PHYSICS-BASED ANIMATION OF LARGE-SCALE SPLASHING LIQUIDS, ELASTOPLASTIC SOLIDS, AND MODEL-REDUCED FLOW

by

Daniel James Gerszewski

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computing

School of Computing

The University of Utah

August 2014

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Daniel James Gerszewski**

has been approved by the following supervisory committee members:

Adam W. Bargteil	, Chair	**5/9/2014** Date Approved
Robert M. Kirby	, Member	**5/8/2014** Date Approved
Charles Hansen	, Member	**5/14/2014** Date Approved
Peter-Pike Sloan	, Member	**5/20/2014** Date Approved
James F. O'Brien	, Member	**6/10/2014** Date Approved

and by **Ross Whitaker**, Chair/Dean of

the Department/College/School of **Computing**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

Physical simulation has become an essential tool in computer animation. As the use of visual effects increases, the need for simulating real-world materials increases. In this dissertation, we consider three problems in physics-based animation: large-scale splashing liquids, elastoplastic material simulation, and dimensionality reduction techniques for fluid simulation.

Fluid simulation has been one of the greatest successes of physics-based animation, generating hundreds of research papers and a great many special effects over the last fifteen years. However, the animation of large-scale, splashing liquids remains challenging. We show that a novel combination of unilateral incompressibility, mass-full FLIP, and blurred boundaries is extremely well-suited to the animation of large-scale, violent, splashing liquids.

Materials that incorporate both plastic and elastic deformations, also referred to as elastioplastic materials, are frequently encountered in everyday life. Methods for animating such common real-world materials are useful for effects practitioners and have been successfully employed in films. We describe a point-based method for animating elastoplastic materials. Our primary contribution is a simple method for computing the deformation gradient for each particle in the simulation. Given the deformation gradient, we can apply arbitrary constitutive models and compute the resulting elastic forces. Our method has two primary advantages: we do not store or compare to an initial rest configuration and we work directly with the deformation gradient. The first advantage avoids poor numerical conditioning and the second naturally leads to a multiplicative model of deformation appropriate for finite deformations.

One of the most significant drawbacks of physics-based animation is that ever-higher fidelity leads to an explosion in the number of degrees of freedom.

This problem leads us to the consideration of dimensionality reduction techniques. We present several enhancements to model-reduced fluid simulation that allow improved simulation bases and two-way solid-fluid coupling. Specifically, we present a basis enrichment scheme that allows us to combine data-driven or artistically derived bases with more general analytic bases derived from Laplacian Eigenfunctions. Additionally, we handle two-way solid-fluid coupling in a time-splitting fashion—we alternately timestep the fluid and rigid body simulators, while taking into account the effects of the fluid on the rigid bodies and vice versa. We employ the vortex panel method to handle solid-fluid coupling and use dynamic pressure to compute the effect of the fluid on rigid bodies.

Taken together, these contributions have advanced the state-of-the art in physics-based animation and are practical enough to be used in production pipelines.

For my parents Jim and Monika and my grandfather Richard

CONTENTS

LIST OF FIGURES

ACRONYMS

Two-Dimensional (2D)

Three-Dimensional (3D)

Graphics Processing Unit (GPU)

Unilateral Incompressibility Constraint (UIC)

Proper Orthogonal Decomposition (POD)

Computational Fluid Dynamics (CDF)

Modified Proportioning with Reduced Gradient Projections (MPRGP)

Modified Incomplete Cholesky (MIC)

Linear Complementary Problem (LCP)

Fluid Implicit Particle (FLIP)

Particle in Cell (PIC)

Smoothed Particle Hydrodynamics (SPH)

Singular Value Decomposition (SVD)

ACKNOWLEDGMENTS

I would like to thank my advisor Adam Bargteil for his guidance in all aspects of getting through graduate school. Your mentorship, patience, knowledge, and words of wisdom have been extremely valuable. I am grateful for the time spent tirelessly going through my drafts, providing feedback on presentations, and correcting me when I go off course.

