

5.4.3 *Performance Analysis of the Anisotropic Transformation*

To evaluate how the refined anisotropic transformation affects the efficiency of the screen space rendering, we carried out the experiment of Figure 5.4 using multiple configurations with various numbers of fluid particles, alternative smoothing filters, and the on/off status of the anisotropic transformation.

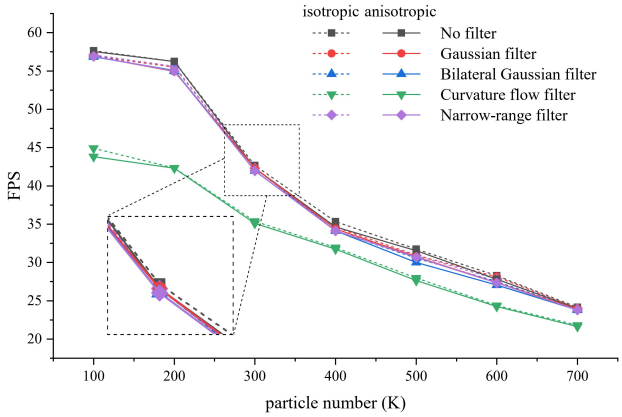


Figure 5.9: Frame rate comparison of different algorithms.

The relationship between particle number and corresponding frame rate under different methods is shown in Figure 5.9. We can see that the anisotropic transformation barely affects the efficiency of the whole screen space rendering pipeline. Although the curvature flow filter can achieve better surface effects with a higher number of iterations, its rendering time also increases. In contrast, the narrow-range filter method not only gives a better surface effect but also keeps the rendering time almost linearly increased with respect to the number of particles.

5.5 DISCUSSION AND CONCLUSION

5.5.1 *Summary*

This work introduced a novel anisotropic real-time surface rendering scheme leveraging the screen space approach, which significantly enhances the visual quality of fluid simulations. Utilizing WPCA, this method achieves an anisotropic transformation of particle point sprites, effectively resolving issues related to jagged edges and uneven surfaces typical in traditional screen space rendering. Through comparative experiments, we have identified the narrow-range filter as the most effective smoothing technique, balancing superior surface rendering with acceptable frame rates.

Looking ahead, we aim to extend this research to encompass the real-time visualization of more complex phenomena, such as multiphase fluid mixing. Our future work will explore advanced techniques for screen-space mapping of material type information to accurately represent the dynamic interactions and visual characteristics of mixed fluids. This advancement could potentially redefine the standards for real-time fluid visualization in virtual environments and interactive applications.

5.5.2 *Limitations*

Anisotropy matrix computation. Although the application of anisotropic transformations for fluid particles minimally impacts the performance of the screen-space rendering procedure, as evidenced in Figure 5.9, the computation of the anisotropy matrices is time-consuming. Achieving real-time performance necessitates precomputing and storing these matrices for each particle at each frame. This performance bottleneck stems from two primary factors. Firstly, computing the anisotropy matrix requires an additional neighbor search procedure for every frame. Secondly, the computational pipeline for the anisotropy matrix itself is inherently time-intensive.

To align this process with real-time demands of the screen space rendering pipeline, further studies on optimizing computation targets are necessary. By limiting the computation of anisotropy matrices to only those particles near the surface and coupling areas— which represent a minor fraction of the total particle count—we can significantly save on both neighbor search and anisotropy computation processes.

Multiphase rendering. The current anisotropic transformation scheme does not account for interphase coupling surfaces that emerge in multiphase fluid simulations. Treating mixed fluid flows as a homogeneous entity without considering these interactions can result in visually bumpy surfaces during transitions between fluid phases. Applying

anisotropic computations separately can create gaps at the interphase areas, disrupting visual continuity and realism.

6 CONCLUSION

This chapter concludes the thesis, which began by identifying the trilemma in fluid simulation—efficiency, accuracy, and versatility—as outlined in Chapter 1. This trilemma served as the driving force behind our studies, leading to the formulation of three research questions, each aimed at balancing these critical factors. We started by revisiting fluid simulation research through a comprehensive survey in Chapter 2, highlighting how current studies address these questions. In pursuit of these research objectives within the context of SPH, we proposed a spatially adaptive mechanism for boundary detail enhancement in Chapter 3, which significantly improves the accuracy of SPH simulations. Furthermore, we introduced an implicit mixture model for multiphase fluid simulation using a volume flux-free SPH solver in Chapter 4, enhancing the versatility of SPH applications while maintaining stability. Lastly, we focused on visualization, proposing an anisotropic screen space rendering scheme in Chapter 5 to efficiently produce high-quality visualizations of simulation results.

6.1 CONTRIBUTIONS OF THIS THESIS

6.1.1 *A Survey on Fluid Simulation*

We conducted a comprehensive survey of fluid simulation within computer graphics (Chapter 2), exploring research trends and challenges by classifying and discussing key works from the past decade. This survey serves as a valuable resource for both novices, by providing an historical overview, an introduction to the physical principles behind the simulations, and the basic discretization schemes commonly employed. It is also intended for seasoned researchers seeking a deeper and broader perspective of the field. We analyze a selection of representative studies, evaluating their methodologies and differences in approach.

However, the survey is limited to developments within the last decade, focusing primarily on the most recent and relevant topics. This temporal scope, while keeping the work concentrated on cutting-edge issues, inadvertently neglects several classic and well-established topics. Notably, an organized discussion on enhancements in various discretization schemes such as MPM, SPH, and Eulerian methods is missing. Such discussions are crucial for a thorough analysis of mathematical formulations and computational approaches in fluid simulation. These older yet foundational topics, published over ten years ago, continue to hold significant relevance in the field.

6.1.2 *An Adaptivity Mechanism with Boundary Refinement for SPH*

We introduced an innovative adaptive mechanism for SPH (Chapter 3) that enhances simulation detail near fluid-solid boundaries. This mechanism utilizes a boundary-distance-based approach, employing a signed-distance field relative to these boundaries to modulate particle resolution across varying spatial positions. The resolution is precisely calibrated to be highest near the boundary, diminishing gradually with increasing distance until it reaches a predetermined threshold. This dynamic adjustment of particle sizes is achieved through an efficient split and merge process. Additionally, we incorporated a wake flow preservation mechanism that temporarily maintains high particle resolution after an interaction with a boundary object, thereby capturing intricate flow dynamics more accurately.

However, the process of determining the appropriate timing and spatial conditions for initiating particle splitting or merging is just one aspect of the broader adaptability challenge. Further research is required to refine the split-merge mechanism and to effectively manage the coupling of particles of varying sizes. Such improvements are critical to prevent energy conservation violations and to mitigate the risk of induced instability in simulations. Furthermore, developing suitable adaptive time-stepping mechanisms tailored for spatial adaptivity represents a promising yet challenging area for future research. Techniques such as local time stepping, error-based adjustment, and predictive-corrective schemes have the potential to optimize the duration of time steps based on the dynamic demands of the simulation, thereby enhancing both accuracy and computational efficiency.

6.1.3 *An Implicitly Stable Mixture Model for Dynamic Multi-fluid Simulations*

We have developed an implicitly stable mixture model for multiphase fluid simulation (Chapter 4) utilizing an implicit SPH solver. This model innovatively calculates phase momentum sources directly from the mixture model, allowing for the derivation of explicit and continuous velocity phase fields. Crucially, it avoids the reliance on an explicit mixture field for dynamic computations and phase transfers between particles. Instead, the mixture field is obtained implicitly using a phase-mixture momentum-mapping mechanism, which ensures the conservation of key physical properties such as incompressibility, mass, and momentum. Furthermore, we introduced a mixture viscosity model that effectively manages viscous interactions between the mixture and individual fluid phases, enhancing stability under conditions of extreme inertia.

While our model operates effectively without requiring local equilibrium assumptions, it shows a propensity for instability as the drift velocity magnitude increases, necessitating the addition of interface vis-

cosity to stabilize the simulation. Additionally, our method currently does not include mechanisms for incorporating diffusion calculations, which limits its ability to simulate nuanced phase-phase interaction effects comprehensively. These areas present opportunities for further enhancement of the model's capabilities and accuracy in simulating complex multi-fluid dynamics.

6.1.4 *An Anisotropic Screen Space Rendering Scheme*

We have developed an anisotropic screen space rendering scheme that applies anisotropic transformations to point sprites in particle-based fluid simulations (Chapter 5). This technique utilizes an anisotropy matrix to stretch the sprites along the direction of feature vectors, effectively smoothing the fluid surfaces. To guide these transformations accurately, we employ a weighted principal component analysis of the particle distribution. Additionally, the visual fidelity of the simulations is further enhanced by integrating advanced screen space filters, such as curvature flow and narrow-range depth processing, with the transformed point sprites.

Although the application of this transformation technique can achieve real-time performance when combined with various smoothing filters, its computation requires considerable computational effort. The process involves extensive calculations to derive the transformation matrices for all fluid particles, which includes both neighbor search and matrix computation steps. This computational intensity can be a significant consideration in scenarios where processing speed is critical.

However, an important observation is that only particles near the fluid surface require transformation, which represents a relatively sparse subset of the total particle count. This presents an opportunity for optimization. Future research could focus on developing methods to efficiently identify these surface-near particles prior to the transformation process. By refining the selection of particles for transformation, it is possible to significantly reduce the computational load and enhance the efficiency of the method, making it more viable for scenarios where processing speed is paramount.

6.2 ADDRESSING OUR RESEARCH QUESTIONS

Below, we revisit the research questions proposed in Chapter 1 and discuss the answers based on the contributions made in this thesis.

RQ1: *How to enhance simulation accuracy without incurring prohibitive computational cost?*

Improving simulation accuracy directly for three-dimensional models typically requires a quadratic increase in computational resources, not to mention additional overheads such as neighbor searching and

cache management issues. Our survey in Chapter 2 demonstrates that enhancing simulation accuracy effectively without incurring prohibitive computational cost involves *dynamically improving simulation quality only in the most critical areas.*

Though the simulation space can be vast, in most scenarios, the crucial details and dynamic motions are present only sparsely within space and time. This observation indicates that high-resolution simulation is necessary only in select parts of the simulation domain, while the remaining areas can be adequately represented with lower resolution. To achieve this in SPH simulations, we identified three key aspects that require thorough investigation:

- *Field evaluation*: To determine whether the current resolution is appropriate for the simulated region.
- *Resolution transformation*: To transition smoothly and accurately between different particle resolutions.
- *Particle coupling*: To effectively manage the interaction between particles of different sizes.

In Chapter 3, we particularly focused on the aspect of field evaluation and proposed a boundary-optimized adaptive solution dedicated to refining the details around fluid-solid interactions. This approach ensures that computational resources are allocated efficiently, enhancing accuracy where it is most needed without overwhelming the system with unnecessary computation.

