Interface Handling in Smoothed Particle Hydrodynamics

vorgelegt von

Nadir Akinci

Dissertation zur Erlangung des Doktorgrades
der Technischen Fakultät,
Albert-Ludwigs-Universität Freiburg im Breisgau

May 2014

1. Gutachter	Prof. Dr. Matthias Teschner
2. Gutachter	Prof. Dr. Nils Thuerey
Dekan	Prof. Dr. Yiannos Manoli
Datum der Disputation	14/05/2014

Abstract

Simulating incompressible flow using Particle-based techniques have been gaining interest of researchers for more than a decade. Smoothed Particle Hydrodynamics (SPH) is a common technique for simulating fluids solely using pairwise forces between particles. SPH has important potential benefits, such as the ability to handle complex boundaries and small-scale phenomena. This dissertation explains some techniques that employ those potential benefits of SPH.

The dissertation starts by reviewing the basic SPH technique and the recent advances in SPH. Secondly, a versatile technique for handling boundaries with two-way rigid-fluid coupling is explained. The technique has several important properties, such as; support for arbitrary rigid objects with large density ratios, support for thin shells including non-manifold geometries, momentum conservation, ability to handle multiphase flow, and allowing large time-steps. Furthermore, the technique addresses particle deficiency related issues of SPH near solid boundaries, which prevents spatial and temporal discontinuities of the physical properties of the fluid. Those benefits are illustrated through a variety of simulation scenarios. Afterwards, in the next chapter, the rigid-fluid coupling technique is extended to support elastic solids with arbitrarily large expansions in SPH, while retaining all of its useful properties. It is shown that this extension produces stable and realistic interactions of SPH fluids with both slowly and rapidly deforming solids.

The next contribution explained in the thesis addresses both fluid-air and fluid-solid interfaces in SPH for more realistic fluid behavior by employing a new surface tension force and a new adhesion force. The surface tension force can handle large surface tensions in a realistic way, which lets it handle challenging real scenarios, such as: water crown formation, various types of fluid-solid interactions, and even droplet simulations. Furthermore, it prevents particle clustering at the fluid-air interface where inter-particle pressure forces are incorrect. Our adhesion force allows plausible two-way attraction of fluids and solids and can be used to model different wetting conditions. It is also shown that the combination of two forces allows simulating a variety of interesting effects in a plausible way.

Lastly, the thesis focuses on the efficient simulation and visualization of fluid-air mixtures, namely, foam to enhance detail. For the foam simulation, physically motivated rules are employed to generate, advect and dissipate foam on an existing SPH simulation as a post processing step. The main contribution explained in detail in that part is a technique for the efficient rendering of large-scale foam data in screen space using a GPU based rendering pipeline. The explained approach employs a multi-pass rendering technique to imitate some of the effects that are commonly accomplished by using expensive ray-tracing based methods. It is demonstrated through different scenarios that the presented pipeline is able to produce convincing foam renderings for large-scale scenarios.

Zusammenfassung

Die Simulation inkompressibler Fluide durch partikelbasierte Techniken weckt bereits seit über einem Jahrzehnt das Interesse der Wissenschaft. Eine verbreitete Technik zur Simulation von Fluiden ist Smoothed Particle Hydrodynamics (SPH). Diese basiert ausschließlich auf Partikeln und zwischen diesen definierten paarweisen Kräften. Dadurch ergeben sich einige wichtige potentielle Vorteile, darunter die Möglichkeit, komplexe Grenzfälle und besonders kleine Phänomene zu behandeln. Diese Dissertation befasst sich mit einigen Techniken, die diese potentiellen Vorteile nutzen.

Die Dissertation beginnt damit, die Grundlagen von SPH und die jüngeren Fortschritte bei der Entwicklung der Technik vorzustellen. Weiterhin wird eine nützliche Technik erklärt, mit der sich die Behandlung von Grenzfällen mit einer zwei-Wege-Kopplung zwischen Starrkörpern und Fluiden realisieren lässt. Diese bietet einige Vorteile, darunter die Erhaltung der Bewegungsenergie und die Möglichkeit, große Zeitschritte zuzulassen und Starrkörper mit beliebigen Dichteverteilungen, dünne Hüllen, einschließlich nicht-mannigfaltiger Geometrien, und mehrphasige Strömungen zu behandeln. Weiterhin befasst sich der Ansatz mit dem Problem fehlender SPH-Partikel in der Nähe fester Grenzschichten, um räumlichen und zeitlichen Diskontinuitäten der physikalischen Eigenschaften des Fluides vorzubeugen. Die genannten Vorteile werden anhand verschiedener Simulationsszenarien illustriert. Im darauffolgenden Kapitel wird die behandelte Starrkörper-Fluid-Technik erweitert, so dass sie auch elastische Körper beliebigen Ausmaßes in SPH behandeln kann, ohne ihre nützlichen Eigenschaften einzubüßen. Es wird gezeigt, dass die erläuterte Erweiterung weiterhin stabile und realistische Interaktionen von SPH-Fluiden sowohl mit sich langsam als auch mit sich schnell verformenden Körpern zulässt.

Der nächste in dieser Arbeit erläuterte Beitrag befasst sich sowohl mit den Grenzschichten zwischen Fluiden und Luft als auch zwischen Fluiden und Festkörpern in SPH. Dabei kommt jeweils eine neue Kraft zur Modellierung der Oberflächenspannung und der Adhäsion zur Anwendung. Diese Oberflächenspannungskraft ist in der Lage, große Oberflächenspannungen auf realistische Weise zu behandeln, wodurch auch schwierige reale Szenarien wie Wasserkronen, verschiedene Arten von Fluid-Festkörper-Interaktionen und sogar Tröpfchensimulationen möglich werden. Weiterhin beugt sie dem Zusammenklumpen von Partikeln an der Fluid-Luft-Grenzschicht, wo die Druckkräfte zwischen Partikeln inkorrekt sind, vor. Unsere Adhäsionskräfte erlauben eine plausible zwei-Wege-Anziehung zwischen Fluiden und Festkörpern, womit verschiedene Benetzungsituationen modelliert werden können. Es wird außerdem gezeigt, dass die Kombination zweier Kräfte die plausible Simulation verschiedener interessanter Effekte zulässt.

Zuletzt konzentriert sich diese Doktorarbeit auf die effiziente Simulation und Visualisierung von Fluid-Luft-Gemischen, insbesondere Schaum, um den Detailreichtum der Darstellung zu erhöhen. Für die Simulation von Schaum werden physikalisch motivierte Regeln angewandt, mit denen Schaum durch einen Nachbearbeitungsschritt innerhalb einer existierenden SPH-Simulation generiert, advektiert und aufgelöst werden kann. Der wesentliche Beitrag hierzu ist eine Screen-Space-Technik zum effizienten Rendern von Schaum in großformatigen Szenen unter Ausnutzung einer GPU-basierten Rendering-Pipeline. Der erläuterte Ansatz nutzt eine mehrstufige Rendertechnik, durch die einige Effekte erzielt werden, für die üblicherweise aufwändige Raytracingmethoden eingesetzt werden.

Anhand verschiedener Szenarien wird gezeigt, dass die vorgestellte Pipeline in der Lage ist, eine überzeugende Darstellung von Schaum für großformatige Szenen zu erzeugen.

Contents

Acknowledgments

First of all, I would like to thank my supervisor Matthias Teschner for giving me the opportunity to do my PhD, reading drafts of my works and giving feedback whenever he can. I also thank Nils Thürey for being the second supervisor of my thesis. Then I would like to thank my colleagues Markus Ihmsen, Jens Cornelis, Marc Gissler and Rüdiger Schmedding for their supports in many areas. I should also thank to the secretaries of our group, Cynthia Findlay and Veria Poppa for helping me complete all of the formal paperwork I faced.

I would like to thank my wife and colleague Gizem for her endless support in the times when I face difficulties. She has always been there to help me. I should also thank to our Defne, who definitely came to the world at the right time and she is now the one that makes us smile and keep going.

Finally, I would like to thank to all of our close relatives for their support in our Germany adventure. They never let us feel lonely here.

1 Introduction

Generating animations involving physical phenomena without employing a simulation model can be a daunting task. While animating plausible looking fluids by hand or combined with procedural techniques can be notoriously difficult, mimicking even the relatively simple looking interaction of a cube with a table can take considerable amount of effort. In the last two decades, advances in computing hardware have opened doors to employing physically-based simulation models for animating even the most complex phenomena. Nowadays, such models are routinely being used in films, commercials and even in real-time applications such as various types of virtual reality simulators and video games. The most obvious difference between simulating the interaction of several solid bodies compared to fluids is the huge difference in the numbers of degrees of freedom, which makes simulating fluids much more expensive. Furthermore, as fluid flow is only roughly predictable, it is usually required to repeat a simulation several times until the desired results are achieved. Therefore, fluid simulations can become one of the most time consuming parts of an animation production process. The choice of the employed simulation techniques also play a vital role in the rapid convergence to the desired fluid motion, which are summarized in the next section.

1.1 Liquid Animation in Computer Graphics

In the early days of liquid animation, most of the research focused on generating simplified equations that mimic the fluid flow, instead of numerically solving the Navier-Stokes equations. Examples include the procedural techniques of Max [Max81] and Peachy [Pea86]. Kass and Miller [KM90] proposed an approximation to the shallow water equations by solving a wave equation on a heightfield. The approach is improved in the work of Chen and Lobo [CL95] by solving 2D Navier-Stokes equations where the third fluid dimension is modeled with a heightfield. The obvious limitation of these works is the fact that, interesting phenomena such as breaking waves, sprays, and splashes cannot be directly captured. Foster and Metaxas [FM96] were the first to solve the full 3D Navier-Stokes equations to animate fluids on a stationary grid. Those works which solve the Navier-Stokes equations at fixed locations on a stationary grid are usually referred to as Eulerian (or grid based) techniques. Later, Stam [Sta99] improved the technique by introducing semi-Lagrangian method for the convection term and implicit solver for both the viscosity and pressure terms, that allowed achieving

unconditional stability. The improvements introduced by Stam forms the basis for most of the subsequent Eulerian techniques. Foster and Fedkiw [FF01] proposed the particle level set scheme, which allows tracking liquid interfaces, thereby allowing plausible looking animations of liquids. Other notable works that improve Eulerian techniques include; the Fluid Implicit Particle method employed in the work of Zhu and Bridson [ZB05], adaptive techniques that allow efficient simulations in very large domains [LGF04, ZLC+13, EQYF13, ATW13], techniques that allow large time steps [KSGF09, LCPF12] and techniques that allow tracking detailed fluid features [TWGT10, ATT12, KTT13], and fluid-solid coupling techniques [BBB07, RMSG+08].

Another popular way to animate liquids is by using Lagrangian approaches. Smoothed Particle Hydrodynamics (SPH) is a commonly used Lagrangian fluid simulation method, which was popularized by the work of Mueller et al. [MCG03], based on the work of Monaghan [Mon94]. The main difference of SPH to Eulerian techniques is that fluid equations of motion are directly solved on the particles. This results in several important benefits over Eulerian techniques. First of all, its purely particle-based nature allows simulating small-scale phenomena e.g. splashes and droplets, and handling complex boundaries. Secondly, since each particle represents a macroscopic portion of the fluid, mass conservation is always guaranteed and momentum conservation is less of a problem since advection is handled by the moving particles. However, there exist inherent challenges in Lagrangian approaches. Firstly, neighborhood search is necessary for the computation of fluid quantities at particle positions, which is a costly operation and commonly considered as one of the main drawbacks of SPH. For a thorough discussion of neighborhood search techniques that are commonly employed in SPH along with a space and memory efficient scheme, we refer the interested reader to the work of Ihmsen et al. [IABT11]. Another challenge of SPH is that satisfying incompressibility requires smaller time steps, when an equation of state is used when computing fluid pressure at particle positions. However, recent works have significantly improved the small time-step limitation of SPH; by using a prediction correction for better approximation of correct pressure forces [SP09], by employing a grid based formulation for computing the pressures [RWT11], by satisfying holonomic kinematic constraints on the fluid density [BLS12], more recently by using an implicit formulation when solving for pressure [ICS+13] and by using an iterative scheme to compute particle density [MM13]. Another common approach for improving the performance of particle-based methods is using adaptive particle radius [APKG07, SG11, HS13].

Besides the performance related aspects, many works have addressed boundary-handling (e.g.[DK01, MST+04, HKK07, Mon05, LD08, MK09, IAGT10]), two-way fluid-solid coupling (e.g. [CBP05, MST+04, SSP07, LD08, BTT09, OKR09b, YLHQ12, DTM+12]), fluid surface tension (e.g. [Mor99, MCG03, TM05, CBP05, BT07, YWTY12, MM13]) and fluid-solid adhesion (e.g. [CBP05, SB12, HLW+12]). This thesis discusses the issues with the mentioned previous work and explains some versatile techniques that address many of those limitations in the context of handling fluid-solid and fluid-air interfaces in SPH.

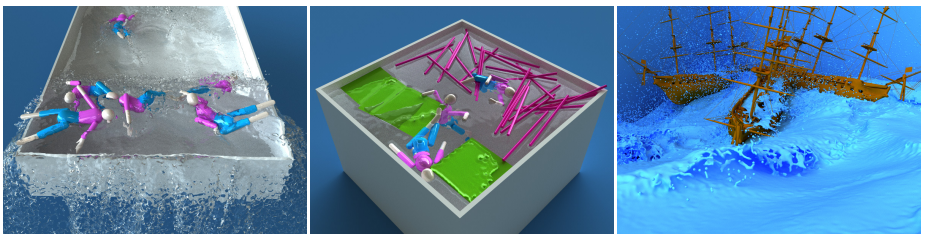


Figure 1.1 – Some scenarios that illustrate the versatility of the explained boundary-handling and rigid-fluid coupling technique. The explained technique is totally based on hydrodynamic forces and can handle the interaction of arbitrary rigid bodies with SPH fluids. Several articulated bodies interact with water (left). Several rigid objects, including lower dimensional objects interact with water (center). Three frigates sailing on wavy water (right). Images are taken from [AIA+12].

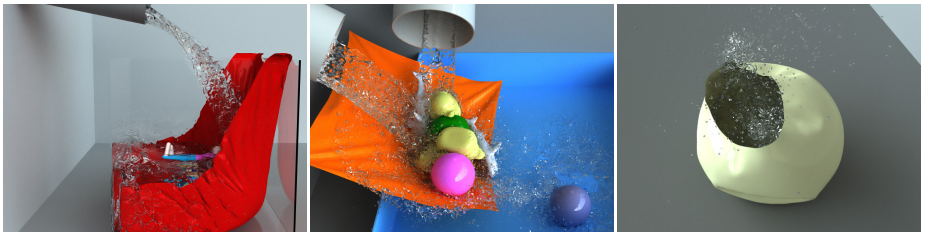


Figure 1.2 – Some scenarios that show the flexibility of the explained elastic-fluid coupling technique. The approach extends the explained two-way rigid-fluid method to make it handle arbitrarily large deformations in an efficient way. An elastic cloth and several rigid objects interact with water (left). Several 3D elastic objects and a cloth interact with water (center). Splash caused by the rapid impact of an elastic bowl filled with water onto the ground (right). Images are taken from [ACAT13].

1.2 Contributions

Rigid-Fluid Two-Way Coupling

The first explained contribution is a momentum-conserving two-way coupling method for SPH fluids and arbitrary rigid objects that is completely based on hydrodynamic forces. The approach samples the surface of rigid bodies with boundary particles that interact with the fluid, preventing deficiency issues and both spatial and temporal discontinuities. The problem of inhomogeneous boundary sampling is addressed by considering the relative contribution of a boundary particle to a physical quantity. This facilitates not only the initialization process but also allows the simulation of multiple dynamic objects. Thin structures consisting of only one layer or one line of boundary particles, and also non-manifold geometries can be handled without any additional treatment. The presented approach is integrated both into different SPH solvers, and its stability and flexibility is demonstrated with several scenarios including multiphase flow. Some examples about this contribution can be seen in Figure 1.1

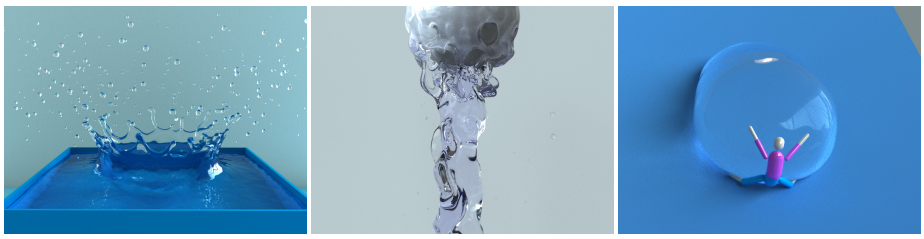


Figure 1.3 – Some experiments that illustrate the possible scenarios that can be created by using the explained surface tension and fluid-solid adhesion techniques. A water-crown emerges as a result of the impact of a water droplet onto a filled water container (left). A water stream realistically flows over a sphere (center). A water droplet (together with a ragdoll stuck onto it) rolls on an inclined plane. Images are taken from [AAT13].

Deformable-Fluid Two-Way Coupling

Building upon the foundation of the mentioned rigid-fluid coupling technique, the next explained contribution is a simple and elegant method for handling elastic solids in SPH fluids. Similarly, the approach samples triangulated surfaces of solids using boundary particles. However, to prevent fluid particle tunneling in case of large expansions, additional boundary particles are adaptively generated to prevent gaps and undesired leakage. Furthermore, as an object compresses, particles are adaptively removed to avoid unnecessary computations. It is demonstrated that the explained approach produces plausible interactions of SPH fluids with both slowly and rapidly deforming solids (see Figure 1.2).

Surface Tension and Two-Way Coupled Fluid-Solid Adhesion

Realistic handling of fluid-air and fluid-solid interfaces in SPH is a challenging problem. The main reason is that some important physical phenomena such as surface tension and adhesion emerge because of inter-molecular forces in a microscopic scale. This is different from scalar fields such as fluid pressure, which can be plausibly evaluated on a macroscopic scale using particles. Although there exist techniques to address this problem for some specific simulation scenarios, there does not yet exist a general approach to reproduce the variety of effects that emerge in reality from fluid-air and fluid-solid interactions. In order to address this problem, a new surface tension force, and a new adhesion force is presented. Different from the existing work, the surface tension force can handle large surface tensions in a realistic way. This property lets the approach handle challenging real scenarios, such as water crown formation, various types of fluid-solid interactions, and even droplet simulations. Furthermore, it prevents particle clustering at the free surface where inter-particle pressure forces are incorrect. The adhesion force allows plausible two-way attraction of fluids and solids and can be used to model different wetting conditions. By using the explained forces, modeling surface tension and adhesion effects do not require involved techniques such as generating a ghost air phase or surface tracking. The forces are applied to the neighboring fluid-fluid and fluid-boundary particle pairs in a symmetric way, which satisfies momentum conservation. Furthermore,

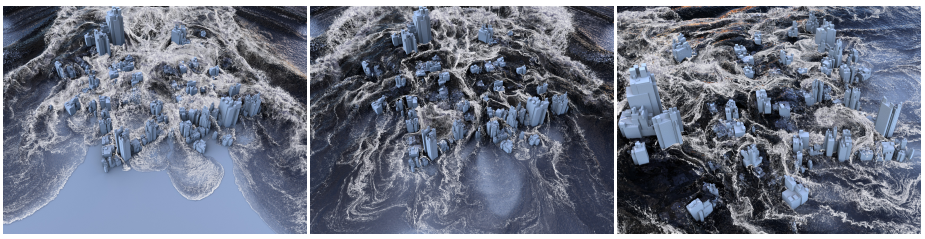


Figure 1.4 – A city is flooded with seawater. The foam is generated by post-processing the SPH particle data, and the generated foam is efficiently rendered using the explained multi-pass technique on the GPU. The foam rendering per frame took approximately 1 second in 1K resolution, where around 29 million foam particles were rendered on average.

it is demonstrated that combining both forces allows simulating a variety of interesting effects in a plausible way (see Figure 1.3).

Efficient Foam Simulation and Rendering

After a brief discussion of foam generation techniques that is tailored to particle based fluid data, a method for the efficient rendering of large scale particle-based foam data in screen space using a GPU based rendering pipeline is presented. The approach employs a multi-pass rendering technique to imitate some of the effects that are commonly accomplished by using expensive ray-tracing based methods. It is demonstrated through different scenarios that the explained pipeline is able to produce convincing foam renderings for large-scale scenarios and it has a significant performance advantage compared to using ray-casting techniques for rendering such particle data.

1.3 Publications

This thesis focuses on the following publications that were published in peer-reviewed journals:

[AIA+12] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, M. Teschner. "Versatile Rigid-Fluid Coupling for Incompressible SPH", *ACM Transactions on Graphics (Proc. SIGGRAPH 2012)*, vol. 31, no. 4, pp. 62:1-62:8, July 2012.

[ACAT12] N. Akinci, J. Cornelis, G. Akinci, M. Teschner. "Coupling Elastic Solids with SPH Fluids", *Journal of Computer Animation and Virtual Worlds (Proc. CASA 2013)*, 24: 195–203. doi: 10.1002/cav.1499

[AAT13] N. Akinci, G. Akinci, M. Teschner. "Versatile Surface Tension and Adhesion for SPH Fluids", *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2013)*, vol. 32, no. 6, pp. 182:1-182:8, November 2013.

[ADAT13] N. Akinci, A. Dippel, G. Akinci, M. Teschner. "Screen Space Foam Rendering", *Journal of WSCG*, Vol.21, No.03, pp.173-182, 2013

The author also contributed to the following publications, where some of these works are also briefly covered in this thesis:

[IAAT12] M. Ihmsen, N. Akinci, G. Akinci, M. Teschner. "Unified Spray, Foam and Bubbles for Particle-based Fluids", *The Visual Computer (Proc. CGI 2012)*, Volume 28, Issue 6-8, pp 669-677, 2012, doi: 10.1007/s00371-012-0697-9

[AIAT12] G. Akinci, M. Ihmsen, N. Akinci, M. Teschner. "Parallel Surface Reconstruction for Particle-based Fluids", *Computer Graphics Forum*, vol. 31, no. 6, pp. 1797-1809, 2012, doi: 10.1111/j.1467-8659.2012.02096.x. *(Presented at Eurographics 2013)*

[IABT11] M. Ihmsen, N. Akinci, M. Becker, M. Teschner. "A Parallel SPH Implementation on Multi-core CPUs", *Computer Graphics Forum*, vol. 30, no. 1, pp. 99-112, 2011, doi: 10.1111/j.1467-8659.2010.01832.

[AAIT12] G. Akinci, N. Akinci, M. Ihmsen, M. Teschner. "An Efficient Surface Reconstruction Pipeline for Particle-Based Fluids", *Proc. VRIPHYS, Darmstadt, Germany,* pp. 61-68, Dec. 6-7, 2012.

[AAIT13] G. Akinci, N. Akinci, E. Oswald, M. Teschner. "Adaptive Surface Reconstruction for SPH Using 3-level Uniform Grids", *WSCG proceedings,* pp.195-204, 2013

[IAGT10] M. Ihmsen, N. Akinci, M. Gissler, M. Teschner. "Boundary Handling and Adaptive Time-stepping for PCISPH", *Proc. VRIPHYS, Copenhagen, Denmark*, pp. 79-88, Nov 11-12, 2010.

1.4 Overview

This thesis is organized as follows: Chapter 2 provides an introduction to the basic SPH technique and provides brief discussions about the recent advances in simulating fluids using SPH. Afterwards, each of the four subsequent chapters firstly thoroughly discuss the related work, then provide detailed explanation of the concept, and present results, discussion and finally gives directions for future work. Chapter 3 thoroughly discusses the versatile boundary-handling and rigid-fluid coupling technique explained in the paper [AIA+12]. Chapter 4 explains the follow-up work [ACAT13], which extends the rigid-fluid coupling approach to elastic-fluid coupling. Chapter 5 discusses the novel surface tension and fluid-solid adhesion technique, which was explained in the paper [AAT13]. Afterwards, Chapter 6 briefly discusses how fluid-air mixtures (i.e. foam) can be efficiently simulated and provides a thorough discussion about how the simulated foam data can be efficiently rendered using the technique explained in [ADAT13]. Finally, Chapter 7 again summarizes the contributions and concludes the thesis.

2

Smoothed Particle Hydrodynamics

This chapter briefly discusses the basic SPH technique applied to solving incompressible flow of Newtonian fluids. The topics covered in this chapter include; particle neighborhood search strategies, computation of basic SPH field variables that are required when solving the equation of motion of fluids (i.e., density, pressure, pressure force, and viscosity force), vorticity confinement techniques, multi-resolution fluid simulation approaches, and finally surface generation strategies. Discussions about; boundary-handling and fluid-solid and fluid-deformable coupling, handling fluid-air interfaces, modeling surface tension and fluid-solid adhesion are deferred to the later chapters; as those topics are covered by the main contributions of this thesis.

2.1 Basic Concept

SPH was originally designed for solving astrophysical problems by Gingold and Monaghan [GM77], and Lucy [Luc77], which is a mesh-free Lagrangian fluid simulation technique. Mesh-free means that it does not require a stationary grid when solving fluid equations of motion, which is in contrast to Eulerian techniques (see Figure 2.1). SPH works by obtaining approximate numerical solutions of the equations of fluid dynamics by representing the fluid with particles, where the physical properties and equations of motion of these particles are based on the continuum equations of fluid dynamics. Further, physical quantities are estimated by interpolating existing fluid quantities using the neighboring particles. In SPH, the integral representation of a field variable A at location $\mathbf{x}_i$

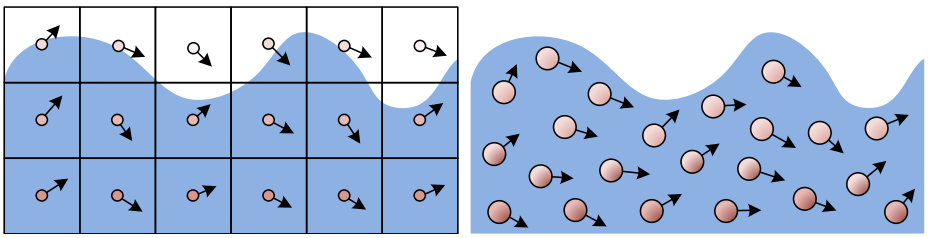


Figure 2.1 – In Eulerian techniques, fluid quantities such as pressure and velocity are measured at fixed locations (left). In Lagrangian approaches such as SPH, those quantities are carried by the moving particles. In the above figures, colors represent pressures and arrows represent velocities for the marked locations.

in domain Ω is defined as

$$A(\mathbf{x}_i) = \int_{\Omega} A(\mathbf{x}_j) W(\mathbf{x}_i - \mathbf{x}_j, h) \, d\mathbf{x}_j, \tag{2.1}$$

where W is a kernel (or weighting) function with influence radius (or smoothing length) h. W basically acts as the weighting factor for the contributions from the neighborhood interpolation points denoted by $\mathbf{x}_j$, where $d\mathbf{x}_j$ denotes the differential volumes represented at each $\mathbf{x}_j$. So as to make (2.1) numerically solvable, the integral in (2.1) can be written by using a finite set of interpolation points by replacing the integral by a summation, and the differential volume element $d\mathbf{x}_j$ by the volume V_j (which is mass m_j divided by density ρ_j) as

$$A_i = \sum_j V_j A_j W_{ij} = \sum_j \frac{m_j}{\rho_j} A_j W_{ij}. \tag{2.2}$$

In the remainder of this thesis, we will use the shorthand W_{ij} for $W(\mathbf{x}_i - \mathbf{x}_j, h)$ and, e.g., A_i for $A(\mathbf{x}_i)$ to make the equations easier to read. When required, the derivative of a field quantity can also be easy computed by simply taking the derivative of the kernel function, e.g.:

$$\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W_{ij}. \tag{2.3}$$

The SPH approximation is illustrated in Figure 2.2. For a detailed explanation and derivation of the above basic SPH equations, we refer the interested reader to the comprehensive annual review of Monaghan [Mon05].

2.2 Smoothing Kernels

Early works in SPH (e.g. [GM77]) used the Gaussian function as the kernel function, which is defined as

$$W_{\text{Gauss}}(r, h) = \sigma^D_{\text{Gauss}} e^{-\frac{r^2}{h^2}} \tag{2.4}$$

where r is the distance to a neighboring interpolation point and σ^D_{Gauss} (the dimensionality factor) is $1/(\pi^{3/2}h^3)$ in 3D. Although 2.4 might be a good choice for SPH approximations, the exponentiation makes it very expensive to compute. Therefore, more efficient functions that mimic the behavior of Gaussian have been created for SPH simulations. Undoubtedly, the most famous of those functions is the ubiquitous cubic B-spline, which was presented by Monaghan and Lattanzio in [ML85]. The kernel can be written as:

$$W_{\text{CSpline}}(r, h) = \sigma^D_{\text{CSpline}} \begin{cases} \frac{6r^3}{h^3} - \frac{6r^2}{h^2} + 1 & 0 \le r \le \frac{h}{2} \\ 2\left(1 - \frac{r}{h}\right)^3 & \frac{h}{2} < r \le h \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where $\sigma^D_{CSpline}$ is a dimensional factor to normalize the kernel for different spatial dimensions and is defined as $3/2\pi h^3$ in 3D. Other Gaussian-like kernels include the Quintic kernel [Wen95], which is defined as

$$W_{\text{Quintic}}(r, h) = \sigma^D_{\text{Quintic}} \begin{cases} \left(1 - \frac{r}{h}\right)^4 \left(\frac{4r}{h} + 1\right) & 0 \le r \le h \\ 0 & \text{otherwise} \end{cases}$$

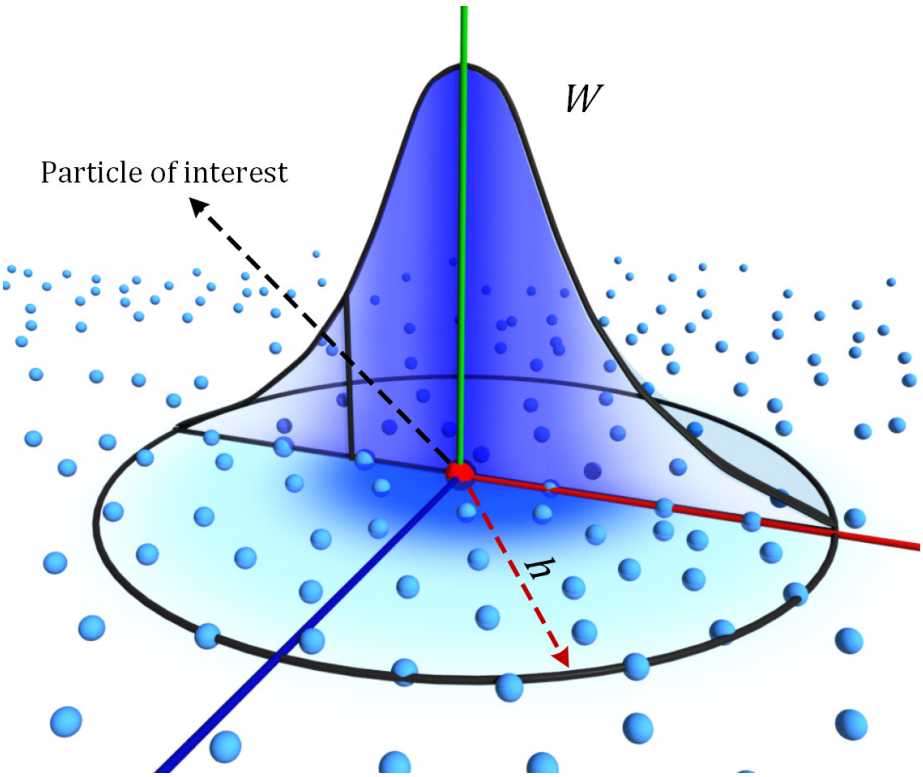


Figure 2.2 – Illustration of SPH approximation for a field variable for the red particle, where W denotes a Gaussian-like interpolation function (a.k.a. SPH kernel), h is the influence radius (a.k.a. smoothing length). Such an SPH kernel is usually used for computing particle density, where contributions from neighboring particles decrease with increasing distance.

where $\sigma^D_{Quintic} = 7/\left(8\pi h^3\right)$ in 3D; and the 6th degree polynomial used in [MCG03], which is defined as

$$W_{\text{Poly6}}\left(r,h\right) = \sigma^D_{\text{Poly6}} \begin{cases} \left(h^2 - r^2\right)^3 & 0 \leq r \leq h \\ 0 & \text{otherwise} \end{cases}$$

where $\sigma^D_{Poly6} = 315/64\pi h^9$ in 3D. How well the kernels approximate the Gaussian kernel is shown in Figure 2.3.

Furthermore, different kernel functions were also used for different applications. For instance, Johnson et al. [JSB96] used a quadratic kernel function to study the high velocity impact problem to prevent particle clustering in compressing areas. Mueller et al. [MCG03] used different kernel functions (that are quite efficient to compute) when computing fluid density, pressure force and viscosity force for real-time applications.

Unless stated otherwise, the cubic spline kernel function and its gradient is used to compute the basic SPH field variables in the presented experiments.

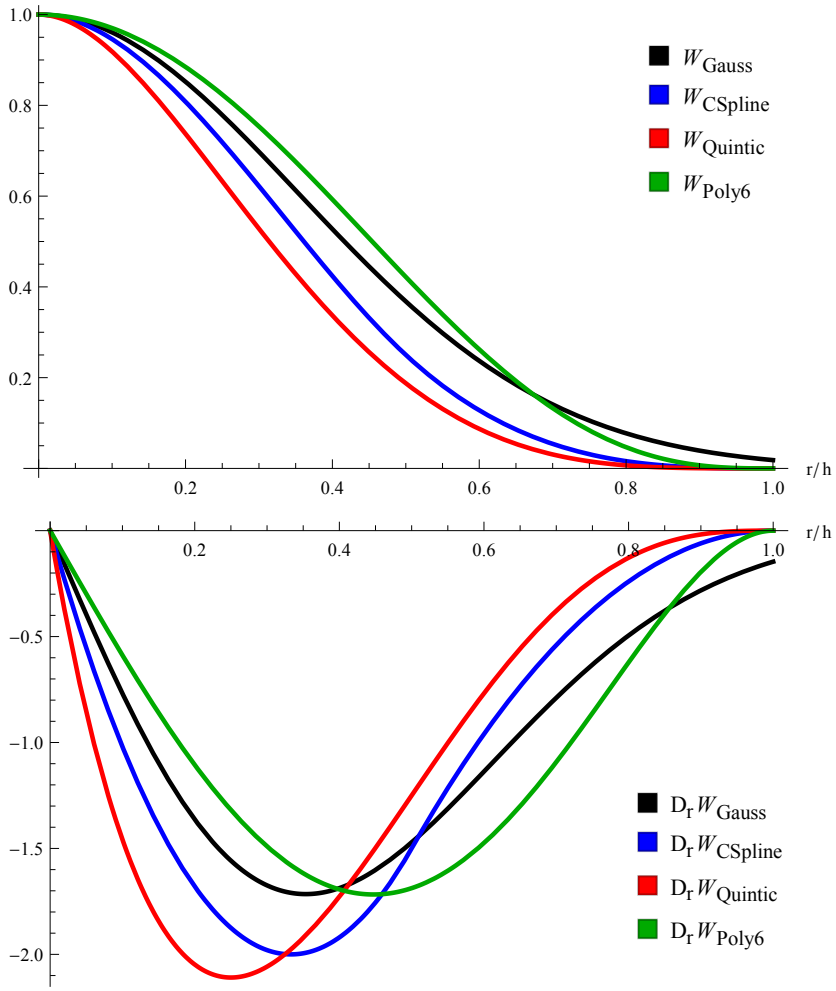


Figure 2.3 – Commonly used SPH kernels functions (top), and their gradients (bottom).