I would also like to thank the members of my committee, Chuck Hansen, Mike Kirby, Peter-Pike Sloan, and James O'Brien. I would like to thank Chuck for his guidance when I was an undergraduate and being instrumental in my decision to attend graduate school. Your early graphics classes sparked my interest in computer graphics research. I would like to thank Mike for his rigorous scientific computing classes that inspired my work greatly. I feel lucky for the opportunity I had to work at Disney Research with Peter-Pike Sloan and Ladislav Kavan. I couldn't have asked for better mentors who taught me more than they know. Thank you Peter-Pike for taking the time to explain the details; you have a way of teaching that opens my mind. I would also like to thank James for all your feedback on my research; your comments and questions were greatly appreciated. I would also like to thank all the students and faculty I had the opportunity to work with and learn from.

Lastly, I would like to thank my friends and family for their love and support. I owe a great deal to my brother-in-law Ryan Petrie whose excitement for computer graphics was infectious. Thank you for teaching me the basics of programming and helping me write my first renderer. I'm grateful for all the awesome relationships and roommates I've had over the years and for the great times and good company. I owe the greatest thanks to my family for their continued love and support and always believing in me.

CHAPTER 1

INTRODUCTION

Over the last decade and a half, physical simulation has become an essential tool in computer animation. With advances in rendering and physics-based animation, many stories where the visuals were previously left to the imagination of readers are now realistically brought to life in film. Nearly every major film uses visual effects in some way. Some blockbuster films use visual effects for nearly every shot, allowing film makers, writers, and directors to tell any story they like. As the demand for visual effects increases, the need for realistically animating real-world materials increases. A major goal in computer animation research is to simulate the behavior of real-world materials, including such phenomena as fracturing rigid and soft bodies, cloth, hair, explosions and smoke, flowing and splashing liquids, and even muscle deformations.

This dissertation considers three problems in physics-based animation: large-scale splashing liquids, elastoplastic material simulation, and dimensionality reduction techniques for fluid simulation. This chapter continues with the introduction of these problems which are covered in detail in later chapters, Large-scale splashing liquids in Chapter 3, elastoplastic material simulation in Chapter 4, and finally reduced fluid simulation in Chapter 5.

Computers have been used to simulate fluids for scientific and engineering applications since the advent of the electronic computer. Over the last fifteen years, fluid simulation has emerged as one of the most effective applications of physics-based approaches to animation and has become a valuable tool for movie production, producing realistic bodies of water, fire, smoke, and other

phenomena. The use of fluid simulation for special effects is now commonplace and the topic has received copious attention in the graphics community and several fluid simulations have won awards. For example, the Visual Effects Society awarded the tidal wave in *The Day After Tomorrow* the award for *The Best Single Visual Effect of the Year*, while Digital Domain, Dreamworks, and Industrial Light and Magic have each received *Academy Awards* for their fluid simulators. However, despite tremendous progress, challenges remain. One such challenge is the simulation of large-scale splashing liquids. Such simulations are challenging for a variety of reasons. Simulating large-scale liquids with fine-scale details can require high grid resolutions, which dramatically increases computational costs. Splashing liquids separate into a variety of scales from large blobs down to fine mist, and capturing these scales can be difficult with surface tracking methods. Allowing liquid to separate and expand freely is also restricted by the standard incompressible fluid solvers.

Because at the spatial and temporal scales we seek to animate, liquids compression is negligible, computer graphics researchers have largely focused on simulating incompressible fluids. Even approaches, such as smoothed particle hydrodynamics (SPH), that are naturally suited to simulate compressible flow are often modified for incompressible flow [63]. While computationally efficient, incompressibility induces an artificial surface tension that prevents liquid near the surface from mixing with the surrounding air. This mixing is important at large scales, especially during violent splashes, such as after underwater explosions. In Chapter 3, we show that for single-phase fluid simulation, such mixing is effectively modeled with unilateral incompressibility [50, 51], which allows positive divergence while prohibiting negative pressures, thus avoiding the pressure oscillations found in compressible simulation, while removing the artificial surface tension caused by bilateral incompressibility.