RQ2: *How to accommodate complex effects in simulations without significantly compromising accuracy and stability?*

Our survey in Chapter 2 has shown that discretization approaches in computer graphics typically aim to integrate versatile effects into a single simulation scenario efficiently, favoring general-purpose solvers over specialized ones. Thus, the key to accommodating complex effects within general solvers is to *simplify the governing equations of these effects to be compatible with widely used simulation solvers.*

In fields like computational fluid dynamics (CFD), accuracy often emerges as the paramount concern. Various optimization schemes are developed specifically to enhance accuracy for critical elements of fluid simulations. In contrast, fluid simulation in computer graphics prioritizes the realism of the output, necessitating not only accuracy but also a rich diversity of simulated elements and scenarios that resonate with everyday experiences. This requires the simulation solver to remain simple and general yet capable of accommodating the governing equations for multiple visual effects. We believe the essential strategies for achieving this in SPH include:

- *Complexity reduction*: For methods derived from CFD, it is crucial to simplify these methods to balance accuracy and visual realism effectively.
- *Mutual non-interference*: For the discretization of new governing equations, it is essential to ensure that these do not compromise the computation of the Navier-Stokes equations.
- *Computing unification*: To enhance computational efficiency, efforts should be made to consolidate shared computational needs.

In the development of our implicit multiphase mixture model in Chapter 4, we concentrated on ensuring the mutual non-interference between the mixture model and the SPH solver. This focus ensured that the established SPH solver remains unaffected by the integration of additional volume fraction schemes. As a result, our approach not only maintained stability but also eliminated the need for Local Equilibrium Approximation (LEA) to ensure stability, demonstrating that our method could sustain the necessary simulation accuracy and stability while incorporating complex effects.

RQ3: *How to refine visualization techniques for enhanced accuracy in an efficient manner?*

Visualization techniques play a pivotal role in conveying the results of simulations with both accuracy and realism. The same simulation can demonstrate varied behaviors when different visualization approaches are employed. Our survey in Chapter 2 has highlighted that while visualization techniques for fluid simulation in computer graphics have advanced significantly, special attention is still needed for specific applications like Smoothed Particle Hydrodynamics (SPH). To enhance both the accuracy and efficiency of visualizing SPH-based fluid simulations, we identified several key factors:

- *Surface smoothing*: Managing to retain both sharp edges and smooth surfaces in SPH simulations, due to their particle-based nature, poses a significant challenge.
- *Sparsity handling*: High-frequency details such as splashing and thin films, where fluid particles are sparse, require special care to prevent visual artifacts.
- *Temporal coherence*: To avoid flickering effects across frames, which do not align with the mass conservation principle of SPH particles, it is crucial to maintain temporal coherence by tracking each particle over time.

We focused specifically on surface smoothing by implementing an anisotropic transformation of fluid particles combined with an efficient

screen space rendering scheme in Chapter 5. This approach was supplemented by applying additional smoothing filters during the depth image derivation, ensuring both sharp detail and smooth transitions on fluid surfaces. This method stands out for its balance of efficiency and visual fidelity, making it a robust solution for enhancing visualization in SPH simulations.

6.3 FUTURE WORK

As we conclude this thesis, we recognize that while several advancements have been made in each contribution area, there is considerable potential in integrating these contributions to develop a cohesive, adaptive multiphase simulation system supplemented with screen space rendering techniques. Here, we outline the avenues for future research and development that could bring these components together into a unified framework.

6.3.1 *Integration of Adaptive Mechanisms and Multiphase Simulations*

A primary challenge in the integration of the adaptive mechanism with the multiphase component is improving the stability of the split-merge process. Currently, the split-merge mechanism experiences slight instabilities, which we have mitigated through a blending method that transitions particle sizes gradually over multiple time steps. Despite these efforts, the system still struggles with stability, particularly with implicit incompressible fluids. Future work will focus on developing new adaptive mechanisms that can maintain stability throughout the split and merge processes. Additionally, exploring more robust particle coupling methods that can handle drastic size ratios between particles will be crucial for advancing this integration.

6.3.2 *Enhancing Screen Space Rendering for Multiphase Fluids*

While the screen space rendering approach effectively handles particles of different sizes, rendering multiphase fluids presents unique challenges. Accurately representing colors after penetration through multiple, interactively complex fluid phases is particularly difficult, especially in real-time scenarios. This complexity is exacerbated by the non-uniform distribution of particles in space, which diverges from the regularity of grids and necessitates additional regularization processes for computation. To address this, we plan to investigate a fast screen space rendering scheme that leverages the consistency of particle movements across space and time, aiming to optimize both the efficiency and accuracy of rendering complex fluid interactions.

6.3.3 *Unified Framework Development*

The ultimate goal of our future work is to develop a unified framework that seamlessly combines adaptive simulation mechanisms with advanced rendering techniques, thereby creating a robust and efficient tool for simulating and visualizing complex fluid dynamics. This integrated system would not only push the boundaries of what is currently achievable in SPH simulations but also enhance the applicability and realism of simulated scenarios, making them more valuable for both academic research and practical applications in visual effects and scientific visualization.

BIBLIOGRAPHY

[1] Mridul Aanjaneya. An efficient solver for two-way coupling rigid bodies with incompressible flow. *Computer Graphics Forum*, 37(8):59–68, 2018.

[2] Mridul Aanjaneya, Ming Gao, Haixiang Liu, Christopher Batty, and Eftychios Sifakis. Power diagrams and sparse paged grids for high resolution adaptive liquids. *ACM Trans. Graph.*, 36(4), July 2017.

[3] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. *ACM Trans. Graph.*, 26(3):48–es, July 2007.

[4] George Biddell Airy. *Tides and waves*. B. Fellowes, 1845.

[5] Muzaffer Akbay, Nicholas Nobles, Victor Zordan, and Tamar Shinar. An extended partitioned method for conservative solid-fluid coupling. *ACM Trans. Graph.*, 37(4), July 2018.

[6] Nadir Akinci, Gizem Akinci, and Matthias Teschner. Versatile surface tension and adhesion for sph fluids. *ACM Trans. Graph.*, 32(6), November 2013.

[7] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph.*, 31(4), July 2012.

[8] Iván Alduán, Angel Tena, and Miguel A. Otaduy. Dyverso: A versatile multi-phase position-based fluids solution for vfx. *Computer Graphics Forum*, 36(8):32–44, 2017.

[9] Ryoichi Ando and Christopher Batty. A practical octree liquid simulator with adaptive surface resolution. *ACM Trans. Graph.*, 39(4), July 2020.

[10] Ryoichi Ando, Nils Thuerey, and Chris Wojtan. A stream function solver for liquid simulations. *ACM Trans. Graph.*, 34(4), July 2015.

[11] Ryoichi Ando, Nils Thurey, and Reiji Tsuruno. Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1202–1214, 2012.

[12] Ryoichi Ando, Nils Thürey, and Chris Wojtan. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph.*, 32(4), July 2013.

[13] Alexis Angelidis. Multi-scale vorticle fluids. *ACM Trans. Graph.*, 36(4), July 2017.

[14] Douglas N Arnold, Franco Brezzi, Bernardo Cockburn, and L Donatella Marini. Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM journal on numerical analysis*, 39(5):1749–1779, 2002.

[15] S. Auer and R. Westermann. A semi-lagrangian closest point method for deforming surfaces. *Computer Graphics Forum*, 32(7):207–214, 2013.

[16] Stefan Auer, Colin B Macdonald, Marc Treib, Jens Schneider, and Rüdiger Westermann. Real-time fluid effects on surfaces using the closest point method. *Computer Graphics Forum*, 31(6):1909–1923, 2012.

[17] Omri Azencot, Orestis Vantzos, and Mirela Ben-Chen. An explicit structure-preserving numerical scheme for epdiff. *Computer Graphics Forum*, 37(5):107–119, 2018.

[18] Omri Azencot, Steffen Weißmann, Maks Ovsjanikov, Max Wardetzky, and Mirela Ben-Chen. Functional fluids on surfaces. *Computer Graphics Forum*, 33(5):237–246, 2014.

[19] Vinicius C. Azevedo, Christopher Batty, and Manuel M. Oliveira. Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Trans. Graph.*, 35(4), July 2016.

[20] Vinicius C. Azevedo and Manuel M. Oliveira. Efficient smoke simulation on curvilinear grids. *Computer Graphics Forum*, 32(7):235–244, 2013.

[21] Kai Bai, Wei Li, Mathieu Desbrun, and Xiaopei Liu. Dynamic upsampling of smoke through dictionary-based learning. *ACM Trans. Graph.*, 40(1), September 2020.

[22] Kai Bai, Chunhao Wang, Mathieu Desbrun, and Xiaopei Liu. Predicting high-resolution turbulence details in space and time. *ACM Trans. Graph.*, 40(6), December 2021.

[23] Stefan Band, Christoph Gissler, Markus Ihmsen, Jens Cornelis, Andreas Peer, and Matthias Teschner. Pressure boundaries for implicit incompressible sph. *ACM Trans. Graph.*, 37(2), February 2018.

[24] Héctor Barreiro, Ignacio García-Fernández, Iván Alduán, and Miguel A. Otaduy. Conformation constraints for efficient viscoelastic fluid simulation. *ACM Trans. Graph.*, 36(6), November 2017.

[25] Christopher Batty, Andres Uribe, Basile Audoly, and Eitan Grinspun. Discrete viscous sheets. *ACM Trans. Graph.*, 31(4), July 2012.

[26] Markus Becker and Matthias Teschner. Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, page 209–217, 2007.

[27] Markus Becker, Hendrik Tessendorf, and Matthias Teschner. Direct forcing for lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):493–503, 2009.

[28] Jan Bender and Dan Koschier. Divergence-free sph for incompressible and viscous fluids. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1193–1206, 2017.

[29] Jan Bender and Dan Koschier. Divergence-free sph for incompressible and viscous fluids. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1193–1206, 2017.

[30] Jan Bender, Dan Koschier, Tassilo Kugelstadt, and Marcel Weiler. A micropolar material model for turbulent sph fluids. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, 2017.

[31] Jan Bender, Tassilo Kugelstadt, Marcel Weiler, and Dan Koschier. Implicit frictional boundary handling for sph. *IEEE Transactions on Visualization and Computer Graphics*, 26(10):2982–2993, 2020.

[32] Jean-Pierre Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, 114(2):185–200, 1994.

[33] John Biddiscombe, Jerome Soumagne, Guillaume Oger, David Guibert, and Jean-Guillaume Piccinali. Parallel computational steering for hpc applications using hdf5 files in distributed shared memory. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):852–864, 2012.

[34] Blender Foundation, Stichting Blender Foundation, Amsterdam. *Blender - a 3D modelling and rendering package*, 2018.

[35] James F. Blinn. Models of light reflection for computer synthesized pictures. volume 11, page 192–198, New York, NY, USA, July 1977. Association for Computing Machinery.

[36] Morten Bojsen-Hansen, Hao Li, and Chris Wojtan. Tracking surfaces with evolving topology. *ACM Trans. Graph.*, 31(4), July 2012.

[37] Morten Bojsen-Hansen and Chris Wojtan. Liquid surface tracking with error compensation. *ACM Trans. Graph.*, 32(4), July 2013.

[38] Morten Bojsen-Hansen and Chris Wojtan. Generalized non-reflecting boundaries for fluid re-simulation. *ACM Trans. Graph.*, 35(4), July 2016.