2.3 Neighborhood Search

SPH requires neighboring sample points to evaluate the approximations of field variables. As those sample points are moving with the fluid, they may change their positions in each simulation step. The most basic neighborhood determination strategy in SPH is using a regular voxel grid with the cell size equivalent to the smoothing length h of the SPH simulation. The grid is usually resized to enclose all the particles in the simulation domain for a given simulation step. In a 3D simulation, up to 27 cells are queried for neighborhood. However, such a strategy has several issues. First of all, memory consumption scales with the AABB (axis aligned bounding box) of the scene, as number of voxels increase proportionally. Although such a data structure seems efficient in computation time at a first glance (e.g. has $O(1)$ access time), for large simulations the memory coherence of the simulated data significantly reduces (therefore the

cache-hit rates), which causes the simulation data to be repeatedly transferred between memory and CPU cache. Because of these important reasons, more sophisticated data structures are commonly preferred for SPH.

2.3.1 More Sophisticated Data Structures

Adams et al. [APKG07] used a kd-tree to search the neighbors in a multi-resolution fluid simulator. In [HKK07], an SPH implementation that runs entirely on the GPU is presented. The method maps a 3D uniform grid onto a 2D texture, where particle indices are stored in RGBA channels. In [OD08], a scheme called index sorting is used to improve the memory coherence of the simulated data. Index sorting first sorts the particles with respect to their cell indices, then indices of the sorted array are stored in each cell. Each cell stores only one reference to the first particle with the corresponding cell index. The same approach is also used in NVIDIA's GPU based SPH solver [Gre08]. Another popular technique to find neighboring SPH particles is the Verlet list method [Ver67, Hie07]. In this approach, a list of potentially neighboring particles is stored for each particle. Potential neighbors are determined based on a threshold distance s, which is commonly chosen much larger than the smoothing length h. The list of potential neighbors is updated only if a particle has moved more than $s - h$. In [KW06], particles are sorted in multiple staggered grids that are created for each simulation dimension. Therefore, the approach does not need to query spatially close cells, but processes each dimension one after another. However, the approach does not scale well for increasing particle numbers.

Compact Hashing Technique

Another more recent strategy to search neighboring particles in shared memory parallel SPH implementations is introduced in [IABT11]. In that work, the authors propose an efficient spatial neighborhood query structure. The approach maps the spatial locality of the particles onto memory by using a Z-curve similar to [WS95]. The work also analyzes and compares basic voxel grids, spatial hashing, and index sorting. The proposed data structures have also been used in GPU-based simulators in [MM13, OK12]. The approach improves upon the spatial hashing procedure explained in [THM+03]. In that approach, the simulation domain is mapped to a finite list, where a hash function that maps a position $\mathbf{x} = (x, y, z)$ to a hash table of size m has the form

$$H(x, y, z) = \left[\left(\frac{x}{d} \cdot p_1\right) \oplus \left(\frac{y}{d} \cdot p_2\right) \oplus \left(\frac{z}{d} \cdot p_3\right)\right] \mathrm{mod} M,$$

where $p1$, $p2$, $p3$ are large prime numbers which are chosen as 73856093, 19349663 and 83492791 respectively in [THM+03]. In [IABT11], spatial hashing is further improved by keeping a compact list of non-empty cells, where the hash cells just store an index to their related used cell. Consequently, the approach has constant memory footprint for the hash table and extra memory only for the used cells. Therefore, the memory consumption scales with the number of particles, but not with the number of cells; which was the case in the basic voxel grid data structure. As another optimization, the complete list of particles is inserted to the data structure only once. In the later simulation steps, only the particles whose cell coordinates change get re-inserted. As insertions cannot be done in parallel

because of race conditions, such an optimization saves significant amount of time, as the amount of particles that change their cell coordinates remains around 2% on average. After the insertion, the particles can be queried in parallel, where only the list of used cells is processed. The overhead caused by hash collisions (where two cells with different coordinates are mapped to the same cell in the hash table) is also low in the approach, as a hash-collision flag is stored in each cell where the hash indices do not need to be computed for cells without hash collisions. As the memory consumption of the data structure scales proportional to the number of particles, but not with the hash table size, the hash table can be set to a large size to enforce a low number of hash collisions. The authors show that having a hash table size that is two times large than the number of particles is sufficient to keep the hash collisions around 2%. As the hash function in the original spatial hashing scheme is designed to map an arbitrarily large simulation domain to a small array, the data is always scattered and spatial locality is lost. This results in low cache hit rates and increased memory transfers. The authors address this issue by sorting the particles according to a Z-curve once in several simulation steps, and then rebuild the compact used-cell list.

For the experiments presented in this thesis, we used the compact hashing scheme, as the approach allows handling arbitrarily large simulation domains in an efficient way, both in terms of computation time and memory consumption.

2.4 Approximating Fluid Equations of Motion with SPH

Navier-Stokes equations are a set of partial differential equations that describe the motion of fluids, which are named after Claude-Louis Navier and George Gabriel Stokes. They are used to model the behavior of various types phenomena whose motion resembles fluids, including: Liquid and gas flow around different objects like cars, ships and aircrafts; motion of ocean currents, weather and even galaxies. When considering the incompressible flow of a Newtonian[1] fluid, the equation can be written in vector form as

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{f}. \tag{2.6}$$

where $\mathbf{v}$ is the flow velocity, $\frac{D\mathbf{v}}{Dt} = \frac{\delta \mathbf{v}}{\delta t} + \mathbf{v} \cdot \nabla \mathbf{v}$, is called convective derivative[2],ρ is the fluid density, p is the pressure, μ is the viscosity coefficient and $\mathbf{f}$ represents body forces acting on the fluid per unit volume. When looking from Lagrangian viewpoint, where the quantities move with the fluid, the convective term in the convective derivative vanishes, which means $\frac{D\mathbf{v}}{Dt} = \frac{d\mathbf{v}}{dt}$, where $\frac{d\mathbf{x}}{dt} = \mathbf{v}$. Finally, for incompressible fluids from Lagrangian perspective, (2.6) becomes

$$\rho \frac{d\mathbf{v}}{dt} = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{f}. \tag{2.7}$$

[1] A fluid is referred to as Newtonian if its viscosity does not change depending on the applied stress or flow velocity. Water is usually considered Newtonian.

[2] Also known as, e.g., “advective derivative” and “Stokes derivative”.

Multiplying both sides of (2.7) with volume V of an infinitesimal fluid particle where the equation is expected to hold, the equation becomes

$$m\frac{d\mathbf{v}}{dt} = \overbrace{-V\nabla p}^{F_{pressure}} + \overbrace{V\mu\nabla^2\mathbf{v}}^{F_{viscosity}} + \overbrace{V\mathbf{f}}^{F_{external}}, \tag{2.8}$$

where m is the mass of an infinitesimal fluid particle inside the fluid. SPH allows the forces on the right hand side to be evaluated, which makes (2.8) easily solvable by using simple numerical ordinary differential equation integration schemes, such as; Euler-Cromer, Verlet or Leap-Frog.

For incompressible flow, the volume conservation equation $\nabla \cdot \mathbf{v} = 0$ is satisfied using the pressure forces in (2.8).

2.5 Density

Fluid density is undoubtedly the most important field variable of SPH simulations, since the pressure force arises as a result of the changes in the fluid density. A well-known way to compute fluid density in SPH is using the summation density approach [Mon05]. It can be easily derived from the basic SPH scheme by substituting fluid density ρ as the field variable A into (2.2), which results in

$$\rho_i = \sum_j m_j W_{ij}. \tag{2.9}$$

The most important issue with the summation density technique is that it results in underestimated density values near fluid interfaces. As will be seen in the incoming sections, reconstruction the density field as correctly as possible is very crucial in SPH, since pressure force that is to satisfy incompressibility solely relies on the density field.

Another way to update density is to use the mass continuity equation:

$$\frac{D\rho}{Dt} + \rho(\nabla \cdot \mathbf{v}) = 0 \tag{2.10}$$

as basis. Expanding the convective derivative and leaving the time rate of change of density alone results in

$$\frac{d\rho}{dt} = -\nabla \cdot (\rho\mathbf{v}) + \mathbf{v} \cdot \nabla\rho.$$

Finally, approximating $\frac{d\rho}{dt}$ using SPH yields

$$\frac{d\rho_i}{dt} = \sum_j m_j(\mathbf{v}_i - \mathbf{v}_j) \cdot \nabla W_{ij}, \tag{2.11}$$

where the rate of change of density is computed based on the relative motion of the particles [Mon92]. (2.11) has computational advantage over (2.9), as all field variables that are necessary to solve fluid equations of motion can be computed in a single loop over the particles. Although (2.11) looks as if it is unaffected by the underestimated densities near fluid interfaces, it has different issues. First of all, the accumulation of numerical errors, and the errors caused by time integration

schemes cause stability issues. The common practice to alleviate these problems is to reinitialize the particles' densities time to time using (2.9), and using a density correction strategy. One of the most common correction strategy is to use Shepard filter [She68] as done in [Pan04]. Another strategy is to use Moving Least Squares (MLS) technique as used in [CL03, Pan04].

Independent of the chosen density computation approach, density underestimation at fluid interfaces is still an important issue, which gives rise to unphysical negative pressures and subsequent particle clustering. The clumping of SPH particles is unphysical, as the repulsive forces between real fluid molecules do not allow such a behavior. One commonly used way to prevent tensile instability in SPH fluids is using artificial pressure forces [Mon00], which are used in [MM13]. Even using the artificial pressure forces, the pressure gradient of the fluid that is in contact with a solid still cannot be reconstructed correctly. To alleviate those issues, [DK01, KAG+05, SSP07, IAGT10, AIA+12, SB12] use particles to represent boundaries, where these particles also contribute when computing fluid density and other field variables.

In this thesis, the summation density approach (2.9) is used, as it conserves mass exactly. Furthermore, the underestimated densities near fluid interfaces and tensile instability issues are handled by using different strategies for fluid-solid and fluid-air interfaces, which are covered in Chapters 3 and 5 respectively.

2.6 Pressure Force

Since the introduction of SPH, many ways to compute pressure of incompressible fluids have been proposed. This section will go through some of those works.

2.6.1 State-Equation-Based SPH

One of the most common and easiest to implement way to compute the pressure of a fluid particle is to use an equation of state to directly relate pressure to density in each simulation step. The most commonly used equation of state has been popularized in [Mon92], which is defined as

$$p_i = \frac{\rho_0 c_s^2}{\gamma}\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right], \tag{2.12}$$

where c_s is the stiffness constant that determines the speed of the caustic waves that are responsible for the numerical propagation of pressure, ρ_0 is the rest density of the fluid and γ is called the polytropic constant which is chosen as $\gamma \geq 1$. Higher c_s and γ values result in faster numerical propagation, and consequently less compressible fluids. However, lower compressibility always means higher computation time when using an equation of state, as smaller time steps have to be used for the simulations to remain stable. In [MCG03], ideal gas equation is used to compute fluid pressure for real-time applications, where some degree of visible compressibility is allowed in favor of higher performance. Ideal gas equation can be easily derived by substituting $\gamma = 1$ into (2.12).

After computing per-particle pressures, the pressure force acting on a particle can be computed using SPH. Approximating the pressure term $-\nabla p$ in 2.7 yields

$$-\nabla p_i = -\sum_j m_j \frac{p_j}{\rho_j} \nabla W_{ij}.$$

Assuming non-viscous fluids and excluding external forces, (2.7) simplifies to Euler equation

$$\frac{d\mathbf{v}}{dt} = -\frac{\nabla p}{\rho}, \tag{2.13}$$

which can be used to write the pressure force on particle i caused by the neighboring particles as

$$\mathbf{F}_i^p = -m_i \frac{1}{\rho_i} \sum_j m_j \frac{p_j}{\rho_j} \nabla W_{ij}. \tag{2.14}$$

An important issue with (2.14) is that it does not conserve momentum, since the pairwise forces between the particles can be asymmetric (i.e. the force from particle i to j is not necessarily equivalent in magnitude to the force from j to i). Fortunately, a symmetrized form can be easily derived by expanding (2.13) as

$$\frac{d\mathbf{v}}{dt} = -\left[\nabla\left(\frac{p}{\rho}\right) + \frac{p}{\rho^2}\nabla\rho\right], \tag{2.15}$$

and applying SPH approximation to (2.15) yields a symmetrized version of (2.14) as

$$\mathbf{F}_i^p = -m_i \sum_j m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2}\right) \nabla W_{ij}. \tag{2.16}$$

Another way to symmetrize the pressure force is explained in [MCG03], where the arithmetic mean of the pressures of the neighboring particles is used as

$$\mathbf{F}_i^p = -m_i \sum_j m_j \left(\frac{p_j + p_i}{2\rho_j}\right) \nabla W_{ij}. \tag{2.17}$$

The state-equation-based SPH (SESPH) algorithm is presented in 2.1. In this thesis, (2.16) is used to compute pressure forces. However, to compute particle pressures, different techniques are used, which are explained next.

2.6.2 Incompressible SPH

For the complete elimination of acoustic waves and compressibility artifacts, some authors proposed to satisfy complete fluid incompressibility. In the works of Cummins and Rudman [CR99], and Premoze et al. [PTB+03] , incompressibility is achieved by computing a divergence-free velocity field. Although these approaches allow larger time steps, a pressure Poisson equation needs to be solved in each simulation step, which makes the performance of the simulations scale inferior than SESPH for increasing number of particles. More recently, some authors proposed to avoid the pressure Poisson solve by using an auxiliary grid to compute pressures [LKO05, LTKF08, RWT11] as in the Eulerian techniques. Afterwards, the pressure values on the grid are projected to the particles, which significantly improves the convergence rate to incompressibility, when compared to SESPH techniques.

Another commonly used way to compute pressure is by using predictor-corrector schemes. In the work of Solenthaler and Pajarola [SP09], density fluctuations are propagated through the fluid and pressure values are updated

Algorithmus 2.1 State-Equation-Based SPH

```
1: while animating do
2:    foreach fluid-particle i do
3:       find neighbors
4:    end foreach
5:    foreach fluid-particle i do
6:       compute density ρ_i(t) = Σ_j m_j W_ij
7:       compute pressure p_i(t) = (ρ_0 c_s^2 / γ) ((ρ_i / ρ_0)^γ − 1)
8:    end foreach
9:    foreach fluid-particle i do
10:      compute and add up all forces
11:   end foreach
12:   foreach fluid-particle i do
13:      compute new velocity v_i(t + Δt)
14:      compute new position x_i(t + Δt)
15:   end foreach
16: end while
```

until the desired compression rate is satisfied. The method uses a convergence loop that is executed in each simulation step, which consists of a prediction and correction. The technique is named Predictive-Corrective Incompressible SPH (PCISPH). More recently, an approach that relies on a similar prediction-correction scheme is presented in the work of He et al. [HLL+12] named Local Poisson SPH (LPSPH). The basic concept of the approach is converting the pressure Poisson equation into an integral form, and then applying the SPH technique to convert the integral into a summation over neighboring fluid particles. The authors show that LPSPH has better convergence than PCISPH.

Another way to satisfy incompressibility in SPH is to derive the incompressibility condition from constraint dynamics. In the work of Ellero et al. [ESE07], a kinematic constraint is used, which enforces that the volumes of the fluid particles remain constant. The authors use Lagrangian multipliers to enforce the constraints. A similar approach is followed in the work of Bodin et al. [BLS12]. In the work of Macklin and Mueller [MM13], an approach with similar types of constraints is demonstrated for real-time scenarios, where the solver is running completely on the GPU.

More recently, Ihmsen et al. [ICS+13] proposed a technique that uses an SPH approximation of the continuity equation to obtain a discretized form of pressure Poisson equation. The authors call the technique Implicit Incompressible SPH (IISPH). Different from the previous projection-based techniques, IISPH also considers the actual computation of the pressure force, which improves the convergence rate of the solver. Additionally, the density deviation in IISPH is computed based on particle velocities instead of positions, which improves the robustness of the solver.

In most of the experiments presented in this thesis, the pressures of fluid particles are computed using PCISPH. A brief discussion of PCISPH is presented in Section 2.6.2.1. Furthermore, a brief discussion of IISPH is also included in the thesis in Section 2.6.2.2, as the presented techniques are also confirmed to work with that pressure solver.

Algorithmus 2.2 PCISPH

while *animating* **do**
  foreach *fluid-particle* i **do**
    find neighbors
  end foreach
  foreach *fluid-particle* i **do**
    compute and add up forces other than the pressure force
    set pressure $p_i(t) = 0$ and pressure force $\mathbf{F}_i^p(t) = \mathbf{0}$
  end foreach
  $k = 0$;
  while $\rho^*_{err_i}(t + \Delta t) > \eta$ **or** $k > minIterations$ **do**
    foreach *fluid-particle* i **do**
      predict velocity $\mathbf{v}_i^*(t + \Delta t)$
      predict position $\mathbf{x}_i^*(t + \Delta t)$
    end foreach
    foreach *fluid-particle* i **do**
      update distances to neighbors
      predict density $\rho_i^*(t + \Delta t)$
      predict density variation $\rho^*_{err_i}(t + \Delta t)$
      update pressure $p_i(t) += \delta\rho^*_{err_i}(t + \Delta t)$
    end foreach
    foreach *fluid-particle* i **do**
      compute pressure force $\mathbf{F}_i^p(t)$
    end foreach
    $k += 1$;
  end while
  foreach *fluid-particle* i **do**
    compute new velocity $\mathbf{v}_i(t + \Delta t)$
    compute new position $\mathbf{x}_i(t + \Delta t)$
  end foreach
end while

2.6.2.1 Predictive-Corrective Incompressible SPH

In PCISPH, density fluctuations are predicted and corrected using pressure forces. The algorithm iteratively computes the pressure $p_i(t)$ for each particle such that the predicted density fluctuation $\rho^*_{err_i}(t + \Delta t)$ is lower than a predefined value. In each convergence iteration, the predicted position $\mathbf{x}_i^*(t + \Delta t)$ and velocity $\mathbf{v}_i^*(t + \Delta t)$ of each particle are estimated based on $\mathbf{x}(t)$, $\mathbf{v}(t)$ and predicted acceleration. Using the predicted positions, the predicted densities $\rho_i^*(t + \Delta t)$ are computed as

$$\rho_i^*(t + \Delta t) = \sum_j m_j W_{ij}^*,$$

where $W_{ij}^* = W\left(\mathbf{x}_i^*(t + \Delta t) - \mathbf{x}_j^*(t + \Delta t), h\right)$. Afterwards, the density change (compression) $\rho^*_{err_i}(t + \Delta t) = \rho_i^*(t + \Delta t) - \rho_0$ is computed to update the pressure that corrects that density error as

$$p_i(t) += \delta\rho^*_{err_i}(t + \Delta t),$$

where δ is a constant computed for a prototype particle with complete neighborhood and defined as

$$\delta = \frac{-1}{\beta \left[-\sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} - \sum_j \left(\nabla W_{ij} \cdot \nabla W_{ij} \right) \right]}$$

and

$$\beta = \Delta t^2 m^2 \frac{2}{\rho_0^2}.$$

Finally, to recompute the predicted positions and velocities for all particles, the pressure force

$$\mathbf{F}_i^p \left(t \right) = -m_i \sum_j m_j \left(\frac{p_i \left(t \right)}{\left(\rho_i^* \right)^2} + \frac{p_j \left(t \right)}{\left(\rho_j^* \right)^2} \right) \nabla W_{ij}^*$$

is used. The procedure is repeated until a user-defined compression threshold value η is met. The steps of PCISPH is given in Algorithm 2.2.

2.6.2.2 Implicit Incompressible SPH

IISPH uses a semi-implicit form of the density prediction using the time rate of change of fluid density. The approach is derived by approximating the continuity equation given in (2.10) at time $t + \Delta t$ using a backward difference for the time derivative of the density, and the divergence of the velocity using SPH as:

$$\frac{\rho_i \left(t + \Delta t \right) - \rho_i \left(t \right)}{\Delta t} = \sum_j m_j \left[\mathbf{v}_i \left(t + \Delta t \right) - \mathbf{v}_j \left(t + \Delta t \right) \right] \nabla W_{ij} \left(t \right). \tag{2.18}$$

(2.18) introduces unknown velocities that depend on unknown pressure forces, which naturally depend on unknown pressure values at time t. By using the Euler-Cromer integration scheme, one can write $\mathbf{v} \left(t + \Delta t \right) = \mathbf{v} \left(t \right) + \Delta t \frac{\mathbf{F}^o(t) + \mathbf{F}^p(t)}{m}$, where $\mathbf{F}^p \left(t \right)$ denotes pressure forces and $\mathbf{F}^o \left(t \right)$ denotes all forces other than pressure. Assuming all other forces are known, the predicted velocities can be approximated as $\mathbf{v}^* \left(t + \Delta t \right) = \mathbf{v} \left(t \right) + \Delta t \frac{\mathbf{F}^o(t)}{m}$. Therefore, similar to (2.18), predicted density can be written as

$$\rho_i^* \left(t + \Delta t \right) = \rho_i \left(t \right) + \Delta t \sum_j m_j \left[\mathbf{v}_i^* \left(t + \Delta t \right) - \mathbf{v}_j^* \left(t + \Delta t \right) \right] \nabla W_{ij} \left(t \right). \tag{2.19}$$

Substituting $\rho_i^* \left(t + \Delta t \right)$ as $\rho_i \left(t \right)$, and ρ_0 as $\rho_i \left(t + 1 \right)$ into (2.18) and simplifying yields

$$\Delta t^2 \sum_j m_j \left(\frac{\mathbf{F}_i^p \left(t \right)}{m_i} - \frac{\mathbf{F}_j^p \left(t \right)}{m_j} \right) \nabla W_{ij} \left(t \right) = \rho_0 - \rho_i^* \left(t + \Delta t \right). \tag{2.20}$$

Substituting the symmetric pressure force in (2.16) into (2.20) results in a linear system with n equations and n unknown pressure values in the form

$$\sum_j a_{ij} p_j = \rho_0 - \rho_i^* \left(t + \Delta t \right). \tag{2.21}$$

The authors show that (2.21) can be solved either using relaxed Jacobi or conjugate gradient. In the relaxed Jacobi implementation, the individual pressure values p_i can be solved as

$$p_i^{l+1} = (1-\omega)\, p_i^l + \omega \frac{\rho_0 - \rho_i^* (t+\Delta t) - \sum_{j \neq i} a_{ij} p_j^l}{a_{ii}}, \tag{2.22}$$

where l is the index of iteration and ω is the relaxation factor. To compute (2.22), $a_{ij}$and $\sum_{j \neq i} a_{ij} p_j^l$ should be computed first. To extract a_{ij}, the displacement caused by pressure force can be reformulated as

$$\begin{aligned} \Delta t^2 \frac{\mathbf{F}_i^p (t)}{m_i} &= -\Delta t^2 \sum_j m_j \left(\frac{p_j}{\rho_j^2 (t)} + \frac{p_i}{\rho_i^2 (t)} \right) \nabla W_{ij} (t) \\ &= \underbrace{\left(-\Delta t^2 \sum_j \frac{m_j}{\rho_i^2 (t)} \nabla W_{ij} (t) \right)}_{\mathbf{d}_{ii}} p_i + \sum_j \underbrace{\left(-\Delta t^2 \frac{m_j}{\rho_j^2 (t)} \nabla W_{ij} (t) \right)}_{\mathbf{d}_{ij}} p_j, \end{aligned} \tag{2.23}$$

where $\mathbf{d}_{ii} p_i$ is the displacement of particle$i$ caused by the pressure p_i and $\mathbf{d}_{ij} p_j$ is the motion caused by the pressure p_j of a neighboring particle j. Substituting (2.23) into (2.20) and representing the neighboring particles j in (2.23) as k results in

$$\rho_0 - \rho_i^* (t+\Delta t) = \sum_j m_j \left(\mathbf{d}_{ii} p_i + \sum_j \mathbf{d}_{ij} p_j - \mathbf{d}_{jj} p_j - \sum_k \mathbf{d}_{jk} p_k \right) \nabla W_{ij} (t)\,. \tag{2.24}$$

Since i and j are neighboring particles, the term $\sum_k \mathbf{d}_{jk} p_k$ also includes pressure values p_i. To separate p_i, one can write $\sum_k \mathbf{d}_{jk} p_k = \sum_{k \neq i} \mathbf{d}_{jk} p_k + \mathbf{d}_{ji} p_i$. Hence, the right-hand side of (2.24) can be separated into two parts: one containing p_i; and another containing p_j and p_k as

$$\begin{aligned} \rho_0 - \rho_i^* (t+\Delta t) = p_i \sum_j m_j \left(\mathbf{d}_{ii} - \mathbf{d}_{ji} \right) \nabla W_{ij} (t) \\ + \sum_j m_j \left(\sum_j \mathbf{d}_{ij} p_j - \mathbf{d}_{jj} p_j - \sum_{k \neq i} \mathbf{d}_{jk} p_k \right) \nabla W_{ij} (t)\,. \end{aligned}$$

Therefore, the coefficients a_{ij} can be computed as

$$a_{ii} = \sum_j m_j \left(\mathbf{d}_{ii} - \mathbf{d}_{ji} \right) \nabla W_{ij} (t)\,,$$

and finally, the pressure can be iteratively computed as

$$p_i^{l+1} = (1-\omega)\, p_i^l + \omega \frac{1}{a_{ii}} \left(\rho_0 - \rho_i^* (t+\Delta t) - D \right) \tag{2.25}$$

where

$$D = \sum_j m_j \left(\sum_j \mathbf{d}_{ij} p_j^l - \mathbf{d}_{jj} p_j^l - \sum_{k \neq i} \mathbf{d}_{jk} p_k^l \right) \nabla W_{ij} (t)\,. \tag{2.26}$$

Algorithmus 2.3 IISPH using relaxed Jacobi iterative solver

while *animating* **do**
  foreach *fluid-particle* i **do**
    find neighbors
  end foreach
  foreach *fluid-particle* i **do**
    compute density $\rho_i(t) = \sum_j m_j W_{ij}$
    compute and add up forces other than the pressure force
    compute velocity $\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_i^o(t)}{m_i}$
    compute displacement $\mathbf{d}_{ii} = \Delta t^2 \sum_j -\frac{m_j}{\rho_i^2(t)} \nabla W_{ij}(t)$
  end foreach
  $l = 0$
  while $\rho_{avg}^l - \rho_0 > \xi$ **or** $l < 2$ **do**
    foreach *fluid-particle* i **do**
      $\sum_j \mathbf{d}_{ij} p_j^l = \Delta t^2 \sum_j -\frac{m_j}{\rho_j^2(t)} p_j^l \nabla W_{ij}(t)$
    foreach *fluid-particle* i **do**
      compute p_i^{l+1}
      $p_i(t) = p_i^{l+1}$
    $l = l + 1$
  end while
  foreach *fluid-particle* i **do**
    compute new velocity $\mathbf{v}_i(t + \Delta t)$
    compute new position $\mathbf{x}_i(t + \Delta t)$
  end foreach
end while

The authors propose to continue the Jacobi iteration until the difference between the average fluid density and the fluid rest density is no longer greater than the threshold ξ. Additionally, the minimum number of iterations is chosen as 2. Algorithm 2.3 summarizes how IISPH using relaxed Jacobi works.

2.7 Viscosity Force

The viscosity term in the Navier-Stokes momentum equation (2.7) is another fundamental term that governs the motion of fluids. In SPH, viscosity forces are commonly employed to be able to model variety of fluids. Viscosity also has an important positive effect on the simulation stability, as it damps relative motion of the particles. For real liquids, viscosity force converts the fluid kinetic energy into heat. Applying SPH to the viscosity term $\mu \nabla^2 \mathbf{v}$ in (2.7) yields

$$\mathbf{F}_i^v = \mu \sum_j m_j \left(\frac{\mathbf{v}_j}{\rho_j} \right) \nabla^2 W_{ij}, \tag{2.27}$$

which again produces an asymmetric formulation similar to directly applying SPH to solve for the pressure force. In the work of Mueller et al. [MCG03], (2.27) is symmetrized using the fact that the viscosity forces only depend on

relative velocities as

$$\mathbf{F}_i^v = \mu \sum_j m_j \left(\frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \right) \nabla^2 W_{ij}. \tag{2.28}$$

However, one issue with (2.28) is that it relies on the second derivative of the kernel function, which is very sensitive to particle disorder.

Another way to compute viscosity is by using the artificial viscosity formulation explained in the work of Monaghan and Gingold [MG83], where the viscosity force on a particle is written as

$$\mathbf{F}_i^v = -m_i \sum_j m_j \Pi_{ij} \nabla W_{ij}, \tag{2.29}$$

where

$$\Pi_{ij} = -\nu \left(\frac{\min(\mathbf{v}_{ij} \cdot \mathbf{x}_{ij}, 0)}{|\mathbf{x}_{ij}|^2 + \epsilon h^2} \right), \tag{2.30}$$

with the viscous factor

$$\nu = \frac{2\alpha h c_s}{\rho_i + \rho_j} \tag{2.31}$$

In (2.30), $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ and $\epsilon = 0.01$ is commonly used to avoid singularities for $|\mathbf{x}_{ij}| = 0$. In (2.31), α is the viscosity constant and c_s denotes the speed of the numerical propagation.

Another popular technique to introduce relative motion damping in SPH is to perform velocity smoothing of the neighboring fluid particles before integration. This technique is called XSPH and has been introduced in the work of Monaghan [Mon89]. XSPH is defined as

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_l + \varepsilon \sum_j \frac{m_j}{\bar{\rho}_{ij}} \mathbf{v}_{ji} W_{ij},$$

where $\bar{\rho}_{ab} = \frac{1}{2}(\rho_a + \rho_b)$ and ϵ is a constant in the interval $(0, 1)$ that determines how fast the velocities are smoothed. In the works of [SB12, MM13], XSPH is used as the sole method to generate viscosity-like damping in SPH simulations.

For the presented experiments in this thesis, the artificial viscosity formulation of Monaghan and Gingold[MG83] (2.29) has been used.

2.8 Vorticity Confinement

Numerical dissipation has the effect of causing additional damping on SPH particles, which results in the generated vortices to rapidly disappear. To counteract this undesired effect, vorticity confinement techniques [SU94] are commonly used. Based on the heuristic representation of the vorticity confinement in [FSJ01]; in the work of Hong et al. [HLYK08], the vorticity at the mass center of two SPH particles is measured and vorticity confinement forces are applied to fluid particles. In the work of Zhu et al. [ZYF10], a high-resolution grid is coupled with each solid object, inside which the turbulence formation is modeled by resolving the local flow by employing a hybrid SPH-Eulerian solver. The method is aimed towards generating enhanced turbulent patterns around solid objects.

In the work of [JBiRBN11], SPH is coupled with multi-level Eulerian grids to combine vortices at different scales. More recently, in the work of Macklin and Mueller [MM13], the vorticity is computed at a particle's position using the vorticity estimator explained in [Mon92], where the vorticity forces are only added to particles, that already has some vorticity present, as in the work of Fedkiw et al. [FSJ01].

The experiments presented in this thesis do not employ any form of vorticity confinement. However, we believe that using a vorticity confinement technique could have a positive influence on flow realism for turbulent scenarios.

2.9 Multi-Resolution Techniques

Multi-resolution techniques in SPH gained popularity with the work of Adams et al. [APKG07]. In that work, a sampling condition based on geometric local feature size is used, which allows using smaller particles for focusing computational resources in geometrically complex regions, while using larger particles deep inside the fluid or near thick flat regions. The transition between different resolution particle sets is performed using split/merge operations. Similar adaptive techniques are later presented in the works of Zhang et al. [ZSP08] and Yan et al. [YWH$^+$09]. More recently, in the work of Solenthaler and Gross. [SG11], a scheme to couple two-resolutions SPH fluids is explained. In their work, frequent particle splitting and merging processes are avoided, which alleviates stability problems of the previous multi-resolution techniques. In the work of Horvath and Solenthaler [HS13], the previous two-scale technique is extended to multiple resolutions. Their approach also supports fine resolution particle generation near fluid surface and close to observer, and can handle complex boundary conditions. Furthermore, the technique improves upon the mass-conservation and stability aspects of the previous multi-resolution techniques.

2.10 Surface Generation

Among all phases in SPH fluid animations, the surface reconstruction is one of the most challenging parts. The main research in this area focuses on obtaining smooth surfaces within reasonable time by reducing memory footprint. Although there are various approaches in the literature, probably the most widely used technique is polygonizing the fluid surface with the Marching Cubes (MC) method of Lorensen and Cline [LC87], which is performed over a proper scalar field on uniform grids.

Within the context of scalar field computation, Blinn initially introduced blobbies method [Bli82], which is efficient to compute, however, produces very bumpy surfaces. Later, Zhu and Bridson proposed to use the signed distance field of the particles [ZB05]. In this approach, each particle contributes to the scalar field along its smoothing radius. This approach alleviates the bumpiness issue; however, it suffers from artifacts in concave regions. Adams et al. [APKG07] addressed these issues by introducing a distance-based surface tracking technique. This technique is able to achieve smoother surfaces; however, its computational complexity makes it less suitable for the surface reconstruction phase. Solenthaler et al. improved the method of Zhu and Bridson in [SSP07], where they correct

the artifacts by considering the movement of the contributing particles' center of mass at a certain query point. This problem is also addressed in the work of Onderik et al. [OCD11]. Recently, in order to obtain smooth surfaces, Yu and Turk [YT10] proposed to use anisotropic kernels to further improve surface smoothness. However, the approach is significantly more expensive when compared to the previous work. Later, Bhattacharya et al. [BGB11] proposed a level set method, where the fluid surface that lies between inner and outer surface approximations is processed by Laplacian smoothing. However, the approach causes most of the surface details to get smoothed out.

To reduce the computational complexity and memory consumption, various narrow band techniques have been proposed, e.g. sparse block grids [Bri03], RLE sparse level sets [HWB04] and dynamic tubular grids [NM06]. Out-of-core techniques, e.g. [NNSM07] and parallel algorithms, e.g. [AIAT12], also gained interest in the recent years to handle large particle counts.

Recent advances in GPU architectures pioneer researchers to investigate interactive fluid rendering techniques. In this context, surface splatting is one of the most popular surface visualization approaches [ZPvBG01, ALD06, vdLGS09, MM13]. The splatting method is quite useful for fast getting fast results. However, generated surfaces only consist of smoothly blended splats, which are not able to produce high quality results. There are also other screen space approaches that can generate a triangle mesh, e.g. [MSD07, FAW10, GSSP10]. The surface reconstruction of SPH fluids invites many researchers for further improvement, e. g. improving surface quality with post-processing [AAIT12] or application of adaptive spatial data structures for memory consumption improvements [ZGHG11, AAET13].

For the presented works in this thesis, the method of Akinci [AIAT12] has been used to construct triangle meshes from particles.

3 Rigid-Fluid Coupling

3.1 Introduction

In fluid animations, the interesting fluid behavior usually emerges when rigid objects are added to a simulation (see Figure 3.1). While the two-way coupling of particle-based fluids and solids seems to be straightforward, there is no general agreement how this should be handled. On one hand, the coupling has to cope with particle deficiency issues at the boundary in order to prevent spatial and temporal discontinuities of physical properties of the particles and sticking artifacts [IAGT10] (see Figure 3.2). On the other hand, lower dimensional geometries, e.g. thin structures and non-manifold surfaces, as well as constrained rigid bodies should be supported. This chapter provides a versatile way to address these issues. In the remainder of this section, we first discuss existing methods for treating boundaries in SPH (Section 3.1.1) as well as for rigid-fluid coupling (Section 3.1.2), and then highlight our contribution (Section 3.1.3).