We use a variant of the fluid-implicit-particle (FLIP) method as our underlying simulation method. However, our approach, which we call mass-full FLIP, attaches mass to the particles and more closely resembles compressible FLIP [9]

than the incompressible variety [76]. Mass-full FLIP is extremely well-suited to the unilateral incompressibility (UIC) solve. In the context of UIC, ensuring conservation of mass becomes difficult—allowing positive divergence can result in significant volume gain. Like SPH methods, mass-full FLIP conserves mass by conserving particles. Additionally, the UIC is most appropriate in highly turbulent simulations where the numerical viscosity associated with semi-Lagrangian and related schemes would be especially inappropriate.

Finally, we treat obstacles and fluid in a unified manner—we discretize obstacles using particles and rasterize their mass onto the background grid using the same trilinear kernel. We additionally employ the variational approach to obstacles endorsed by Batty and colleagues [4]. Combined with unilateral incompressibility, our treatment of boundaries easily allows liquids to separate from obstacles, avoiding the common visual artifact of liquid gliding along the ceiling.

In Chapter 3, we show that a novel combination of unilateral incompressibility, mass-full FLIP, and blurred boundaries is extremely well-suited to the animation of such liquids and avoids common artifacts such as artificial surface tension, volume loss/gain, and fluid sticking to obstacles. We demonstrate our approach on several examples, such as the flooding of a city. Side-by-side comparisons with incompressible simulations clearly demonstrate the different behavior afforded by our approach. In general, the more tumultuous the motion, the more different the results. While we do not expect our approach to replace bilateral incompressibility, we believe the rich behavior afforded by it will prove an important tool for animating large-scale splashing liquids.

Materials that incorporate both plastic and elastic deformations such as chewing gum, toothpaste, shaving cream, sauces, bread dough, and modeling clay are frequently encountered in everyday life and have been successfully used in special effects such as the honey in *Bee Movie* [58] and the food in *Ratatouille* [26]. In fact, the later work won the Visual Effects Society award for *Outstanding Effects in an Animated Motion Picture*. At the same time, point-based simulation methods

have increased dramatically in popularity and sophistication in recent years. These methods are capable of modeling a wide range of materials in a variety of contexts, from real-time fluid simulations [11] to fracturing solids [54]. Their versatility makes them especially attractive for computer graphics applications.

We describe a point-based approach for animating elastoplastic materials. Our primary contribution is a simple method for computing the deformation gradient for each particle in the simulation. The deformation gradient is computed for each particle by finding the affine transformation that best approximates the motion of neighboring particles over a single timestep. This transformation is found using a least-squares fit to the positions of neighboring particles at the beginning and end of the timestep. These transformations are then multiplicitively composed to compute the total deformation gradient that describes the deformation around a particle over the course of the simulation.

Our approach has two primary advantages. First, we do not store and compare to an initial rest state. Under large plastic deformations, the mapping from an initial rest state to the current state becomes numerical ill-conditioned. By storing only the elastic part of the deformation, we avoid these numerical problems. Second, instead of working with a strain metric, we work directly with the deformation gradient. By focusing on the deformation gradient, our approach can handle arbitrary constitutive models [35]. More importantly, working with the deformation gradient naturally leads to a multiplicative formulation of deformation, which is more suitable to finite deformations than the additive models from classical plasticity that are often used in graphics [61]. In Chapter 4, we demonstrate our approach on a number of examples that exhibit a wide range of material behaviors.

One of the most significant drawbacks of physics-based animation is "the curse of dimensionality"—the quest for ever-higher fidelity leads to an explosion in the number of degrees of freedom. This problem naturally leads to the consideration of dimensionality reduction techniques. By constructing a problem-specific model of our fluid, we can reduce the number of degrees of freedom to only

the ones needed for our specific problem. While this is less accurate than a full dynamics simulation, we can tailor the simulation to our specific time and computational requirements.

In Chapter 5, we present several enhancements to the basic reduced fluid simulation pipeline. Specifically, we present a basis enrichment scheme for combining analytic, data-driven, and artistically authored bases as well as a new approach to two-way solid-fluid coupling that scales to a large number of rigid bodies.