[39] Landon Boyd and Robert Bridson. Multiflip for energetic two-phase fluid simulation. *ACM Trans. Graph.*, 31(2), April 2012.

[40] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[41] J.U. Brackbill and H.M. Ruppel. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65(2):314–343, 1986.

[42] Christopher Brandt, Leonardo Scandolo, Elmar Eisemann, and Klaus Hildebrandt. The reduced immersed method for real-time fluid-elastic solid interaction and contact simulation. *ACM Trans. Graph.*, 38(6), November 2019.

[43] Christopher E Brennen. *Fundamentals of multiphase flow*. Cambridge University Press, 2005.

[44] Robert Bridson. *Fluid Simulation for Computer Graphics*. 2008.

[45] Oleksiy Busaryev, Tamal K. Dey, Huamin Wang, and Zhong Ren. Animating bubble interactions in a liquid foam. *ACM Trans. Graph.*, 31(4), July 2012.

[46] José A. Canabal, David Miraut, Nils Thuerey, Theodore Kim, Javier Portilla, and Miguel A. Otaduy. Dispersion kernels for water wave simulation. *ACM Trans. Graph.*, 35(6), November 2016.

[47] Yadi Cao, Yunuo Chen, Minchen Li, Yin Yang, Xinxin Zhang, Mridul Aanjaneya, and Chenfanfu Jiang. An efficient b-spline lagrangian/eulerian method for compressible flow, shock waves, and fracturing solids. *ACM Trans. Graph.*, 41(5), May 2022.

[48] Yue Chang, Shusen Liu, Xiaowei He, Sheng Li, and Guoping Wang. Semi-analytical solid boundary conditions for free surface flows. *Computer Graphics Forum*, 39(7):131–141, 2020.

[49] Jingyu Chen, Victoria Kala, Alan Marquez-Razon, Elias Gueidon, David A. B. Hyde, and Joseph Teran. A momentum-conserving implicit material point method for surface tension with contact angles and spatial gradients. *ACM Trans. Graph.*, 40(4), July 2021.

[50] Xiao-Song Chen, Chen-Feng Li, Geng-Chen Cao, Yun-Tao Jiang, and Shi-Min Hu. A moving least square reproducing kernel particle method for unified multiphase continuum simulation. *ACM Trans. Graph.*, 39(6), November 2020.

[51] Yi-Lu Chen, Jonathan Meier, Barbara Solenthaler, and Vinicius C. Azevedo. An extended cut-cell method for sub-grid liquids tracking with surface tension. *ACM Trans. Graph.*, 39(6), November 2020.

[52] Yixin Chen, Wei Li, Rui Fan, and Xiaopei Liu. Gpu optimization for high-quality kinetic fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 28(9):3235–3251, 2022.

[53] Zhili Chen, Byungmoon Kim, Daichi Ito, and Huamin Wang. Wetbrush: Gpu-based 3d painting simulation at the bristle level. *ACM Trans. Graph.*, 34(6), October 2015.

[54] Nuttapong Chentanez and Matthias Mueller-Fischer. A multigrid fluid pressure solver handling separating solid boundary conditions. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1191–1201, 2012.

[55] Nuttapong Chentanez, Matthias Müller, Miles Macklin, and Tae-Yong Kim. Fast grid-free surface tracking. *ACM Trans. Graph.*, 34(4), July 2015.

[56] Nuttapong Chentanez, Matthias Müller, and Tae-Yong Kim. Coupling 3d eulerian, heightfield and particle methods for interactive simulation of large scale liquid phenomena. *IEEE Transactions on Visualization and Computer Graphics*, 21(10):1116–1128, 2015.

[57] Junghyun Cho and Hyeong-Seok Ko. Geometry-aware volume-of-fluid method. *Computer Graphics Forum*, 32(2pt3):379–388, 2013.

[58] Alexandre Joel Chorin. The numerical solution of the navier-stokes equations for an incompressible fluid. *Bulletin of the American Mathematical Society*, 73(6):928–931, 1967.

[59] Jieyu Chu, Nafees Bin Zafar, and Xubo Yang. A schur complement preconditioner for scalable parallel fluid simulation. *ACM Trans. Graph.*, 36(5), July 2017.

[60] Mengyu Chu and Nils Thuerey. Data-driven synthesis of smoke flows with cnn-based feature descriptors. *ACM Trans. Graph.*, 36(4), July 2017.

[61] Pascal Clausen, Martin Wicke, Jonathan R. Shewchuk, and James F. OBrien. Simulating liquids and solid-liquid interactions with lagrangian meshes. *ACM Trans. Graph.*, 32(2), April 2013.

[62] A. Clebsch. Ueber die integration der hydrodynamischen gleichungen. *Journal für die reine und angewandte Mathematik*, 56:1–10, 1859.

[63] MM Conde, C Vega, and A Patrykiejew. The thickness of a liquid layer on the free surface of ice as obtained from computer simulation. *The Journal of chemical physics*, 129(1):014702, 2008.

[64] Jens Cornelis, Markus Ihmsen, Andreas Peer, and Matthias Teschner. Iisph-flip for incompressible fluids. *Computer Graphics Forum*, 33(2):255–262, 2014.

[65] N Cornells and Luc Van Gool. Real-time connectivity constrained depth map computation using programmable graphics hardware. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 1099–1104. IEEE, 2005.

[66] Richard Courant, Kurt Friedrichs, and Hans Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische annalen*, 100(1):32–74, 1928.

[67] Qiaodong Cui, Timothy Langlois, Pradeep Sen, and Theodore Kim. Spiral-spectral fluid simulation. *ACM Trans. Graph.*, 40(6), December 2021.

[68] Qiaodong Cui, Pradeep Sen, and Theodore Kim. Scalable laplacian eigenfluids. *ACM Trans. Graph.*, 37(4), July 2018.

[69] Fang Da, Christopher Batty, and Eitan Grinspun. Multimaterial mesh-based surface tracking. *ACM Trans. Graph.*, 33(4), July 2014.

[70] Fang Da, Christopher Batty, Chris Wojtan, and Eitan Grinspun. Double bubbles sans toil and trouble: Discrete circulation-preserving vortex sheets for soap films and foams. *ACM Trans. Graph.*, 34(4), July 2015.

[71] Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. Surface-only liquids. *ACM Trans. Graph.*, 35(4), July 2016.

[72] F. Dagenais, J. Gagnon, and E. Paquette. Detail-preserving explicit mesh projection and topology matching for particle-based fluids. *Computer Graphics Forum*, 36(8):444–457, 2017.

[73] Fernando de Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun. Power particles: An incompressible fluid solver based on power diagrams. *ACM Trans. Graph.*, 34(4), July 2015.

[74] Tyler De Witt, Christian Lessig, and Eugene Fiume. Fluid simulation using laplacian eigenfunctions. *ACM Trans. Graph.*, 31(1), February 2012.

[75] Yitong Deng, Mengdi Wang, Xiangxin Kong, Shiying Xiong, Zangyueyang Xian, and Bo Zhu. A moving eulerian-lagrangian particle method for thin film and foam simulation. *ACM Trans. Graph.*, 41(4), July 2022.

[76] Mathieu Desbrun and Marie-Paule Gascuel. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '96*, pages 61–76, 1996.

[77] M.-L. Eckert, W. Heidrich, and N. Thuerey. Coupled fluid density and motion from single views. *Computer Graphics Forum*, 37(8):47–58, 2018.

[78] Marie-Lena Eckert, Kiwon Um, and Nils Thuerey. Scalarflow: A large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Trans. Graph.*, 38(6), November 2019.

[79] Essex Edwards and Robert Bridson. Detailed water with coarse grids: Combining surface meshes and adaptive discontinuous galerkin. *ACM Trans. Graph.*, 33(4), July 2014.

[80] R. Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. Chimera grids for water simulation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, page 85–94, 2013.

[81] Abbas Fakhari, Diogo Bolster, and Li-Shi Luo. A weighted multiple-relaxation-time lattice boltzmann method for multiphase flows and its application to partial coalescence cascades. *Journal of Computational Physics*, 341:22–43, 2017.

[82] Yu Fang, Yuanming Hu, Shi-Min Hu, and Chenfanfu Jiang. A temporally adaptive material point method with regional time stepping. *Computer Graphics Forum*, 37(8):195–204, 2018.

[83] Yu Fang, Ziyin Qu, Minchen Li, Xinxin Zhang, Yixin Zhu, Mridul Aanjaneya, and Chenfanfu Jiang. Iq-mpm: An interface quadrature material point method for non-sticky strongly two-way coupled nonlinear solids and fluids. *ACM Trans. Graph.*, 39(4), July 2020.

[84] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Proc. ACM SIGGRAPH*, pages 15–22, 2001.

[85] Yun (Raymond) Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. A multi-scale model for simulating liquid-fabric interactions. *ACM Trans. Graph.*, 37(4), July 2018.

[86] Yun (Raymond) Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. A multi-scale model for coupling strands with shear-dependent liquid. *ACM Trans. Graph.*, 38(6), November 2019.

[87] Yun (Raymond) Fei, Henrique Teles Maia, Christopher Batty, Changxi Zheng, and Eitan Grinspun. A multi-scale model for simulating liquid-hair interactions. *ACM Trans. Graph.*, 36(4), July 2017.

[88] Fan Feng, Jinyuan Liu, Shiying Xiong, Shuqi Yang, Yaorui Zhang, and Bo Zhu. Impulse fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2022.

[89] Florian Ferstl, Ryoichi Ando, Chris Wojtan, Rüdiger Westermann, and Nils Thuerey. Narrow band flip for liquid simulations. *Computer Graphics Forum*, 35(2):225–232, 2016.

[90] Florian Ferstl, Rüdiger Westermann, and Christian Dick. Large-scale liquid simulation on adaptive hexahedral grids. *IEEE Transactions on Visualization and Computer Graphics*, 20(10):1405–1417, 2014.

[91] Sean Flynn, Parris Egbert, Seth Holladay, and Bryan Morse. Fluid carving: Intelligent resizing for fluid simulation data. *ACM Trans. Graph.*, 38(6), November 2019.

[92] Sean Flynn, David Hart, Bryan Morse, Seth Holladay, and Parris Egbert. Generalized fluid carving with fast lattice-guided seam computation. *ACM Trans. Graph.*, 40(6), December 2021.

[93] Zahra Forootaninia and Rahul Narain. Frequency-domain smoke guiding. *ACM Trans. Graph.*, 39(6), November 2020.

[94] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.

[95] Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. A polynomial particle-in-cell method. *ACM Trans. Graph.*, 36(6), November 2017.

[96] Makoto Fujisawa and Kenjiro T. Miura. An efficient boundary handling with a modified density calculation for sph. *Computer Graphics Forum*, 34(7):155–162, 2015.

[97] Jonathan Gagnon, Julián E. Guzmán, David Mould, and Eric Paquette. Patch erosion for deformable lapped textures on 3d fluids. *Computer Graphics Forum*, 40(2):367–374, 2021.