3.1.1 Boundary-Handling in SPH

For SPH boundary-handling, distance-based penalty methods with boundary particles have been commonly used [MST+04, Mon05, HKK07, MK09]. These approaches, however, require large penalty forces that restrict the time step. Furthermore, particles tend to stick to the boundary due to the lack of fluid neighbors. In [HKK07], this problem is alleviated by employing a wall weight function for approximating density contributions for planar boundaries.

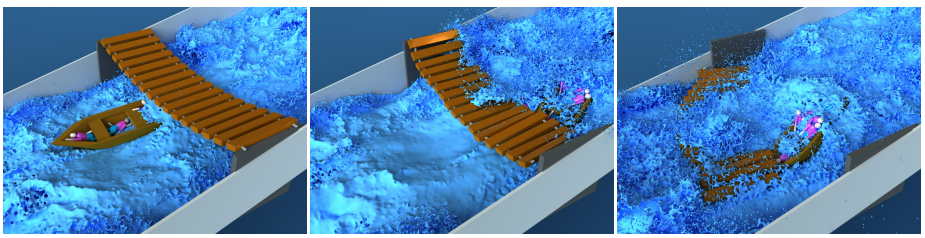


Figure 3.1 – Fluid-rigid interaction in a large-scale setting. A boat with ragdolls passes a bridge (left). A second boat with ragdolls collides with the bridge due to an increased flow rate and the bridge is released (middle and right). Images are taken from [AIA+12].

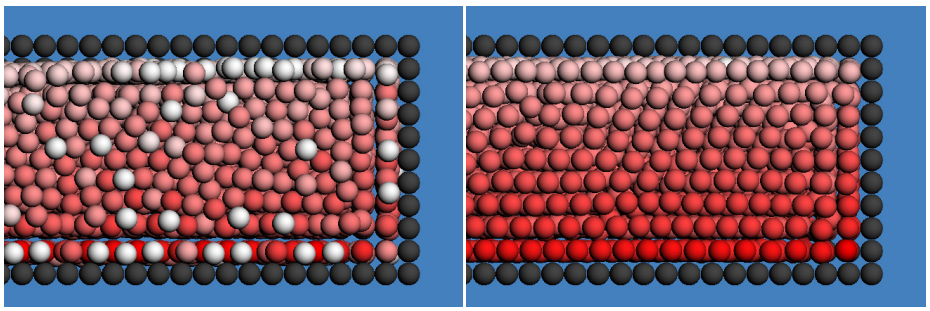


Figure 3.2 – Pressure profile of a box filled with particles where the front side is clipped for illustration purposes. The fluid particle pressures are color-coded and proportional to their red saturation. Black particles denote boundary particles. While [BTT09] leads to pressure noise and sticking artifacts (left), our method avoids these problems (right). Images are taken from [AIA+12].

The sticking of particles is avoided with frozen and ghost particles based models, e.g. [LP91, HA06, SB12]. Frozen particles are fluid particles that are constrained to static positions, whereas ghost particles are mirrored across the boundary on the fly. Figure 3.3 provides an illustration to the sticking problem in SPH. In order to guarantee non-penetration, either more than one layer of frozen particles should be used [DK01], or the positions of penetrating particles should be corrected [IAGT10]. Since this class of methods samples the boundary with fluid particles, the relevant field variables can be well approximated with SPH. This results in continuous pressure gradients, which consequently prevents sticking of fluid particles to the boundaries. However, handling the interaction of the fluid with thin shells is problematic in these approaches since the elevated density on one side of a boundary particle affects potential fluid particles on the other side. The interaction of fluids with deformable thin shells has been demonstrated in [LD08] with additional distance-based non-symmetric forces to prevent the penetration of fluid particles.

Alternative to particles, boundaries can also be efficiently represented with triangles. In this case, however, it is challenging to handle discontinuous surface normals and non-manifold structures that cause spatial and temporal discontinuities of the fluid properties.

3.1.2 Two-Way Fluid-Rigid Coupling in SPH

For the two-way coupling of SPH fluids and rigid bodies, only few approaches have been proposed so far. In [CBP05], the fluid is considered as a collection of rigid spheres exchanging impulses with surrounding rigid bodies. In [ODAF06, KAD+06], the pressure at the boundary is taken into account for two-way coupled fluid-rigid interaction. In those models, however, dynamic forces, e.g. viscosity, are neglected. More recently, an impulse-based approach for simulating the two-way coupling of SPH fluids with particle-based rigid bodies has been proposed in [OKR09a]. This approach, however, is not purely based on hydrodynamic forces, relies on normal information for rigid bodies, and does not guarantee non-penetration for thin shells.

In [BTT09], direct forcing has been employed for both one- and two-way

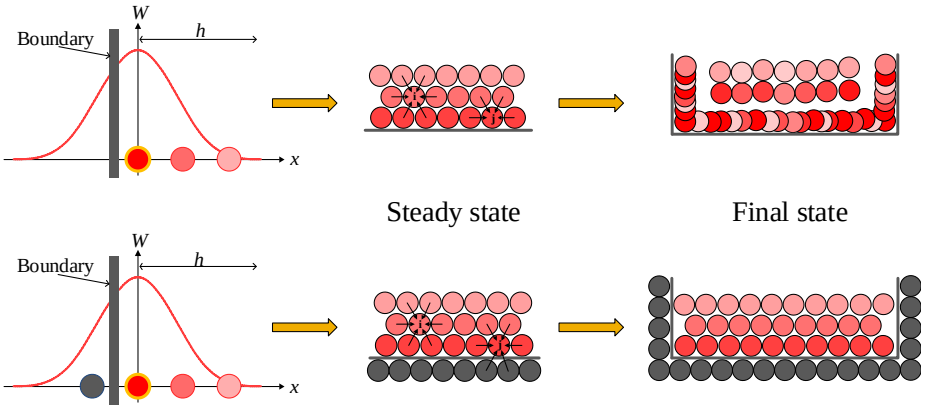


Figure 3.3 – Illustration of the boundary deficiency problem. Since particle i has forces on itself from all directions, it is able to move to any direction based on the pressure change of the neighboring fluid particles. Particle j, however, does not have a full neighborhood that surround it, which restricts its motion to the boundaries. Resuming from these steady states would result in the configurations illustrated in the right of the figure.

fluid-rigid coupling. This method uses a predictor-corrector scheme to compute forces that constrain particle positions and velocities to specific values. Non-penetration is guaranteed by using position correction. Different slip conditions are realized by including a non-symmetric friction model. The position correction and the non-symmetric friction forces are, however, not momentum-conserving. Another issue is that timestep-dependent operations are used that require careful parameter evaluation for each setup. Finally, it requires two additional neighbor queries for two-way coupling, which is rather expensive. This method has not yet been extended to handle the interaction of a particular fluid particle with multiple rigid bodies or the simultaneous contact among the bodies.

There exist impressive two-way coupling approaches for Eulerian and semi-Lagrangian schemes (e.g. [CMT04, GSLF05, CGFO06, BBB07, RMSG+08]), as well as for 2D heightfield models (e.g. [CM10]). A thorough discussion of these methods is, however, beyond the scope of this chapter.

3.1.3 Contributions

We present a novel, versatile method for the two-way coupling of SPH fluids and rigid bodies. We use boundary particles to sample the surface of rigid objects, which has several benefits. First, the use of particles allows us to derive a model that can cope with different shapes, including lower-dimensional rigid bodies consisting of one layer (referred to as thin shells) or one line of boundary particles (referred to as rods), as well as non-manifold geometries. Second, the inclusion of boundary particles successfully alleviates the particle deficiency problem of SPH near boundaries, preventing density (and consequently pressure) discontinuities at the boundary and particle sticking artifacts.

Our model addresses the problem of inhomogeneous particle sampling at the boundary by deriving new equations that consider the relative contribution of a boundary particle to a physical quantity. This does not only facilitate

the particle initialization at complex boundaries, but also enables the use of multiple dynamic objects where the boundary sampling in the neighborhood may change due to contacts. A friction model is additionally included to simulate various slip conditions and drag effects. All pressure and viscous forces that are applied between fluid and boundary particles are symmetric, conserving linear and angular momentum. The approach is designed such that even very large density ratios between fluids and rigid bodies can be handled.

3.2 Formalism

This section firstly explains our corrected density computation concept at the rigid-fluid interface in Section 3.2.1. Afterwards, Sections 3.2.2 and 3.2.3 describe novel pressure and friction forces for pairs of boundary and fluid particles, while Section 3.2.4 discusses the overall forces and the symmetry of these forces.

3.2.1 Corrected Density Computation

The density summation approach (2.9) approximates the density of a fluid particle correctly only if a particle is spherically surrounded by particles with the same initial density. Therefore, densities of fluid particles near the boundaries are underestimated. In order to alleviate this underestimation, we set the densities of such particles to the rest density of the fluid. Even though this simple correction scheme significantly improves the situation, the density gradient still remains discontinuous near the boundaries. Additionally, since the particles near the boundaries do not have neighbors that spherically surround them, forces on such particles constrain their movements to the boundaries, which causes sticking artifacts. To avoid this problem, we take the neighboring boundary particles into account when computing densities and forces for fluid particles, similar to [IAGT10].

Since we focus on the interaction of fluids with non-deformable rigid bodies without melting effects, particles do not necessarily need to be generated inside a rigid. Therefore, we generate particles as a single layer at the surface similar to [BYM05, BIT09, IAGT10]. This approach saves memory and improves performance. The particle representations of rigid bodies in the framework are computed either directly (e.g. for analytical shapes) or from mesh representations. Particle representations of triangle meshes are generated based on [BYM05], which permits placing particles at an arbitrary offset to the surface mesh and yields a quite homogenous sampling. However, at regions with high-curvature, the particle distribution usually remains non-homogenous, resulting in a denser sampling in such areas (e.g., see Figure 3.4). We observed similar issues using the remeshing algorithm of [BK04] and placing particles at the vertex positions. Fortunately, neither of the algorithms results in under-sampled regions. Now, we can say that each boundary particle b_i represents a volume V_{b_i} at the surface of a rigid:

$$V_{b_i} = \frac{m_{b_i}}{\rho_{b_i}} = \frac{m_{b_i}}{\sum_k m_{b_k} W_{ik}},$$

where k denotes boundary particle neighbors. In [SP08], it has been shown that the density summation approach in (2.9) causes stability issues for large

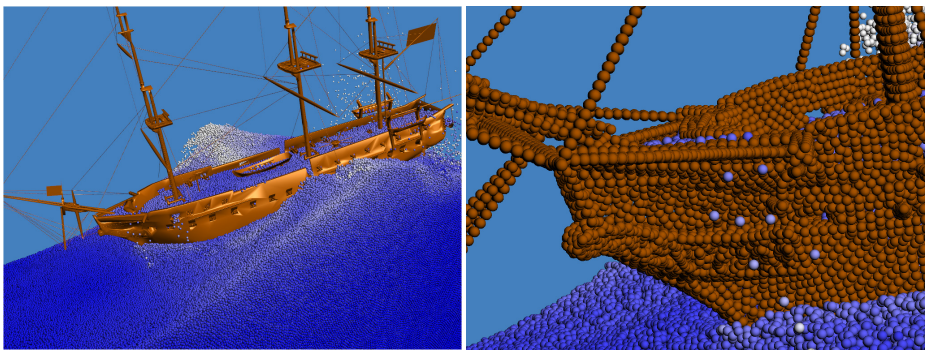


Figure 3.4 – A frigate is sailing on wavy sea. The right image shows the irregular sampling of boundary particles.

density ratios due to erroneous density estimations for particles at the interface. Therefore, the density of a fluid particle can be written as

$$\rho_{f_i} = m_{f_i} \sum_j W_{ij} + m_{f_i} \sum_k W_{ik}, \tag{3.1}$$

where j denotes fluid particle neighbors. Applying this idea to the volume of a boundary particle results in

$$V_{b_i} = \frac{m_{b_i}}{m_{b_i} \sum_k W_{ik}} = \frac{1}{\delta_{b_i}}, \tag{3.2}$$

with $\delta_{b_i} = \sum_k W_{ik}$. Finally, (3.2) implies that the volume of a boundary particle gets smaller for densely sampled areas and larger for sparsely sampled areas. We now derive the fluid density computation based on the boundary particle volume V_{b_i}.

Even though (3.1) addresses discontinuities for uniformly sampled particles, it does not account for a variable sampling of particles. Therefore, since the homogeneity of rigid sampling is not guaranteed, (3.1) causes fluid particles to get large contributions from overly sampled regions. Those overestimated densities cause large pressure forces and therefore stability issues. This is due to the fact that the contribution of boundary particles in (3.1) does not consider the volume of a particle. This contradicts with the SPH concept, where the contribution of a particle in the approximation of any field variable should be governed by its volume (see (2.2)). Therefore, we write the contribution of a boundary particle to a fluid particle by taking the volume of the boundary particle into account as

$$\Psi_{b_i}(\rho_0) = \rho_0 V_{b_i}, \tag{3.3}$$

where ρ_0 denotes the rest density of the fluid that the rigid is interacting with. Finally, the corrected density of a fluid particle can be written in the form

$$\rho_{f_i} = m_{f_i} \sum_j W_{ij} + \sum_k \Psi_{b_k}(\rho_{0_i}) W_{ik}, \tag{3.4}$$

which computes the densities correctly regardless of the boundary particle sampling. Note that Ψ_b increases the contributions of boundary particles by

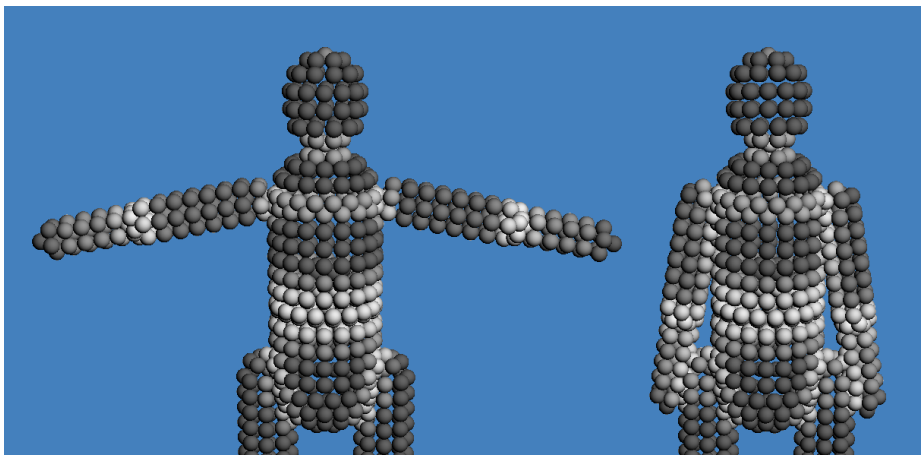


Figure 3.5 – Color-coded contributions of boundary particles for two ragdolls with different poses. Black and white particles represent large and small contributions, respectively. The contribution is defined by the particular sampling density. Please note, e.g., the smaller contribution of rigid parts in contact. Image is taken from [AIA+12].

the amount of the volume ratio of boundary and fluid particles in a uniformly sampled case (by a factor of ~ 1.4). Since Gaussian like kernels are commonly used for SPH simulations, the weight of the next layer of particles is significantly lower compared to the closer layer. Therefore, using a single layer of boundary particles with (3.4) and taking the missing particles into account in (3.3) is a decent approximation in practice. See Figure 3.5 for an illustration of the particle contributions. Our approach updates the contributions of boundary particles for changing boundary configurations with a minimal influence on the fluid particles that are in contact with the boundary. We experimentally verified that even dynamically moving and overlapping boundaries can be handled, which is illustrated in Figure 3.6.

Even though boundary particles are precomputed, a boundary particle is included in the simulation only if it is in the neighborhood of a fluid particle, similar to [IAGT10]. For moving boundary particles and all neighboring boundary particles, the represented particle volumes are recomputed for handling the case of overlapping object parts or objects in close proximity (e.g. for dynamic or kinematic rigid bodies).

3.2.2 Boundary-Fluid Pressure Force

In SPH, the pressure force between two particles can be directly derived as

$$\mathbf{F}^p_{i \leftarrow j} = -m_i m_j \left(\frac{p_j}{\rho_i \rho_j} \right) \nabla W_{ij}, \tag{3.5}$$

where p denotes pressure of a particle [Mon05]. For purely incompressible flow one can say that,

$$\lim_{\eta \to 0} (\rho_i - \rho_j) = 0 \quad \text{and} \quad \lim_{\eta \to 0} (p_i - p_j) = 0$$

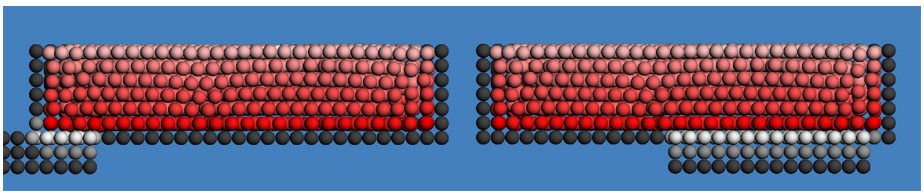


Figure 3.6 – Handling of overlapping boundaries. Pressures on particles are proportional to their red saturation and black particles denote boundary particles Front side of the box has been clipped to make the fluid visible. For boundary particles, whiteness denotes smaller contributions. Fluid gets very small amount of disturbance even in such challenging scenarios. Image is taken from [AIA+12].

where η denotes the density fluctuation of the fluid. Therefore, for weakly compressible fluids, we can assume that $\rho_i \approx \rho_j$ and $p_i \approx p_j$. Consequently, (3.5) can be approximated as

$$\mathbf{F}^p_{i \leftarrow j} = -m_i m_j \left(\frac{p_x}{\rho_x^2} \right) \nabla W_{ij}, \tag{3.6}$$

where x can be either i or j. A similar assumption has been also used in the derivation of PCISPH [SP09].

In practice, the applied pressure from fluid to some region of the rigid does not have any kinematic influence on the nearby fluid particles. Based on this fact, we write the pressure force applied from a boundary particle b_j to a fluid particle f_i as

$$\mathbf{F}^p_{f_i \leftarrow b_j} = -m_{f_i} \Psi_{b_j}(\rho_{0_i}) \left(\frac{p_{f_i}}{\rho_{f_i}^2} \right) \nabla W_{ij}, \tag{3.7}$$

by substituting m_j with $\Psi_{b_j}(\rho_{0_i})$ as done in (3.4), and using the fluid particle's density and pressure only. The symmetric pressure force from a fluid particle to a boundary particle is

$$\mathbf{F}^p_{b_j \leftarrow f_i} = -\mathbf{F}^p_{f_i \leftarrow b_j}. \tag{3.8}$$

In (3.7) and (3.8), the idea is making use of a fluid particle's own pressure when computing the boundary force. Magnitudes of the boundary forces are based on the pressure of the fluid particle, which increase as the particle gets closer to a boundary. Since the pressure of a fluid particle near a boundary would result in a pressure force to the boundary, that force can be counteracted by a force that is proportional to the pressure of the fluid particle. Therefore, this formulation eliminates sticking artifacts and prevents penetration of fluid particles to the boundaries without using extra forces or position correction. It also eliminates the need for normal information for our boundary particles. Additionally, densities and pressures for boundary particles are not required. Figure 3.7 illustrates how our boundary model works.

3.2.3 Boundary-Fluid Friction Force

Inspired by the viscosity-based friction model proposed in [MST+04], friction between interacting fluid and boundary particles is generated by employing the laminar artificial viscosity model explained in the work of Monaghan and

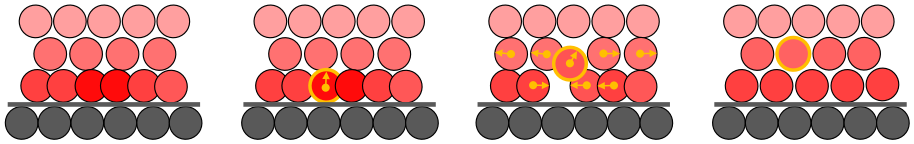


Figure 3.7 – Step-by-step illustration of how our approach prevents sticking artifacts. Yellow arrows denote movement directions for selected particles. (a) As the pressure of a fluid particle near the boundary increases, the boundary force acting on it also increases. (b) The particle is pushed away from the boundary. (c) It moves to the next layer of fluid particles. (d) High density region is resolved.

Gingold [MG83]. Therefore, based on (2.29), we define the viscosity force from a boundary particle to a fluid particle as

$$\mathbf{F}^v_{f_i \leftarrow b_j} = -m_{f_i} \Psi_{b_j}(\rho_{0_i}) \Pi_{ij} \nabla W_{ij}, \tag{3.9}$$

with the reformulated viscous factor

$$\nu = \frac{\sigma h c_s}{2\rho_{f_i}}, \tag{3.10}$$

where σ is the viscosity coefficient between fluid and rigid. From (3.10), ρ_{b_i} is eliminated based on the same assumption that was used when deriving (3.7). When computing the viscosity force, the velocity of a boundary particle can be easily computed based on the kinematic properties of the rigid body it belongs to.

The symmetric friction force from a fluid particle to a boundary particle can be written as

$$\mathbf{F}^v_{b_j \leftarrow f_i} = -\mathbf{F}^v_{f_i \leftarrow b_j}. \tag{3.11}$$

(3.11) results in drag effects on the rigid (see Figure 3.8). This idea was also presented in [BTT09]. However, in their work, friction forces are not momentum-conserving.

3.2.4 Total Force and Force Symmetry

Our boundary particles are transformed based on the position and orientation returned by the rigid solver before computing all relevant forces. Based on the derived forces, the total boundary force acting on a fluid particle and the total force acting on a boundary particle from its fluid neighbors can be written as

$$\mathbf{F}^{total}_{f_i} = \sum_j \left(\mathbf{F}^p_{f_i \leftarrow b_j} + \mathbf{F}^v_{f_i \leftarrow b_j} \right),$$
$$\mathbf{F}^{total}_{b_i} = \sum_j \left(\mathbf{F}^p_{b_i \leftarrow f_j} + \mathbf{F}^v_{b_i \leftarrow f_j} \right).$$

Since the pairwise forces between particles are symmetric (i.e. $\mathbf{F}^p_{f_i \leftarrow b_j} + \mathbf{F}^p_{b_i \leftarrow f_j} = 0$ and $\mathbf{F}^v_{f_i \leftarrow b_j} + \mathbf{F}^v_{b_i \leftarrow f_j} = 0$), the total boundary and viscosity forces are symmetric as well, i.e. $\sum_i \mathbf{F}^{total}_{f_i} = \sum_i \mathbf{F}^{total}_{b_i}$.

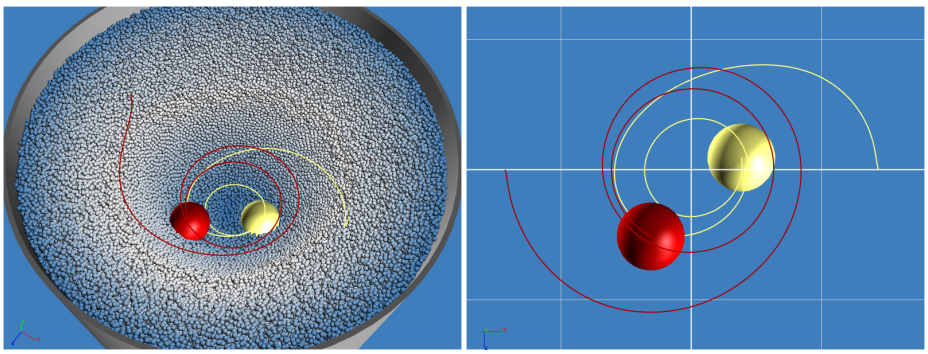


Figure 3.8 – Two spheres with different fluid viscosities ($\sigma = 0$ for yellow and $\sigma = 8$ for red) are dragged differently by the vortex. Fluid particles are colored according to speed where blue denotes slow particles. The curves visualize the trajectories of the spheres. Images are taken from [AIA+12].

Afterwards, for each boundary particle i which belongs to a dynamic rigid and has a fluid neighborhood, $\mathbf{F}_{b_i}^{total}$ is converted to total force and torque for the rigid body as

$$\mathbf{F}_{rigid} = \sum_i \mathbf{F}_{b_i}^{total},$$
$$\tau_{rigid} = \sum_i (\mathbf{x}_i - \mathbf{x}^{cm}) \times \mathbf{F}_{b_i}^{total},$$

where $\mathbf{x}^{cm}$ is the center of mass of the rigid body, and $\mathbf{x}_i$ is the position of a boundary particle. Finally, $\mathbf{F}_{rigid}$ and τ_{rigid} are applied to the rigid body.

3.3 Implementation Details

For the presented experiments in this chapter; in order to compute the pressure from the density, we employ either PCISPH [SP09] or SESPH [Mon05, BT07]. The employed pressure and viscosity forces are based on [Mon05]. For generating surface tension effects, we rely on [BT07]. Finally, for simulating multiphase fluids, we use [SP08]. For the SPH interpolations, we employ the cubic spline kernel [Mon05]. We use the Euler-Cromer scheme for time integration. We further employ the adaptive time-stepping schemes explained in [MK99] for SESPH, and [IAGT10] for PCISPH, where the shock handling criteria in the latter is also added to the former. We also included the velocities of boundary particles inside the time step estimation criteria so as to prevent fluid particles from passing through (i.e. tunneling). Even though we presented all underlying equations based on Monaghan's pressure and viscosity terms [Mon05], the same assumptions that we used in our derivations can be applied to different formulations as well, e.g. the force terms in [MCG03].

We used Bullet [Cou11] for simulating rigid bodies. However, because of the clear fluid-rigid solver decoupling, any rigid solver might be used as well. For finding neighboring particles, we employed compact hashing as proposed in [IABT11]. The application of our two-way coupling approach to SPH is presented in Algorithm 3.1.

Algorithmus 3.1 Simulation update with our boundary-handling model

while *animating* **do**
  foreach *moving-rigid-body* i **do**
    synchronize boundary particles with rigid body state
  end foreach
  foreach *fluid-particle* i **do**
    find fluid and boundary neighbors
    activate neighboring boundary particles
  end foreach
  foreach *fluid-particle* i **do**
    compute density $\rho_i(t)$
    compute pressure $p_i(t)$ (e.g. SESPH, PCISPH, IISPH)
  end foreach
  foreach *fluid-particle* i **do**
    add fluid forces $\mathbf{F}_i^{p,\nu,c,ext}(t)$
    add forces exerted by boundary particles $\mathbf{F}_{f_i}^{total}$
  end foreach
  foreach *active-boundary-particle* i **do**
    add forces exerted by fluid particles $\mathbf{F}_{b_i}^{total}$
  end foreach
  foreach *rigid-body* i **do**
    compute the total force exerted by fluids $\mathbf{F}_{rigid_i}$
    compute the total torque exerted by fluids τ_{rigid_i}
  end foreach
  foreach *fluid-particle* i **do**
    update $\mathbf{x}_i, \mathbf{v}_i$
  end foreach
  update rigid bodies (e.g. Bullet)
end while

Integration into IISPH

As summarized in the previous chapter, IISPH is a recent promising technique to efficiently compute particle pressures. Therefore, we found it worthwhile to explain how our approach can be also used with IISPH. First of all, the density estimation 2.18 in IISPH should be extended by also including the contributions of the boundary particles as

$$\begin{aligned}\rho_i\left(t+\Delta t\right) = &\sum_j m_j W_{ij} + \sum_b \Psi_b\left(\rho_{0_i}\right) W_{ib} \\ &+ \Delta t \sum_J m_j \mathbf{v}_{ij}\left(t+\Delta t\right) \nabla W_{ij} \\ &+ \Delta t \sum_b \Psi_b\left(\rho_{0_i}\right)\left(\mathbf{v}_i\left(t+\Delta t\right) - \mathbf{v}_b\left(t+\Delta t\right)\right) \nabla W_{ib},\end{aligned}$$

where ρ_{0_i} is the density of the fluid particle i that the boundary particle b interacts with. Assuming constant rigid velocity $\mathbf{v}_b$ throughout the pressure

iterations of IISPH, the density without pressure forces can be predicted as

$$\begin{aligned}\rho_i^*(t+\Delta t) = &\sum_j m_j W_{ij} + \sum_b \Psi_b(\rho_{0_i}) W_{ib} \\ &+ \Delta t \sum_J m_j \left(\mathbf{v}_i^*(t+\Delta t) - \mathbf{v}_i^*(t+\Delta t)\right) \nabla W_{ij} \\ &+ \Delta t \sum_b \Psi_b(\rho_{0_i}) \left(\mathbf{v}_i^*(t+\Delta t) - \mathbf{v}_b(t+\Delta t)\right) \nabla W_{ib}.\end{aligned}$$

Therefore, analogous to 2.20, pressure forces are expected to correct the density field based on:

$$\begin{aligned}\rho_i(t+\Delta t) = \rho_i^*(t+\Delta t) + &\sum_j m_j \left(\Delta t^2 \frac{\mathbf{F}_i^p(t)}{m_i} - \Delta t^2 \frac{\mathbf{F}_j^p(t)}{m_j}\right) \nabla W_{ij} \\ &+ \sum_b \Psi_b(\rho_{0_i}) \left(\Delta t^2 \frac{\mathbf{F}_i^p(t)}{m_i}\right) \nabla W_{ib}.\end{aligned}$$

Accordingly, particle displacement due to pressure is computed as

$$\begin{aligned}\Delta t^2 \frac{\mathbf{F}_i^p}{m_i} &= \sum_j \underbrace{-\Delta t^2 \frac{m_j}{\rho_j^2} \nabla W_{ij}}_{\mathbf{d}_{ij}} p_j + \\ &= \underbrace{\left(-\Delta t^2 \sum_j \frac{m_j}{\rho_i^2} \nabla W_{ij} - \Delta t^2 \sum_b \Psi_b(\rho_{0_i}) \frac{1}{\rho_i^2} \nabla W_{ib}\right)}_{\mathbf{d}_{ii}} p_i,\end{aligned}$$

Finally, the pressure update with relaxed Jacobi solver can be written as

$$p_i^{l+1} = (1-\omega) p_i^l + \omega \frac{1}{a_{ii}} \left(\rho_0 - \rho_i^*(t+\Delta t) - D - B\right)$$

where D is given in (2.26) and B is defined as

$$B = \sum_b \Psi_b(\rho_{0_i}) \sum_j \mathbf{d}_{ij} p_j^l \nabla W_{ib}.$$

3.4 Results

In this section, we demonstrate the versatility of our approach in various simulation settings. If not stated otherwise, we used PCISPH [SP09] as the basic fluid simulation model, where the maximum permissible degree of compression was kept at 1%. In our simulations, we used different particle radii r for different scenarios. The SPH smoothing length was always chosen as $4r$. All simulated fluids had low laminar viscosity ($\alpha = 0.01$) and surface tension ($\kappa = 0.05$), which were determined experimentally to approach the behavior of water. The employed adaptive time-stepping schemes produced time steps roughly between 10^{-2} and 10^{-4}. For all scenes, the computation overhead of the rigid-fluid coupling was mainly between 5-10%. The overhead varied based on the number

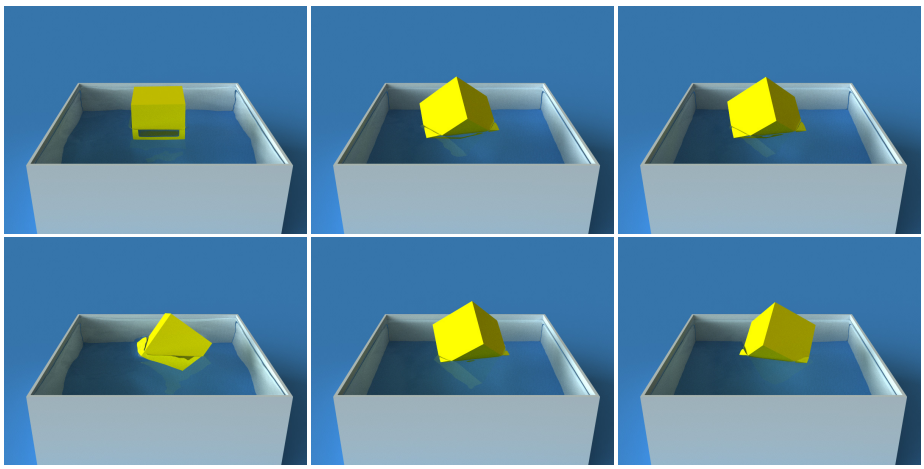


Figure 3.9 – Frame by frame comparison of our approach (top) to [BTT09] (bottom). This figure illustrates that our boundary-handling method does not introduce non-physical energy to the simulation. In the approach of Becker et al., the cube starts spinning in the water. Whereas in our approach, the cube stays in a balanced state. Images are taken from [AIA+12].

of fluid-boundary particle pairs. The overhead of the rigid body simulation was usually below 1%. Fluid surfaces were generated using a parallelized implementation [AIAT12] of the method proposed in [SSP07]. Renderings were done using mental ray v3.9.4 [NVI11]. The simulations and renderings were run on an Intel Xeon X5690 with 12 GB RAM. Average computation time per frame (note that for one frame several simulation steps are computed) was 1 second to 2 minutes depending on the complexity of the presented scene. These timings exclude surface reconstruction and rendering.

We firstly compare our approach to [BTT09] in a simple setting where a cube with density $400\frac{kg}{m^3}$ was dropped into a container with 200K fluid particles that have a rest density of $1000\frac{kg}{m^3}$. We used SESPH in this example. In contrast to [BTT09], our pairwise forces are symmetric. One frame could be computed in 6 seconds on average with our model, compared to 36 seconds with [BTT09]. The reasons are twofold; our boundary-handling allows to use larger time steps (in this experiment three times larger on average), and it does not require additional neighbor queries. This experiment is shown in Figure 3.9.

Our viscosity model can simulate drag effects. This is shown in Figure 3.8 where two spheres with different fluid viscosities were dropped into a whirlpool. The sphere with zero viscosity moved faster to the center of the vortex, while the sphere with high viscosity was dragged by the velocity field of the fluid. In Figure 3.10, we further show in a simple setting that the symmetric force can be used to generate friction on the fluid particles.

One main advantage of our method is that the interaction with lower dimensional rigid bodies can be simulated. Figure 3.11 shows an example where rods and planes, sampled by a single layer of boundary particles, were dropped into a fluid with 300K particles. Further, we dropped ragdolls that are modeled by multiple capsules connected with different constraints. Even in such challenging scenarios, our two-way coupling approach produces plausible results. Related

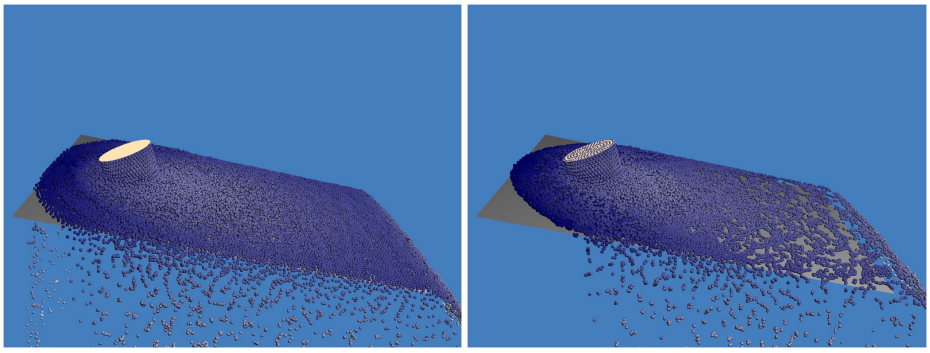


Figure 3.10 – Different slip conditions can be realized by using different viscosity constants for the rigid. Left image illustrates $\sigma = 0$ and right image illustrates $\sigma = 4$. Note that fluid particles do not penetrate to the plane that is sampled using a single layer of boundary particles. Also, note that fluid particles are able to slide over the plane smoothly without getting any disturbance from the underlying boundary particles. Images are taken from [AIA+12].