The analytic bases act somewhat like regularization, allowing our approach to generalize outside the training data and thus requiring significantly less training data without the risk of over-fitting. We treat two-way solid-fluid coupling in a time-splitting fashion—we first compute the effect of the solid on the fluid and then compute the effect of the fluid on the solid. We employ a *vortex panel method* to compute obstacles' effects on the fluid and dynamic pressure to compute forces induced on the obstacle by the surrounding fluid. In a precomputation step, we account for the geometric boundary of each object, which involves assembling and inverting a dense "panel matrix;" however, at runtime, solid-fluid coupling reduces to a matrix multiplication for each object. We handle multiple obstacles by iteratively computing the coupling in a way similar to Schwarz alternating methods [68]. Fluid-solid coupling is achieved using dynamic pressure to compute forces on solid objects from fluid velocities; these forces are then treated as external forces in a rigid body simulator. Our results demonstrate that our enhancements are practical for two-way coupled reduced fluid simulation with rigid bodies.

1.1 Thesis Statement

The choice of the degrees of freedom for simulating real-world phenomena is important. By combining degrees of freedom like particles, grids, different global bases, and vortex panels, we can make tradeoffs in simulation speed and detail to effectively simulate various real-world phenomena.

CHAPTER 2

RELATED WORK

2.1 Fluid Simulation

The most popular approach for animating fluids has been to discretize the governing equations on a regular Cartesian grid and staggering the velocity and pressure samples, known as the "staggered-grid" [30]. In graphics, the first use of this approach for a grid-based liquid simulation in 3D was introduced by Foster and Metaxas [24], who pioneered its use for graphics applications. Semi-Lagrangian advection was introduced by Stam [65], which allowed simulations to remain stable even with very large timesteps, but also led to excessive numerical dissipation. To address excessive dissipation, Fedkiw *et al.* [22] used vorticity confinement and higher order interpolation; additionally, this method was incorporated into a liquid solver with a combination of level sets and marker particles for surface tracking [23]. This approach was later extended by Enright *et al.* [21], who used marker particles on both sides of the interface as well as velocity extrapolation into the air.

The goal of our method is to simulate large-scale splashing liquids without sacrificing small-scale details, which is very similar to that of Losasso and colleagues [43], who coupled incompressible flow to a particle system to achieve some of the first really convincing animations of breaking waves. Their particle system is quite similar to our mass-full FLIP, though they do not create particles deep in the body of the fluid. More significantly, like us, they adopted variable densities and are able to achieve fluid expansion in the form of spray and foam. However, they used these variable densities as targets for bilateral

incompressibility, whereas we adopt UIC. McAdams and colleagues [47] adopted this density targeting approach to precondition collisions in hair simulation and Solenthaler and colleagues [63] also used density targeting to reduce fluctuations in SPH simulations. Raveendran and colleagues [57] advocated using a coarse grid projection to reduce compressibility in SPH simulations. Lentine and colleagues [39] similarly use coarse grids to resolve large-scale divergence and then perform finer scale projections to achieve high detail. Bodin and colleagues [7] introduced inequality constraints to incorporate boundaries into incompressibility solves in SPH fluid simulations.

The most closely related work to ours from a technical standpoint is the work of Narain and colleagues, who introduced unilateral incompressibility and applied it to two-dimensional crowd simulation [50] and to the animation of granular materials [51]. The latter approach was extended to the PCISPH framework by Alduán and colleagues [2]. These impressive results prompted us to consider the application of unilateral incompressibility to liquids. The chief technical innovation required in the context of liquids is the unified treatment of liquid and obstacles through particle sampling and rasterization with the same trilinear filter.

Recently, Schechter and Bridson [60] introduced *Ghost SPH* to address artificial surface tension in SPH simulations. Their approach involves using improved boundary conditions through the introduction of ghost particles. This technique alleviates particle clumping that results when particles do not have sufficient neighbors to reach their target density—by introducing ghost particles, the target density is reached. Another solution to this problem was presented by Macklin and Müller [46]. Our solution is different. Instead of modifying the free-surface boundary conditions, we place a one-sided constraint on the divergence, which is more analogous to turning off SPH pressure forces if the density were below the target.