[98] Jonathan Gagnon, Julián E. Guzmán, Valentin Vervondel, François Dagenais, David Mould, and Eric Paquette. Distribution update of deformable patches for texture synthesis on the free surface of fluids. *Computer Graphics Forum*, 38(7):491–500, 2019.

[99] Ming Gao, Andre Pradhana, Xuchen Han, Qi Guo, Grant Kot, Eftychios Sifakis, and Chenfanfu Jiang. Animating fluid sediment mixture in particle-laden flows. *ACM Trans. Graph.*, 37(4), July 2018.

[100] Ming Gao, Andre Pradhana Tampubolon, Chenfanfu Jiang, and Eftychios Sifakis. An adaptive generalized interpolation material point method for simulating elastoplastic materials. *ACM Trans. Graph.*, 36(6), November 2017.

[101] Ming Gao, Xinlei Wang, Kui Wu, Andre Pradhana, Eftychios Sifakis, Cem Yuksel, and Chenfanfu Jiang. Gpu optimization of material point methods. *ACM Trans. Graph.*, 37(6), December 2018.

[102] Yang Gao, Shuai Li, Aimin Hao, and Hong Qin. Simulating multiscale, granular materials and their transitions with a hybrid euler-lagrange solver. *IEEE Transactions on Visualization and Computer Graphics*, 27(12):4483–4494, 2021.

[103] Yang Gao, Quancheng Zhang, Shuai Li, Aimin Hao, and Hong Qin. Accelerating liquid simulation with an improved data-driven method. *Computer Graphics Forum*, 39(6):180–191, 2020.

[104] Yue Gao, Chen-Feng Li, Bo Ren, and Shi-Min Hu. View-dependent multiscale fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 19(2):178–188, 2013.

[105] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389, December 1977.

[106] Christoph Gissler, Andreas Peer, Stefan Band, Jan Bender, and Matthias Teschner. Interlinked sph pressure solvers for strong fluid-rigid coupling. *ACM Trans. Graph.*, 38(1), January 2019.

[107] Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming Lin. Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Trans. Graph.*, 31(6), November 2012.

[108] Ryan Goldade, Mridul Aanjaneya, and Christopher Batty. Constraint bubbles and affine regions: Reduced fluid models for efficient immersed bubbles and flexible spatial coarsening. *ACM Trans. Graph.*, 39(4), July 2020.

[109] Ryan Goldade, Christopher Batty, and Chris Wojtan. A practical method for high-resolution embedded liquid surfaces. *Computer Graphics Forum*, 35(2):233–242, 2016.

[110] Ryan Goldade, Yipeng Wang, Mridul Aanjaneya, and Christopher Batty. An adaptive variational finite difference framework for efficient symmetric octree viscosity. *ACM Trans. Graph.*, 38(4), July 2019.

[111] Prashant Goswami and Christopher Batty. Regional Time Stepping for SPH. In Eric Galin and Michael Wand, editors, *Eurographics 2014 - Short Papers*, 2014.

[112] Simon Green. Screen space fluid rendering for games. In *Proceedings for the Game Developers Conference*. Moscone Center San Francisco, CA, 2010.

[113] L Greengard and V Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[114] James Gregson, Ivo Ihrke, Nils Thuerey, and Wolfgang Heidrich. From capture to simulation: Connecting forward and inverse problems in fluids. *ACM Trans. Graph.*, 33(4), July 2014.

[115] Jie Guo, Mengtian Li, Zijing Zong, Yuntao Liu, Jingwu He, Yanwen Guo, and Ling-Qi Yan. Volumetric appearance stylization with stylizing kernel prediction network. *ACM Trans. Graph.*, 40(4), July 2021.

[116] Yulong Guo, Xiaopei Liu, and Xuemiao Xu. A unified detail-preserving liquid simulation by two-phase lattice boltzmann modeling. *IEEE Transactions on Visualization and Computer Graphics*, 23(5):1479–1491, 2017.

[117] Francis H Harlow. The particle-in-cell method for numerical solution of problems in fluid dynamics. Technical report, March 1962.

[118] Francis H Harlow. The particle-in-cell computing method for fluid dynamics. *Methods Comput. Phys.*, 3:319–343, 1964.

[119] Francis H. Harlow and J. Eddie Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids*, 8(12):2182–2189, 1965.

[120] S He and R. W. H. Lau. Synthetic controllable turbulence using robust second vorticity confinement. *Computer Graphics Forum*, 32(1):27–35, 2013.

[121] Xiaowei He, Ning Liu, Guoping Wang, Fengjun Zhang, Sheng Li, Songdong Shao, and Hongan Wang. Staggered meshless solid-fluid coupling. *ACM Trans. Graph.*, 31(6), November 2012.

[122] Xiaowei He, Huamin Wang, and Enhua Wu. Projective peridynamics for modeling versatile elastoplastic materials. *IEEE Transactions on Visualization and Computer Graphics*, 24(9):2589–2599, 2018.

[123] Xiaowei He, Huamin Wang, Fengjun Zhang, Hongan Wang, Guoping Wang, and Kun Zhou. Robust simulation of sparsely sampled thin features in sph-based free surface flows. *ACM Trans. Graph.*, 34(1), December 2015.

[124] Xiaowei He, Huamin Wang, Fengjun Zhang, Hongan Wang, Guoping Wang, Kun Zhou, and Enhua Wu. Simulation of fluid mixing with interface control. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '15, page 129–135, 2015.

[125] David J. Hill and Ronald D. Henderson. Efficient fluid simulation on the surface of a sphere. *ACM Trans. Graph.*, 35(2), April 2016.

[126] Hendrik Hochstetter and Andreas Kolb. Evaporation and condensation of sph-based fluids. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, 2017.

[127] Darryl D Holm, Tanya Schmah, and Cristina Stoica. *Geometric mechanics and symmetry: from finite to infinite dimensions*, volume 12. 2009.

[128] Christopher Horvath and Barbara Solenthaler. Mass preserving multi-scale sph. *Pixar Technical Memo #13-04*, 01 2013.

[129] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. Graph.*, 37(4), July 2018.

[130] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: A language for high-performance computation on spatially sparse data structures. *ACM Trans. Graph.*, 38(6), November 2019.

[131] Yuanming Hu, Jiafeng Liu, Xuanda Yang, Mingkuan Xu, Ye Kuang, Weiwei Xu, Qiang Dai, William T. Freeman, and Frédo

Durand. Quantaichi: A compiler for quantized simulations. *ACM Trans. Graph.*, 40(4), July 2021.

[132] Libo Huang, Torsten Hädrich, and Dominik L. Michels. On the accurate large-scale simulation of ferrofluids. *ACM Trans. Graph.*, 38(4), July 2019.

[133] Libo Huang and Dominik L. Michels. Surface-only ferrofluids. *ACM Trans. Graph.*, 39(6), November 2020.

[134] Libo Huang, Ziyin Qu, Xun Tan, Xinxin Zhang, Dominik L. Michels, and Chenfanfu Jiang. Ships, splashes, and waves on a vast ocean. *ACM Trans. Graph.*, 40(6), December 2021.

[135] Weizhen Huang, Julian Iseringhausen, Tom Kneiphof, Ziyin Qu, Chenfanfu Jiang, and Matthias B. Hullin. Chemomechanical simulation of soap film flow on spherical bubbles. *ACM Trans. Graph.*, 39(4), July 2020.

[136] M. Huber, B. Eberhardt, and D. Weiskopf. Boundary handling at cloth–fluid contact. *Computer Graphics Forum*, 34(1):14–25, 2015.

[137] David A. B. Hyde, Steven W. Gagniere, Alan Marquez-Razon, and Joseph Teran. An implicit updated lagrangian formulation for liquids with large surface energy. *ACM Trans. Graph.*, 39(6), November 2020.

[138] Hikaru Ibayashi, Chris Wojtan, Nils Thuerey, Takeo Igarashi, and Ryoichi Ando. Simulating liquids on dynamically warping grids. *IEEE Transactions on Visualization and Computer Graphics*, 26(6):2288–2302, 2020.

[139] Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. Implicit incompressible sph. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):426–435, 2014.

[140] Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. SPH Fluids in Computer Graphics. In Sylvain Lefebvre and Michela Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014.

[141] Jaeho Im, Hanwook Park, Jong-Hyun Kim, and Chang-Hun Kim. A particle-grid method for opaque ice formation. *Computer Graphics Forum*, 32(2pt3):371–377, 2013.

[142] T. Inglis, M.-L. Eckert, J. Gregson, and N. Thuerey. Primal-dual optimization for fluids. *Computer Graphics Forum*, 36(8):354–368, 2017.

[143] Sadashige Ishida, Peter Synak, Fumiya Narita, Toshiya Hachisuka, and Chris Wojtan. A model for soap film dynamics with evolving thickness. *ACM Trans. Graph.*, 39(4), July 2020.

[144] Sadashige Ishida, Masafumi Yamamoto, Ryoichi Ando, and Toshiya Hachisuka. A hyperbolic geometric flow for evolving films and foams. *ACM Trans. Graph.*, 36(6), November 2017.

[145] Andrei C. Jalba, Jacek Kustra, and Alexandru C. Telea. Surface and curve skeletonization of large 3d models on the gpu. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1495–1508, 2013.

[146] Ondřej Jamriška, Jakub Fišer, Paul Asente, Jingwan Lu, Eli Shechtman, and Daniel Sýkora. Lazyfluids: Appearance transfer for fluid animations. *ACM Trans. Graph.*, 34(4), July 2015.

[147] Taekwon Jang, Heeyoung Kim, Jinhyuk Bae, Jaewoo Seo, and Junyong Noh. Multilevel vorticity confinement for water turbulence simulation. *The Visual Computer*, 26(6–8):873–881, 2010.

[148] S. Jeschke, C. Hafner, N. Chentanez, M. Macklin, M. Müller-Fischer, and C. Wojtan. Making procedural water waves boundary-aware. *Computer Graphics Forum*, 39(8):47–54, 2020.

[149] Stefan Jeschke, Tomáš Skřivan, Matthias Müller-Fischer, Nuttapong Chentanez, Miles Macklin, and Chris Wojtan. Water surface wavelets. *ACM Trans. Graph.*, 37(4), July 2018.

[150] Stefan Jeschke and Chris Wojtan. Water wave animation via wavefront parameter interpolation. *ACM Trans. Graph.*, 34(3), May 2015.

[151] Stefan Jeschke and Chris Wojtan. Water wave packets. *ACM Trans. Graph.*, 36(4), July 2017.

[152] Chenfanfu Jiang, Theodore Gast, and Joseph Teran. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Trans. Graph.*, 36(4), July 2017.

[153] Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. The affine particle-in-cell method. *ACM Trans. Graph.*, 34(4), July 2015.

[154] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, SIGGRAPH '16, 2016.

[155] Y. Jiang and Y. Lan. A dynamic mixture model for non-equilibrium multiphase fluids. *Computer Graphics Forum*, 40(7):85–95, 2021.

[156] Y. Jiang, C. Li, S. Deng, and S. M. Hu. A divergence-free mixture model for multiphase fluids. *Computer Graphics Forum*, 39(8):69–77, 2020.