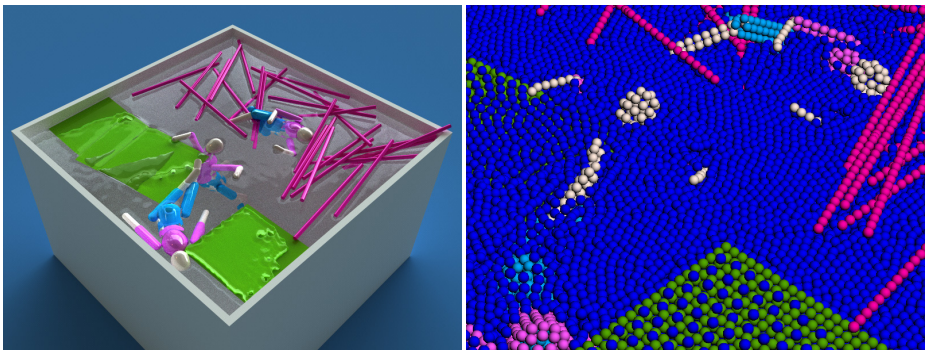


Figure 3.11 – Handling of lower dimensional objects. The right image shows the underlying particles. Note that planes and rods are modeled with single particle layers. Images are taken from [AIA+12].

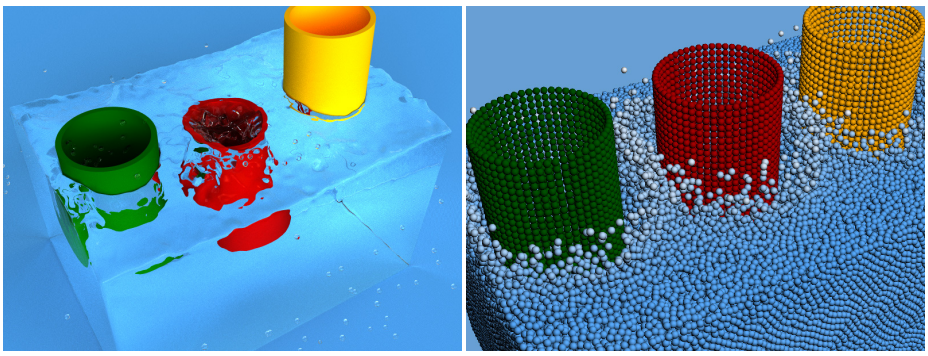


Figure 3.12 – Thin shells of different densities, each of them represented by a single particle layer. Images are taken from [AIA+12].

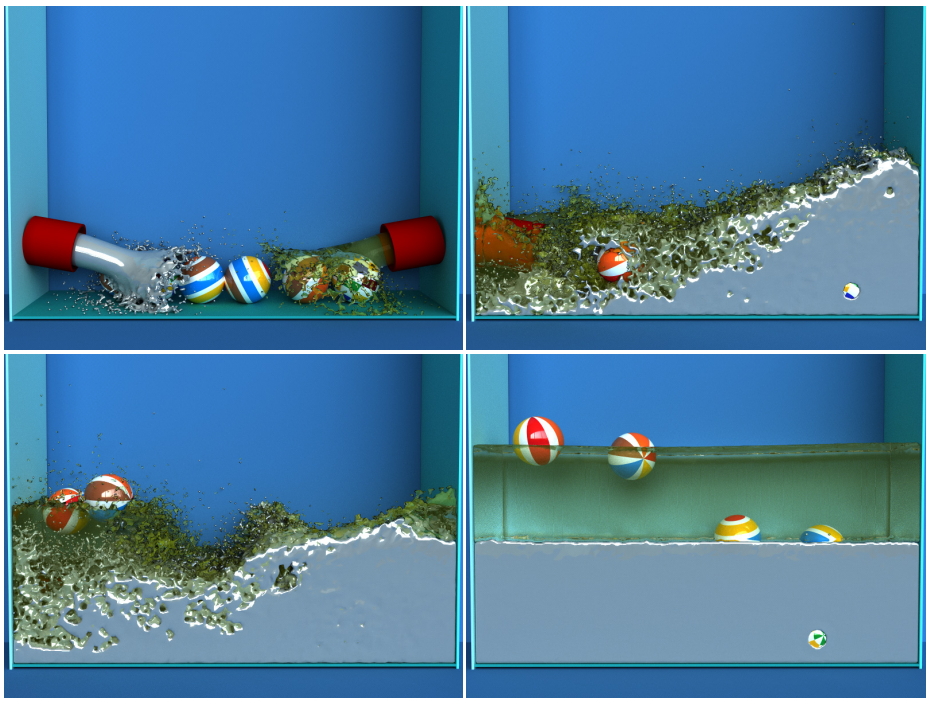


Figure 3.13 – Six balls with different densities are behaving differently in a multiphase setting. Images are taken from [AIA+12].

to the previous example, Figure 3.12, bottom-right, shows that our approach is able to interact with shell-like structures without interpenetration. In this scene, three cylindrical shells with different densities (which were represented with one layer of boundary particles) interacted with 200K fluid particles.

In order to show that our approach works with multiphase fluids, we simulated two fluids with a density ratio of 1:3 and several spheres with different densities (see Figure 3.13).

Our approach can handle large density ratios between fluid and rigid bodies. A scene is illustrated in Figure 3.14, where a sphere with variable density interacted with 90K fluid particles. The density of the sphere was changed from $1\frac{kg}{m^3}$ to $10\frac{kg}{m^3}$, $100\frac{kg}{m^3}$ and $500\frac{kg}{m^3}$. Note that at this point of the sequence, exactly half of the sphere was below the water surface. Finally, the sphere density was changed to $50000\frac{kg}{m^3}$.

A more complex ragdoll example is shown in Figure 3.15 where 2M fluid particles were used. Due to the dynamics of the individual rigid parts of the ragdoll, the boundary particle sampling can dynamically change in the neighborhood of a fluid particle. The sampling density is, however, considered in our equations so that discontinuities and large forces are prevented. Furthermore, we show a non-uniformly sampled frigate traveling on wavy sea that was simulated using 4M particles (see Figure 3.4). We also extended the experiment by placing three frigates behind a dam. After the dam broke, the frigates were hit by turbulent waves and the cuboid pieces of the dam. 20M fluid particles were used for this experiment (see Figure 3.16).

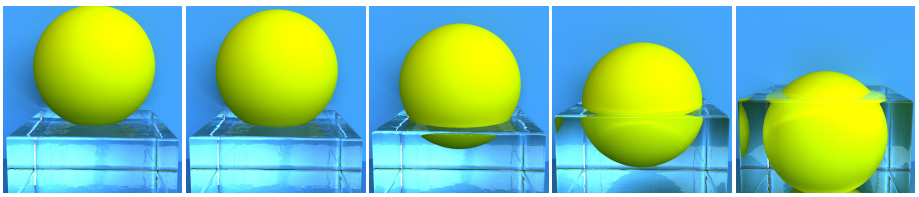


Figure 3.14 – From left to right, the density of a sphere was slowly increased from $1\frac{kg}{m^3}$ to $50000\frac{kg}{m^3}$. Images are taken from [AIA+12].

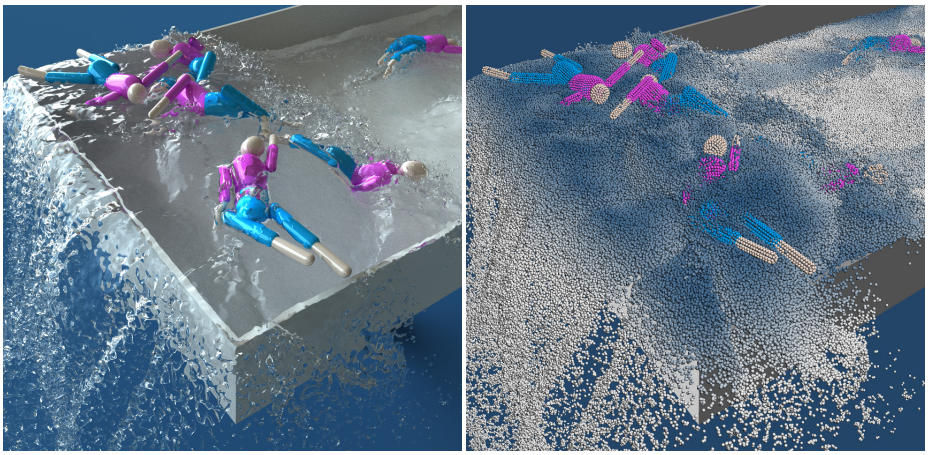


Figure 3.15 – Dynamically moving rigid bodies in close proximity can change the sampling density in neighborhood of a fluid particle. Since our method takes variable boundary sampling into account, discontinuous forces are prevented. Images are taken from [AIA+12].

In another example, two towers were modeled using cubes and cylinders. Each cylinder and cube had a density of $600\frac{kg}{m^3}$ and $1500\frac{kg}{m^3}$, respectively. Due to their different densities, primitives interacted differently with the flow that was simulated with up to 2.5M particles (see Figure 3.17). Finally we show a complex scenario with various rigid objects (see Figure 3.1). In this scene, two boats filled with ragdolls were dropped into a river like flow. While the second boat was floating, we collapsed the bridge by removing the constraints on both sides of the bridge. Up to 6M fluid particles were used in that simulation.

3.5 Discussion and Future Work

In our simulations, we generated boundary particles for all rigid bodies including static planar objects (e.g. water containers). For such regions, it would be more efficient to use a wall weight function to approximate the density and force contributions, as done in [HKK07]. For complex boundaries with varying triangle size or in non-manifold regions, however, defining a wall weight function is difficult. In those cases, the triangle mesh should be re-meshed to get a nearly isotropic triangle distribution. Our boundary particles can be related to such an isotropic triangle mesh, without topology and normal information.

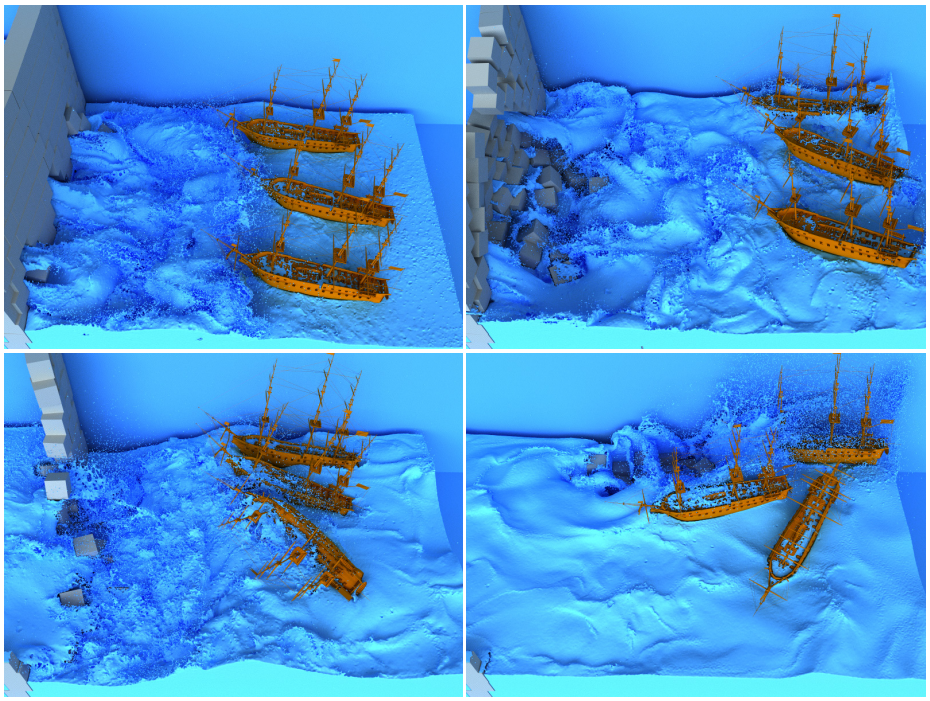


Figure 3.16 – Three frigates are exposed to a dam brake scenario.

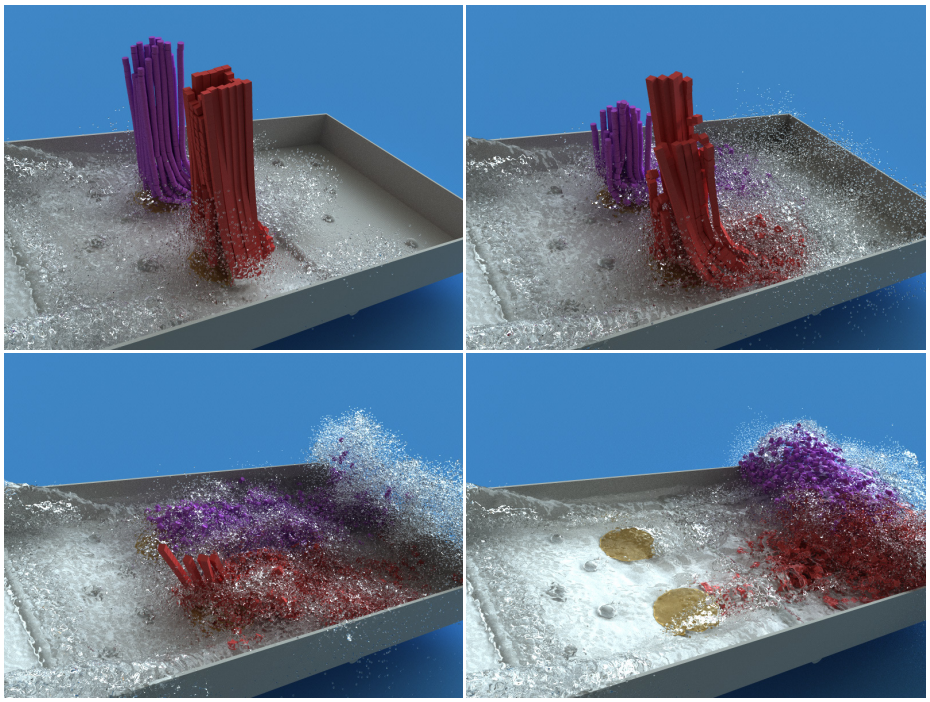


Figure 3.17 – The approaching water collapses two towers that were modeled by individual cylinders and cubes of different densities. Images are taken from [AIA+12].

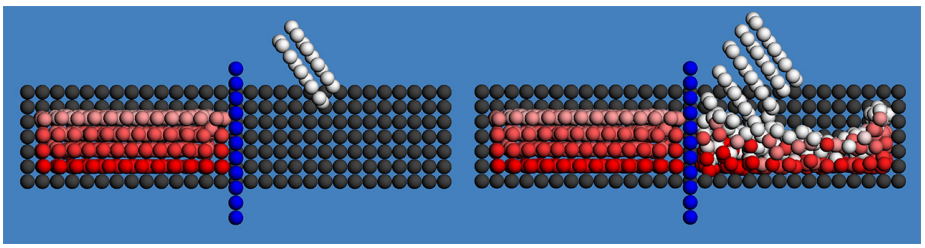


Figure 3.18 – The negligible interaction of fluid particles on opposite sides of a thin shell. Pressures on particles are proportional to their red saturation, black particles denote boundary particles, front side of the box has been clipped to make the fluid visible. Images are taken from [AIA+12].

Our approach computes boundary forces based on the pressure of fluid particles. While the boundary forces are appropriate for incompressible and weakly compressible fluids, the forces might not be sufficient to prevent interpenetrations when used with compressible fluids. This is particularly the case, if the computed pressure based on a certain density is much smaller for a compressible fluid compared to an incompressible fluid. Further, our approach is limited in terms of a minimal object size that can be handled. The diameter of rods and the thickness of shells cannot be smaller than the diameter of a fluid particle. Thus, the minimum possible object size is defined by the fluid particle resolution. Another issue of our approach is the interaction of fluid particles on opposite sides of thin boundaries. However, due to the coupling of minimal thickness of boundaries and particle resolution, these interactions hardly influence the behavior of the fluid as demonstrated in Figure 3.18. For improperly large time steps (that are larger than what is estimated by the employed adaptive time-stepping schemes), fluid particle tunneling may occur. However, this is a general problem that also exists for fluid-fluid interaction.

We generated the boundary particles such that they are completely enclosed by the rigid. Generating boundary particles exactly at the surface would cause stability issues when fluid particles stuck between two layers of overlapping boundary particles. These issues could be prevented by detecting such fluid particles and treating them differently. In our approach, fluid particles are not immediately updated after the collisions in the final rigid update of Algorithm 3.1. However, in the next iteration, based on the updated position of the rigid, forces are generated, and the fluid particles are updated. Although our approach outperforms the global approach in [BTT09], we believe that the investigation of alternative global approaches for a simultaneous coupling similar to [CGFO06] is a very promising direction for future research in particle-based fluids.

Our method could also be integrated into previously presented SPH frameworks. One way to employ our approach in [SG11] would be using two boundary samplings, one for the low-resolution simulation and another for the high-resolution simulation. Existing unified SPH models such as [SSP07, LD08] could be used in combination with our model to simulate the interaction of fluid, rigid, and deformable models, including thin shells such as cloth.

4
Deformable-Fluid Coupling

4.1 Introduction

As explained previously for the case of rigid objects, the basic SPH interpolation works well, only if the neighboring particles have similar properties. For all other cases, boundary-handling techniques that are specific to the problem of interest need to be applied. The boundary-handling and two-way rigid-fluid coupling method explained in the previous chapter offers several important useful features. First of all, the approach is completely based on hydrodynamic forces and conserves momentum. Secondly, it addresses the inhomogeneous boundary sampling by governing particle contributions based on the represented volumes of the boundary particles. The approach also supports fluid interaction with thin structures and non-manifold geometry. However, deformable objects are not addressed in that work, as the boundary particle configurations are precomputed and remain unchanged for the whole simulation. This chapter extends the boundary-handling and two-way rigid-fluid coupling method explained in the previous chapter to deformable-fluid coupling. For handling the interaction of SPH fluids and deformable solids, only few works exist [MST+04, LD08, DTM+12, YLHQ12]. When handling deformable boundaries, some important problems arise. On one hand, particle deficiency issues at the boundary need to be handled so as to prevent spatial and temporal discontinuities of the physical properties (e.g. density, pressure and velocity) of the fluid particles. On the other hand, in the case of large deformations, leakage through the boundaries must be prevented. Additionally, pairwise forces need to be symmetric for avoiding temporal artifacts. Our work addresses these open issues by adapting the rigid boundary-handling method in explained in Chapter 3 for deformation handling. In the remainder of this section, we discuss the existing works about deformable boundary-handling in SPH (Section 4.1.1), and then summarize our contribution (Section 1.2).

4.1.1 Handling Deformable Boundaries in SPH

Triangles are one of the most practical mesh representation primitives for defining solids in computer graphics. In SPH simulations, directly coupling triangle meshes with fluid particles poses some challenges. First of all, density estimation of fluid particles near boundaries is only possible for simpler boundaries [HKK07], but is difficult for arbitrarily complex ones. Furthermore, for meshes with ambiguous surface orientation (e.g. non-manifold meshes), computing fluid-

solid forces correctly is a challenging problem. These issues cause spatial and temporal discontinuities of the physical properties of fluid particles, which in turn introduce falsified forces to the solids in the case of two-way coupling. For a detailed discussion about these issues, we refer the reader to Chapter 3.

Interaction of SPH fluids with deformable objects is firstly presented in [MST+04], where the employed boundary forces are based on the Lennard-Jones potential. In this approach, boundary particles are automatically generated per triangle according to Gaussian quadrature rules. Boundary particles generated in this fashion, however, are not optimal for computing field variables (e.g. density and forces) for the nearby SPH particles, since the boundary particles are distributed non-homogeneously for different triangles with different sizes and aspect ratios. In [LD08], two way coupling of SPH with thin deformable shells is realized. In this work, deformable objects are also simulated using SPH, based on [SSP07]. This approach combines SPH forces with the explicit collision handling scheme of [BYM05] and position correction to prevent leakage in the case of deformations where the boundary particle spacing only changes marginally. The authors also state that for thin shells with low Young's moduli, their approach causes leaks. Recently, an SPH and cloth coupling method using continuous collision detection to prevent tunneling [DTM+12], and an SPH and Finite Element Method coupling method have been proposed [YLHQ12]. However, both techniques use penalty forces for the coupling, and do not handle the fluid densities correctly near the boundary. Therefore, they are prone to the problems we have described in the beginning of this section. An image based fluid-deformable interaction model has been briefly discussed in [ACF11]. In the context of Eulerian fluids, two-way fluid-shell coupling have been demonstrated in [RMSG+08].

4.2 Formalism

As mentioned in the previous section, the approach explained in [AIA+12] works well for fluid-rigid coupling. However, a static setup of boundary particles is not sufficient for deformable objects, since large deformations may cause gaps between boundary particles, which may cause undesired fluid leakage. In order to avoid this problem, dynamic boundary particle generation is necessary. Our algorithm takes the triangle mesh of the deformable object as input and generates boundary particles for the mesh based on its vertices, edges, and triangles, respectively. This section explains the main parts of our two-way fluid-elastic coupling algorithm.

4.2.1 Boundary Forces

As a quick reminder, this section will first summarize how the boundary forces explained in Chapter 3 works. In [AIA+12], the contribution of a boundary particle is related to its volume, which is estimated by using the inverse of the number density of a boundary particle b_i based on its boundary particle neighbors as

$$V_{b_i} = \frac{1}{\delta_{b_i}} = \frac{1}{\sum_k W_{ik}}, \tag{4.1}$$

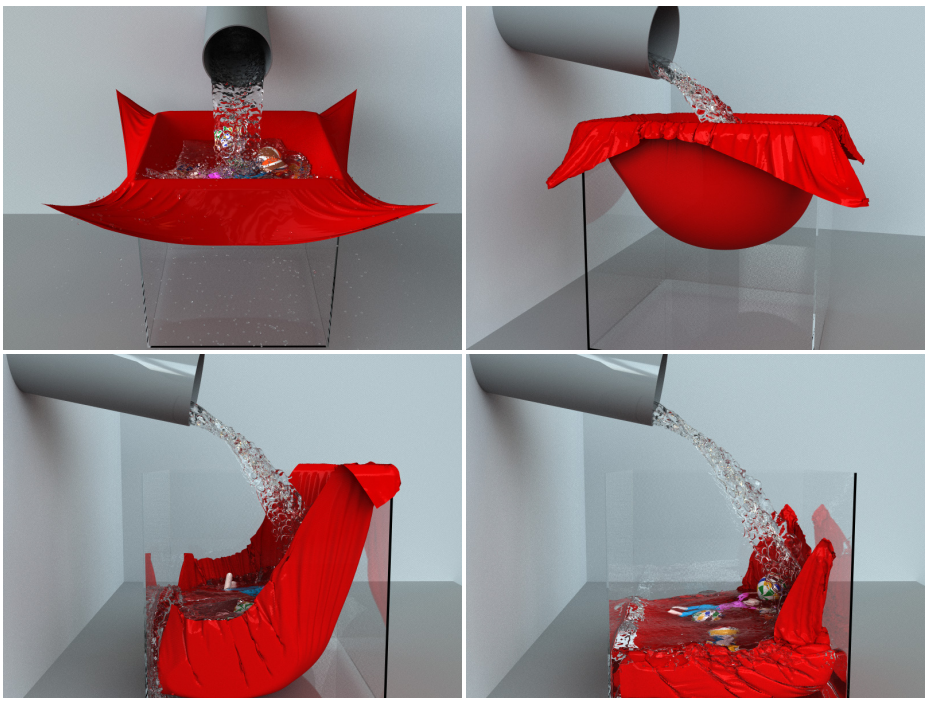


Figure 4.1 – An elastic cloth stretches as it is filled with water. Our approach prevents leakage even in the case of large expansions. Images are taken from [ACAT13].

where δ_{b_i} denotes number density of the particle, k denotes boundary neighbors, W is a kernel function with smoothing length h. Hereby, the contribution of a boundary particle in an SPH interpolation of a nearby fluid particle is given as $\Psi_{b_i}(\rho_0) = \rho_0 V_{b_i}$, where ρ_0 is the rest density of the fluid the boundary particle is interacting with. Using this volume based contribution concept, the density of a fluid particle is given as $\rho_{f_i} = \sum_j m_{f_i} W_{ij} + \rho_{f_i \leftarrow b}$, where j denotes fluid particle neighbors, and the density contributed by boundary neighbors is given as $\rho_{f_i \leftarrow b} = \sum_k \Psi_{b_k}(\rho_{0_i}) W_{ik}$. Using ρ_{f_i}, the resultant pressure is computed by either using [BT07], [SP09] or [ICS+13]. Afterwards, pressure and viscosity forces are applied to both types of particles. We refer the reader to Chapter 3 for those force computations.

4.2.2 Force Transfer

For rigid bodies, the net force and torque terms can be easily computed by accumulating the forces on all boundary particles of the rigid. For deformable objects, however, deformations result in many more degrees of freedom than just position, orientation, linear and angular momentum. Here, we will assume a mass-point-based deformable solver, where the vertices represent mass points and edges represent constraints (e.g. as in [Jak01]) . When generating the boundary particles per triangle as described in Sections 4.2.4 and 4.2.5, the barycentric coordinates describing the position of a particle with respect to the positions of the corresponding triangle's vertices are stored with the particle. As the mesh deforms, the particles are synchronized with the mesh according

to these barycentric coordinates similar to [MST+04]. When distributing the forces from boundary particles to the mass points, barycentric coordinates are again used for force weighting.

4.2.3 Boundary Requirements

The above density and force approximations are valid only if the contribution of boundary particles to a fluid particle $\rho_{f_i \leftarrow b}$ is defined uniformly on the surface of the solid, since this term or its derivative is used in all force terms. We determined that, for a 2D simple cubic boundary particle alignment with a spacing equivalent to fluid particle spacing, $\rho_{f_i \leftarrow b} \approx 0.35\rho_0$ when a fluid particle's normal distance to a boundary is less than $h/2$. When $\rho_{f_i \leftarrow b}$ is considerably smaller than this value at some point on the surface, fluid particles can leak through the solid at that point. We will refer to such boundary particle alignments as under-sampled. Such alignments can frequently occur when simulating expanding deformable objects, as the distances between neighboring boundary particles get larger. The inverse situation, i.e. over-sampling, occurs when there are too many boundary particles that do not further improve the approximation of $\rho_{f_i \leftarrow b}$. Although [AIA+12] can handle over-sampling by adapting boundary particle to fluid particle contributions according to (4.1), over-sampling causes performance overhead since the number of neighboring particles increases. It should be noted that, since $\rho_{f_i \leftarrow b}$ essentially depends on the boundary particle number density δ_b, we can use δ_b at boundary particle positions as a metric to decide if the resampling of a region is necessary. We experimentally determined that $3.6\,\Gamma(h)$ and $10\,\Gamma(h)$ are appropriate thresholds for detecting under-sampling and over-sampling respectively, where $\Gamma(h)$ is SPH kernel function dimensional factor and defined as $\Gamma(h) = \frac{1}{h^d}$ and d denotes the simulation dimension. Our algorithm starting with Section 4.2.4 deals with creating such a sampling without causing any under-sampling, and with minimal over-sampling. The analysis of our approach in terms of field variable approximation is given in Section 4.2.6.

4.2.4 Initial Boundary Particle Generation

As the boundary forces explained in Section 4.2.1 are SPH based, we aim to generate an initial boundary sampling, where we keep the sampling density of the boundary close to fluid particle sampling density. We observed that in incompressible SPH simulations, spacing between fluid particles remains around half of the SPH smoothing length. This implies that for SPH based fluid-boundary interaction, the sampling spacing of the boundary particles should be as close to fluid sampling spacing as possible to achieve the same sampling density. This motivated us to create a geometric boundary particle generation scheme, where the spacing between boundary particles is close to fluid particle spacing. Our algorithm not only satisfies the under/over-sampling requirements given in the previous section (i.e. $\rho_{f_i \leftarrow b} \approx 0.35\rho_0$ and $3.6\,\Gamma(h) < \delta < 10\,\Gamma(h)$), but it is also very efficient and suitable for locally adaptive resampling (see Figures 4.3 and 4.4).

For the initial boundary particle sampling, our algorithm generates particles in three steps: Placing particles at vertices, particle generation for edges and sampling triangle interiors with particles.

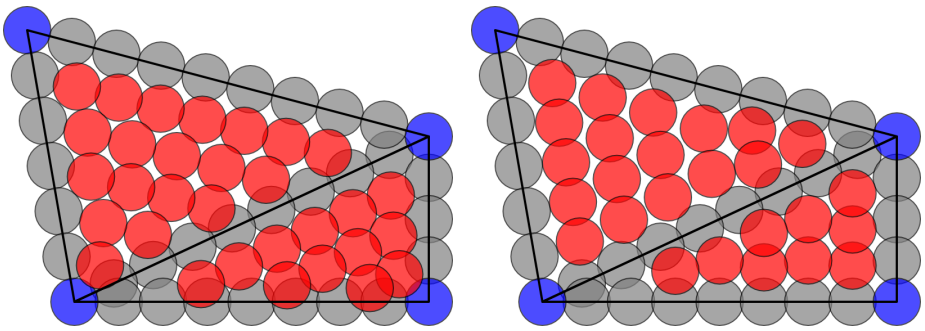


Figure 4.2 – Two adjacent triangles sampled with scan-lines in the direction of their longest edges (left). Same triangles sampled in the direction of their shortest edges (right), which results in better particle distribution, with less over-sampling. Blue, gray and red particles denote vertex, edge and interior boundary particles respectively. Images are taken from [ACAT13].

The first step in edge sampling is determining the number of particles n_e to generate, which is simply done by dividing the edge length $|\mathbf{e}| = |\mathbf{v}_a - \mathbf{v}_b|$ by the particle diameter $d = \frac{h}{2}$ as $n_e = \left\lfloor \frac{|\mathbf{e}|}{d} \right\rfloor$. The displacement vector $\mathbf{p}_e$ between the particles is computed as $\mathbf{p}_e = \frac{\mathbf{e}}{n_e}$. Afterwards, particles can be placed along the direction of the edge starting from any of its vertices with $\mathbf{p}_e$ displacements.

After sampling triangle edges, the next step is sampling the triangle interiors. The approach we employ is similar to scanline algorithms known from rendering. In our algorithm, firstly the shortest edge $\mathbf{e}_s$ is determined, and its normal direction $\widehat{\mathbf{s}}$ in the direction of the triangle interior is computed as $\widehat{\mathbf{s}} = \frac{\mathbf{e}_s \times (\mathbf{e}_l \times \mathbf{e}_s)}{|\mathbf{e}_s \times (\mathbf{e}_l \times \mathbf{e}_s)|}$, where $\mathbf{e}_l$ is the longest edge. Afterwards, the number of required sweeping steps n_t for the triangle is computed as $n_t = \left\lfloor \frac{h_t}{d} \right\rfloor$, where h_t is the height of the triangle in the sweeping direction $\widehat{\mathbf{s}}$, which is computed by projecting $\mathbf{e}_l$ onto the sweeping direction $\widehat{\mathbf{s}}$ as $h_t = \widehat{\mathbf{s}} \cdot \mathbf{e}_l$. For each sweeping step, the line intersection positions $\mathbf{i}_1$ and $\mathbf{i}_2$ with the other two edges are computed as explained in [Hil94]. Finally, particles are placed between the edge intersections with a displacement vector $\mathbf{p}_s = \frac{\mathbf{i}_1 - \mathbf{i}_2}{n_s}$ where $n_s = \left\lfloor \frac{|\mathbf{i}_1 - \mathbf{i}_2|}{d} \right\rfloor$.

According to our experiments, choosing the shortest edge as the scan direction results in a better boundary particle distribution with less over-sampling (see Figure 4.2). The employed step sizes in the particle generation result in a boundary sampling that is sufficiently dense according to the criterion introduced in Section 4.2.3. Furthermore, separate handling of triangle vertices and edges from triangle interiors allows robust handling of arbitrary mesh configurations without any under-sampling. On the other hand, the approach generates some over-sampling, especially between the edges of skinny triangles. However, due to the employed boundary-handling scheme that can handle over-sampled boundaries (despite some performance overhead), intersections between particles can be handled when computing boundary forces.

4.2.5 Resampling of Deformed Regions

One way to completely avoid the under-sampling problem is to regenerate all boundary particles in each simulation step using the algorithm explained in

Section 4.2.4. However, in stable solid simulations, deformations are temporally coherent. We utilize this information by introducing measures for determining whether a triangle needs to be resampled, or if the existing sampling is sufficient. Therefore, we avoid complete boundary sampling of a deformable object in each simulation step by using samplings from previous simulation steps as possible.

As discussed in Section 4.2.3, when under- or over-sampling is detected (i.e. $3.6\,\Gamma(h) < \delta < 10\,\Gamma(h)$) , the triangle enclosing the boundary particle, and all its edges can be resampled. Even though the resampling criterion based on δ_b is closely related to the SPH concept, it has some important disadvantages. First of all, since it relies on δ_b values, lazy updating of δ_b is not possible (i.e. updating only when a boundary particle is in the neighborhood of the fluid). Furthermore, it needs an additional evaluation of the δ_b values, which requires an additional neighborhood search among boundary particles. So as to avoid these performance issues, we derived a heuristic for resampling detection that is purely based on geometric measures, which aims to generate a full boundary neighborhood for fluid particles. In our geometric approach, triangle edges and interiors are handled and resampled separately. For detecting whether edge resampling is necessary, the number of required boundary particles n_e is computed (see Section 4.2.4). Afterwards, the edges where n_e changes between two consecutive simulation steps are resampled. For detecting whether triangle interior resampling is necessary, we employ additional rules. In each simulation step, firstly the shortest edge $\mathbf{e}_s$ of a triangle is determined. When $\mathbf{e}_s$ becomes another edge between two simulation steps, the triangle is resampled. If not, the number of necessary scanline steps n_t is computed. By multiplying the number of particles sampled onto the smallest edge n_{e_s} of the triangle with n_t, the number of particles n_{pa} that are necessary to sample the considered parallelogram is computed. When n_{pa} for a triangle changes between two consecutive simulation steps, the triangle's interior is resampled. A comparison of the two resampling strategies (i.e. the δ_b based and the geometric heuristic based) is shown in Figures 4.3 and 4.4. Note that our geometric approach results in similar samplings when compared to δ_b based sampling, where $3.6\,\Gamma(h) < \delta < 10\,\Gamma(h)$ for any point in a triangle is always satisfied.

4.2.6 Comparison to Other Boundary Sampling Strategies and Analysis

In [IAGT10] and [AIA+12], the remeshing algorithm explained in [BK04] is used for improving isotropy of triangle meshes. After the remeshing step, boundary particles are placed at vertex positions of the resultant mesh. Even though this approach might generate better samplings compared to our method, especially in low-curvature regions (see Figure 4.5), the performance cost of the approach makes it unsuitable for the application to deformable objects, which might require boundary particle generation in each simulation step. Poisson disk sampling has also been used for boundary particle generation in [SB12], which however requires relaxation iterations and neighborhood search when placing particles. Furthermore, similar to the remeshing based method, it is designed for producing a global boundary particle distribution, which makes it inappropriate for deformable objects, where local particle insertions and deletions are usually sufficient and very efficient.