However, the underlying causes of artificial surface tension in SPH and FLIP simulation are similar, but different. Artificial surface tension in SPH results

from small particle neighborhoods clumping together to try and achieve a target density, while in grid-based and FLIP simulations, artificial surface tension is caused by negative pressures that result from disallowing positive divergence.

Chentanez and Müller-Fischer [14] also set out to create large-scale effects. To do so in real-time they developed an Eulerian "tall-cell" simulation system that runs on a GPU. They also incorporated important secondary effects, including wave textures and spray, mist, and foam particles. Incorporating such elements into our approach would likely lead to a much richer visual experience. The same year, they also achieved separation from solid boundaries by solving a linear complementarity problem [13]. They adopted a multigrid solver and enforced non-negative pressures only in obstacle boundaries, whereas we use preconditioned conjugate gradient wrapped in an active-set method and disallow negative pressures in all cells near the liquid surface.

We also draw on the work of Batty and colleagues [4] for handling boundary conditions. Similar to their work, we take into account the volume occupied by obstacles to adjust the amount of fluid that can enter a cell. However, while they use a box filter for obstacles, we use the same trilinear filter for rasterizing obstacles as we use for particles. We also note that they were the first to cast wall-separation as a linear complementarity problem, which they solved with a PATH solver that did not scale to large problems.

There is a rich body of work on fluid simulation in computer graphics. A complete survey is beyond the scope of this dissertation, but we wholeheartedly refer the interested reader to the book by Bridson [10] or, for the more mathematically oriented, the text by Chorin and Marsden [15].

2.2 Point-based Methods and Elastoplastic Materials

For a complete survey of point-based methods, we heartily recommend the book by Gross and Pfister [28]. We focus our attention on methods for animating elastoplastic solids and viscoelastic fluids. Terzopoulos and Fleisher [67] introduced inelastic deformations, including viscoelasticity, plasticity,

and fracture, to the graphics community. O'Brien and colleagues [52] incorporated a similar plasticity model into a finite element simulation to animate ductile fracture. To avoid problems with poorly conditioned or tangled elements, they demonstrated only limited amounts of plastic deformation. We also note that they used an additive model of plasticity. While such a model is appropriate when considering infinitesimal deformations, as Irving and colleagues [35] pointed out, a multiplicative model is more appropriate in the context of finite plastic deformation. Clavet *et al.* [16] modeled viscoelastic fluids with a mass-spring system in which the springs are dynamically inserted and removed. Their springs explicitly model viscous and elastic forces and include a model of plastic flow. Goktekin *et al.* [25] took an alternative approach and added elastic forces to an Eulerian fluid simulation. In their approach, a linear strain rate is integrated through time and undergoes plastic decay. Their approach used a linear model of elastic deformation that is not invariant to rotations. This shortcoming was addressed by Losasso and colleagues [42], who applied a rotation to the advected elastic strain to account for rotations in the velocity field. However, as noted by Irving [34], because this model is not based on the deformation gradient, it is unable to model hyperelastic materials.

Müller *et al.* [49] introduced a point-based method for animating elastic, plastic, and melting objects. They broke the possible deformations into two separate regimes. Deformations that were largely elastic were treated by comparing the current configuration of neighboring particles to a rest configuration, while large plastic deformations were handled by updating a strain measure in a way similar to Goktekin *et al.* [25]. Their approach to primarily elastic deformation uses moving least-squares to fit a transformation that maps neighbors in a reference shape to the current shape. While we use a very similar moving least-squares fit to compute the deformation gradient, we fit the deformation over individual timesteps and compose the deformations to arrive at the total deformation. While this approach invariably leads to drift in the deformation gradient over time, it is able to handle changing neighborhoods and large plastic flow in a unified

way. Keiser *et al.* [37] also developed a unified approach by substituting fluid dynamics for the large plastic deformation regime of Müller *et al.* [49]. Like ours, their approach is able to model a wide variety of materials in a unified way.