[157] Richard Jones and Richard Southern. Physically-based droplet interaction. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, 2017.

[158] T. Keeler and R. Bridson. Ocean waves animation using boundary integral equations and explicit mesh tracking. *SCA 2014 - Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 11–19, 2014.

[159] Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. Transport-based neural style transfer for smoke simulations. *ACM Trans. Graph.*, 38(6), November 2019.

[160] Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. Lagrangian neural style transfer for fluids. *ACM Trans. Graph.*, 39(4), July 2020.

[161] Jong-Hyun Kim, Jung Lee, Sungdeok Cha, and Chang-Hun Kim. Efficient representation of detailed foam waves by incorporating projective space. *IEEE Transactions on Visualization and Computer Graphics*, 23(9):2056–2068, 2017.

[162] Theodore Kim and John Delaney. Subspace fluid re-simulation. *ACM Trans. Graph.*, 32(4), July 2013.

[163] Theodore Kim, Jerry Tessendorf, and Nils Thürey. Closest point turbulence for liquid surfaces. *ACM Trans. Graph.*, 32(2), April 2013.

[164] T. Koike, S. Morishima, and R. Ando. Asynchronous eulerian liquid simulation. *Computer Graphics Forum*, 39(2):1–8, 2020.

[165] Dan Koschier and Jan Bender. Density maps for improved sph boundary handling. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, 2017.

[166] Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids. *Eurographics 2019 - Tutorials*, 2019.

[167] L'ubor Ladický, SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, and Markus Gross. Data-driven fluid simulations using regression forests. *ACM Trans. Graph.*, 34(6), October 2015.

[168] Junyu Lai, Yangang Chen, Yu Gu, Christopher Batty, and Justin W.L. Wan. Fast and scalable solvers for the fluid pressure equations with separating solid boundary conditions. *Computer Graphics Forum*, 39(2):23–33, 2020.

[169] Lev Davidovich Landau and Evgenii Mikhailovich Lifshitz. *Fluid Mechanics*, volume 6. 2013.

[170] Timothy R. Langlois, Changxi Zheng, and Doug L. James. Toward animating water with complex acoustic bubbles. *ACM Trans. Graph.*, 35(4), July 2016.

[171] Egor Larionov, Christopher Batty, and Robert Bridson. Variational stokes: A unified pressure-viscosity solver for accurate viscous liquids. *ACM Trans. Graph.*, 36(4), July 2017.

[172] Minjae Lee, David Hyde, Michael Bao, and Ronald Fedkiw. A skinned tetrahedral mesh for hair animation and hair-water interaction. *IEEE Transactions on Visualization and Computer Graphics*, 25(3):1449–1459, 2019.

[173] Minjae Lee, David Hyde, Kevin Li, and Ronald Fedkiw. A robust volume conserving method for character-water interaction. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '19, 2019.

[174] Michael Lentine, Mridul Aanjaneya, and Ronald Fedkiw. Mass and momentum conservation for fluid simulation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 91–100, 2011.

[175] Anat Levin, Rob Fergus, Frédo Durand, and William T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Trans. Graph.*, 26(3):70–es, July 2007.

[176] Chen Li, Sheng Qiu, Changbo Wang, and Hong Qin. Learning physical parameters and detail enhancement for gaseous scene design based on data guidance. *IEEE Transactions on Visualization and Computer Graphics*, 27(10):3867–3880, 2021.

[177] Wei Li, Kai Bai, and Xiaopei Liu. Continuous-scale kinetic fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 25(9):2694–2709, 2019.

[178] Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. Fast and scalable turbulent flow simulation with two-way coupling. *ACM Trans. Graph.*, 39(4), July 2020.

[179] Wei Li, Daoming Liu, Mathieu Desbrun, Jin Huang, and Xiaopei Liu. Kinetic-based multiphase flow simulation. *IEEE Transactions on Visualization and Computer Graphics*, 27(7):3318–3334, 2021.

[180] Wei Li, Yihui Ma, Xiaopei Liu, and Mathieu Desbrun. Efficient kinetic simulation of two-phase flows. *ACM Trans. Graph.*, 41(4), July 2022.

[181] Xiaosheng Li, Xiaowei He, Xuehui Liu, Jian J. Zhang, Baoquan Liu, and Enhua Wu. Multiphase interface tracking with fast semi-lagrangian contouring. *IEEE Transactions on Visualization and Computer Graphics*, 22(8):1973–1986, 2016.

[182] Xiangyun Liao, Weixin Si, Zhiyong Yuan, Hanqiu Sun, Jing Qin, Qiong Wang, and Pheng-Ann Heng. Animating wall-bounded turbulent smoke via filament-mesh particle-particle method. *IEEE Transactions on Visualization and Computer Graphics*, 24(3):1260–1273, 2018.

[183] Beibei Liu, Gemma Mason, Julian Hodgson, Yiying Tong, and Mathieu Desbrun. Model-reduced variational fluid simulation. *ACM Trans. Graph.*, 34(6), October 2015.

[184] C. Liu, L. Wang, Z. Li, S. Quan, and Y. Xu. Real-time lighting estimation for augmented reality via differentiable screen-space rendering. *IEEE Transactions on Visualization & Computer Graphics*, (01):1–1, January 5555.

[185] Haixiang Liu, Nathan Mitchell, Mridul Aanjaneya, and Eftychios Sifakis. A scalable schur-complement fluids solver for heterogeneous compute platforms. *ACM Trans. Graph.*, 35(6), November 2016.

[186] Shusen Liu, Xiaowei He, Wencheng Wang, and Enhua Wu. Adapted simple algorithm for incompressible sph fluids with a broad range viscosity. *IEEE Transactions on Visualization and Computer Graphics*, 28(9):3168–3179, 2022.

[187] Sinuo Liu, Xiaokun Wang, Xiaojuan Ban, Yanrui Xu, Jing Zhou, Jiří Kosinka, and Alexandru C. Telea. Turbulent details simulation for sph fluids via vorticity refinement. *Computer Graphics Forum*, 40(1):54–67, 2021.

[188] Xiaopei Liu, Wai-Man Pang, Jing Qin, and Chi-Wing Fu. Turbulence simulation by adaptive multi-relaxation lattice boltzmann modeling. *IEEE Transactions on Visualization and Computer Graphics*, 20(2):289–302, 2014.

[189] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, August 1987.

[190] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph.*, 23(3):457–462, August 2004.

[191] Jia-Ming Lu, Xiao-Song Chen, Xiao Yan, Chen-Feng Li, Ming Lin, and Shi-Min Hu. A rigging-skinning scheme to control fluid simulation. *Computer Graphics Forum*, 38(7):501–512, 2019.

[192] Leon B Lucy. A numerical approach to the testing of the fission hypothesis. *The astronomical journal*, 82:1013–1024, 1977.

[193] Chaoyang Lyu, Wei Li, Mathieu Desbrun, and Xiaopei Liu. Fast and versatile fluid-solid coupling for turbulent flow simulation. *ACM Trans. Graph.*, 40(6), December 2021.

[194] Miles Macklin and Matthias Müller. Position based fluids. *ACM Trans. Graph.*, 32(4), July 2013.

[195] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33(4), July 2014.

[196] Mikko Manninen, Veikko Taivassalo, and Sirpa Kallio. On the mixture model for multiphase flow. *VTT Publications*, (288):3–67, 1996.

[197] P.-L. Manteaux, C. Wojtan, R. Narain, S. Redon, F. Faure, and M.-P. Cani. Adaptive physically based models in computer graphics. *Computer Graphics Forum*, 36(6):312–337, 2017.

[198] Omid Mashayekhi, Chinmayee Shah, Hang Qu, Andrew Lim, and Philip Levis. Automatically distributing eulerian and hybrid fluid simulations in the cloud. *ACM Trans. Graph.*, 37(2), June 2018.

[199] Olivier Mercier, Cynthia Beauchemin, Nils Thuerey, Theodore Kim, and Derek Nowrouzezahrai. Surface turbulence for particle-based liquid simulations. *ACM Trans. Graph.*, 34(6), October 2015.

[200] Olivier Mercier and Derek Nowrouzezahrai. Local bases for model-reduced smoke simulations. *Computer Graphics Forum*, 39(2):9–22, 2020.

[201] Marek Krzysztof Misztal, Kenny Erleben, Adam Bargteil, Jens Fursund, Brian Bunch Christensen, Jakob Andreas Bærentzen, and Robert Bridson. Multiphase flow of immiscible fluids on unstructured moving meshes. *IEEE Transactions on Visualization and Computer Graphics*, 20(1):4–16, 2014.

[202] Joe J Monaghan. Smoothed particle hydrodynamics. *Reports on progress in physics*, 68(8):1703, 2005.

[203] Joseph J Monaghan and John C Lattanzio. A refined particle method for astrophysical problems. *Astronomy and Astrophysics (ISSN 0004-6361), vol. 149, no. 1, Aug. 1985, p. 135-143.*, 149:135–143, 1985.

[204] D. Morgenroth, S. Reinhardt, D. Weiskopf, and B. Eberhardt. Efficient 2d simulation on moving 3d surfaces. *Computer Graphics Forum*, 39(8):27–38, 2020.

[205] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, page 141–151, Goslar, DEU, 2004. Eurographics Association.

[206] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *J. Vis. Comun. Image Represent.*, 18(2):109–118, April 2007.

[207] Ken Museth. Vdb: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph.*, 32(3), July 2013.

[208] Mohammad Sina Nabizadeh, Stephanie Wang, Ravi Ramamoorthi, and Albert Chern. Covector fluids. *ACM Trans. Graph.*, 41(4), July 2022.

[209] Kentaro Nagasawa, Takayuki Suzuki, Ryohei Seto, Masato Okada, and Yonghao Yue. Mixing sauces: A viscosity blending model for shear thinning fluids. *ACM Trans. Graph.*, 38(4), July 2019.

[210] Rafael Nakanishi, Filipe Nascimento, Rafael Campos, Paulo Pagliosa, and Afonso Paiva. Rbf liquids: An adaptive pic solver using rbf-fd. *ACM Trans. Graph.*, 39(6), November 2020.

[211] Liordino dos S Rocha Neto and Antonio L Apolinário Jr. Real-time screen space cartoon water rendering with the iterative separated bilateral filter. *SBC Journal on Interactive Systems*, 8(1):20–32, 2017.

[212] Xingyu Ni, Bo Zhu, Bin Wang, and Baoquan Chen. A level-set method for magnetic substance simulation. *ACM Trans. Graph.*, 39(4), July 2020.

[213] Xiaoying Nie, Yong Hu, Zhiyuan Su, and Xukun Shen. Fluid reconstruction and editing from a monocular video based on the sph model with external force guidance. *Computer Graphics Forum*, 40(6):62–76, 2021.

[214] Michael B. Nielsen and Ole Østerby. A two-continua approach to eulerian simulation of water spray. *ACM Trans. Graph.*, 32(4), July 2013.

[215] Michael B Nielsen, Andreas Söderström, and Robert Bridson. Synthesizing waves from animated height fields. *ACM Transactions on Graphics (TOG)*, 32(1):1–9, 2013.