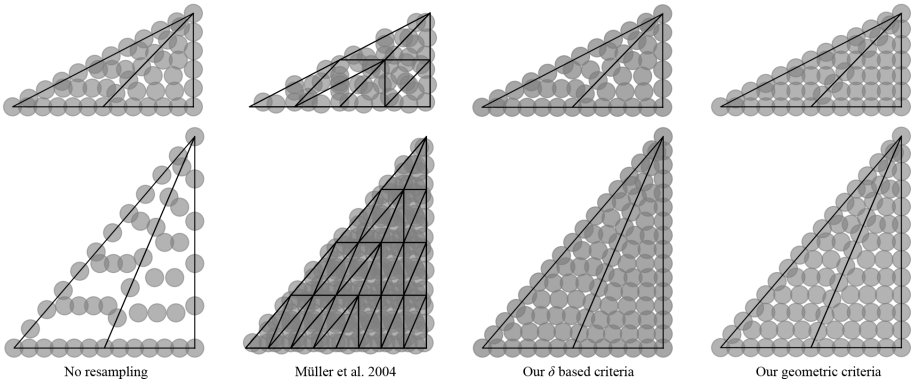


Figure 4.3 – The initial sampling of two adjacent triangles with boundary particles based on four different boundary particle resampling schemes are shown in the top row. After the triangles are deformed, the resultant samplings are shown in the bottom row. The left most column shows the outcome when no new particles are generated. The seven point rule used in [MST+04] tends to generate non-uniform sampling patterns. Note that our geometric resampling criterion is in good agreement with our δ based criterion and both produce good sampling patterns without under-sampling and with minimal over-sampling. (The plotted particle diameter is equivalent to $\frac{h}{2}$). Image is taken from [ACAT13].

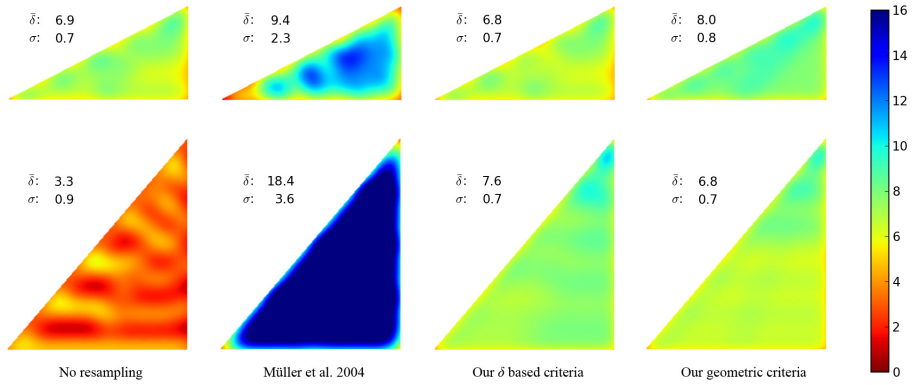


Figure 4.4 – The number density δ is used as a metric to evaluate under- and over-sampling. The δ value for $h = 1$ is computed at each point inside the triangles in Figure 4.3, where $\bar{\delta}$ denote arithmetic mean and σ standard deviation. $\delta \lessapprox 3.6$ denotes under-sampling and may result in fluid particle tunneling through the boundary. Whereas, $\delta \gtrapprox 10$ denotes over-sampling and causes unnecessary computational overhead. Note that the underlying formulation of [AIA+12] results in smooth boundary forces as long as the under-sampled regions are avoided. Using [MST+04] results in a non-uniform δ field, with variable $\bar{\delta}$ and σ as the triangles deform. Such large variances can cause disturbance to the nearby fluid particles. Both of our approaches result in a uniform δ field, with minimal temporal variance of $\bar{\delta}$ and σ as the triangles deform. These properties make our approach a good match for generating boundary particles in [AIA+12]. Images are taken from [ACAT13].

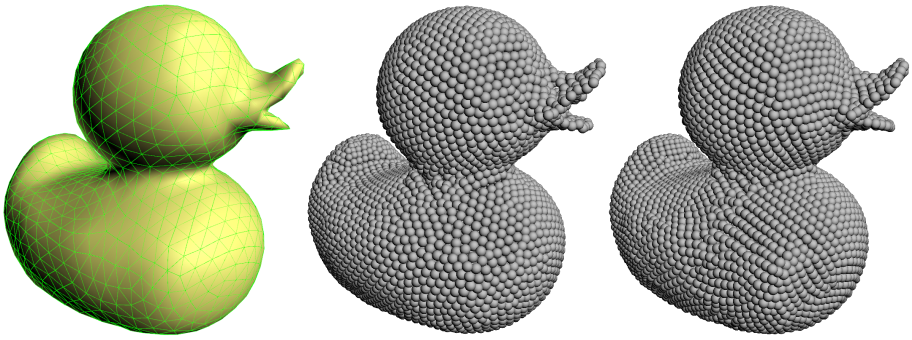


Figure 4.5 – A duck model composed of 2160 triangles (left), sampled with boundary particles by remeshing the model using [BK04] and placing particles at vertex positions in 1.4 s (as done in [IAGT10, AIA+12]) (center). The same mesh is sampled using our algorithm in 15 ms (right). Image is taken from [ACAT13].

[MST+04] proposes a seven-point rule to generate boundary particles. Based on a user defined threshold, which is chosen relative to the smoothing length h of the fluid particles, a triangle is subdivided into four smaller triangles, which are again sampled with seven boundary particles recursively. Because of the employed subdivision technique, the number of particles sampled per triangle is $4^d \times 7$, where d is the recursion depth of the triangle subdivisions. Therefore, for small deformations, the approach cannot generate only few particles per triangle, but it generates four times more particles in each subdivision. If the sampling threshold is chosen small, it may result in severe over-sampling of the triangles, while a large threshold may cause gaps between the boundary particles. Whereas, our approach only adds or removes small number of particles in case of small deformations. The difference between both methods applied to two neighboring triangles in both the initial pose and after a deformation are depicted in Figure 4.3. As can be seen, our method results in more homogenous boundary particle sampling. In contrast, [MST+04] results in under-sampling and over-sampling for different parts of the triangles. Even though smaller sampling thresholds might be used to prevent under-sampling, it might cause further over-sampling for already over-sampled triangles. One important disadvantage of over-sampling is that the required time for both neighborhood search and force computations increases significantly. Furthermore, for approximating field variables of fluid particles near the boundary, our method provides better boundary particle distributions without under-sampling and with much less over-sampling. To quantify the difference, in Figure 4.4, we computed the number density δ at each point inside the triangles based on the generated boundary particles for the boundary configurations given in Figure 4.3. Note that our strategy results in a more homogenous distribution of δ. Whereas, if [MST+04] is used, the deviation is considerably larger. These properties make our approach a good match for using with [AIA+12] for boundary particle generation.

Another triangle sampling strategy that can be applied to local triangle sampling has been briefly explained in [Cor05] in the context of point-based level sets. However, the case of deforming triangles is not explained in that work.

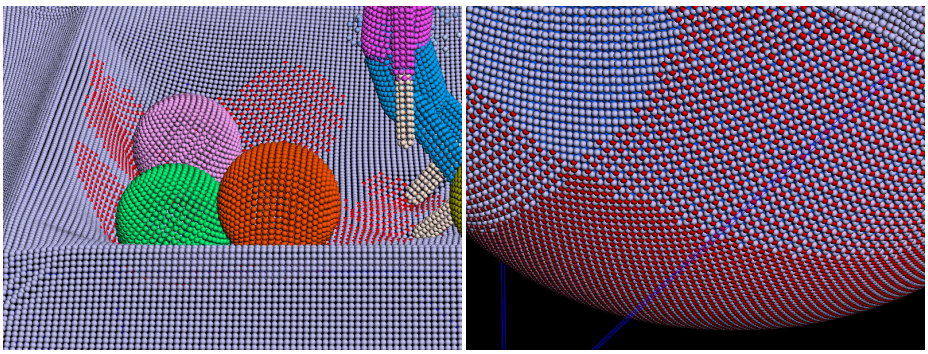


Figure 4.6 – Underlying boundary particles of the stretching cloth in Figure 4.1. The red regions denote the additional particles added to the initial boundary sampling. Image is taken from [ACAT13].

4.3 Implementation Details

The simulation loop in our approach is similar to the one used in [AIA+12] (Algorithm 3.1), with some differences that are related to the deformable-boundary interface as discussed in Section 4.2.2. For simulating rigid and deformable bodies, we use Bullet Physics [Cou11]. For SPH simulations, we use the predictive-corrective formulation presented in [SP09] for computing fluid pressures. For time step selection, we use the scheme presented in [IAGT10], where the velocities of boundary particles are also included in the time step estimation criteria. Finally, fluid surfaces are reconstructed using [AIAT12], and final renderings are done using mental ray [NVI11].

4.4 Results

In this section, we demonstrate the versatility of our method using different scenarios. For all experiments, we used the geometric approach for resampling detection because of its performance benefits. Several simulation steps were computed for each frame. For each animation sequence, the average computation time per frame was 10 to 30 seconds, excluding surface reconstruction and rendering, depending on the frame complexity. The employed adaptive time-stepping scheme produced time steps roughly between 5×10^{-3} and 10^{-4} seconds. In our simulations, we used different particle radii r for different scenarios, where the SPH smoothing length h was always chosen as $4r$. Fluid-boundary adhesion constant β was chosen as 0.05. The additional overhead of our approach over [AIA+12] was below 5% in all scenes, excluding deformable simulation. The simulations were run on an Intel Xeon X5690 with 12 GB RAM.

In the scene corresponding to Figure 4.1, an elastic cloth with density $1200 \frac{kg}{m^3}$ consisting of 90K uniform triangles was stretched above a box using point constraints at its corners. Afterwards, several rigid objects with different densities were dropped onto the cloth, which was later exposed to a steady fluid flow from its top, consisting of up to 1M fluid particles. The weight of the rigid objects and the fluid caused the cloth to slowly stretch, which was handled by updating the boundary particles in order to prevent the leakage of

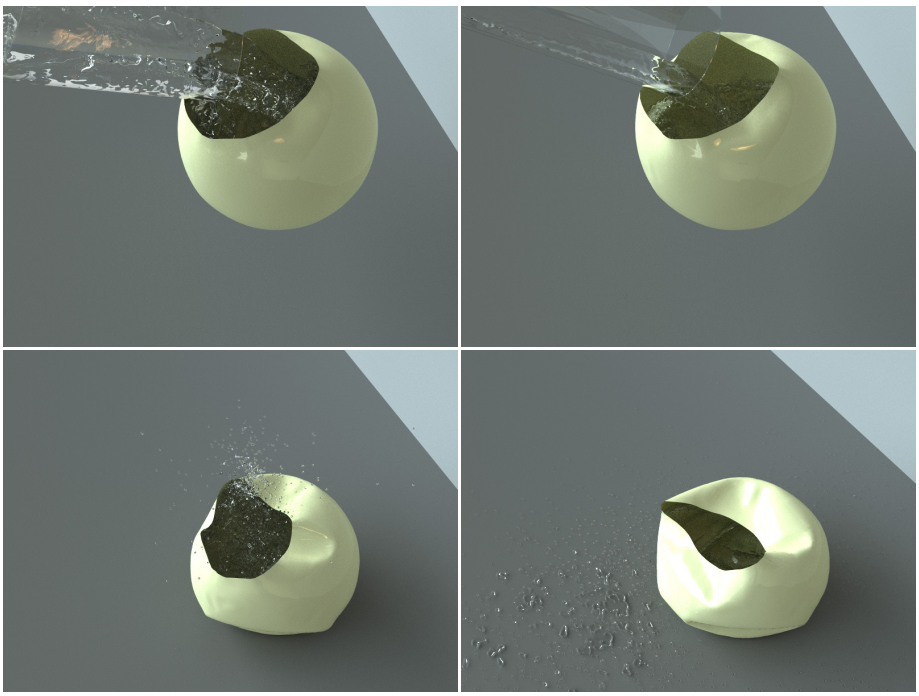


Figure 4.7 – A thin bowl is filled with water and dropped onto the ground. Our approach can handle rapid deformations. Images are taken from [ACAT13].

fluid particles through the cloth (see Figure 4.6). Afterwards, when the cloth was released and shrunk back to its original size, all additional boundary particles were automatically removed. Our method ensures a sufficient boundary particle sampling that prevents leakage. Because of the regular structure of the cloth, our algorithm generated one boundary particle for each vertex for the initial sampling of the surface, which resulted in 46K boundary particles in 22 ms. Whereas, the number of boundary particles for the cloth's maximum stretched state was 62K. The average computation time for the resampling was 9 ms per simulation step.

The presented method is also applicable to fast deformations (see Figure 4.7). In this scene, a thin bowl with density $1000\frac{kg}{m^3}$ consisting of 6754 triangles was constrained and filled with 110K fluid particles. The initial sampling of the bowl took 14 ms that generated 42K boundary particles. When the bowl was filled with fluid, the constraints were released and the bowl was dropped onto the ground. The impact caused a sudden deformation, and consequently a rapid change in the number of boundary particles up to 44K. In this example, the average computation time for resampling was 2.5 ms per simulation step.

In order to show that our boundary particle sampling also works with volumetric deformable objects, we created another scenario (see Figure 4.8). Several volumetric deformable objects and a thin shell cloth with density $300\frac{kg}{m^3}$ interacted with a fluid that was represented by 1.2M fluid particles. The application of our geometric resampling approach to a volumetric deformable object took under 1 ms on average.

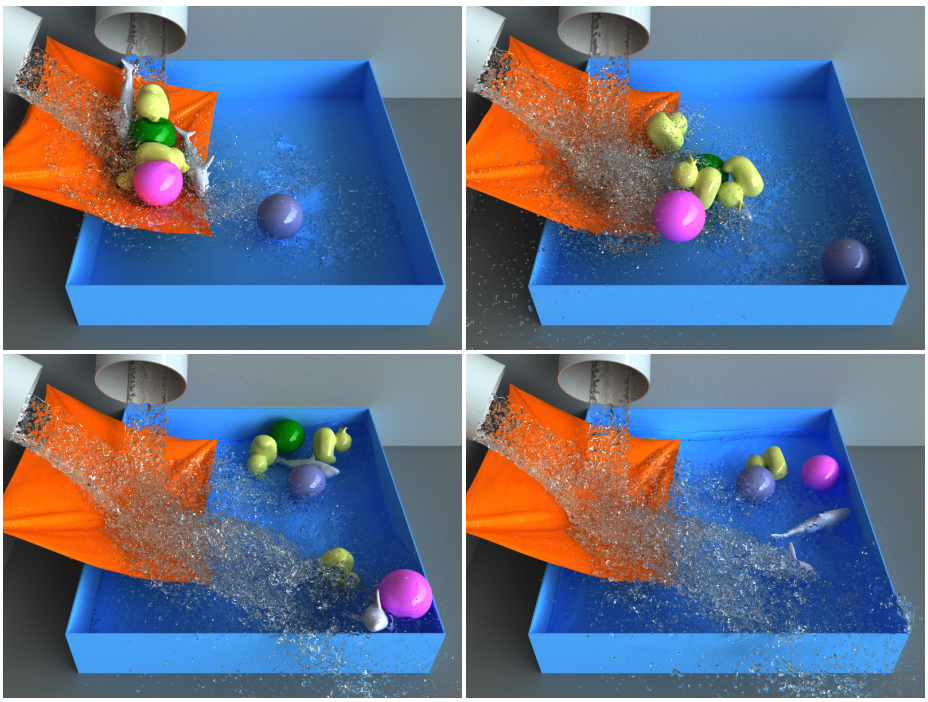


Figure 4.8 – Interaction of fluid with a cloth and several deformable solids. Images are taken from [ACAT13].

4.5 Discussion and Future Work

Since our method always places particles at vertex positions, it can cause over-sampling when the average edge length of a triangle mesh is less than the particle diameter. So as to prevent such over-sampled areas from affecting the simulation performance negatively, such meshes can be remeshed before the simulation.

Besides the deformation model employed in the chapter, we believe that our approach can work with different deformation models, as long as the deformations can be mapped to a triangle mesh and the forces can be transferred to the deformation model through the mesh vertices.

Our approach is also applicable for the boundary particle sampling of static and dynamic rigid objects with significant performance benefits over existing methods, as discussed in Section 4.2.6.

Because of the local nature of our boundary particle generation approach, it might be possible to extend it such that boundary particles are adaptively generated only for the triangles that are in the smoothing length h of the fluid particles. This can allow SPH simulations in very large solid domains (e.g. terrains, whole cities), where the boundary particle sampling of the whole domain might be infeasible because of large memory and computational requirements.

5

Surface Tension and Fluid-Solid Adhesion

5.1 Introduction

Surface tension is a ubiquitous effect in daily life. For instance, when pouring water into a glass, the force that keeps liquid molecules together is the surface tension force. It is caused by cohesive forces among neighboring fluid molecules. Inside the fluid, each molecule is pulled equally by its neighbors, resulting in a zero net force. However, as the free surface does not have neighbors on all sides, the molecules in such regions are pulled inwards. Furthermore, surface tension minimizes surface area according to Laplace's law, which causes droplets of water to form a sphere when external forces are excluded. Another effect that is again caused by molecular interaction is adhesion. Adhesion allows fluids to get attracted by other materials. For instance, the unique appearance of dew on plants and the ability of water striders to stay atop water are caused by the interplay of surface tension and adhesion forces. In this chapter, we focus on simulating those two molecular interaction related phenomena in the context of computer animation, more specifically for SPH fluids.

One important issue that arises at fluid-air and fluid-solid interfaces in SPH is density underestimation, where densities of the particles are erroneously computed as less than the rest density when the density summation approach is used. Those wrong density values result in negative pressures and cause the particles to cluster, which is known as tensile instability in SPH. This phenomenon can be alleviated by using artificial pressure forces [Mon00, MM13], which, however, result in spurious surface tensions. For this reason, either a density correction technique [She68], or simply not allowing negative pressures are other common practices for avoiding tensile instability. However, this still does not solve the problem of sticking particles at the fluid interface, since the pressure field is still not reconstructed in a physically sensible way. [AIA+12] addresses this issue for fluid-solid interface by pre-computing a single layer of boundary particles for the solid boundaries, which also extends to two-way fluid-solid coupling. In [SB12], ghost SPH particles are dynamically generated at both fluid-solid and fluid-air interfaces.

For modeling surface tension in SPH, additional techniques are generally preferred. These can be listed as: curvature based external forces on particles (e.g. [MCG03]), pairwise forces based on cohesion (e.g. [BT07]), a modified SPH formulation [CBP05], and more recently forces based on surface mesh curvature [YWTY12]. However, there is no single approach, which can handle very large surface tensions, avoids particle clustering at the free surface, minimizes surface

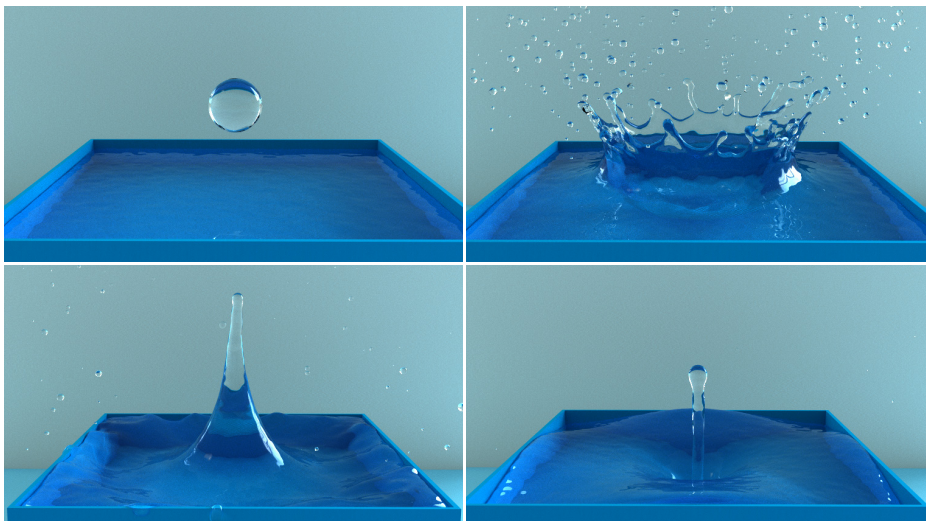


Figure 5.1 – A water crown emerges as a result of the impact of a water droplet into a filled container. Our surface tension force allows realistic simulation of such natural phenomena. Images are taken from [AAT13].

curvature, and conserves momentum at the same time. Adhesion of fluids to solids is modeled in SPH by using a distance based attraction force [CBP05] and by using the combination of ghost solid particles and XSPH [SB12] between the fluid and the solid boundary. However, neither of the methods is capable of simulating some important real world scenarios such as different wetting effects. Furthermore, the effect of adhesion forces on the solids is neglected in both works. Our surface tension and adhesion schemes are the first to meet all these requirements, and allow plausible simulations of variety of effects that emerge in reality. Additionally, our surface tension force avoids particle clustering at the free surface without generating ghost air particles using [SB12] or artificial pressure forces [Mon00]. In the remainder of this section, we first discuss existing works that model surface tension and adhesion effects in fluid animation with an emphasis on SPH fluids, and then we highlight the benefits of our techniques in comparison to the existing work.

5.1.1 Surface Tension in Fluid Simulation

As surface tension has a quite significant role in the appearance of liquids, many researchers have investigated techniques for incorporating surface tension to fluid simulations. In the context of SPH, the early surface tension techniques focus on applying forces to minimize surface curvature [Mor99, MCG03]. Those approaches compute normals for each particle, which determine the direction of the force and can be computed using the gradient of the smoothed color field. The magnitude of the force is based on the curvature at a particle position, which can be computed by taking the second derivative of the smoothed color field, or the divergence of the normal field. However, there exist important issues with such techniques. First of all, for the particles that are inside of the fluid, normalizing the smoothed color gradient can result in arbitrary normals for inner

particles. This problem has been avoided by applying the surface tension force to the particles whose smoothed color gradient has a magnitude larger than a threshold value (which however results in discontinuous forces). The second problem is that curvature estimation is very sensitive to particle disorder in SPH because of requiring the second derivative. The third problem with those techniques is that the forces are applied to the fluid particles as external forces in a non-symmetric way, which invalidates momentum conservation.

Because of the important limitations of the approaches that are based on surface curvature and normal information, researchers proposed new techniques to address the surface tension problem on a molecular level by using cohesion forces between neighboring fluid particles [TM05, BT07]. Therefore, these techniques avoid both normal computation and the erroneous curvature estimation. Another benefit of these techniques is that they trivially conserve momentum as the applied forces are pairwise symmetric. However, only using cohesion forces between fluid particles does not guarantee surface area minimization as the forces can trivially balance each other in a form that does not necessarily correspond to the smallest surface area. As we will show later in the chapter, large cohesion forces between particles can also result in unrealistic fluid patterns, such as cobweb-like elongating fluid structures. For large surface tensions, such structures do not easily break as using attraction forces alone does not minimize surface area of the fluid, but strengthens the already existing structures. Another approach that applies attraction forces between neighboring fluid particles to generate surface tension is [CBP05]. In this work, the basic SPH scheme is reformulated by using double density relaxation, where the surface tension force is computed based on the negative pressures that arise at the free surface similar to [Mon00]. However, as the surface tension arises as a side effect, different surface tension behaviors cannot be modeled with this work. Furthermore, this work also does not take surface area minimization into account.

More recently, [YWTY12] proposed another solution to the surface tension problem in SPH. In this work, the curvature is estimated on the fluid surface mesh, but the generated surface tension forces are applied to the adjacent fluid particles enclosed by the mesh. They show that when the surface mesh has more samples than the fluid (i.e. more vertices than the particles); the curvature computation is not as error prone as it is when computing curvature from the particles. Furthermore, a nice comparison of the method to [BT07] is given for the experiment where a cubic droplet deforms to a sphere. However, there exist some limitations of the technique. First of all, the employed surface tracking scheme may fail to detect isolated fluid pieces in regions where the mesh resolution remains coarse; this prevents generating surface tension for such areas. Furthermore, the quality of the surface tension depends on the tracked mesh resolution. Finally, requiring an explicit representation of the fluid surface in each simulation step is an overhead for the cases where an explicit surface is not required (e.g., for interactive scenarios such as [MM13], or when an efficient view dependent surface reconstruction scheme is preferred [FAW10]).

In the context of grid based fluids, there exist many different techniques for incorporating surface tension: e.g., from a level set function [KFL00], by employing an octree structure for more accurate force evaluation [LGF04], by treating surface tension as discontinuous boundary conditions [HK05], based on surface energy [MBE$^+$10, BUAG12], and based on the surface mesh [BBB10, TWGT10]. As we focus on fully Lagrangian flow, a detailed discussion of these

works is beyond the scope of this chapter.

5.1.2 Solid Adhesion in Fluid Simulation

[SCED04] proposed a fully Lagrangian approach for the simulation of viscous liquids, including adhesion to solids. They define adhesion properties of different types of materials using distance-dependent forces. However, the employed linear density kernel and strict anti-penetration constraints limit their approach to highly viscous liquids. Later in [CBP05], adhesion of viscoelastic SPH fluids to solids is modeled by using a distance based attractive force term, which is added as an impulse to the fluid particles. The authors demonstrated interesting scenarios such as droplet formation and sticking of fluids to solids. More recently in [SB12], fluid to solid adhesion is accomplished by computing a ghost velocity at each solid particle by combining solid's own velocity and the tangential component of the nearest fluid particle's velocity. After this step, fluid to solid adhesion is generated using an XSPH based artificial viscosity term. In [HLW$^+$12], sticking of fluid particles to solids arises as a side effect of the employed velocity constraints to realize different slip conditions, which makes it difficult to model different adhesion related effects.

In the context of Eulerian approaches, adhesion of fluids to solids can be accomplished by adjusting the velocity or pressure constraints enforced along the boundaries (e.g. [GBO04]). In addition to the general techniques, adhesion of fluids is modeled to animate variety of interesting scenarios such as sticking of viscous threads [BAV$^+$10] and sheets [BUAG12] to solids, wetting of hair [RKN12], animating droplets on a glass surface [CCW12] and sintering of snow [TF12].

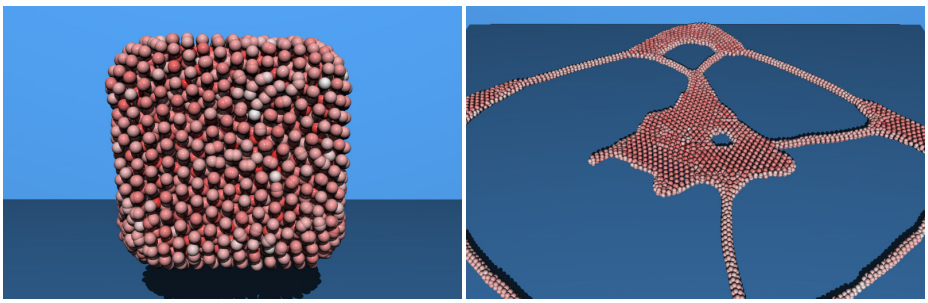
5.1.3 Contributions

We present a surface tension force and a fluid-solid adhesion force for the improved treatment of fluid-air and fluid-solid interfaces in SPH fluids. Our surface tension approach can handle large surface tensions by minimizing surface area in all scales while conserving momentum. Furthermore, our surface tension force also generates repulsion forces for close distances, which prevents the particle-clustering problem at the free surface without requiring additional treatment such as generating ghost air particles or artificial pressure forces. Our adhesion force allows physically plausible solid-fluid adhesion effects without requiring additional handling such as ghost SPH. Furthermore, our approach allows simulating interesting phenomena such as different wetting effects and two-way adhesion. Both of our forces can be easily added to an existing SPH solver as additional force terms without any extra effort. By combining our surface tension and adhesion forces, we are able to simulate a variety of effects that can be observed in nature.

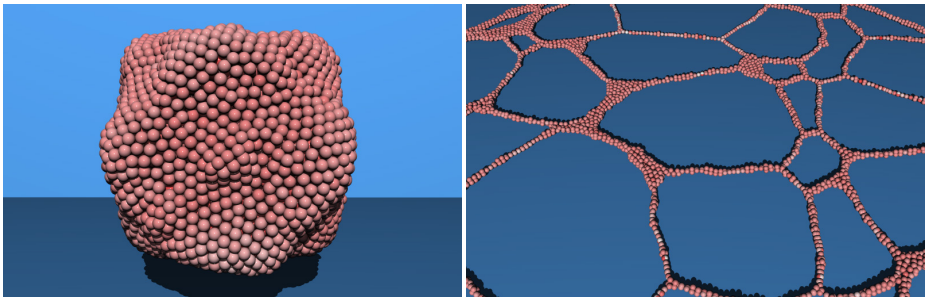
5.2 Formalism

5.2.1 Surface Tension Model

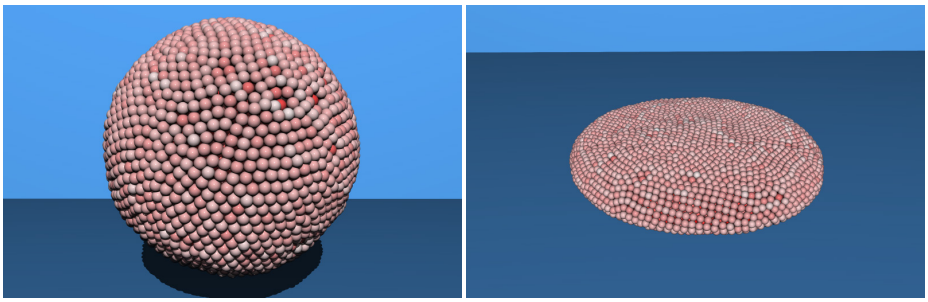
Surface tension in liquids arises as a result of molecular cohesion. However, as SPH simulates fluids on a macroscopic level with a finite support radius h,



(a) [Tartakovsky and Meakin 2005]



(b) [Becker and Teschner 2007]



(c) Our surface tension model.

Figure 5.2 – A fluid droplet in the shape of a cube is left to transform to a sphere and then dropped on the ground. Particle clustering and non-uniform particle alignment is visible in both (a) and (b), which is addressed with our surface tension model (c). Furthermore, neither of the cohesion-only models is able to simulate the large surface tension possible with our model, but they result in cobweb-like elongating structures, no matter how large the cohesion forces are. Particles are colored according to pressure. Images are taken from [AAT13].

cohesion forces between SPH particles do not reproduce the surface tension that we observe in reality. When only cohesion forces are applied between particles, the neighboring particles just attract each other which can result in any arbitrary configuration depending on the initial configuration of the particles (see e.g. Figures 5.2a, 5.2b). Therefore, surface area minimization is not guaranteed. Additionally, we also experienced that only using curvature minimization terms [Mor99, MCG03] results in more severe particle clustering for the experiment depicted in Figure 5.2, causing the droplet to break into many smaller droplets. To address these issues, we propose a new surface tension force that takes

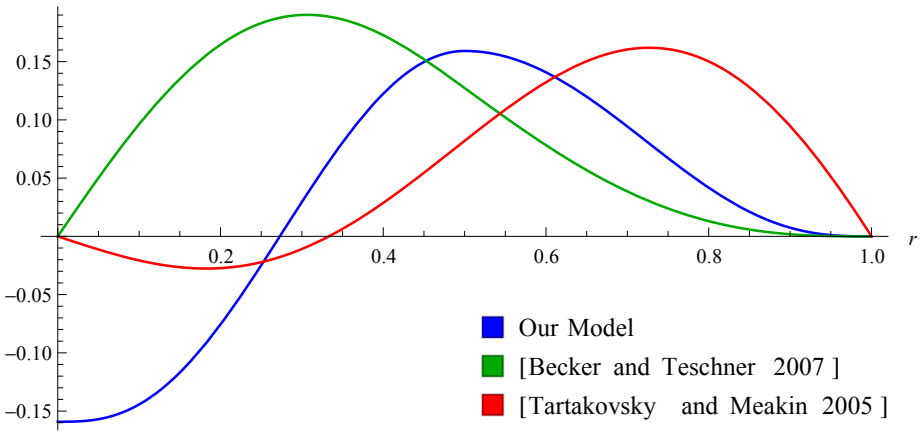


Figure 5.3 – Comparison of the shape of our cohesion force (blue) to [TM05] (red) and [BT07] (green) inside the SPH support radius $h = 1$. Graphs taken from [AAT13].

both molecular cohesion and surface area minimization into account and allows handling large surface tensions properly (see Figure 5.2c).

5.2.1.1 Cohesion Term

[TM05] is the first work to employ molecular cohesion forces to generate surface tension in SPH. They adjusted the cosine function to generate attraction for distant particles, and repulsion for close particles. However, we observed that the function used in [TM05] results in clustering throughout the fluid (see Figure 5.2a). Later in [BT07], instead of the cosine function, the SPH kernel function is used. However, since the method lacks a repulsion term, it also results in severe clustering; especially in regions with underestimated pressures (see Figure 5.2b-right). Both methods are applied through the displacement vector between the neighboring particles. This causes the forces to vanish for very close neighbors, which is another reason why clustering occurs with these methods. The forces applied by those two methods for a support radius $h = 1$ can be seen in Figure 5.3.

Because of the issues with the existing cohesion forces, we propose an alternative cohesion force defined as:

$$\mathbf{F}_{i \leftarrow j}^{cohesion} = -\gamma \, m_i m_j C\left(|\mathbf{x}_i - \mathbf{x}_j|\right) \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|}, \tag{5.1}$$

where i and j are neighboring fluid particles, m denotes mass and $\mathbf{x}$ denotes position of the respective particle, γ is the surface tension coefficient and C is a spline function that we created for a 3D SPH simulation as:

$$C\left(r\right) = \frac{32}{\pi h^9} \begin{cases} (h-r)^3 r^3 & 2r > h \wedge r \leq h \\ 2(h-r)^3 r^3 - \frac{h^6}{64} & r > 0 \wedge 2r \leq h \\ 0 & \text{otherwise} \end{cases}. \tag{5.2}$$

The term h^9 in the denominator of (5.2) is a normalization factor to make the force result in the same acceleration for different support radii (e.g. like the

SPH pressure force). The constant term in the beginning of (5.2) is basically used to shift the practical γ values close to 1. Similar to [TM05], our cohesion force also has both a positive and a negative part to result in repulsion (see Figure 5.3). For the attraction term, we chose a maximum around the particle rest distance $h/2$, where the attraction smoothly vanishes to 0 until the support radius h. For fluid neighbors closer than the rest distance, the force smoothly decreases to a negative value, which results in repulsion forces for the particles that are too close to each other. Furthermore, both the attraction and repulsion forces behave like a Gaussian to avoid clustering, which is in contrast to [TM05]. Additionally, our repulsion force does not vanish to 0 for close neighbors, which prevents particle clustering in regions with underestimated pressures (see Figure 5.2c). Our cohesion term also has similarities to the Lennard-Jones potential [Jon24], which is a commonly used model to approximate the interaction between a pair of molecules. However, the main difference of our cohesion term to the Lennard-Jones potential is that our term stops increasing as the particles move closer, which helps to avoid too stiff forces and resultant stability issues.