Solenthaler and colleagues [64] have replaced the moving least-squares approach used by Müller *et al.* [49] and Keiser *et al.* [37] with an SPH formulation. Their approach is able to model fluids, elastic, and rigid objects as well as objects that have parts of different types. Additionally, they include melting and solidification, merging and splitting, and plasticity using the model of O'Brien *et al.* [52]. More recently, Becker *et al.* [5] extended the approach of Solenthaler and colleagues by employing a corotated SPH formulation that extracts the local orientations of the object from the deformation field and calculating the elastic forces in a rotated configuration. Hieber and Koumoutsakos [32] described a Lagrangian particle method for simulating linear and nonlinear elastic solids that does not require a rest configuration. Instead of performing a least-squares fit to the deformation in every timestep, they update the deformation gradient by integrating the gradient of the velocity field. In contrast to these meshless methods, Bargteil *et al.* [3] introduced a finite element method for animating large viscoplastic flow. Their approach relied on a robust remeshing operation to maintain well-conditioned elements. Wojtan and Turk [73] improved on this approach by using embedded surface meshes, producing highly detailed animations of heavily deformed objects. By using embedded meshes, they were also able to adopt a fast and simple remeshing procedure. These last two papers are the only work in graphics that shares both the main advantages of our approach. However, our approach has the advantage of being meshless, allowing us to avoid remeshing and the consequent resampling and smoothing of simulation variables. More recently, Wicke *et al.* [70] introduced a dynamic meshing algorithm that attempts to replace as few elements as possible while still maintaing high element quality even under gross mesh deformation.

2.3 Dimensionally Reduced Fluids

Dimensionality reduction for computational fluid dynamics (CFD) has been well studied in engineering. These methods are variously referred to in the literature as proper orthogonal decomposition (POD), Karhunen-Loéve decomposition, or subspace integration. The POD method has been used in many applications involving dimensionality reduction of complex flows and to investigate coherent structures in turbulent flows [44, 45, 33]. The snapshot POD method introduced by Sirovich [62] for the study of coherent structures can be used to create a reduced model from a series of snapshots of a simulation. Treuille and colleagues [69] introduced this snapshot technique to graphics and described how each step of a fluid simulation can be performed in the reduced space.

Since that work, researchers have also developed modular techniques by connecting fluid *tiles*, which capture specific boundary conditions, at runtime to create large novel reduced fluid simulations [71]. Additionally, researchers have also experimented with different bases for fluid simulation. Gupta *et al.* [29] used the Legendre polynomials for both simulation and rendering of participating media. Long *et al.* [40] improve upon Fourier-based solutions by shifting to the discrete sine/cosine transform to handle boundary conditions. However, this method is limited to simple domains. More recently, DeWitt *et al.* [18] used static analysis of the domain to construct a basis from Eigenfuctions of the Laplacian. In some simple domains, this basis even has a closed form. In our work, we combine this basis with the snapshot POD method. Other researchers have extended reduced fluid methods by applying a cubature approach for nonlinear functions [38] and including inverse operators for solid-fluid coupling [66].

Treuille *et al.* [69] handled solid obstacles by defining a local basis on a fixed size grid surrounding each obstacle. This local basis was created by computing the velocity field that cancels the flow into the obstacle induced by each mode of the fluid simulation basis and then applying the same snapshot POD technique to compute a compressed basis. This process was repeated for a number of

translations and rotations of the obstacle. Additionally, the rigid body motion of each object was also sampled to incorporate object movement into the local basis. At runtime, the local basis for canceling the normal flow is determined based on the location, rotation, and movement of the object. In contrast to this approach, our approach to solid-fluid coupling, based on vortex panel methods, does not limit interaction to a small region around the obstacle, has a small runtime memory footprint, avoids expensive precomputation, and allows for direct interaction between obstacles.

Other approaches for handling moving boundary conditions in reduced fluid simulations involve taking the difference of the normal velocity and the desired normal velocity, and projecting it onto the velocity basis and then subtracting the result from the reduced state [18]. This method approximates the forces up to the resolution representable by the basis modes. In contrast, our method is able to increase the resolution of our boundary conditions independent of our fluid basis.