[216] Young Jin Oh and In-Kwon Lee. Two-step temporal interpolation network using forward advection for efficient smoke simulation. *Computer Graphics Forum*, 40(2):355–365, 2021.

[217] Makoto Okabe, Yoshinori Dobashi, Ken Anjyo, and Rikio Onai. Fluid volume modeling from sparse multi-view images by appearance transfer. *ACM Trans. Graph.*, 34(4), July 2015.

[218] Felipe Oliveira and Afonso Paiva. Narrow-band screen-space fluid rendering. volume 41, pages 82–93, 2022.

[219] Jens Orthmann, Hendrik Hochstetter, Julian Bader, Serkan Bayraktar, and Andreas Kolb. Consistent surface model for sph-based fluid transport. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, page 95–103, 2013.

[220] Jens Orthmann and Andreas Kolb. Temporal blending for adaptive sph. *Computer Graphics Forum*, 31(8):2436–2449, 2012.

[221] Marcel Padilla, Albert Chern, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. On bubble rings and ink chandeliers. *ACM Trans. Graph.*, 38(4), July 2019.

[222] Zherong Pan, Jin Huang, Yiying Tong, and Hujun Bao. Wake synthesis for shallow water equation. *Computer Graphics Forum*, 31(7):2029–2036, 2012.

[223] Zherong Pan, Jin Huang, Yiying Tong, Changxi Zheng, and Hujun Bao. Interactive localized liquid motion editing. *ACM Trans. Graph.*, 32(6), November 2013.

[224] Zherong Pan and Dinesh Manocha. Efficient solver for spacetime control of smoke. *ACM Trans. Graph.*, 36(5), July 2017.

[225] Jonathan Panuelos, Ryan Goldade, and Christopher Batty. Efficient Unified Stokes using a Polynomial Reduced Fluid Model. In Dominik L. Michels, editor, *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation - Posters*, 2020.

[226] Saket Patkar, Mridul Aanjaneya, Dmitriy Karpman, and Ronald Fedkiw. A hybrid lagrangian-eulerian formulation for bubble generation and dynamics. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, page 105–114, 2013.

[227] Saket Patkar and Parag Chaudhuri. Wetting of porous solids. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1592–1604, 2013.

[228] Andreas Peer, Christoph Gissler, Stefan Band, and Matthias Teschner. An implicit sph formulation for incompressible linearly elastic solids. *Computer Graphics Forum*, 37(6):135–148, 2018.

[229] Andreas Peer, Markus Ihmsen, Jens Cornelis, and Matthias Teschner. An implicit viscosity formulation for sph fluids. *ACM Trans. Graph.*, 34(4), July 2015.

[230] Andreas Peer and Matthias Teschner. Prescribed velocity gradients for highly viscous sph fluids with vorticity diffusion. *IEEE Transactions on Visualization and Computer Graphics*, 23(12):2656–2662, 2017.

[231] Stéphane Popinet. Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries. *Journal of Computational Physics*, 190(2):572–600, 2003.

[232] Daniel J. Price. Smoothed particle hydrodynamics and magnetohydrodynamics. *Journal of Computational Physics*, 231(3):759–794, 2012. Special Issue: Computational Plasma Physics.

[233] Hang Qu, Omid Mashayekhi, Chinmayee Shah, and Philip Levis. Accelerating distributed graphical fluid simulations with micropartitioning. *Computer Graphics Forum*, 39(1):375–388, 2020.

[234] Karthik Raveendran, Nils Thuerey, Chris Wojtan, and Greg Turk. Controlling liquids using meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, page 255–264, 2012.

[235] Karthik Raveendran, Chris Wojtan, Nils Thuerey, and Greg Turk. Blending liquids. *ACM Trans. Graph.*, 33(4), July 2014.

[236] William T. Reeves and Ricki Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *SIGGRAPH Comput. Graph.*, 19(3):313–322, July 1985.

[237] Stefan Reinhardt, Markus Huber, Bernhard Eberhardt, and Daniel Weiskopf. Fully asynchronous sph simulation. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, 2017.

[238] Bo Ren, Wei He, Chenfeng Li, and Xu Chen. Incompressibility enforcement for multiple-fluid sph using deformation gradient. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2021.

[239] Bo Ren, Chenfeng Li, Xiao Yan, Ming C. Lin, Javier Bonet, and Shi-Min Hu. Multiple-fluid sph simulation using a mixture model. *ACM Trans. Graph.*, 33(5), September 2014.

[240] Bo Ren, Ben Xu, and Chenfeng Li. Unified particle system for multiple-fluid flow and porous material. *ACM Trans. Graph.*, 40(4), July 2021.

[241] Bo Ren, Tailing Yuan, Chenfeng Li, Kun Xu, and Shi-Min Hu. Real-time high-fidelity surface flow simulation. *IEEE Transactions on Visualization and Computer Graphics*, 24(8):2411–2423, 2018.

[242] Bruno Roy, Pierre Poulin, and Eric Paquette. Neural upflow: A scene flow learning approach to increase the apparent resolution of particle-based liquids. *Proc. ACM Comput. Graph. Interact. Tech.*, 4(3), September 2021.

[243] Liangwang Ruan, Jinyuan Liu, Bo Zhu, Shinjiro Sueda, Bin Wang, and Baoquan Chen. Solid-fluid interaction with surface-tension-dominant contact. *ACM Trans. Graph.*, 40(4), July 2021.

[244] W. Rungjiratananon, Y. Kanamori, and T. Nishita. Wetting effects in hair simulation. *Computer Graphics Forum*, 31(7):1993–2002, 2012.

[245] Michael Rusin. The structure of nonlinear blending models. *Chemical Engineering Science*, 30(8):937–944, 1975.

[246] Steven J Ruuth and Barry Merriman. A simple embedding method for solving partial differential equations on surfaces. *Journal of Computational Physics*, 227(3):1943–1961, 2008.

[247] Marcos Sandim, Douglas Cedrim, Luis Gustavo Nonato, Paulo Pagliosa, and Afonso Paiva. Boundary detection in particle-based fluids. *Computer Graphics Forum*, 35(2):215–224, 2016.

[248] Syuhei Sato, Yoshinori Dobashi, Theodore Kim, and Tomoyuki Nishita. Example-based turbulence style transfer. *ACM Trans. Graph.*, 37(4), July 2018.

[249] Syuhei Sato, Yoshinori Dobashi, and Tomoyuki Nishita. Editing fluid animation using flow interpolation. *ACM Trans. Graph.*, 37(5), September 2018.

[250] T. Sato, C. Wojtan, N. Thuerey, T. Igarashi, and R. Ando. Extended narrow band flip for liquid simulations. *Computer Graphics Forum*, 37(2):169–177, 2018.

[251] Hagit Schechter and Robert Bridson. Ghost sph for animating water. *ACM Trans. Graph.*, 31(4), July 2012.

[252] Arnaud Schoentgen, Pierre Poulin, Emmanuelle Darles, and Philippe Meseure. Particle-based liquid control using animation templates. *Computer Graphics Forum*, 39(8):79–88, 2020.

[253] Camille Schreck, Christian Hafner, and Chris Wojtan. Fundamental solutions for water wave animation. *ACM Trans. Graph.*, 38(4), July 2019.

[254] Camille Schreck and Chris Wojtan. Coupling 3d liquid simulation with 2d wave propagation for large scale water surface animation using the equivalent sources method. *Computer Graphics Forum*, 41(2):343–353, 2022.

[255] Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. Spgrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Trans. Graph.*, 33(6), November 2014.

[256] C. Shah, D. Hyde, H. Qu, and P. Levis. Distributing and load balancing sparse fluid simulations. *Computer Graphics Forum*, 37(8):35–46, 2018.

[257] Han Shao, Libo Huang, and Dominik Michels. A current loop model for the fast simulation of ferrofluids. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–12, October 2022.

[258] Han Shao, Libo Huang, and Dominik L. Michels. A fast unsmoothed aggregation algebraic multigrid framework for the large-scale simulation of incompressible flow. *ACM Trans. Graph.*, 41(4), July 2022.

[259] X. Shao, Z. Zhou, N. Magnenat-Thalmann, and W. Wu. Stable and fast fluid–solid coupling for incompressible sph. *Computer Graphics Forum*, 34(1):191–204, 2015.

[260] Christian Sigg, Tim Weyrich, Mario Botsch, and Markus Gross. Gpu-based ray-casting of quadratic surfaces. In *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, SPBG'06, page 59–65, Goslar, DEU, 2006. Eurographics Association.

[261] Tom Sito. *Moving Innovation: A History of Computer Animation*. The MIT Press, 2013.

[262] Tomas Skrivan, Andreas Soderstrom, John Johansson, Christoph Sprenger, Ken Museth, and Chris Wojtan. Wave curves: Simulating lagrangian water waves on dynamically deforming surfaces. *ACM Trans. Graph.*, 39(4), July 2020.

[263] B. Solenthaler and R. Pajarola. Density contrast sph interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pages 211–218, 2008.

[264] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible sph. *ACM Trans. Graph.*, 28(3), July 2009.

[265] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible sph. *ACM Trans. Graph.*, 28(3), July 2009.

[266] Barbara Solenthaler and Markus Gross. Two-scale particle simulation. *ACM Trans. Graph.*, 30(4), July 2011.

[267] Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 121–128, 1999.

[268] Jos Stam and Eugene Fiume. Turbulent wind fields for gaseous phenomena. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, page 369–376, 1993.

[269] Matt Stanton, Yu Sheng, Martin Wicke, Federico Perazzi, Amos Yuen, Srinivasa Narasimhan, and Adrien Treuille. Non-polynomial galerkin projection on deforming meshes. *ACM Trans. Graph.*, 32(4), July 2013.

[270] Alexey Stomakhin, Craig Schroeder, Chenfanfu Jiang, Lawrence Chai, Joseph Teran, and Andrew Selle. Augmented mpm for phase-change and varied materials. *ACM Trans. Graph.*, 33(4), July 2014.

[271] Alexey Stomakhin and Andrew Selle. Fluxed animated boundary method. *ACM Trans. Graph.*, 36(4), July 2017.

[272] Haozhe Su, Tao Xue, Chengguizi Han, Chenfanfu Jiang, and Mridul Aanjaneya. A unified second-order accurate in time mpm formulation for simulating viscoelastic liquids with phase change. *ACM Trans. Graph.*, 40(4), July 2021.

[273] Deborah Sulsky, Shi-Jian Zhou, and Howard L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87(1):236–252, 1995. Particle Simulation Methods.

[274] Yuchen Sun, Xingyu Ni, Bo Zhu, Bin Wang, and Baoquan Chen. A material point method for nonlinearly magnetized materials. *ACM Trans. Graph.*, 40(6), December 2021.

[275] Yunxin Sun, Tamar Shinar, and Craig Schroeder. Effective time step restrictions for explicit mpm simulation. *Computer Graphics Forum*, 39(8):55–67, 2020.

[276] D. F. Swinehart. The beer-lambert law. *Journal of Chemical Education*, 39(7):333, 1962.