5.2.1.2 Surface Area Minimization Term

Although our cohesion force solves some important issues of previous cohesion models, it is still not sufficient for minimizing the fluid surface area because of the reasons we discussed previously. Therefore, we use an additional force term to counteract surface curvature to minimize the surface area. Since computing curvature from particles is error-prone in SPH, different from the previous models, we avoid computing surface curvature explicitly. We firstly compute normal information by applying the SPH approximation to the gradient of the smoothed color field as:

$$\mathbf{n}_i = h \sum_j \frac{m_j}{\rho_j} \nabla W \left(|\mathbf{x}_i - \mathbf{x}_j| \right),$$

where W is the SPH kernel function and ρ_j is neighboring particle density. Different from [Mor99] and [MCG03], we have the factor h to make the computed normal scale independent. At this point, we use the fact that the magnitude of $\mathbf{n}_i$ is proportional to the curvature, where its value is close to zero for the inner parts of the fluid, but large at the free surface proportional to the curvature. We utilize this information and finally create a symmetric force as:

$$\mathbf{F}_{i \leftarrow j}^{curvature} = -\gamma \, m_i \left(\mathbf{n}_i - \mathbf{n}_j \right). \tag{5.3}$$

One can easily confirm that (5.3) is zero in flat regions (as $\mathbf{n}_i - \mathbf{n}_j = \mathbf{0}$) and inside the fluid (as $\mathbf{n}_i \approx \mathbf{0}$ and $\mathbf{n}_j \approx \mathbf{0}$), but it gets larger as the curvature increases. Therefore, our surface area minimization force avoids two important issues of previous techniques: Normalization of $\mathbf{n}_i$ (that is erroneous inside the fluid), and explicit curvature computation (that is very sensitive to particle disorder).

5.2.1.3 Combined Surface Tension Force

Before discussing the combined surface tension force, we will discuss another important particle deficiency related issue in SPH. For the fluids in reality, attractions between fluid molecules occur at a microscopic scale, where each

molecule interacts with many other molecules. In SPH, however, particles represent macroscopic volumes of the fluid. If we consider the simple example of two neighboring particles, the net attraction force that affects each particle is smaller than in a configuration of three neighboring particles, which makes the two particles separate easier than the three particles in the case of external forces. If those macroscopic particles would have been sampled with real water molecules, there would not be such a difference. This error in SPH manifests itself as too many isolated particles, since particles with smaller neighborhood get isolated easier than the rest of the fluid. There exist techniques to address this issue, such as corrected SPH (e.g. [BKDG98]) and adaptive SPH [SMVO96]. However, these works add too much computational overhead to basic SPH. Although [Mon00] (also used in [MM13]) implicitly addresses this issue by generating a spurious surface tension, such an approach is not desirable for our purposes since it would interfere with our refined surface tension model. We provide an explicit solution to this problem by creating the following symmetrized correction factor:

$$K_{ij} = \frac{2\rho_0}{\rho_i + \rho_j}, \tag{5.4}$$

where ρ_0 is the rest density of the fluid, and ρ_i and ρ_j are the densities of the neighboring fluid particles. As we do not correct the fluid particle densities, a fluid particle with less than full neighborhood has $K_{ij} > 1$, and a fluid particle with full neighborhood has $K_{ij} \approx 1$. Therefore, K_{ij} amplifies forces for the particles with neighborhood deficiency, while the forces remain the same for the particles with appropriate neighborhood. The final surface tension force can be written as:

$$\mathbf{F}^{st}_{i \leftarrow j} = K_{ij} \left(\mathbf{F}^{cohesion}_{i \leftarrow j} + \mathbf{F}^{curvature}_{i \leftarrow j} \right). \tag{5.5}$$

Note that the terms in (5.5) are fully symmetrized, and the total force is applied to the particle pairs. This is, however, not the case in previous surface area minimization techniques (e.g. [Mor99, MCG03, YWTY12]), as they apply the forces to the particles as external forces. The effect of combining both terms can be seen in Figure 5.2c. Note the large surface tension possible with our approach. Although we used larger cohesion forces for both [TM05] and [BT07], they failed to generate the large surface tension that we wanted to achieve. We also observed that using larger surface tension forces with the cohesion-only models does not further improve the quality of the generated surface tension behavior, but only results in stiffer fluids with similar spurious structures.

5.2.2 Adhesion Model

Different from cohesion, adhesion occurs as a result of molecular interaction of dissimilar materials. In our work, we focus on two-way fluid-solid adhesion in SPH simulations.

For boundary-handling and two-way fluid-solid coupling, again we use [AIA+12] (explained in Chapter 3.1), where solid surfaces are sampled using boundary particles. As a quick summary, in this approach, the volume of a boundary particle is approximated as $V_{b_i} = \frac{1}{\delta_{b_i}}$, where δ_{b_i} is the number density of a boundary particle computed according to the neighboring boundary particles. The contribution of a boundary particle to a fluid particle (and vice versa) is based on the volume of a boundary particle and written as $\Psi_{b_i}(\rho_0) = \rho_0 V_{b_i}$,

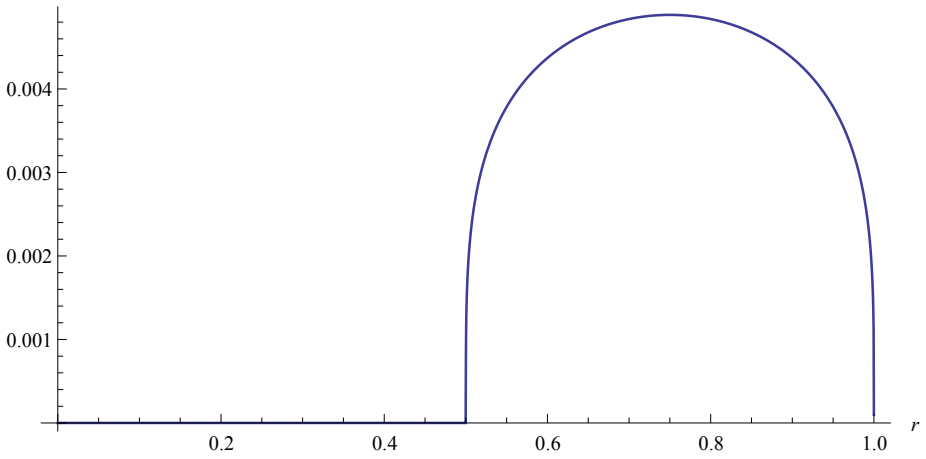


Figure 5.4 – The shape of our adhesion function inside the SPH support radius $h = 1$. Graph is taken from [AAT13].

where Ψ_{b_i} is used in place of the mass of a boundary particle when computing fluid density, pressure forces and viscosity forces. Therefore, the approach addresses the sticking problem of SPH near solid boundaries, and allows two-way fluid-solid coupling with different slip conditions. Adhesion effects, however, are not addressed in [AIA+12].In our work, we compute adhesion forces between the fluid and the boundary particles as:

$$\mathbf{F}_{i\leftarrow k}^{adhesion} = -\beta\, m_i \Psi_{b_k} A\left(|\mathbf{x}_i - \mathbf{x}_k|\right) \frac{\mathbf{x}_i - \mathbf{x}_k}{|\mathbf{x}_i - \mathbf{x}_k|}, \tag{5.6}$$

where k denotes boundary particles, $\mathbf{x}$ denotes position of the respective particle, β is the adhesion coefficient and A is a spline function that we created for a 3D SPH simulation as:

$$A(r) = \frac{0.007}{h^{3.25}} \begin{cases} \sqrt[4]{-\frac{4r^2}{h} + 6r - 2h} & 2r > h \wedge r \leq h \\ 0 & \text{otherwise} \end{cases}. \tag{5.7}$$

Similar to our cohesion force, the term $h^{3.25}$ in the denominator of (5.7) is a normalization factor to make the force result in the same acceleration for different support radii. The scalar term in front of (5.7) is used to be able to select β values in the similar range of γ values, where $\beta \approx \gamma$ models a moderate hydrophilic behavior. Since the boundary forces used in [AIA+12] already prevent clustering near the solid boundaries, we designed our adhesion force to only attract particles with distances between $h/2$ and h. Furthermore, we tried to make our attraction force large in this interval, while keeping the force continuous (see Figure 5.4). Initially, we started with a Gaussian-like shape for the adhesion force. However, such a force was causing most of the fluid (except the closest fluid layer) to unrealistically detach from the solid, regardless of the magnitude of the adhesion force. Finally, we came up with such a steep parabolic function, which generates strong attractions for most of the neighboring fluid particles.

Note that similar to our surface tension force, our adhesion force is also fully symmetric, where $\mathbf{F}_{k\leftarrow i}^{adhesion} = -\mathbf{F}_{i\leftarrow k}^{adhesion}$. Our adhesion force allows simulating interesting scenarios, such as different wetting conditions and two-way adhesion.

5.3 Implementation Details

We use [SP09] for computing SPH pressures. However, our surface tension and adhesion forces can be integrated into any SPH solver (e.g. [BT07, BLS12, MM13, ICS+13]) since the forces are computed from the particles and are directly applied to the neighboring pairs. Furthermore, we do not apply pressure forces that arise from negative pressures. For time-step selection, we use the adaptive scheme in [IAGT10]. For boundary-handling and two-way solid-fluid coupling, we employ [AIA+12] and simulate dynamic objects using Bullet [Cou11]. We use the artificial viscosity model described in [Mon05] both for fluid-fluid and fluid-solid interactions as done in [AIA+12]. Because of the particle deficiency related reason highlighted in Section 5.2.1.3, we also multiply viscosity forces with (5.4). We use the cubic spline kernel function [Mon05] for our SPH simulations. For SPH neighborhood search, we use the compact hashing technique explained in [IABT11].

If not stated otherwise, we used $\gamma = 1$ for all experiments to mimic the surface tension of water. Furthermore, we used $\beta = \gamma$ to model moderate hydrophilic interactions. In all of our simulations, we kept fluid compressibility below 0.1%. Depending on the scale of the simulated setting, we used different particle radii r, where the particle spacings and support radii were $2r$ and $4r$ respectively. We used very low artificial viscosity for the simulated fluids (~ 0.01) to mimic the viscosity of water. If not stated otherwise, all solids in our experiments also have the same viscosity to model a moderate slip condition when interacting with the fluids. We reconstructed the fluid surfaces using the efficient implementation explained in [AIAT12]. The renderings were performed using mental ray [NVI11]. All simulations and renderings were run on an Intel Xeon X5690 with 16 GB RAM. For the presented scenes, the average simulation time per frame was between 0.1 to 15 seconds, depending on the complexity of the scene, where several simulations steps were computed for each frame.

5.4 Results

In this section, we demonstrate the versatility of our surface tension and adhesion models in different simulation scenarios. Our experiments show plausible fluid-fluid and fluid-solid interactions, even at the scale of a single droplet.

To show that our approach can handle realistic fluid-fluid interactions, we dropped a 6.5 cm^3 fluid droplet into a $15 \times 4 \times 15$ cm^3 container filled with 1M fluid particles (Figure 5.1). After the fluid droplet hits the main fluid body in the container, a realistic water crown emerged. Also, note how the splashes form spherical droplets with various sizes, which is a phenomenon observed in reality. Note both the thin features around the crown and the smooth fingerings that emerge at its top. Later in the simulation, a vertical finger occurred, which is an effect that occurs after a water crown collapses. With the previous surface tension models that are only based on cohesion, the experiment reveals many spurious fluid structures (see Figure 5.5).

Our next experiment shows that, by using our surface tension and adhesion models, it is possible to simulate the impact of a large fluid droplet on a solid object in a realistic way. A 0.5 m^3 water volume consisting of 100K fluid particles was dropped on a $1.5 \times 0.12 \times 1.5$ m^3 solid (Figure 5.6). The adhesion of solid

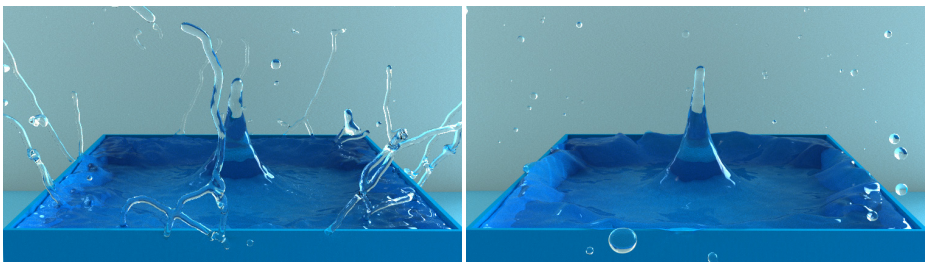


Figure 5.5 – Comparison of a selected frame from the water crown experiment using [BT07] (left) and our surface tension model (right). Note the spurious surface tension and bumpy fluid surface on the left image. Such effects are avoided with our approach as the fluid surface area is properly minimized. Images are taken from [AAT13].

was chosen as $\beta = 0.6$ to model a reduced hydrophilic behavior. Because of the surface tension, sheets of fluid broke into fingers, which then transformed into spherical droplets (Figure 5.6 middle-left). The adhesion of the table resulted in sliding droplets dripping from the side of the table (Figure 5.6 middle-right). Furthermore, the adhesion prevented the fluid to completely merge to a single water body at the end (Figure 5.6 right).

In the next experiment, we show how the combination of our surface tension and adhesion forces can be used to simulate a scenario where a vertical water stream realistically flows over a sphere with diameter 5 cm (Figure 5.7). There were up to 50K fluid particles in the scene. We are able to simulate such vertical flows with solid adhesion realistically without requiring ghost SPH.

In another experiment we show that by using different adhesion and surface tension constants (where $1 \geq \gamma, \beta \geq 0.001$) , we were able to simulate different wetting conditions in the same simulation scale between no wetting up to perfect wetting for a 1 cm^3 fluid droplet consisting of 750 particles (see Figure 5.8).

We created our next experiment to demonstrate how fluids react differently to two solid objects with different adhesion in the same environment (Figure 5.9). In this scene, a 1 cm^3 water droplet consisting of 4K particles was resting in a highly hydrophilic box with $\beta = 2$. Afterwards, another object with zero adhesion split the droplet into two parts. Note how the droplet sticks to the box, while its contact area with the splitting object remains small.

In our final experiment, we show how two-way adhesion force can be used to create interesting scenarios (Figure 5.10). In this scene, we dropped a 27 cm^3 fluid droplet consisting of 14K particles with $\gamma = 3$, on an inclined plane. The plane had $\beta = 1.2$ to make the droplet create a large contact angle with the plane. Additionally, we set the viscosity of the inclined plane to 1 (which was 0.01 in the other experiments). Such a large viscosity makes the droplet roll on the surface, instead of slide. In this experiment, we also dropped ragdolls on the fluid droplet with a density two times larger than the fluid density. This experiment also shows that large surface tension forces prevent the solid to penetrate into the fluid, which is a phenomenon observed in nature. Furthermore, pink ragdolls (with $\beta = 1$) stick to the droplet, whereas the green ragdolls (with $\beta = 0$) bounce and slip from the droplet.

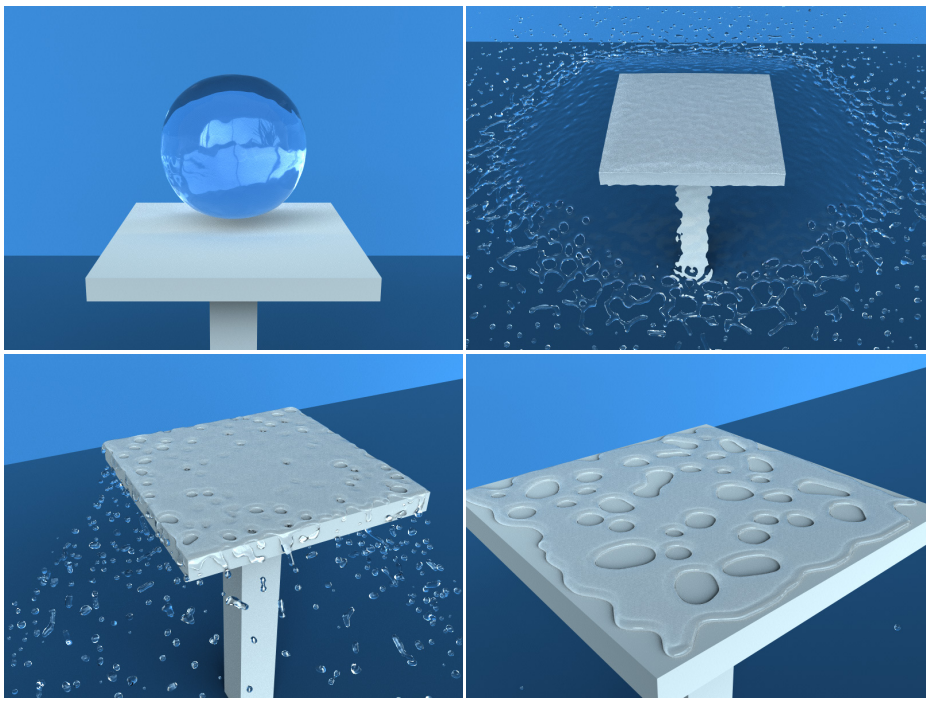


Figure 5.6 – A spherical water volume collides with a planar solid object. The interplay of our surface tension and adhesion forces allows realistic interactions in such scenarios. Images are taken from [AAT13].

5.5 Discussion and Future Work

In the presented experiments, when the surface tension is the dominant force acting on a particle, time steps are limited by the surface tension. This was the case for some of our experiments that were performed in droplet scales. However, for most of the larger scale experiments, the pressure force was the dominant force. However, as we used the adaptive time stepping scheme explained in [IAGT10] that also takes the total force on a particle into account, we did not run into stability issues when performing our experiments.

In reality, surface tension arises because of cohesion forces between fluid molecules of the same fluid phase, independent of what is beyond the free surface (e.g. air, another liquid or vacuum). Therefore, modeling surface tension does not require an explicit second fluid phase. One ubiquitous effect that arises in reality because of multi-phase interactions is air bubbles inside fluids. Since simulating density ratios in the order of ρ_{water}/ρ_{air} would require considerably more computational effort, there exist alternative air bubble generation techniques to avoid multi-phase simulations (e.g. [HLYK08, IBAT11, BDWR12]). We believe that the realism of our results can be further improved by using such a method.

Recently, [YWTY12] demonstrated that sub-particle scale capillary waves that are directly simulated on the fluid surface mesh can add a significant amount of detail to an existing simulation. We believe that our results would further

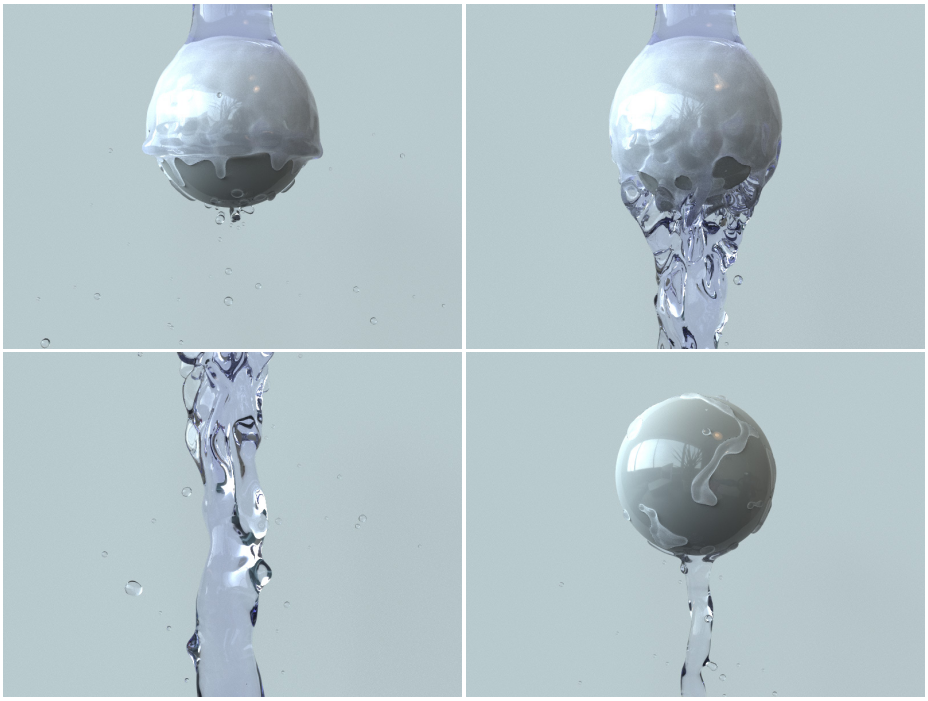


Figure 5.7 – Pouring water on a sphere. Our forces allow simulating a realistic stream flowing over a sphere without using ghost SPH. Images are taken from [AAT13].

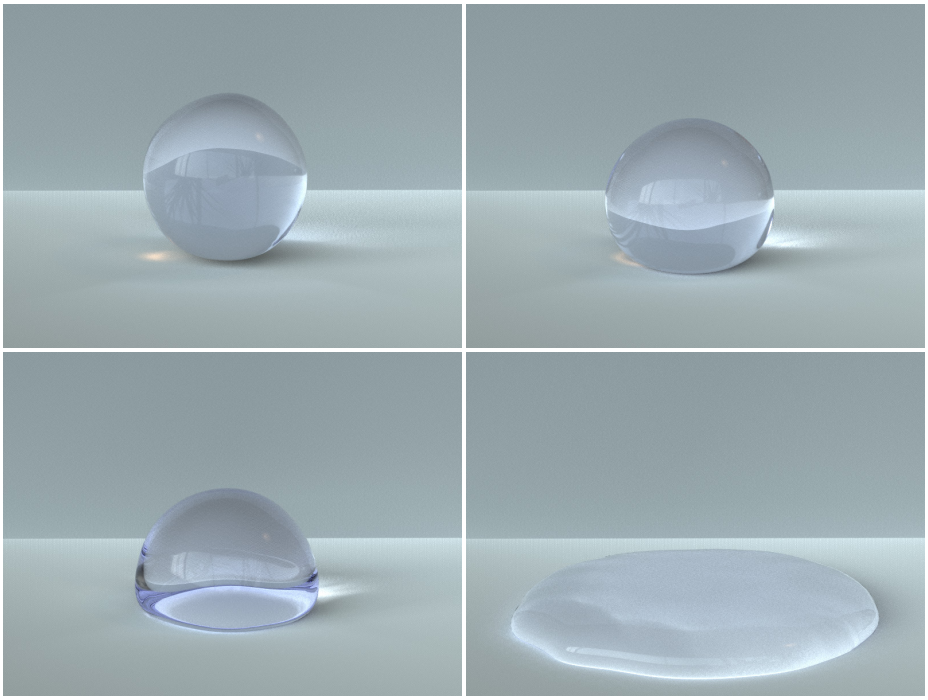


Figure 5.8 – Combination of our surface tension and adhesion forces allows simulating different wetting effects. Images are taken from [AAT13].

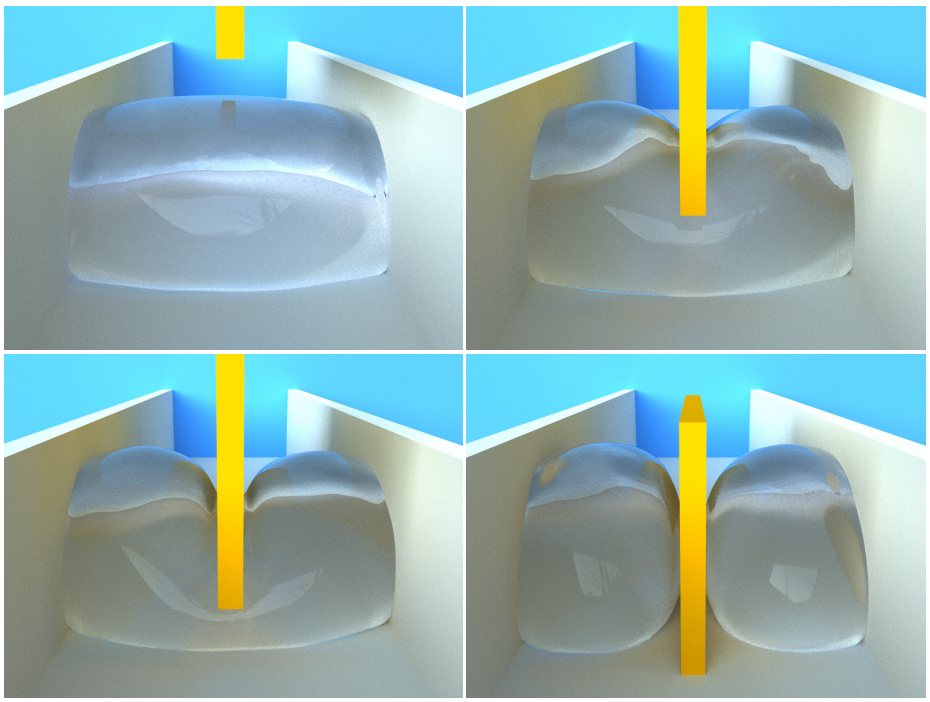


Figure 5.9 – A droplet in an adhesive box is split into two with an object with zero adhesion. Note how the fluid sticks to the box but maintains a much smaller contact area with the splitting object because of the wetting difference. Images are taken from [AAT13].

improve by using their surface tracking and capillary wave simulation approach.

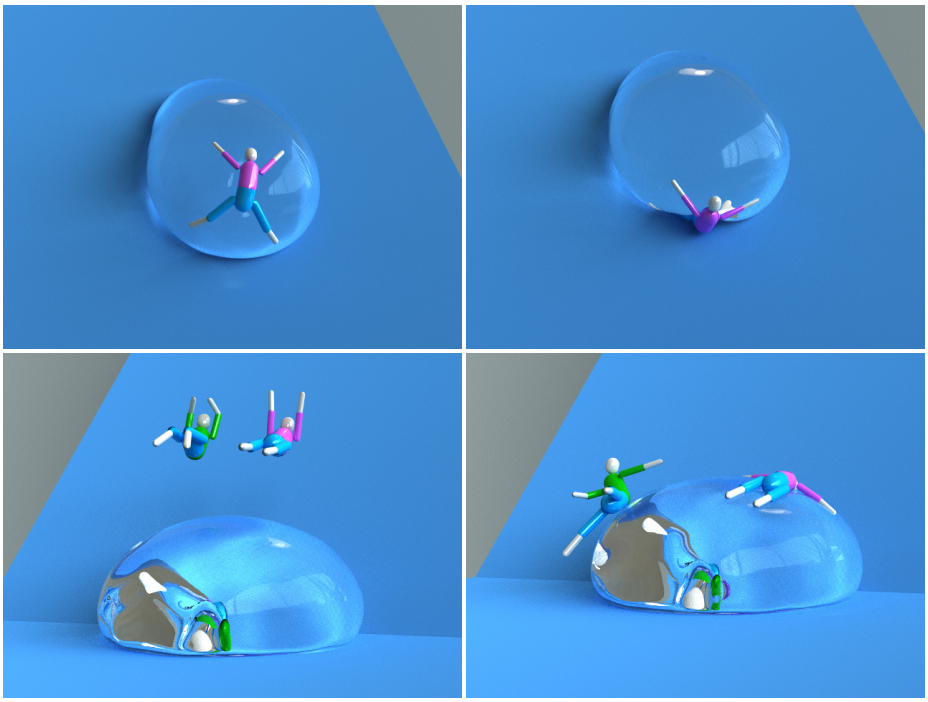


Figure 5.10 – Using the right balance of surface tension, adhesion, and fluid-solid viscosity forces allows us to simulate rolling water droplets. In this scene, ragdolls with different adhesion properties interact differently with the droplet. Images are taken from [AAT13].

6 Handling of Fluid-Air Mixtures

6.1 Introduction

Foam is a complex phenomenon whose behavior and appearance is challenging to simulate in computer graphics. When viewed from a close distance, foam is composed of many air bubbles sticking to each other. It can occur inside most fluids as a result of trapped air. One can observe milky white foam caused by dashing waves on seashores. For most semi-transparent materials, it is an interesting observation that, even though the underlying material may have a color, the foam usually looks whitish to the viewer. The reason for this behavior is that the foam is composed of thin films of fluid containing air. As the number of such thin films increase per unit volume, all incoming light is reflected without allowing any light to penetrate beneath it. This optical phenomenon makes the foam look brighter than the material itself, to the point that it looks almost white. This chapter focuses on the efficient rendering of such white foam by approximating some important effects in screen space, that are otherwise time consuming to compute in a physically correct way. Our technique is specifically useful for complex large-scale scenarios, where large amount of foam data need to be rendered. In the remainder of this section, we first summarize the existing works about GPU accelerated rendering of fluid data (Section 6.1.1), foam simulation and rendering (Section 6.1.2) and then highlight our contribution (Section 6.1.3).

6.1.1 GPU Rendering of Fluids

For non-interactive applications, fluid surfaces are generally visualized by triangulating the isosurface of the particle data (e.g. [ZB05, YT10, AIAT12]) and then rendering the resultant mesh using ray-tracing based techniques to produce convincing results. For real-time applications, the computational overhead of those approaches remains too high. Therefore, for the efficient GPU accelerated visualization of fluid surfaces, several methods have been proposed in the recent years, e.g., using screen space surface construction [MSD07, FAW10], height field techniques [CM10] and methods that are based on particle splatting [vdLGS09, BSW10]. Even though foam is actually composed of the molecules of the underlying fluid, its characteristic appearance requires it to be handled using different rendering approaches, which will be explained in the next section.

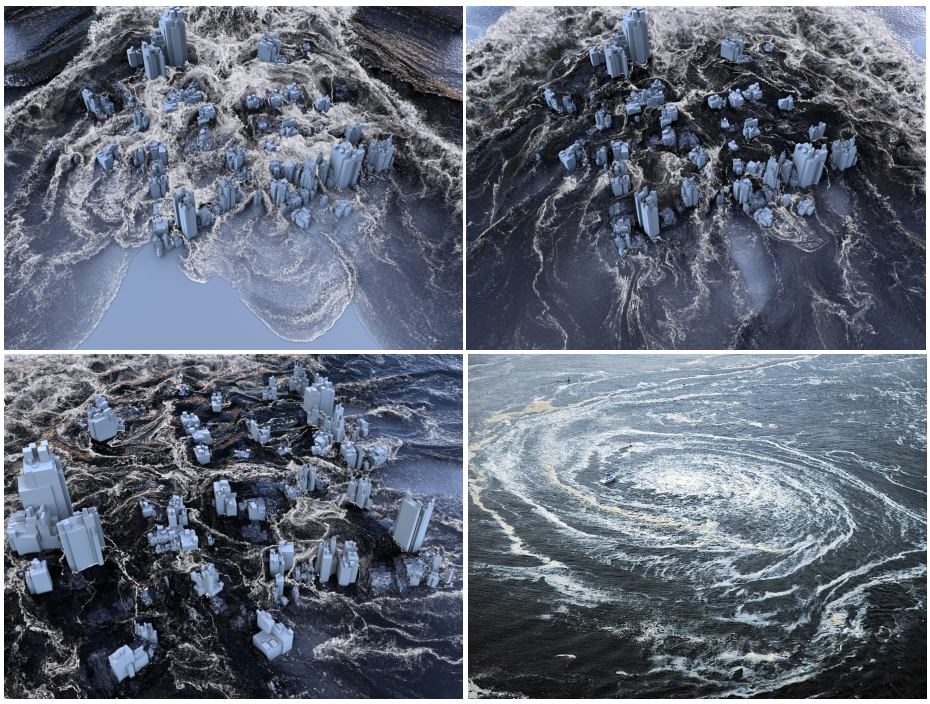


Figure 6.1 – A flood scenario. Foam is rendered using our technique and composited with the rest of the scene (left and middle). Picture of real sea foam caused by a whirlpool (right) (©Reuters). Images are taken from [ADAT13].

6.1.2 Foam Simulation and Rendering

In computer graphics, foam generation techniques are used to enhance the realism of existing fluid simulations. High quality foam simulation and rendering techniques are commonly encountered in movies [GLR+06, BSK+07] and in commercial fluid simulation and visualization packages [hyb11]. In those works, however, the underlying foam generation and rendering stages are usually described briefly. Although foam is composed of fluid and air mixture, some of the existing research also focuses on generating foam particles, usually in a scale smaller than the fluid particles to be able to enhance the flow detail [TFK+03, GLR+06, LTKF08, MMS09, IAAT12]. For foam generation, we employ [IAAT12]. The approach generates and processes three types of diffuse material, i.e. air bubbles, surface foam and spray. All types of diffuse material are represented with particles that are generated, advected and dissipated according to physically-motivated rules. The approach adds diffuse material to particle-based fluid simulations in a post-processing step. For the details of the technique, we refer the interested reader to [IAAT12].

For high quality foam renderings, ray-tracing methods are commonly preferred for both the fluid and the foam [GLR+06]. Although the fluid surface can be rendered efficiently using ray-tracing, non-homogenous phenomena such as foam require expensive volume rendering techniques. In [IAAT12], the authors employed a volume ray-casting method, which accounts for absorption and emission of radiance but neglecting light scattering effects. In that method,

each traced ray is sampled using equally spaced intervals; and according to the measured foam density at each sample point, the computed radiance is attenuated. The employed ray-casting approach, however, is time consuming to compute, especially for scenes with many millions of foam particles. The performance of volume ray-casting can be significantly improved by using the GPU-based method explained in [FAW10].

In [vdLGS09, BSW10], alternative to generating new particles, selected fluid particles are visualized as foam particles using GPU-based techniques for real-time applications. In [BSW10], Weber number thresholding is used to separate fluid and foam. Furthermore, the method also takes volumetric effects into account by rendering foam and fluid layers from back to front order. Therefore, it can visualize effects such as foam inside the fluid. Furthermore, based on the thickness of the foam, it generates foam color between two user defined colors. The approach, however, neglects information such as occlusion and irradiance from the environment when rendering foam, which limits its applicability to non-photorealistic real-time renderings.

There also exist methods for the modeling of larger scale foam effects by using air bubbles (e.g. see [KVG02, KLL+07, HLYK08, IBAT11, BDWR12]). In these works, air phase is either visualized by rendering spheres [KVG02, BDWR12], or by reconstructing the surface of the modeled air phase [KLL+07, HLYK08, IBAT11]. Since we are focusing on large-scale scenarios, where the single air bubbles inside the foam are not clearly noticeable, such methods are beyond the scope of this chapter.

6.1.3 Contribution

We present an efficient method for large-scale foam rendering. In our approach, foam is rendered using a novel multi-pass rendering algorithm and finally composited with the pre-rendered images of the scene without foam. In comparison to volume ray-casting methods that compute only absorption and emission of radiance (e.g. [FAW10, IAAT12]), our approach is significantly faster as the foam particles are directly rendered. Furthermore, when compared to [BSW10], our pipeline takes the scene occlusion and lighting into account and therefore produces more convincing results that can be composited with realistic renderings. Results show that our new pipeline generates convincing large scale foam renderings (e.g. see Figure 6.1) using modern GPU-based rendering architectures.