[277] Tetsuya Takahashi and Christopher Batty. Monolith: A monolithic pressure-viscosity-contact solver for strong two-way rigid-rigid rigid-fluid coupling. *ACM Trans. Graph.*, 39(6), November 2020.

[278] Tetsuya Takahashi and Christopher Batty. Frictionalmonolith: A monolithic optimization-based approach for granular flow with contact-aware rigid-body coupling. *ACM Trans. Graph.*, 40(6), December 2021.

[279] Tetsuya Takahashi, Yoshinori Dobashi, Issei Fujishiro, Tomoyuki Nishita, and Ming C. Lin. Implicit formulation for sph-based viscous fluids. *Computer Graphics Forum*, 34(2):493–502, 2015.

[280] Tetsuya Takahashi, Yoshinori Dobashi, Tomoyuki Nishita, and Ming C. Lin. An efficient hybrid incompressible sph solver with interface handling for boundary conditions. *Computer Graphics Forum*, 37(1):313–324, 2018.

[281] Tetsuya Takahashi and Ming C. Lin. A multilevel sph solver with unified solid boundary handling. *Computer Graphics Forum*, 35(7):517–526, 2016.

[282] Tetsuya Takahashi and Ming C. Lin. A geometrically consistent viscous fluid solver with two-way fluid-solid coupling. *Computer Graphics Forum*, 38(2):49–58, 2019.

[283] Tetsuya Takahashi and Ming C. Lin. Video-guided real-to-virtual parameter transfer for viscous fluids. *ACM Trans. Graph.*, 38(6), November 2019.

[284] Andre Pradhana Tampubolon, Theodore Gast, Gergely Klár, Chuyuan Fu, Joseph Teran, Chenfanfu Jiang, and Ken Museth. Multi-species simulation of porous sand and water mixtures. *ACM Trans. Graph.*, 36(4), July 2017.

[285] Jingwei Tang, Vinicius C. Azevedo, Guillaume Cordonnier, and Barbara Solenthaler. Honey, i shrunk the domain: Frequency-aware force field reduction for efficient fluids optimization. *Computer Graphics Forum*, 40(2):339–353, 2021.

[286] Michael Tao, Christopher Batty, Mirela Ben-Chen, Eugene Fiume, and David I. W. Levin. Vempic: Particle-in-polyhedron fluid simulation for intricate solid boundaries. *ACM Trans. Graph.*, 41(4), July 2022.

[287] Yun Teng, David I. W. Levin, and Theodore Kim. Eulerian solid-fluid coupling. *ACM Trans. Graph.*, 35(6), November 2016.

[288] Jerry Tessendorf. Simulating ocean water. *Simulating nature: realistic and interactive techniques. SIGGRAPH*, 1(2):5, 2001.

[289] Nils Thuerey. Interpolations of smoke and liquid simulations. *ACM Trans. Graph.*, 36(1), September 2016.

[290] A. Treuille, A. Lewis, and Z. Popović. Model reduction for real-time fluids. *ACM Transactions on Graphics*, 25(3):826–834, 2006.

[291] Nghia Truong and Cem Yuksel. A narrow-range filter for screen-space fluid rendering. *Proc. ACM Comput. Graph. Interact. Tech.*, 1(1), July 2018.

[292] Nghia Truong, Cem Yuksel, Chakrit Watcharopas, Joshua Aaron Levine, and Robert M. Kirby. Particle merging-and-splitting. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2021.

[293] Kiwon Um, Xiangyu Hu, and Nils Thuerey. Perceptual evaluation of liquid simulation methods. *ACM Trans. Graph.*, 36(4), July 2017.

[294] R. Vacondio, B.D. Rogers, P.K. Stansby, and P. Mignosa. Variable resolution for sph in three dimensions: Towards optimal splitting and coalescing for dynamic adaptivity. *Computer Methods in Applied Mechanics and Engineering*, 300:442–460, 2016.

[295] Orestis Vantzos, Omri Azencot, Max Wardeztky, Martin Rumpf, and Mirela Ben-Chen. Functional thin films on surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1179–1192, 2017.

[296] Orestis Vantzos, Saar Raz, and Mirela Ben-Chen. Real-time viscous thin films. *ACM Trans. Graph.*, 37(6), December 2018.

[297] Mauricio Vines, Ben Houston, Jochen Lang, and Won-Sook Lee. Vortical inviscid flows with two-way solid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics*, 20(2):303–315, 2014.

[298] Huamin Wang, Peter J. Mucha, and Greg Turk. Water drops on surfaces. *ACM Trans. Graph.*, 24(3):921–929, July 2005.

[299] Hui Wang, Yongxu Jin, Anqi Luo, Xubo Yang, and Bo Zhu. Codimensional surface tension flow using moving-least-squares particles. *ACM Trans. Graph.*, 39(4), July 2020.

[300] Mengdi Wang, Yitong Deng, Xiangxin Kong, Aditya H. Prasad, Shiying Xiong, and Bo Zhu. Thin-film smoothed particle hydrodynamics fluid. *ACM Trans. Graph.*, 40(4), July 2021.

[301] Xiaojun Wang, Shiguang Liu, and Yiying Tong. Stain formation on deforming inelastic cloth. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3214–3224, 2018.

[302] Xiaokun Wang, Xiaojuan Ban, Xu Liu, Yalan Zhang, and Lipeng Wang. Effective reconstructing surfaces algorithm of anisotropic kernels orienting sph fluids. *Journal of computer-Aided Design & Computer Graphics*, 28(9):1497–1505, 2016.

[303] XiaoKun Wang, XiaoJuan Ban, Xu Liu, YaLan Zhang, and LiPeng Wang. Efficient extracting surfaces approach employing anisotropic kernels for sph fluids. *Journal of Visualization*, 19(2):301–317, 2016.

[304] Xinlei Wang, Yuxing Qiu, Stuart R. Slattery, Yu Fang, Minchen Li, Song-Chun Zhu, Yixin Zhu, Min Tang, Dinesh Manocha, and Chenfanfu Jiang. A massively parallel and scalable multi-gpu material point method. *ACM Trans. Graph.*, 39(4), July 2020.

[305] Daniel Weber, Johannes Mueller-Roemer, André Stork, and Dieter Fellner. A cut-cell geometric multigrid poisson solver for fluid simulation. *Computer Graphics Forum*, 34(2):481–491, 2015.

[306] Marcel Weiler, Dan Koschier, Magnus Brand, and Jan Bender. A physically consistent implicit viscosity solver for sph fluids. *Computer Graphics Forum*, 37(2):145–155, 2018.

[307] Jakub Wejchert and David Haumann. Animation aerodynamics. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, page 19–22, 1991.

[308] S. Wiewel, M. Becher, and N. Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. *Computer Graphics Forum*, 38(2):71–82, 2019.

[309] S. Wiewel, B. Kim, V. C. Azevedo, B. Solenthaler, and N. Thuerey. Latent space subdivision: Stable and controllable time predictions for fluid flow. *Computer Graphics Forum*, 39(8):15–25, 2020.

[310] R. Winchenbach and A. Kolb. Multi-level memory structures for simulating and rendering smoothed particle hydrodynamics. *Computer Graphics Forum*, 39(6):527–541, 2020.

[311] Rene Winchenbach, Rustam Akhunov, and Andreas Kolb. Semi-analytic boundary handling below particle resolution for smoothed particle hydrodynamics. *ACM Trans. Graph.*, 39(6), November 2020.

[312] Rene Winchenbach, Hendrik Hochstetter, and Andreas Kolb. Constrained Neighbor Lists for SPH-based Fluid Simulations. In Ladislav Kavan and Chris Wojtan, editors, *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, 2016.

[313] Rene Winchenbach, Hendrik Hochstetter, and Andreas Kolb. Infinite continuous adaptivity for incompressible sph. *ACM Trans. Graph.*, 36(4), July 2017.

[314] Rene Winchenbach and Andreas Kolb. Optimized refinement for spatially adaptive sph. *ACM Trans. Graph.*, 40(1), January 2021.

[315] Joel Wretborn, Sean Flynn, and Alexey Stomakhin. Guided bubbles and wet foam for realistic whitewater simulation. *ACM Trans. Graph.*, 41(4), July 2022.

[316] Kui Wu, Nghia Truong, Cem Yuksel, and Rama Hoetzlein. Fast fluid simulations with sparse volumes on the gpu. *Computer Graphics Forum*, 37(2):157–167, 2018.

[317] Xiaoyue Wu, Xubo Yang, and Yang Yang. A novel projection technique with detail capture and shape correction for smoke simulation. *Computer Graphics Forum*, 32(2pt4):389–397, 2013.

[318] Xiangyun Xiao, Hui Wang, and Xubo Yang. A cnn-based flow correction method for fast preview. *Computer Graphics Forum*, 38(2):431–440, 2019.

[319] Xiangyun Xiao, Yanqing Zhou, Hui Wang, and Xubo Yang. A novel cnn-based poisson solver for fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 26(3):1454–1465, 2020.

[320] Yuwei Xiao, Szeyu Chan, Siqi Wang, Bo Zhu, and Xubo Yang. An adaptive staggered-tilted grid for incompressible flow simulation. *ACM Trans. Graph.*, 39(6), November 2020.

[321] You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. Tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Trans. Graph.*, 37(4), July 2018.

[322] Shiying Xiong, Rui Tao, Yaorui Zhang, Fan Feng, and Bo Zhu. Incompressible flow simulation on vortex segment clouds. *ACM Trans. Graph.*, 40(4), July 2021.

[323] Shiying Xiong, Zhecheng Wang, Mengdi Wang, and Bo Zhu. A clebsch method for free-surface vortical flow simulation. *ACM Trans. Graph.*, 41(4), July 2022.

[324] Yanrui Xu. Implementation of multiphase flow simulation. https://github.com/sakamotoyan/TiSPH_multiphase, 2023.

[325] Tao Xue, Haozhe Su, Chengguizi Han, Chenfanfu Jiang, and Mridul Aanjaneya. A novel discretization and numerical solver for non-fourier diffusion. *ACM Trans. Graph.*, 39(6), November 2020.

[326] Guowei Yan, Zhili Chen, Jimei Yang, and Huamin Wang. Interactive liquid splash modeling by user sketches. *ACM Trans. Graph.*, 39(6), November 2020.

[327] X. Yan, C-F. Li, X-S. Chen, and S-M. Hu. Mpm simulation of interacting fluids and solids. *Computer Graphics Forum*, 37(8):183–193, 2018.

[328] Xiao Yan, Yun-Tao Jiang, Chen-Feng Li, Ralph R. Martin, and Shi-Min Hu. Multiphase sph simulation for interactive fluids and solids. *ACM Trans. Graph.*, 35(4), July 2016.

[329] Lipeng Yang, Shuai Li, Aimin Hao, and Hong Qin. Realtime two-way coupling of meshless fluids and nonlinear fem. *Computer Graphics Forum*, 31(7):2037–2046, 2012.