6.2 Formalism and Implementation

As more air bubble layers implies more light scattering, we relate the foam thickness to the foam intensity as usually done in volume ray-casting. Later, we determine the regions on screen space, which should receive, and therefore scatter less light using ambient occlusion and attenuate the foam intensity according to the occlusion factor. Afterwards, we approximate per-pixel foam irradiance to colorize the foam color according to the environment. Finally, the generated results are composited with the rest of the scene. We realized our approach using a seven pass rendering algorithm. The technical steps of our pipeline (illustrated in Figure 6.2 and 6.3) can be summarized as:

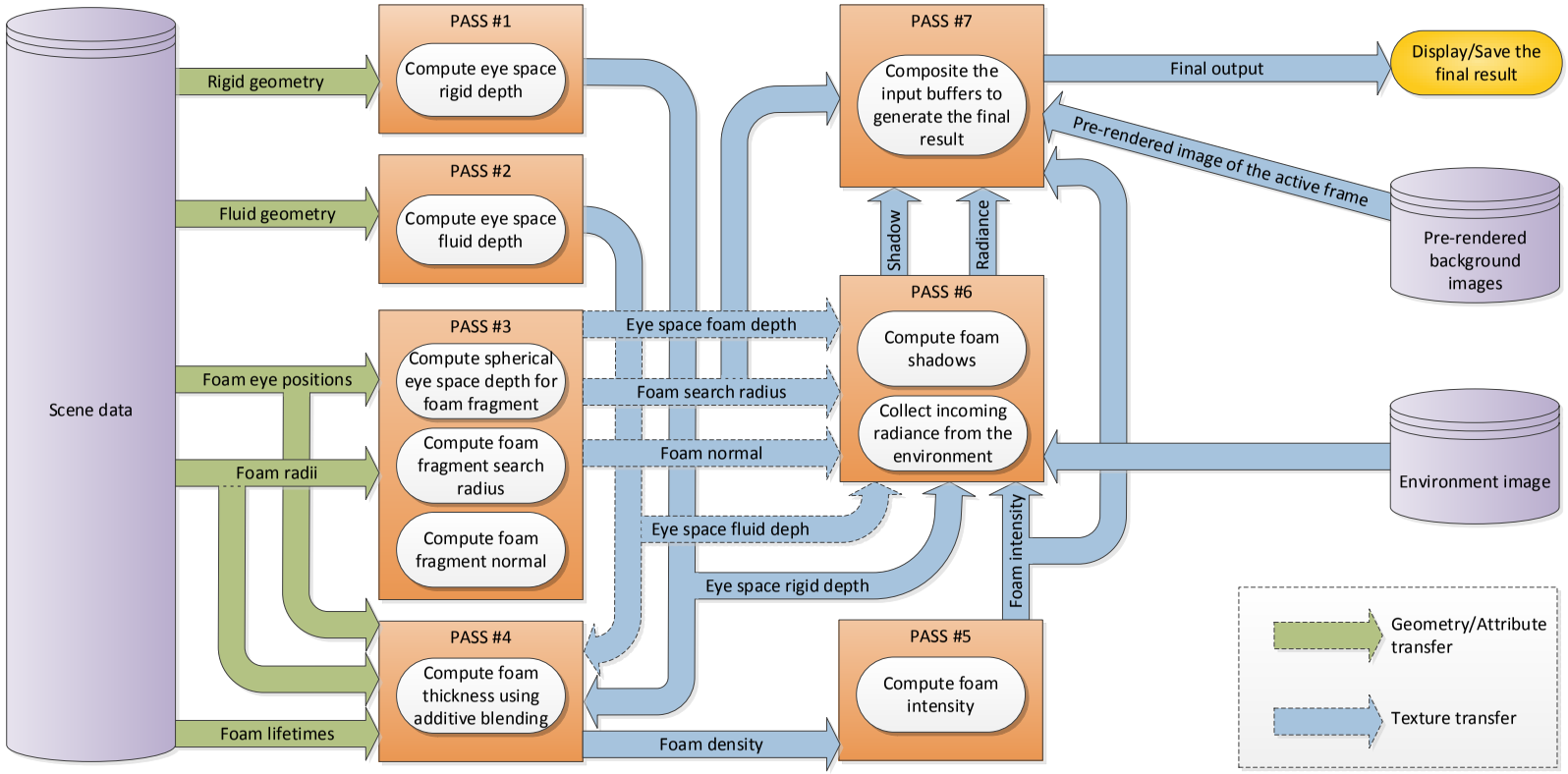
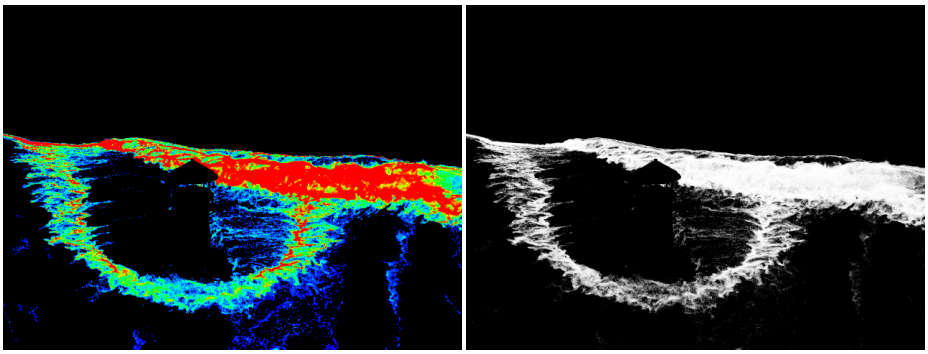
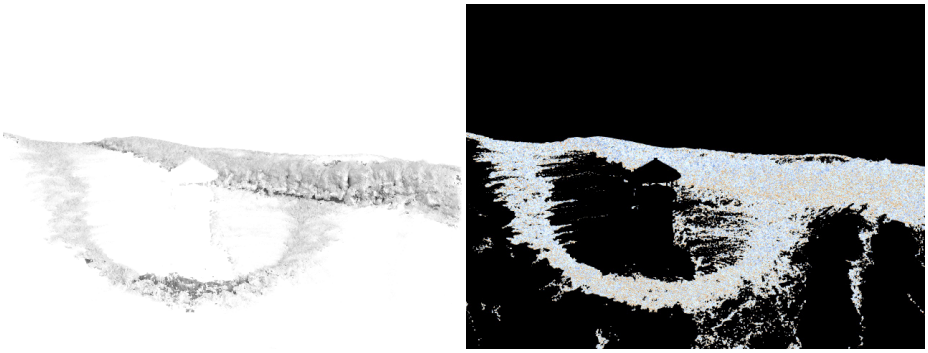


Figure 6.2 – Diagram of our foam composition pipeline. Orange boxes denote the render passes and the arrows in between denote data flow and dependencies. For each frame, the render passes from #1 to #7 are executed. Each pass produces data explained in the enclosed rounded rectangles, which is then transferred through arrows to the subsequent passes. All of the generated textures have the same resolution as the final output. Diagram taken from [ADAT13].



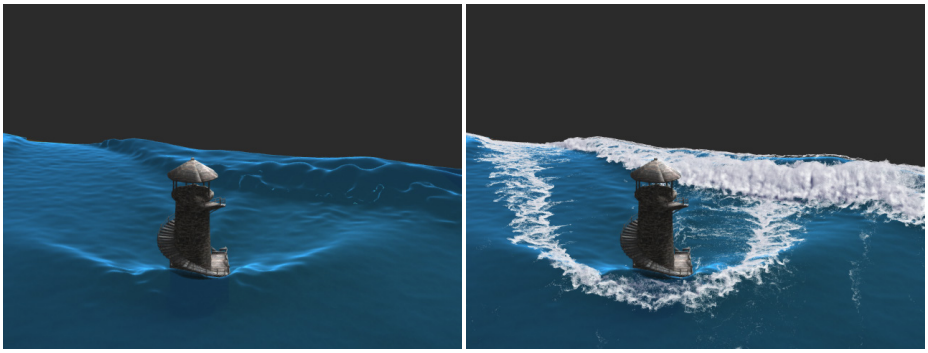
(a) Foam thickness (red denotes thickest, blue denotes thinnest parts)

(b) Foam intensity



(c) Foam shadow (inverted)

(d) Foam irradiance



(e) Pre-rendered image

(f) Image composited with foam

Figure 6.3 – Some of the intermediate textures from our foam composition pipeline (a-e) and the final composited result (f). Images are taken from [ADAT13].

- *PASS #1 and #2:* Storing eye space depth images of solid and fluid meshes in two textures, which are used to compute occlusion of foam fragments by those primitives in the later stages.

- *PASS #3:* Storing an eye space depth image of the foam particles in a texture, which is used in different parts of our pipeline. This pass also

stores a search radius for each foam fragment, in whose range neighboring fragments are later considered for screen space ambient occlusion and final composition (Section 6.2.1). Additionally, this pass computes a normal for each foam fragment, which is used when approximating irradiance at the fragment location.

- *PASS #4:* Accumulating foam particles via additive blending to approximate per-pixel foam thickness. This pass also discards foam fragments that are occluded by solids and attenuates foam fragments that are inside of the fluid based on the fluid transparency (Section 6.2.2).
- *PASS #5:* Conversion of per-pixel foam thickness to per-pixel foam intensity (Section 6.2.3).
- *PASS #6:* Determination of foam fragments that should receive and scatter less light using screen space ambient occlusion (SSAO) and shadow generation for such regions (Section 6.2.4.1). This pass also approximates the irradiance at each foam fragment from an environment texture if the scene is illuminated using image based lighting (Section 6.2.4.2).
- *PASS #7:* Post processing of the foam and final composition with a pre-rendered image of the scene (Section 6.2.5).

Since the first step of the pipeline is relatively straightforward, we will focus on the remaining steps throughout this section. The following render passes are implemented using OpenGL Shading Language (GLSL).

6.2.1 Smoothed Depth and Search Radius Computation

We use point sprites instead of spheres for rendering foam particles. A regular point sprite has the same depth values for all of its fragments. However, to produce convincing results in the later steps of our pipeline, we modify the fragment depth values similar to [vdLGS09, BSW10], such that the spherical shapes of the particles are regained.

To create the initial depth information, foam particles with ids i and radii r_i in world space are rendered with depth testing and depth masking enabled. In [IAAT12], foam particles are separated to three different types, namely: spray, surface-foam and bubble particles. For bubble particles, we use half of r_i to make them less visible. Furthermore, particle radii are randomized as $r_i = \frac{r_i}{(i \bmod 5)+1}$ to make the particles look irregular between the scales ${}^{r_i}/_5$ and r_i.

The vertex shader computes eye space and projection space coordinates of the sprites and passes the resultant data to the fragment shader for further processing. In the fragment shader, the distance of the fragment position to the point sprite center is calculated using the sprite's texture coordinates to discard fragments that are outside of the circle. Afterwards, the flat depth values of the point sprite are transformed to spherical depth values. In this context, the first step is solving for the w coordinate of a unit sphere for the fragment's texture coordinates in uvw space as $w = \sqrt{1 - u^2 - v^2}$, where u and v denote texture coordinates of the fragment. Subsequently, the eye space z coordinate of the fragment is simply modified as

$$\mathbf{e}^{frag}_{foam_z} = \mathbf{e}^{frag}_{foam_z} + w \cdot r_i.$$

In contrast to [vdLGS09, BSW10], we do not apply filtering to the generated depth values since it would reduce the effect of ambient occlusion.

In the same render pass, the vertex shader also projects the search radius h_i for each particle as

$$h_i = \frac{r_i}{\tan\left(\frac{\alpha}{2}\right)\left|\mathbf{e}^{vert}_{foam_z}\right|},$$

where α is the field of view of the camera and $\mathbf{e}^{vert}_{foam_z}$ denotes z coordinate of the eye position of the point sprite (i.e., distance of the sprite to the camera). Afterwards, the search radius is passed to the fragment shader as h^{frag} to be written to a texture. The depth information and the search radius are essential when rendering the SSAO pass and when doing the final composition.

This pass also computes a world space normal for each fragment $\mathbf{n}_{frag}$ by transforming (u, v, w) using the transpose of the normal matrix, and stores the normals in a texture. Per fragment normals will be required when estimating irradiance in Section 6.2.4.2.

6.2.2 Thickness Estimation

Before estimating the intensity of foam at a given pixel position, we estimate the foam thickness for each pixel. In this step, foam particles are rendered again as point sprites with the spherical depth modification as in the previous render pass. Similar to [vdLGS09, BSW10], the foam fragments are blended additively to estimate thickness. Different from [vdLGS09, BSW10], however, depth buffer read and write is disabled as we do not require the frontmost particles to be visible.

As foam particles are separated to spray, surface-foam and bubble particles, we also employ this knowledge to render foam fragments differently by using a falloff function with different arguments, where the falloff is based on the fragment's distance to the particle center in texture coordinates. The falloff function f is defined as

$$f(x,\, b,\, n,\, m) = \begin{cases} \left[1 - \left(\frac{x}{b}\right)^n\right]^m & \frac{x}{b} \leq 1 \\ 0 & \text{otherwise} \end{cases}, \tag{6.1}$$

where x is the distance to the center, b is the maximum allowed distance, and $n \geq 0$ and $m \geq 0$ are exponents which determine the shape of the function (e.g., $n = 1$ and $m = 1$ result in linear falloff). When rendering spray, surface-foam and bubble fragments, we used $f_{spray} = f(x,\, 1,\, 1.5,\, 1)$, $f_{sfoam} = f(x,\, 1,\, 2.25,\, 1)$ and $f_{bubble} = 1 - f(x,\, 1,\, 2,\, 1)$ respectively. These different falloff functions are illustrated in Figure 6.4 and the corresponding particle intensities are shown in Figure 6.5. We preferred a larger overall intensity for surface foam particles to increase their visibility. Whereas, we preferred a comparatively smaller intensity value for the spray particles to make them relatively less visible. Furthermore, we used hollow circle like structures for the bubble particles to make their appearance more convincing under water.

In this step, the intensities of the foam particles are further modulated based on two additional factors. The first of these factors is the lifetime of the particle. For this purpose, we use $f_{lifetime} = f(l_i,\, 1,\, 2,\, 0.4)$, where $0 < l_i < 1$ denotes the normalized lifetime of a particle. Such a function allows a foam particle to

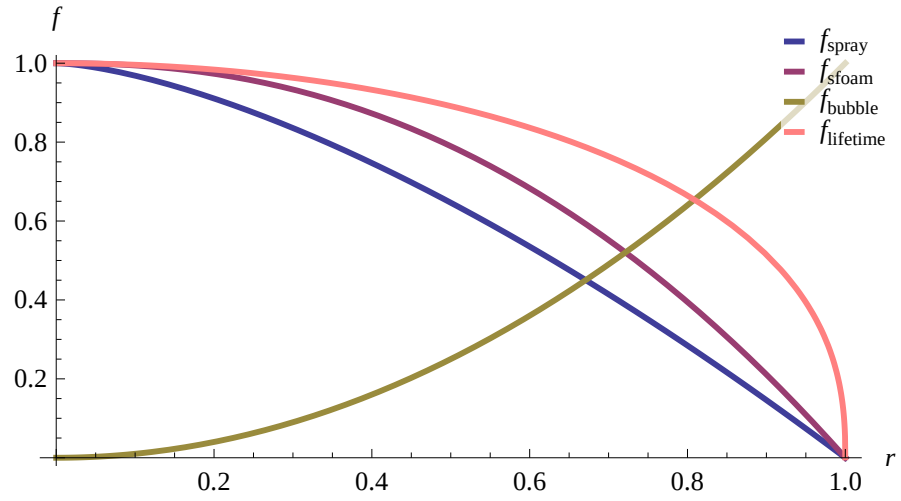


Figure 6.4 – Different forms of the falloff function given in (6.1) that are used in our experiments. Graph is taken from [ADAT13].

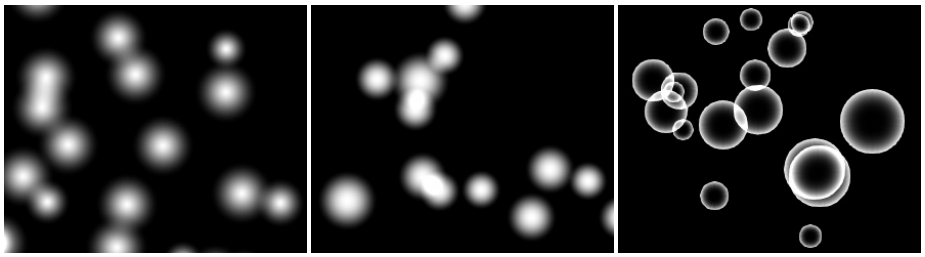


Figure 6.5 – Intensity distributions of different types of foam particles, namely: spray particles (left), surface foam particles (middle) and air bubble particles (right). Images are taken from [ADAT13].

remain visible for a sufficiently long time and fade smoothly near the end of its lifetime. Furthermore, when a particle lies in the back of the closest fluid surface (i.e. $0 < \mathbf{e}^{frag}_{fluid_z} < \mathbf{e}^{frag}_{foam_z}$, where $\mathbf{e}^{frag}_{fluid_z}$ is the eye space z coordinate of the fluid surface), we apply an additional falloff to its intensity, which is defined as

$$f_{att} = f(\mathbf{e}^{frag}_{foam_z} - \mathbf{e}^{frag}_{fluid_z}, \eta_{max}, \eta_n, \eta_m),$$

with the limiting distance η_{max}, where the foam fragment completely fades to invisible, and η_n and η_m are the exponents for shaping the attenuation curve.

At the end of this render pass, the final foam thickness values are stored in a texture (see Figure 6.3a). In the next pass, the computed thickness values are processed and converted to normalized intensity values to lie between 0 and 1. For all subsequent passes, a screen-filling quad is rendered to further process the relevant information that are saved in the textures.

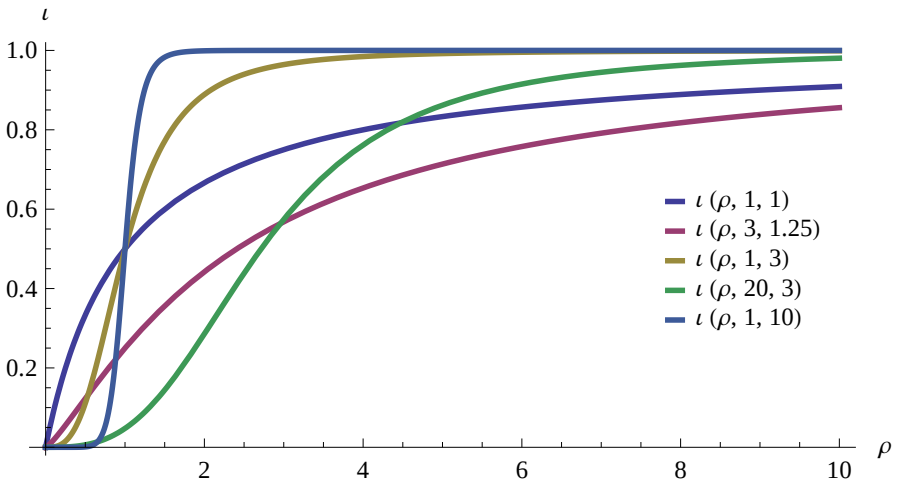


Figure 6.6 – Different forms of the sigmoid function that can be applied to the accumulated foam densities. The function can be used to create different distributions as well. For instance, to reduce the intensities below some threshold, $\rho_{exp} \geq 2$, can be used. We use the $\iota(\rho, 3, 1.25)$ form in our experiments. Graph is taken from [ADAT13].

6.2.3 Intensity Estimation

As foam is composed of more bubble layers, it scatters more of the incoming light. We use this knowledge to relate the foam intensity proportional to foam thickness. A texel from the previous render pass may have any value between $[0, \infty)$. In this render pass, we scale the values taken from that texture to the interval $[0, 1]$. However, scaling the values linearly to the target interval would make sparse areas invisible. We expect the foam to become completely opaque after some thickness threshold. Therefore, to increase the effective range of the thinner regions, to reduce the range of thicker regions and to normalize the intensities, we define the following sigmoid function ι to non-linearly scale a pixel thickness value ρ as

$$\iota(\rho, \rho_{mod}, \rho_{exp}) = \frac{\rho^{\rho_{exp}}}{\rho_{mod} + \rho^{\rho_{exp}}},$$

where $\rho_{mod} > 0$ and $\rho_{exp} > 0$ control how fast the function grows. Note that if $\rho > 0$ and $\rho_{exp} > 0$, $0 < \iota < 1$. ι is illustrated in Figure 6.6 for different parameters. Furthermore, Figure 6.7-top shows the effect of using different ρ_{mod} values.

At the end of this step, the normalized intensities are saved in a texture, which will be used in the following steps (see Figure 6.3b).

6.2.4 Foam Radiance Estimation

Since foam is composed of many transparent layers of air bubbles, light can travel through it and then scatter. Until the current stage of our pipeline, we assume that foam scatters light uniformly, where the intensity of the light was

Figure 6.7 – Application of different parameters for the setting presented in Figure 6.1-middle. top-left: $\rho_{mod} = 1$; top-right: $\rho_{mod} = 5$; bottom-left: $AO_{ShScale}$ =0.1; bottom-right: $AO_{ShScale} = 2$. Images are taken from [ADAT13].

only related to the foam thickness. In this section, we determine the regions which should receive, and therefore scatter less light using ambient occlusion (AO), and generate shadows for these regions (Section 6.2.4.1). Furthermore, the intensities that are computed in the previous section do not employ any knowledge about the actual illumination that comes from the scene. In this render pass, we will also use a very rough screen space approximation of the irradiance from the surrounding environment, which is used to colorize the foam fragments (Section 6.2.4.2).

This render pass again gets the textures that have been computed in the previous step as input and computes two additional textures, one for the shadow and another for the illumination of the foam (see Figure 6.3).

6.2.4.1 Shadow Generation

As object space AO methods (e.g. [ZIK98, Bun05, RWS+06]) are very expensive to compute, especially for complex dynamical phenomena such as foam, we investigated SSAO techniques [TCM06, Mit07, SA07, RGS09, BS09, HL10]. Finally, we decided to build our SSAO approach upon the basic concept explained in [Mit07] because of its efficiency and simplicity. One important difference of our method in comparison to [Mit07] is that we apply multiple sample collection iterations to capture both small scale and large-scale occlusions. Instead of increasing search radii, [HL10] used multiple depth maps with decreasing resolution to achieve the same effect. The search radii and total number of passes

are controlled by three parameters: the initial search radius factor $AO_{InitSRFac}$, which is a factor for h^{frag} to capture small scale occlusions; the search radius increment factor $AO_{SRIncFac}$, which is another factor for h^{frag} to determine how much the search radius increases in each sample collection step; and finally $AO_{\#Passes}$, which limits the total number of SSAO passes. For each fragment, 3d samples are generated within the fragment search radius:

$$h_{pass}^{frag} = h^{frag} \left(AO_{InitSRFac} + AO_{SRIncFac} \cdot AO_{Pass} \right),$$

where AO_{pass} increases by 1 in each sample collection pass and $AO_{pass} \leq AO_{\#Passes}$. In our experiments we used: $AO_{InitSRFac} = 1$, $AO_{SRIncFac} = 7$ and $AO_{\#Passes} = 3$.

The total number of samples ν in each sample collection pass is controlled by a user defined sampling density parameter AO_{SDens} as

$$\nu = \text{clamp} \left(\frac{3}{4} \pi {h_{pass}^{screen}}^3 \, AO_{SDens}, \, AO_{\#MinSamp}, \right.$$
$$\left. AO_{\#MaxSamp} \right),$$

where h_{pass}^{screen} is the search radius projected to fragment coordinates. Since h^{frag} can be very small for distant fragments, a minimum value $AO_{\#MinSamp}$ is used for ν. An upper limit $AO_{\#MaxSamp}$ is also introduced to prevent too many samples from being generated for fragments that are very close to the viewer. In our experiments, we used $AO_{SDensity} = 0.5$, $AO_{\#MinSamp} = 16$ and $AO_{\#MaxSamp} = 512$. The samples are created inside a cube in the range $[-1, 1]$ on all axes using the Halton sampling algorithm with a constant seed [Hal64], which produces low-discrepancy sequences. Subsequently, the samples are mapped to a sphere by simply neglecting the samples that lie outside of the sphere in the range $[-1, 1]$.

Additionally, the occlusion contribution λ of a sample $\mathbf{s}$ depends on its distance to the fragment and we compute it using a quadratic falloff as

$$\lambda = \left(1 - |\mathbf{s}|\right)^2.$$

Furthermore, if a sample is occluded by a fragment with a distance larger than the user defined $AO_{MaxOcclDist}$, the sample does not contribute to the visibility of the fragment. This effect is necessary to prevent occlusion by distant fragments and is controlled using a quadratic falloff function as

$$\delta = \max \left[\left(1 - \frac{\left| \mathbf{e}_{foam_z}^{frag} - s_z \right|}{AO_{MaxOcclDist}} \right), 0 \right]^2,$$

where $AO_{MaxOcclDist} = 5$ is used in our experiments. The sample $\mathbf{s}$ is used to look up the occlusion in eye space by other fragments (e.g. foam, fluid and solid fragments) in the scene. Based on the knowledge collected so far, the occlusion k of a sample is defined as

$$k = \begin{cases} 1 & \left[\left(s_z > \mathbf{e}_{foam_z}^{frag} \vee s_z > \mathbf{e}_{fluid_z}^{frag} \vee \right. \right. \\ & \left. \left. s_z > \mathbf{e}_{rigid_z}^{frag} \right) \wedge (0 < \delta < 1) \right] \\ 0 & \text{otherwise} \end{cases},$$

which basically states that; if a sample is occluded by any other fragment in the scene and if the occlusion distance is not larger than $AO_{MaxOcclDist}$, the sample is occluded.

Afterwards, we compute the occlusion factor ω of a fragment as

$$\omega = \frac{\sum_{AO_{pass}=1}^{AO_{\#Passes}} \left(\sum_{i=1}^{\psi} \lambda_i \cdot \delta_i \cdot k_i \cdot a_i \right)}{\sum_{AO_{pass}=1}^{AO_{\#Passes}} \left(\sum_{i=1}^{\psi} \lambda_i \right)},$$

where for the pass AO_{pass}, i iterates over all generated samples that are inside the render area (denoted as ψ), and a_i is the transparency of the sampled fragment, which is equivalent to ι_i for foam fragments. For rigid and fluid fragments, a_i is equivalent to the fragment's transparency. Additionally, if there are multiple overlapping transparent fragments at a sample position, a_i is computed by adding all of the transparency values.

Finally, so as to be more flexible about the appearance of the generated shadows, we compute the final shadow value ζ clamped into $[0, 1]$ as

$$\zeta = \text{clamp}\left[(\omega \cdot AO_{ShScale})^{AO_{ShExp}} + AO_{ShOffset},\, 0,\, 1 \right]$$

which is controlled by three self-explanatory user defined parameters. In the presented scenarios, we used: $AO_{ShScale} = 1$, $AO_{ShExp} = 1.5$ and $AO_{ShOffset} = -0.05$. The ambient occlusion step especially improves the regions that have similar intensities, which would look totally flat otherwise (e.g., see Figure 6.8, top-middle). Furthermore, Figure 6.7-bottom shows the effect of different $AO_{ShScale}$ values. The computed ζ values are written to a texture to be further used by the final composition step (see Figure 6.3c).

6.2.4.2 Irradiance

If the scene is illuminated using image based lighting, we approximate the direct illumination of each foam fragment by looking up the environment map that has been used as the light source. Using the fragment normal $\mathbf{n}^{frag}$, we create a hemisphere around the normal and use the already generated samples from the SSAO step to create direction vectors $\mathbf{n}_i^{sample}$ that are used for looking up the intensity $\mathbf{P} = (r, g, b)$ at an environment map position. Finally, the irradiance that is coming from the environment to a fragment location is simply computed in a cosine weighted fashion as

$$\mathbf{I} = \left(\frac{\sum_{i=1}^{\psi} \mathbf{P} \cdot \left(\mathbf{n}_i^{sample} \cdot \mathbf{n}^{frag} \right)}{\psi} \right),$$

where i iterates only over the samples that are generated for the first sample collection pass. The sole purpose of this step is to reflect the hue of the environment onto the foam fragments to make the foam not look too distinct from the rest of the scene. Finally, the computed $\mathbf{I}$ values are written to another texture to be used by the next and the final render pass (see Figure 6.3d). The performance of this step can be improved by using an irradiance environment map and making color look-up once for every $\mathbf{n}^{frag}$.

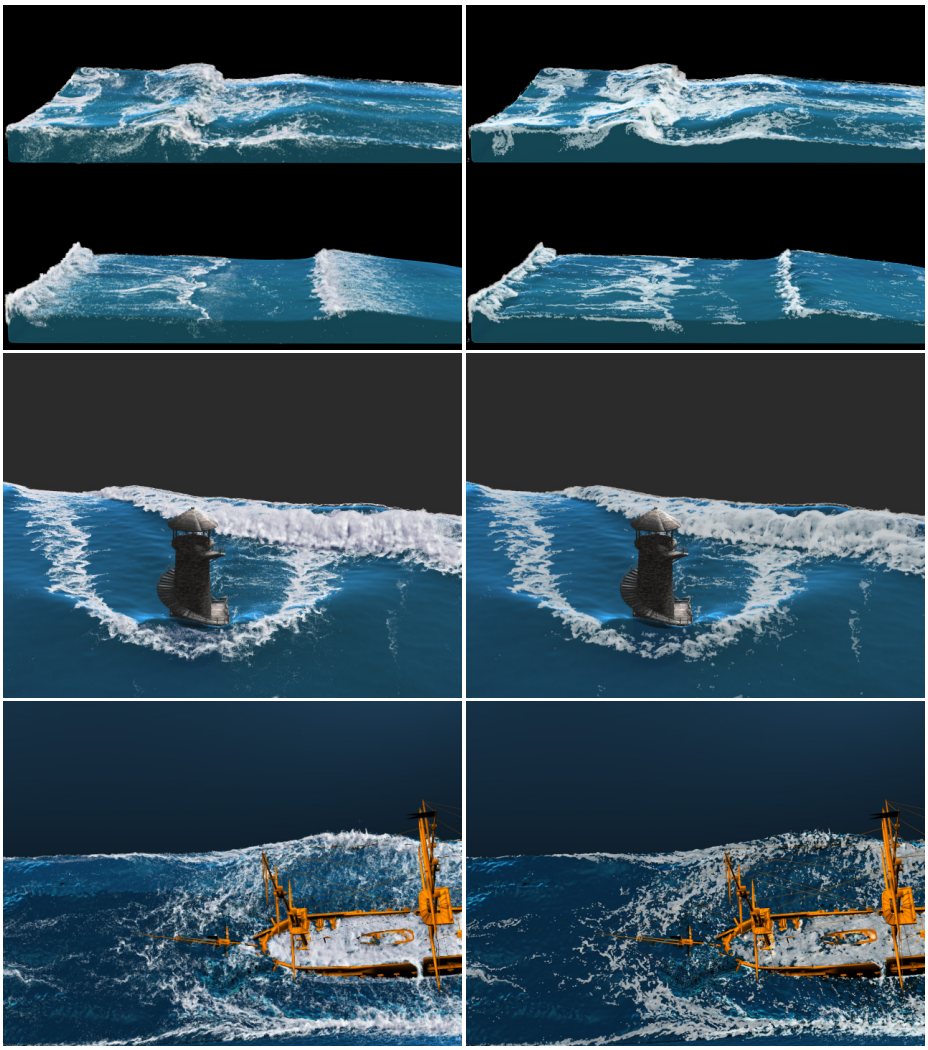


Figure 6.8 – Comparisons of our method (left) to volume ray-casting that computes emission and absorption only (right). As our method approximates shadows in concave regions, the foam looks more volumetric and detailed. The scenes are named from top to bottom as: Wave, Lighthouse and Ship. Images are taken from [ADAT13].

Scene	# Foam particles	Resolution	Average foam rendering time per frame			
			[IAAT12]	[FAW10]	Ours	
					Intensity	Total
Wave	up to 820K	800 × 600	2 min 10 s	235 ms	8 ms	52 ms
Ship	up to 9M	800 × 600	4 min 20 s	760 ms	16 ms	102 ms
Lighthouse	up to 15M	800 × 600	7 min 3 s	1 s	21 ms	150 ms
Flood	up to 29M	1280 × 960	16 min 19 s	1.7 s	33 ms	235 ms

Table 6.1 – Performance analysis of the example scenes.

6.2.5 Composition

In this render pass, the information that has been created in the previous steps and the pre-rendered images of the scene without foam are composited to generate a final image of the scene with foam (see Figure 6.2).

Depending on the user defined $AO_{\#MaxSamples}$, the shadow and radiance values computed in the previous section can include high frequency noise. In order to alleviate this problem, we apply Gaussian blur with a filter radius of $\frac{3}{2}h_{pass}^{screen}$ to both textures to generate per-pixel $\zeta_{filtered}$ and $\mathbf{I}_{filtered}$ before doing the composition.

Afterwards, to compute a final shadow color ζ_{final} for a pixel, the filtered shadow values are modulated with a user defined color $\mathbf{C}_{ShadowColor}$ and clamped into $[0, 1]$ as

$$\boldsymbol{\zeta}_{final} = \text{clamp}\left[\zeta_{filtered} \cdot \left(\mathbf{C}_{white} - \mathbf{C}_{ShadowColor}\right),\, 0,\, 1\right],$$

where $\mathbf{C}_{white} = (1,\, 1,\, 1)$. We select $\mathbf{C}_{ShadowColor}$ similar to the visible color of the fluid that the foam is generated on, and it was chosen in our experiments as $(0,\, 0,\, 0.2)$ because of the dark blue appearance of the fluids in our renderings. Since $\boldsymbol{\zeta}_{final}$ will be subtracted when doing the composition, the $\mathbf{C}_{ShadowColor}$ term is subtracted from white to invert it. From our experiences, colorizing shadows makes the foam blend better with the underlying fluid.

As foam is composed of many air-liquid interfaces, it has a very large scattering albedo, which causes it to scatter most of the incoming light, but absorb only a small amount of it. Therefore, it is usually observed very bright. We control this phenomenon by linearly scaling the irradiance values $\mathbf{I}$ using a user defined parameter $C_{IrrScale}$, whose value depends on the desired foam brightness and the color range of the environment map used. Afterwards, we clamp the resulting color into the $[0, 1]$ interval to compute

$$\mathbf{I}_{final} = \text{clamp}\left(C_{IrrScale} \cdot \mathbf{I}_{filtered},\, 0,\, 1\right).$$

Finally, the composited pixel color $\mathbf{C}$ is computed as

$$\mathbf{C} = (1 - \iota)\, \mathbf{C}_{bg} + \iota \left(\mathbf{I}_{final} - \boldsymbol{\zeta}_{final}\right)$$

where $\mathbf{C}_{bg}$ is the color at the corresponding pixel position of the background image on which the foam is composited (see Figure 6.3f).

6.3 Results

In this section, we demonstrate the versatility of our approach in different animation sequences. For all presented scenes, the underlying fluid has been simulated using the methods referred in [IAAT12], and the fluid surfaces have been reconstructed based on [SSP07, AIAT12, AAIT12]. The scenes were rendered using mental ray [NVI11] on an Intel Xeon X5690 CPU with 12 GB RAM, and the foam composition pipeline was implemented using GLSL and ran on an NVIDIA 480 GTX GPU with 1.5 GB RAM. The ray-casting code used in [IAAT12] was implemented as a mental ray shader and ran on the CPU, and an optimized version based on [FAW10] was implemented on the GPU. All scenes were illuminated using image based lighting with a clear sky environment map.

For all scenes, foam was simulated using [IAAT12] and the same foam data were used for the rendering comparisons to [IAAT12]. For the comparisons shown in Figure 6.8, the ray-casting technique explained in [IAAT12] took 9 s to 20 min depending on the complexity of the frame, excluding the other scene geometry and loading of the foam data. Using the optimized volume ray-casting scheme, the computation time has been reduced down to 90 ms to 2.5 s. Using our pipeline, the foam rendering of a frame took 30 ms to 270 ms depending on the complexity of the foam in the scene being rendered, excluding the time spent for loading of the foam data from secondary storage to the GPU memory. The results produced by using a basic volume ray-casting scheme that only accounts for absorption and emission of radiance is similar to the results we achieve excluding the additional effects that are described in Section 6.2.4 (see also Figure 6.3b). Excluding those additional effects, our pipeline took between 5 ms to 39 ms per frame. See Table 6.1 for additional information about each scene. As our pipeline also takes additional effects into account (i.e. ambient occlusion and irradiance estimation), our presented foam renderings look volumetric and blend with the rest of the scene (see Figure 6.8). Note that in [IAAT12], the fluid surface has been constructed only for the fluid particles that have more than five neighbors. For our comparisons to [IAAT12], however, we used the whole fluid surface for our renderings to better estimate the SSAO of the foam by the fluid surface. Therefore, differences between the two fluid surfaces can be noticeable.