[330] Lipeng Yang, Shuai Li, Aimin Hao, and Hong Qin. Hybrid particle-grid modeling for multi-scale droplet/spray simulation. *Computer Graphics Forum*, 33(7):199–208, 2014.

[331] Meng Yang, Juntao Ye, Frank Ding, Yubo Zhang, and Dong-Ming Yan. A semi-explicit surface tracking mechanism for multi-phase immiscible liquids. *IEEE Transactions on Visualization and Computer Graphics*, 25(10):2873–2885, 2019.

[332] Shuqi Yang, Shiying Xiong, Yaorui Zhang, Fan Feng, Jinyuan Liu, and Bo Zhu. Clebsch gauge fluid. *ACM Trans. Graph.*, 40(4), July 2021.

[333] Tao Yang, Jian Chang, Ming C. Lin, Ralph R. Martin, Jian J. Zhang, and Shi-Min Hu. A unified particle system framework for multi-phase, multi-material visual simulations. *ACM Trans. Graph.*, 36(6), November 2017.

[334] Tao Yang, Jian Chang, Bo Ren, Ming C. Lin, Jian Jun Zhang, and Shi-Min Hu. Fast multiple-fluid simulation using helmholtz free energy. *ACM Trans. Graph.*, 34(6), October 2015.

[335] Tao Yang, Ralph R. Martin, Ming C. Lin, Jian Chang, and Shi-Min Hu. Pairwise force sph model for real-time multi-interaction applications. *IEEE Transactions on Visualization and Computer Graphics*, 23(10):2235–2247, 2017.

[336] Wencong Yang and Chengying Gao. A completely parallel surface reconstruction method for particle-based fluids. *The Visual Computer*, 36(10):2313–2325, 2020.

[337] Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Trans. Graph.*, 32(1), February 2013.

[338] Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Trans. Graph.*, 32(1), February 2013.

[339] Jihun Yu, Chris Wojtan, Greg Turk, and Chee Yap. Explicit mesh surfaces for particle based fluids. *Computer Graphics Forum*, 31(2pt4):815–824, 2012.

[340] Yonghao Yue, Breannan Smith, Christopher Batty, Changxi Zheng, and Eitan Grinspun. Continuum foam: A material point method for shear-dependent flows. *ACM Trans. Graph.*, 34(5), November 2015.

[341] Yonghao Yue, Breannan Smith, Peter Yichen Chen, Maytee Chantharayukhonthorn, Ken Kamrin, and Eitan Grinspun. Hybrid grains: Adaptive coupling of discrete and continuum simulations of granular media. *ACM Trans. Graph.*, 37(6), December 2018.

[342] Omar Zarifi and Christopher Batty. A positive-definite cut-cell method for strong two-way coupling between fluids and deformable bodies. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, 2017.

[343] Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. An advection-reflection solver for detail-preserving fluid simulation. *ACM Trans. Graph.*, 37(4), July 2018.

[344] Xiao Zhai, Fei Hou, Hong Qin, and Aimin Hao. Inverse modelling of incompressible gas flow in subspace. *Computer Graphics Forum*, 36(6):100–111, 2017.

[345] Xiao Zhai, Fei Hou, Hong Qin, and Aimin Hao. Fluid simulation with adaptive staggered power particles on gpus. *IEEE Transactions on Visualization and Computer Graphics*, 26(6):2234–2246, 2020.

[346] Xinxin Zhang and Robert Bridson. A pppm fast summation method for fluids and beyond. *ACM Trans. Graph.*, 33(6), November 2014.

[347] Xinxin Zhang, Robert Bridson, and Chen Greif. Restoring the missing vorticity in advection-projection fluid solvers. *ACM Trans. Graph.*, 34(4), July 2015.

[348] Xinxin Zhang, Minchen Li, and Robert Bridson. Resolving fluid boundary layers with particle strength exchange and weak adaptivity. *ACM Trans. Graph.*, 35(4), July 2016.

[349] Yizhong Zhang, Huamin Wang, Shuai Wang, Yiying Tong, and Kun Zhou. A deformable surface model for real-time water

drop animation. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1281–1289, 2012.

[350] Yubo Zhang and Kwan-Liu Ma. Spatio-temporal extrapolation for fluid animation. *ACM Trans. Graph.*, 32(6), November 2013.

[351] Yi Zheng, Yanyun Chen, Guangzheng Fei, Julie Dorsey, and Enhua Wu. Simulation of textile stains. *IEEE Transactions on Visualization and Computer Graphics*, 25(7):2471–2481, 2019.

[352] Bo Zhu, Minjae Lee, Ed Quigley, and Ronald Fedkiw. Codimensional non-newtonian fluids. *ACM Trans. Graph.*, 34(4), July 2015.

[353] Bo Zhu, Wenlong Lu, Matthew Cong, Byungmoon Kim, and Ronald Fedkiw. A new grid structure for domain extension. *ACM Trans. Graph.*, 32(4), July 2013.

[354] Bo Zhu, Ed Quigley, Matthew Cong, Justin Solomon, and Ronald Fedkiw. Codimensional surface tension flow on simplicial complexes. *ACM Trans. Graph.*, 33(4), July 2014.

[355] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Trans. Graph.*, 24(3):965–972, July 2005.

ACKNOWLEDGMENTS

First and foremost, I extend my sincere gratitude to my supervisors, Jiří Kosinka, Alexandru C. Telea, and Steffen Frey from the University of Groningen, and Xiaojuan Ban from the University of Science and Technology Beijing.

I am profoundly grateful to Jiří for orchestrating my entire study within the SVCG group. Without your steadfast dedication, this thesis would not have been possible. Your passion for detail and insightful ideas have profoundly shaped my research approach.

I deeply appreciate Alexandru's guidance throughout my studies. Your fervor for articulating ideas has inspired me to ambitiously delve deeper into my research. Your feedback has been an invaluable resource, providing new insights with each encounter.

Heartfelt thanks are also due to Steffen for his intelligent mentorship. Your innovative ideas and insightful advice during our weekly meetings have significantly contributed to my progress. Furthermore, your grounded and rigorous approach to scientific research has deeply influenced me, enriching my academic journey beyond the acquisition of knowledge.

I cannot thank Xiaojuan enough for your patience and support, which have provided ample space for me to thrive in my research endeavors, enabling me to perform to the best of my abilities.

Further gratitude is extended to all members of the SVCG group. The support I received from this incredible community has been invaluable. We have consistently supported and assisted each other when needed, and I am incredibly proud to be a member of such a collaborative and encouraging group.

I also wish to express my gratitude to the University of Groningen and the Chinese Scholarship Council for their financial support, which enabled me to conduct my research. Special thanks go to the Faculty of Science and Engineering and the Bernoulli Institute at the University of Groningen for providing the necessary space and equipment. I am also grateful for the continuous support from all staff members.

Lastly, my deepest gratitude goes to my parents, who have supported me unconditionally in my pursuit of research. The credit for this work is shared with you.

SHORT RÉSUMÉ

Yanrui Xu

PERSONAL DATA

Name:	Yanrui Xu
Email:	xuyanruiedw@me.com
Orcid:	0000-0002-2154-1178

SCIENTIFIC EDUCATION

Current | 02/2023	PhD candidate in Computer Science (Joint-PhD program) University of Groningen, Groningen, the Netherlands Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence
09/2020	University of Science and Technology Beijing, Beijing, China School of Intelligence Science and Technology
07/2020 | 09/2018	Master of Engineering in Computer Science and Technology University of Science and Technology Beijing, Beijing, China School of Intelligence Science and Technology
07/2018 | 09/2014	Bachelor of Engineering in Information Security University of Science and Technology Beijing, Beijing, China School of Intelligence Science and Technology

SELECTED PUBLICATIONS

X. Ye, X. Wang, **Y. Xu**, J. Kosinka, A. C. Telea, L. You, J. J. Zhang, and J. Chang. Monte Carlo Vortical Smoothed Particle Hydrodynamics for Simulating Turbulent Flows. *Computer Graphics Forum*, Volume 43, Issue 2. Wiley, 2024. doi: 10.1111/cgf.15024

Y. Xiong, X. Wang, **Y. Xu**, Y. Zhang, J. Chang, J. J. Zhang, and X. Ban. Dual-mechanism surface tension model for SPH-based simulation. *The Visual Computer*, Volume 40, 4765–4776. Springer, 2024. doi: 10.1007/s00371-024-03474-4

X. Wang*, **Y. Xu***, S. Liu, B. Ren, J. Kosinka, A. C. Telea, J. Wang, C. Song, J. Chang, C. Li, J. J. Zhang, and X. Ban. Physics-based fluid simulation in Computer Graphics: Survey, research trends, and challenges. *Computational Visual Media*. Springer, 2024. doi: 10.1007/s41095-023-0368-y

T. Wang*, **Y. Xu***, R. Li, H. Wang, Y. Xiong, and X. Wang. Simulating hyperelastic materials with anisotropic stiffness models in a particle-based framework. *Computers & Graphics*, Volume 116, 437-447. Elsevier, 2023. doi: 10.1016/j.cag.2023.09.007

X. Wang, T. Wang, J. Wang, **Y. Xu**, X. Ban, H. Huang, Z. Zhu, J. Chang, and J. J. Zhang. Implicit smoothed particle hydrodynamics model for simulating incompressible fluid-elastic coupling. *Computer Animation and Virtual Worlds*, Volume34, Issue5. Wiley, 2023. doi: 10.1002/cav.2146

Y. Xu, C. Song, X. Wang, X. Ban, J. Wang, Y. Zhang, and J. Chang. Spatial adaptivity with boundary refinement for smoothed particle hydrodynamics fluid simulation. *Computer Animation and Virtual Worlds*, Volume 34, Issue 5. Wiley, 2023. doi: 10.1002/cav.2136

Y. Xu, X. Wang, J. Wang, C. Song, T. Wang, Y. Zhang, J. Chang, J. J. Zhang, J. Kosinka, A. C. Telea, and X. Ban. An Implicitly Stable Mixture Model for Dynamic Multi-fluid Simulations. In *SIGGRAPH Asia 2023 Conference Paper*. ACM, 2023. doi: 10.1145/3610548.3618215

Y. Xu, Y. Xu, Y. Xiong, D. Yin, X. Ban, X. Wang, J. Chang, and J. J. Zhang. Anisotropic screen space rendering for particle-based fluid simulation. *Computers & Graphics*, Volume 110, 118-124. Elsevier, 2023. doi: 10.1016/j.cag.2022.12.007

S. Liu, X. Wang, X. Ban, **Y. Xu**, J. Zhou, J. Kosinka, and A. C. Telea. Turbulent Details Simulation for SPH Fluids via Vorticity Refinement. *Computer Graphics Forum*, Volume 40, Issue 1. Wiley, 2021. doi: 10.1111/cgf.14095

X. Wang, S. Liu, X. Ban, **Y. Xu**, J. Zhou, and J. Kosinka. Robust turbulence simulation for particle-based fluids using the Rankine vortex model. *The Visual Computer*, Volume 36, 2285–2298. Springer, 2020. doi: 10.1007/s00371-020-01914-5

* Equal contribution

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Final Version as of December 11, 2024 (`classicthesis`).