For all of our scenes, most of the rendering time has been spent on the foam radiance estimation pass (between 50-80%). Whereas, the computational overheads of the rest of the render passes were significantly lower.

6.4 Discussion and Future Work

Taking a closer look at sea foam from a distance less than a few meters, one may observe the underlying air bubbles at varying sizes that form the foam. Rendering of such scenarios is not handled by our approach. However, using an air bubble generation technique like [BDWR12] for such close-ups might be an interesting direction for future research.

For scenes where most of the light is coming from a specific direction at shallow angles (e.g. sunset scenarios), the currently employed SSAO based shadow generation technique can fail to capture the resultant potentially large shadows cast by distant objects. For such cases, an explicit shadow generation algorithm which can handle image based lighting such as the one explained in [CK09], or explicit shadow source selection as discussed in [Bjo04] can be employed. Since we assume that foam scatters most of the incident light randomly, we omitted Fresnel effect. However, it might be desirable to make the foam reflect the environment, when it is viewed from a shallow angle.

Our algorithm neglects many physical effects that could be otherwise simulated by using modern ray-tracing techniques. Those effects include; scattering of light inside the foam, influence of the foam on the appearance of the surrounding objects and vice versa. However, for large scale scenarios (e.g. as in Figure 6.1), those effects have less significance on the appearance of the foam, and our approximations can efficiently generate convincing results. However, for close-ups, the effects that we have omitted have more significance on the final outcome. For

those cases, using a volume ray-casting method that simulates light scattering can definitely yield more convincing results (e.g. [RNGF03, GLR+06]).

Although we demonstrated our rendering scheme only for the particle data generated by the method explained in [IAAT12], we believe that our pipeline is mostly applicable to the rendering of other particle based foam simulation techniques.

7 Conclusions

This thesis has presented a set of versatile techniques that allow improved interface handling in SPH simulations. First of all, we have presented a novel boundary-handling method for incompressible SPH fluids that is applicable to both one-way and two-way fluid-rigid coupling. While particle-based solvers offer the benefit that complex boundaries can be handled in a simple way, there had been no general agreement about how solid-fluid interaction should be handled. Compared to existing techniques like frozen or ghost particles, direct mesh interaction, or penalty forces, our method offers several benefits: Sampling the solids with our proposed boundary particles allows including thin and non-manifold geometries into simulations, since normal information is not needed. Our method does not require a uniform boundary sampling, which facilitates the particle initialization, especially when dealing with complex geometries. Our solution does not only account for the inhomogeneous sampling, but also considers density (and consequently pressure) discontinuities at the boundary as well as symmetry of the forces. Overall, our method adheres to the concept of SPH, is efficient to compute, and allows versatile fluid-rigid coupling.

Afterwards, we have extended our two-way rigid-fluid coupling method described to elastic-fluid coupling. Our approach firstly generates initial boundary particle setups on solids with appropriate sampling. As the simulated objects deform, our approach efficiently evaluates the boundary sampling and efficiently resamples only the necessary primitives to prevent both undesired fluid leakage in the case of under-sampling, and performance issues in the case of over-sampling. Since our approach adjusts boundary particle contributions, adding new boundary particles does not result in discontinuous forces. Our approach offers several benefits over existing works. First of all, it can handle large deformations. Since our approach generates a more uniform boundary particle sampling compared to the previous efficient sampling strategies, field variables near the boundary can be better approximated. Furthermore, it does not use additional distance based force terms. As have been shown in the experiments, our approach allows versatile two-way interaction of fluids with both slowly and rapidly deforming solids.

Next, we have presented a new surface tension force and a new fluid-solid adhesion force to improve the handling of fluid-air and fluid-solid interfaces in SPH simulations. Our surface tension force allows simulating large surface tension, minimizes surface curvature, addresses the particle-clustering problem in SPH, and conserves momentum, all at the same time. Our adhesion force allows physically plausible fluid-solid adhesion effects, including symmetric adhesion,

and can be used to model different surface wettings. Furthermore, our forces can be easily added to an existing SPH solver. Combining both forces allowed us to simulate a variety of interesting scenarios that have not been shown in the graphics literature yet.

Finally, we have presented an efficient screen-space foam rendering pipeline that can render large particle-based foam data sets on the GPU. Our approach uses a multi-pass rendering scheme, where different effects are added to the foam rendering incrementally, and the final foam rendering is composited with a pre-rendered image of the scene. The presented method can be used as an efficient alternative to ray-casting techniques for the rendering of large-scale particle-based foam data.

Curriculum Vitae

Personal Information

Name	Nadir Akinci
Date of birth	August 11, 1984
Place of birth	Ankara, Turkey

Education

2010-2014	Phd. student in the Computer Graphics Group, Informatik Institute, University of Freiburg, Germany
2007-2010	MSc. degree in the Informatik Institute, University of Freiburg, Germany
2002-2007	BSc. degree in the Computer Enginnering Department, Atilim University, Turkey
1999-2002	Yildirim Beyazit Anatolian High School

Publications

[AAT13] N. Akinci, G. Akinci, M. Teschner. "Versatile Surface Tension and Adhesion for SPH Fluids", *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2013)*, vol. 32, no. 6, pp. 182:1-182:8, November 2013.

[ADAT13] N. Akinci, A. Dippel, G. Akinci, M. Teschner. "Screen Space Foam Rendering", *Journal of WSCG*, Vol.21, No.03, pp.173-182, 2013

[AAOT13] G. Akinci, N. Akinci, E. Oswald, M. Teschner. "Adaptive Surface Reconstruction for SPH Using 3-level Uniform Grids", *WSCG proceedings,* pp.195-204, 2013

[ACAT12] N. Akinci, J. Cornelis, G. Akinci, M. Teschner. "Coupling Elastic Solids with SPH Fluids", *Journal of Computer Animation and Virtual Worlds (Proc. CASA 2013)*, 24: 195–203. doi: 10.1002/cav.1499

[AIA+12] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, M. Teschner. "Versatile Rigid-Fluid Coupling for Incompressible SPH", *ACM Transactions on Graphics (Proc. SIGGRAPH 2012)*, vol. 31, no. 4, pp. 62:1-62:8, July 2012.

[IAAT12] M. Ihmsen, N. Akinci, G. Akinci, M. Teschner. "Unified Spray, Foam and Bubbles for Particle-based Fluids", *The Visual Computer (Proc. CGI 2012)*, Volume 28, Issue 6-8, pp 669-677, 2012, doi: 10.1007/s00371-012-0697-9

[AIAT12] G. Akinci, M. Ihmsen, N. Akinci, M. Teschner. "Parallel Surface Reconstruction for Particle-based Fluids", *Computer Graphics Forum*, vol. 31, no. 6, pp. 1797-1809, 2012, doi: 10.1111/j.1467-8659.2012.02096.x. *(Presented at Eurographics 2013)*

[AAIT12] G. Akinci, N. Akinci, M. Ihmsen, M. Teschner. "An Efficient Surface Reconstruction Pipeline for Particle-Based Fluids", *Proc. VRIPHYS, Darmstadt, Germany,* pp. 61-68, Dec. 6-7, 2012.

[IABT11] M. Ihmsen, N. Akinci, M. Becker, M. Teschner. "A Parallel SPH Implementation on Multi-core CPUs", *Computer Graphics Forum*, vol. 30, no. 1, pp. 99-112, 2011, doi: 10.1111/j.1467-8659.2010.01832.

[IAGT10] M. Ihmsen, N. Akinci, M. Gissler, M. Teschner. "Boundary Handling and Adaptive Time-stepping for PCISPH", *Proc. VRIPHYS, Copenhagen, Denmark*, pp. 79-88, Nov 11-12, 2010.

Bibliography

[AAET13] G. Akinci, N. Akinci, Oswald E., and M. Teschner. Adaptive surface reconstruction for sph using 3-level uniform grids. In *Proc. WSCG*, 2013.

[AAIT12] G. Akinci, N. Akinci, M. Ihmsen, and M. Teschner. An efficient surface reconstruction pipeline for particle-based fluids. In *Workshop on Virtual Reality Interaction and Physical Simulation*, pages 61–68. The Eurographics Association, 2012.

[AAT13] N. Akinci, G. Akinci, and M. Teschner. Versatile surface tension and adhesion for SPH. *ACM Trans. on Graphics (SIGGRAPH Asia Proc.)*, 32(6), 2013.

[ACAT13] N. Akinci, J. Cornelis, G. Akinci, and M Teschner. Coupling elastic solids with smoothed particle hydrodynamics fluids. *ournal of Computer Animation and Virtual Worlds (CAVW)*, 24(3-4):195–203, 2013.

[ACF11] J. Allard, H. Courtecuisse, and F. Faure. Implicit FEM and fluid coupling on GPU for interactive multiphysics simulation. In *SIGGRAPH Talks*, 2011.

[ADAT13] N. Akinci, A. Dippel, G. Akinci, and Teschner. Screen space foam rendering. *Journal of WSCG*, 21(03):173–182, 2013.

[AIA+12] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, and M. Teschner. Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, 31(4):62:1–62:8, 2012.

[AIAT12] Gizem Akinci, Markus Ihmsen, Nadir Akinci, and Matthias Teschner. Parallel surface reconstruction for particle-based fluids. *Computer Graphics Forum*, 31:1797–1809, 2012.

[ALD06] B. Adams, T. Lenaerts, and P. Dutre. Particle Splatting: Interactive Rendering of Particle-Based Simulation Data. Technical Report CW 453, Katholieke Universiteit Leuven, 2006.

[APKG07] B. Adams, M. Pauly, R. Keiser, and L.J. Guibas. Adaptively sampled particle fluids. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, 26(3):48–54, 2007.

[ATT12] Ryoichi Ando, Nils Thurey, and Reiji Tsuruno. Preserving fluid sheets with adaptively sampled anisotropic particles. *Visualization and Computer Graphics, IEEE Transactions on*, 18(8):1202–1214, 2012.

[ATW13] Ryoichi Ando, Nils Thürey, and Chris Wojtan. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Transactions on Graphics (TOG)*, 32(4):103, 2013.

[BAV+10] Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. Discrete viscous threads. *ACM Trans. on Graphics*, 29(4):116, 2010.

[BBB07] C. Batty, F. Bertails, and R. Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 26(3):100–106, 2007.

[BBB10] Tyson Brochu, Christopher Batty, and Robert Bridson. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, 29(4):47, 2010.

[BDWR12] O. Busaryev, T.K. Dey, H. Wang, and Z. Ren. Animating bubble interactions in a liquid foam. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, 31(4):63, 2012.

[BGB11] H. Bhattacharya, Y. Gao, and A. W. Bargteil. A level-set method for skinning animated particle data. In *In roceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2011.

[BIT09] M. Becker, M. Ihmsen, and M. Teschner. Corotated SPH for deformable solids. *Eurographics Workshop on Natural Phenomena*, pages 27–34, 2009.

[Bjo04] K. Bjorke. *Image-based lighting.* GPU Gems. NVIDIA, 2004.

[BK04] M. Botsch and L. Kobbelt. A remeshing approach to multiresolution modeling. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symposium on Geometry processing*, pages 185–192, 2004.

[BKDG98] T. Belytschko, Y. Kronagauz, J. Dolbow, and C. Gerlach. On the completeness of meshfree particle methods. *Int. J. Numer. Meth. Engng.*, 43:785–819, 1998.

[Bli82] J.F. Blinn. A Generalization of Algebraic Surface Drawing. *ACM Trans. Graph.*, 1(3):235–256, 1982.

[BLS12] Kenneth Bodin, Claude Lacoursiere, and Martin Servin. Constraint fluids. *Visualization and Computer Graphics, IEEE Transactions on*, 18(3):516–526, 2012.

[Bri03] R. Bridson. *Computational aspects of dynamic surfaces.* PhD thesis, Stanford University, 2003.

[BS09] L. Bavoil and M. Sainz. Multi-layer dual-resolution screen-space ambient occlusion. In *SIGGRAPH 2009: Talks*, page 45. ACM, 2009.

[BSK+07] R. Bredow, D. Schaub, D. Kramer, M. Hausman, D. Dimian, and R.S. Duguid. Surf's up: the making of an animated documentary. In *International Conference on Computer Graphics and Interactive Techniques: ACM SIGGRAPH 2007 courses: San Diego, California*, volume 5, 2007.

[BSW10] F. Bagar, D. Scherzer, and M. Wimmer. A layered particle-based fluid model for real-time rendering of water. In *Computer Graphics Forum*, volume 29, pages 1383–1389. Wiley Online Library, 2010.

[BT07] M. Becker and M. Teschner. Weakly compressible SPH for free surface flows. In *Proc. of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 209–217, 2007.

[BTT09] M. Becker, H. Tessendorf, and M. Teschner. Direct forcing for Lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):493–503, 2009.

[BUAG12] Christopher Batty, Andres Uribe, Basile Audoly, and Eitan Grinspun. Discrete viscous sheets. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, 31(4):113, 2012.

[Bun05] M. Bunnell. Dynamic ambient occlusion and indirect lighting. *Gpu gems*, 2(2):223–233, 2005.

[BYM05] N. Bell, Y. Yu, and P. J. Mucha. Particle-based simulation of granular materials. In *Proc. of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 77–86, 2005.

[CBP05] S. Clavet, P. Beaudoin, and P. Poulin. Particle-based viscoelastic fluid simulation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 219–228, New York, NY, USA, 2005. ACM Press.

[CCW12] Kai-Chun Chen, Pei-Shan Chen, and Sai-Keung Wong. A hybrid method for water droplet simulation. In *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*, pages 341–344. ACM, 2012.

[CGFO06] Nuttapong Chentanez, Tolga Goktekin, Bryan Feldman, and James. O'Brien. Simultaneous coupling of fluids and deformable bodies. In *Proc. of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 83–89, 2006.

[CK09] Mark Colbert and Jaroslav Krivanek. Real-time dynamic shadows for image-based lighting. *ShaderX 7 - Advanced Rendering Techniques*, page Section 4.3, 2009.

[CL95] J.X. Chen and N.V. Lobo. Toward Interactive-Rate Simulation of Fluids with Moving Obstacles Using Navier-Stokes Equations. *Graphical Models and Image Processing*, 57(2):107–116, 1995.

[CL03] A. Colagrossi and M. Landrini. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *J. Comp. Phys.*, 191(2):448–475, 2003.

[CM10] N. Chentanez and M. Müller. Real-time simulation of large bodies of water with small scale details. In *Proc. of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 197–206, 2010.

[CMT04] M. Carlson, P.J. Mucha, and G. Turk. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 23(3):377–384, 2004.

[Cor05] R.D. Corbett. Point–based level sets and progress towards unorganised particle based fluids. Master's thesis, The University of British Columbia, 2005.

[Cou11] E. Coumans. Bullet physics library (version 2.78) [software], 2011. http://www.bulletphysics.org.

[CR99] S.J. Cummins and M. Rudman. An SPH Projection Method. *Journal of Computational Physics*, 152(2):584–607, 1999.

[DK01] R.A. Dalrymple and O. Knio. SPH modeling of water waves. In *Proc. Coastal Dynamics*, pages 779–787, 2001.

[DTM+12] Peng Du, Min Tang, Chang Meng, Ruofeng Tong, and Lanfen Lin. A fluid/cloth coupling method for high velocity collision simulation. VRCAI '12, pages 309–314. ACM, 2012.

[EQYF13] 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, pages 85–94, New York, NY, USA, 2013. ACM.

[ESE07] Marco Ellero, Mar Serrano, and Pep Espanol. Incompressible smoothed particle hydrodynamics. *Journal of Computational Physics*, 226(2):1731–1752, 2007.

[FAW10] R. Fraedrich, S. Auer, and R. Westermann. Efficient high-quality volume rendering of sph data. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1533–1540, 2010.

[FF01] N. Foster and R. Fedkiw. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 23–30, New York, NY, USA, 2001. ACM Press.

[FM96] N. Foster and D. Metaxas. Realistic animation of liquids. *Graph. Models Image Process.*, 58(5):471–483, 1996.

[FSJ01] R. Fedkiw, J. Stam, and H.W. Jensen. Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22, New York, NY, USA, 2001. ACM.

[GBO04] T.G. Goktekin, A.W. Bargteil, and J.F. O'Brien. A method for animating viscoelastic fluids. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, pages 463–468, 2004.

[GLR+06] W. Geiger, M. Leo, N. Rasmussen, F. Losasso, and R. Fedkiw. So real it'll make you wet. In *ACM SIGGRAPH*, 2006.

[GM77] R.A. Gingold and J.J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–398, 1977.

[Gre08] Simon Green. Particle-based fluid simulation. In *Game Developers Conference*, pages 1–33, 2008.

[GSLF05] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 24(3):973–981, 2005.

[GSSP10] Prashant Goswami, Philipp Schlegel, Barbara Solenthaler, and Renato Pajarola. Interactive SPH Simulation and Rendering on the GPU. In *SCA '10: Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2010.

[HA06] X.Y. Hu and N.A. Adams. A multi-phase SPH method for macroscopic and mesoscopic flows. *Journal of Computational Physics*, 213(2):844–861, 2006.

[Hal64] J.H. Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Communications of the ACM*, 7(12):701–702, 1964.

[Hie07] S.E. Hieber. *Particle methods for flow-structure interactions*. PhD thesis, Michigan Technological University, 2007.

[Hil94] J.F.S. Hill. The pleasures of 'perp dot' products. *Graphics gems IV*, pages 138–148, 1994.

[HK05] J.-M. Hong and C.-H. Kim. Discontinuous fluids. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, pages 915–920, 2005.

[HKK07] T. Harada, S. Koshizuka, and Y. Kawaguchi. Smoothed particle hydrodynamics on GPUs. In *Proc. of Computer Graphics International*, pages 63–70, 2007.

[HL10] T.D. Hoang and K.L. Low. Multi-resolution screen-space ambient occlusion. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, pages 101–102. ACM, 2010.

[HLL+12] Xiaowei He, Ning Liu, Sheng Li, Hongan Wang, and Guoping Wang. Local poisson sph for viscous incompressible fluids. In *Computer Graphics Forum*, volume 31, pages 1948–1958. Wiley Online Library, 2012.

[HLW+12] Xiaowei He, Ning Liu, Guoping Wang, Fengjun Zhang, Sheng Li, Songdong Shao, and Hongan Wang. Staggered meshless solid-fluid coupling. *ACM Trans. on Graphics (SIGGRAPH Asia Proc.)*, 31(6):149, 2012.

[HLYK08] Jeong-Mo Hong, Ho-Young Lee, Jong-Chul Yoon, and Chang-Hun Kim. Bubbles alive. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, 27:48:1–48:4, 2008.

[HS13] Christopher Horvath and Barbara Solenthaler. Mass preserving multi-scale sph. Pixar Technical Memo 13-04, Pixar, Pixar Animation Studios, Emeryville, July 2013.

[HWB04] B. Houston, M. Wiebe, and C. C. Batty. Rle sparse level sets. In *In Proceedings of the SIGGRAPH 2004 conference on sketches & applications*, New York, NY, USA, 2004.

[hyb11] Next Limit Technologies: Realflow 2012, Hybrido. White Paper. 2011.

[IAAT12] Markus Ihmsen, Nadir Akinci, Gizem Akinci, and Matthias Teschner. Unified spray, foam and air bubbles for particle-based fluids. *The Visual Computer*, pages 1–9, 2012. 10.1007/s00371-012-0697-9.

[IABT11] Markus Ihmsen, Nadir Akinci, Markus Becker, and Matthias Teschner. A parallel SPH implementation on multi-core CPUs. *Computer Graphics Forum*, 30(1):99–112, 2011.

[IAGT10] M. Ihmsen, N. Akinci, M. Gissler, and M. Teschner. Boundary handling and adaptive time-stepping for PCISPH. In *Proc. of VRIPHYS*, pages 79–88, 2010.

[IBAT11] M. Ihmsen, J. Bader, G. Akinci, and M. Teschner. Animation of air bubbles with SPH. In *International Conference on Graphics Theory and Application*, pages 225–234, 2011.

[ICS+13] Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. Implicit incompressible sph. *IEEE Transactions on Visualization and Computer Graphics*, 99(PrePrints):1, 2013.

[Jak01] T. Jakobsen. Advanced character physics. In *Game Developers Conference*, pages 383–401, 2001.

[JBiRBN11] Taekwon Jang, Roger Blanco i Ribera, Jinhyuk Bae, and Junyong Noh. Simulating sph fluid with multi-level vorticity. *International Journal of Virtual Reality*, 10(1):21, 2011.

[Jon24] JE Jones. On the determination of molecular fields. ii. from the equation of state of a gas. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 106(738):463–477, 1924.

[JSB96] G.R. Johnson, R.A. Stryk, and S.R. Beissel. SPH for high velocity impact computations. *Computer Methods in Applied Mechanics and Engineering*, 139(1-4):347–373, 1996.

[KAD+06] R. Keiser, B. Adams, P. Dutré, L.J. Guibas, and M. Pauly. Multiresolution particle-based fluids. Technical report, ETH Zurich, 2006.

[KAG+05] R. Keiser, B. Adams, D. Gasser, P. Bazzi, P. Dutré, and M. Gross. A Unified Lagrangian Approach to Solid-Fluid Animation. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, pages 125–134, 2005.

[KFL00] Myungjoo Kang, Ronald P Fedkiw, and Xu-Dong Liu. A boundary condition capturing method for multiphase incompressible flow. *Journal of Scientific Computing*, 15(3):323–360, 2000.

[KLL+07] Byungmoon Kim, Yingjie Liu, Ignacio Llamas, Xiangmin Jiao, and Jarek Rossignac. Simulation of bubbles in foam with the volume control method. *ACM Trans. Graph.*, 26(3), July 2007.

[KM90] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. *ACM SIGGRAPH Computer Graphics*, 24(4):49–57, 1990.

[KSGF09] Nipun Kwatra, Jonathan Su, Jón T Grétarsson, and Ronald Fedkiw. A method for avoiding the acoustic time step restriction in compressible flow. *Journal of Computational Physics*, 228(11):4146–4161, 2009.

[KTT13] Theodore Kim, Jerry Tessendorf, and Nils Thuerey. Closest point turbulence for liquid surfaces. *ACM Transactions on Graphics (TOG)*, 32(2):15, 2013.

[KVG02] Hendrik Kück, Christian Vogelgsang, and Günther Greiner. Simulation and rendering of liquid foams. In *In Proc. Graphics Interface '02*, pages 81–88, 2002.

[KW06] P. Kipfer and R. Westermann. Realistic and interactive simulation of rivers. *Proceedings of the 2006 conference on Graphics interface*, pages 41–48, 2006.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21:163–169, 1987.

[LCPF12] Michael Lentine, Matthew Cong, Saket Patkar, and Ronald Fedkiw. Simulating free surface flow with very large time steps. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation*, pages 107–116. Eurographics Association, 2012.

[LD08] T. Lenaerts and P. Dutré. Unified SPH model for fluid-shell simulations. In *ACM SIGGRAPH 2008 posters*, SIGGRAPH '08, pages 12:1–12:1, 2008.

[LGF04] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, pages 457–462, 2004.

[LKO05] Jie Liu, Seiichi Koshizuka, and Yoshiaki Oka. A hybrid particle-mesh method for viscous, incompressible, multiphase flows. *J. Comput. Phys.*, 202(1):65–93, 2005.

[LP91] L. Libersky and A. Petschek. Smooth particle hydrodynamics with strength of materials. *Advances in the Free-Lagrange Method Including Contributions on Adaptive Gridding and the Smooth Particle Hydrodynamics Method*, 395:248–257, 1991.

[LTKF08] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-Way Coupled SPH and Particle Level Set Fluid Simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):797–804, 2008.

[Luc77] L.B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013–1024, 1977.

[Max81] N.L. Max. Vectorized procedural models for natural terrain: Waves and islands in the sunset. In *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, pages 317–324, New York, NY, USA, 1981. ACM Press.

[MBE+10] Marek Krzysztof Misztal, Robert Bridson, Kenny Erleben, Jakob Andreas Bærentzen, and François Anton. Optimization-based fluid simulation on unstructured meshes. *Proceedings of Virtual Reality Interaction and Physical Simulation, VRIPHYS*, 2010.

[MCG03] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 154–159, 2003.

[MG83] JJ Monaghan and RA Gingold. Shock simulation by the particle method SPH. *Journal of Computational Physics*, 52(2):374–389, 1983.

[Mit07] M. Mittring. Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 courses*, pages 97–121. ACM, 2007.

[MK99] JJ Monaghan and A. Kos. Solitary waves on a Cretan beach. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 125:145, 1999.

[MK09] J.J. Monaghan and J.B. Kajtar. SPH particle boundary forces for arbitrary boundaries. *Computer Physics Communications*, 180(10):1811–1820, 2009.

[ML85] JJ Monaghan and JC Lattanzio. A refined particle method for astrophysical problems. *Astronomy and astrophysics*, 149:135–143, 1985.

[MM13] M. Macklin and M. Mueller. Position Based Fluids. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, page To appear, 2013.

[MMS09] V. Mihalef, D. Metaxas, and M. Sussman. Simulation of two-phase flow with sub-scale droplet and bubble effects. In *Computer Graphics Forum*, volume 28, pages 229–238. Wiley Online Library, 2009.

[Mon89] JJ Monaghan. On the problem of penetration in particle methods. *Journal of Computational Physics*, 82(1):1–15, 1989.

[Mon92] J.J. Monaghan. Smoothed particle hydrodynamics. *Ann. Rev. Astron. Astrophys.*, 30:543–574, 1992.

[Mon94] J.J. Monaghan. Simulating free surface flows with SPH. *Journal of Computational Physics*, 110(2):399–406, 1994.

[Mon00] JJ Monaghan. SPH without a tensile instability. *Journal of Computational Physics*, 159(2):290–311, 2000.

[Mon05] J.J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68(8):1703–1759, 2005.

[Mor99] J.P. Morris. Simulating surface tension with smoothed particle hydrodynamics. *Int. J. Numer. Meth. Fluids*, 33(3):333–353, 1999.

[MSD07] M. Müller, S. Schirm, and S. Duthaler. Screen Space Meshes. In *Proceedings of ACM SIGGRAPH / EUROGRAPHICS Symposium on Computer Animation (SCA)*, 2007.

[MST+04] M. Müller, S. Schirm, M. Teschner, B. Heidelberger, and M. Gross. Interaction of fluids with deformable solids. *Computer Animation and Virtual Worlds*, 15(34):159–171, 2004.

[NM06] M. B. Nielsen and K. Museth. Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets. *J. Scient. Comput.*, 26(3):261–299., 2006.

[NNSM07] M. B. Nielsen, O. Nilsson, A. Söderström, and K. Museth. Out-of-core and compressed level set methods. *ACM Transactions on Graphics*, 26(4), 2007.

[NVI11] NVIDIA ARC. mental ray 3.9 [software]. http://www.mentalimages.com/products/mental-ray/about-mental-ray.html, 2011.

[OCD11] J. Onderik, J., M. Chladek, and R. Durikovic. Sph with small scale details and improved surface reconstruction. In *In In SCCG 2011: Proceedings of the Spring Conference on Computer graphics*, 2011.

[OD08] J. Onderik and R. Durikovic. Efficient Neighbor Search for Particle-based Fluids. *Journal of the Applied Mathematics, Statistics and Informatics (JAMSI)*, 4(1):29–43, 2008.

[ODAF06] G. Oger, M. Doring, B. Alessandrini, and P. Ferrant. Two-dimensional SPH simulations of wedge water entries. *Journal of Computational Physics*, 213(2):803–822, 2006.

[OK12] Jens Orthmann and Andreas Kolb. Temporal blending for adaptive sph. In *Computer Graphics Forum*, volume 31, pages 2436–2449. Wiley Online Library, 2012.

[OKR09a] S. Oh, Y. Kim, and B.S. Roh. Impulse-based rigid body interaction in SPH. *Computer Animation and Virtual Worlds*, 20(2-3):215–224, 2009.

[OKR09b] Seungtaik Oh, Younghee Kim, and Byung-Seok Roh. Impulse-based rigid body interaction in SPH. *Comput. Animat. Virtual Worlds*, 20(2-3):215–224, 2009.

[Pan04] A. Panizzo. *Physical and numerical modelling of subaerial landslide generated waves*. PhD thesis, Universita degli studi di L'Aquila, L'Aquila, 2004.

[Pea86] D.R. Peachey. Modeling waves and surf. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 65–74, New York, NY, USA, 1986. ACM Press.

[PTB+03] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R.T. Whitaker. Particle-Based Simulation of Fluids. *Computer Graphics Forum (Proc. of Eurographics)*, 22:401–410, 2003.

[RGS09] T. Ritschel, T. Grosch, and H.P. Seidel. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 75–82. ACM, 2009.

[RKN12] Witawat Rungjiratananon, Yoshihiro Kanamori, and Tomoyuki Nishita. Wetting effects in hair simulation. In *Computer Graphics Forum*, volume 31, pages 1993–2002. Wiley Online Library, 2012.

[RMSG+08] A. Robinson-Mosher, T. Shinar, J. Gretarsson, J. Su, and R. Fedkiw. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 27:46:1–46:9, 2008.

[RNGF03] N. Rasmussen, D.Q. Nguyen, W. Geiger, and R. Fedkiw. Smoke simulation for large scale phenomena. *ACM Transactions on Graphics (TOG)*, 22(3):703–707, 2003.

[RWS+06] Z. Ren, R. Wang, J. Snyder, K. Zhou, X. Liu, B. Sun, P.P. Sloan, H. Bao, Q. Peng, and B. Guo. Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 977–986. ACM, 2006.

[RWT11] K. Raveendran, C. Wojtan, and G. Turk. Hybrid smoothed particle hydrodynamics. In *Proc. of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 33–42, 2011.

[SA07] P. Shanmugam and O. Arikan. Hardware accelerated ambient occlusion techniques on gpus. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 73–80. ACM, 2007.

[SB12] H. Schechter and R. Bridson. Ghost sph for animating water. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, 31:61:1–61:8, 2012.

[SCED04] Kevin Steele, David Cline, Parris K Egbert, and Jonathan Dinerstein. Modeling and rendering viscous liquids. *Computer Animation and Virtual Worlds*, 15(3-4):183–192, 2004.

[SG11] Barbara Solenthaler and Markus Gross. Two-scale particle simulation. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, 30(4):81:1–81:8, 2011.

[She68] D. Shepard. A two-dimensional interpolation function for irregularly spaced points. In *Proceedings of the 23rd ACM national conference*, pages 517–524, 1968.

[SMVO96] Paul R Shapiro, Hugo Martel, Jens V Villumsen, and J Michael Owen. Adaptive smoothed particle hydrodynamics, with application to cosmology: methodology. *The Astrophysical Journal Supplement Series*, 103:269, 1996.

[SP08] B. Solenthaler and R. Pajarola. Density Contrast SPH Interfaces. In *Proc. of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 211–218, 2008.

[SP09] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible SPH. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, 28(3):1–6, 2009.

[SSP07] B. Solenthaler, J. Schläfli, and R. Pajarola. A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds*, 18(1):69–82, 2007.

[Sta99] J. Stam. Stable fluids. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128, 1999.

[SU94] John Steinhoff and David Underhill. Modification of the euler equations for "vorticity confinement": Application to the computation of interacting vortex rings. *Physics of Fluids*, 6:2738, 1994.

[TCM06] M. Tarini, P. Cignoni, and C. Montani. Ambient occlusion and edge cueing for enhancing real time molecular visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):1237–1244, 2006.

[TF12] Tetsuya Takahashi and Issei Fujishiro. Particle-based simulation of snow trampling taking sintering effect into account. In *ACM SIGGRAPH 2012 Posters*, page 7. ACM, 2012.

[TFK+03] T. Takahashi, H. Fujii, A. Kunimatsu, K. Hiwada, T. Saito, K. Tanaka, and H. Ueki. Realistic Animation of Fluid with Splash and Foam. *Computer Graphics Forum*, 22(3):391–400, 2003.

[THM+03] M. Teschner, B. Heidelberger, M. Müller, D. Pomeranets, and M. Gross. Optimized Spatial Hashing for Collision Detection of Deformable Objects. In *Proc. of Vision, Modeling, Visualization (VMV)*, pages 47–54, 2003.

[TM05] A. Tartakovsky and P. Meakin. Modeling of surface tension and contact angles with smoothed particle hydrodynamics. *Physical Review E*, 72(2):26301, 2005.

[TWGT10] Nils Thürey, Chris Wojtan, Markus Gross, and Greg Turk. A multiscale approach to mesh-based surface tension flows. *ACM Trans. on Graphics (SIGGRAPH Proc.)*, 29(4):48, 2010.

[vdLGS09] W.J. van der Laan, S. Green, and M. Sainz. Screen space fluid rendering with curvature flow. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 91–98. ACM, 2009.

[Ver67] L. Verlet. Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.*, 159(1):98–103, 1967.

[Wen95] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in computational Mathematics*, 4(1):389–396, 1995.

[WS95] M.S. Warren and J.K. Salmon. A portable parallel particle program. *Computer Physics Communications*, 87(1-2):266–290, 1995.

[YLHQ12] L. Yang, S. Li, A. Hao, and H. Qin. Realtime two-way coupling of meshless fluids and nonlinear fem. In *CGF*, volume 31, pages 2037–2046, 2012.

[YT10] J. Yu and G. Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 217–225. Eurographics Association, 2010.

[YWH+09] H. Yan, Z. Wang, J. He, X. Chen, C. Wang, and Q. Peng. Real-time fluid simulation with adaptive SPH. *Computer Animation and Virtual Worlds*, 20(2-3):417–426, 2009.

[YWTY12] Jihun Yu, Chris Wojtan, Greg Turk, and Chee Yap. Explicit mesh surfaces for particle based fluids. In *Computer Graphics Forum (Eurographics Proc.)*, volume 31, pages 815–824. Wiley Online Library, 2012.

[ZB05] Y. Zhu and R. Bridson. Animating sand as a fluid. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 965–972, New York, NY, USA, 2005. ACM Press.

[ZGHG11] K. Zhou, M. Gong, X. Huang, and B. Guo. Data-parallel octrees for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):669–681, 2011.

[ZIK98] S. Zhukov, A. Iones, and G. Kronin. An ambient light illumination model. *Rendering techniques*, 98:45–55, 1998.

[ZLC+13] Bo Zhu, Wenlong Lu, Matthew Cong, Byungmoon Kim, and Ronald Fedkiw. A new grid structure for domain extension. *ACM Transactions on Graphics (TOG)*, 32(4):63, 2013.

[ZPvBG01] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, New York, NY, USA, 2001. ACM Press.

[ZSP08] Y. Zhang, B. Solenthaler, and R. Pajarola. Adaptive Sampling and Rendering of Fluids on the GPU. In *Proceedings Symposium on Point-Based Graphics*, pages 137–146, 2008.

[ZYF10] Bo Zhu, Xubo Yang, and Ye Fan. Creating and preserving vortical details in sph fluid. In *Computer Graphics Forum*, volume 29, pages 2207–2214. Wiley Online Library, 2010.