Our approach to solid-fluid coupling makes use of the vortex panel method, which was developed to study flow around airfoils [31, 17] and was introduced to graphics by Park and Kim [53] to handle obstacles in a vortex particle method. More recent variations have been used to simulate smoke as a surface [55, 12].

CHAPTER 3

PHYSICS-BASED ANIMATION OF LARGE-SCALE SPLASHING LIQUIDS

Our approach brings together several components: unilateral incompressibility, mass-full FLIP, and blurred obstacles. While some of these techniques have been employed before inside and outside of the graphics literature, we demonstrate that their novel combination is especially effective for computer animation of large-scale splashing liquids.

3.1 Unilateral Incompressibility

For completeness, we briefly describe the unilateral incompressibility constraint (UIC) and discuss practical issues in its application to simulating liquids. The Euler equations describe the motion of inviscid fluids by stating that mass and momentum are conserved:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{u}) \tag{3.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} - \frac{\nabla p}{\rho} + \frac{\mathbf{f}}{\rho}, \tag{3.2}$$

where ρ denotes the density of the fluid, t time, $\mathbf{u}$ the velocity, p the pressure, and $\mathbf{f}$ external forces such as gravity. For incompressible flow, the constraint that density be a constant leads to the solution of a Poisson equation to determine the pressure that will lead to a divergence-free velocity field, while for compressible flow, an equation of state determines pressure as a function of density [6]. In the case of *unilaterally incompressible* flows, we place an upper bound on the density of fluid in any given cell, $\rho_{\max}$, and require that pressures be non-negative.

Additionally, we require that for any given cell, either the density is $\rho_{\max}$ or that the pressure is zero. Intuitively, this last constraint requires that the cell be full of liquid or be treated like air.

These constraints can be formulated as a linear complementarity problem,

$$\mathbf{Ap} + \mathbf{b} \geq 0 \tag{3.3}$$

$$\mathbf{p} \geq 0 \tag{3.4}$$

$$\mathbf{p}^T(\mathbf{Ap} + \mathbf{b}) = 0 \tag{3.5}$$

with the substitutions

$$\mathbf{Ap} = -\Delta t \nabla \cdot \left(v_f \nabla \mathbf{p}\right) \tag{3.6}$$

$$\mathbf{b} = \rho_{\max} - \rho_c + \Delta t \nabla \cdot \left(\rho_f \mathbf{u}\right) \tag{3.7}$$

Here, v_f is a face-centered volume fraction (see Section 3.3) representing the fraction of the volume around the face that may be occupied by liquid, and ρ_c and ρ_f are cell-centered and face-centered liquid densities, respectively (see Section 3.2). Intuitively, $\mathbf{b}$ estimates the amount of free space (air) in a cell at the end of the timestep if pressure was zero.

We solve this system using the *modified proportioning with reduced gradient projections* (MPRGP) method as described by Dostàl [20, 19], with the Modified Incomplete Cholesky (MIC(0)) preconditioner as described by Bridson [10]. MPRGP is an active-set method where the active set includes cells where the pressure is currently zero and the free set contains cells where the pressure is positive. The method interleaves conjugate gradient steps with *expansion* steps that increase the size of the active set and *proportioning* steps that add cells to the free set. The method requires an estimate of the induced norm of the matrix (the largest eigenvalue) to use as a bound on step sizes. In practice, this can be found using power iterations. We have found that warm-starting these iterations with the eigenvector from a previous matrix can dramatically reduce the resulting number of matrix multiplications. The method is more general than the problem we have described—it also allows for constraints of the form $p_i \geq l_i$. We can take

advantage of this functionality to allow for negative pressures (suction) away from the surface of the liquid.

After solving for the pressure, we update the velocity field,

$$\mathbf{u}^{n+1} = \mathbf{u}^{*} - v_f \frac{\nabla \mathbf{p}^n}{\rho_f}. \tag{3.8}$$

It is worth noting that this formulation is not unique. In particular, the volume fractions, v_f in Equation (3.8) could be moved to the right-hand side and/or the densities, ρ_f, could be included in the matrix, both of which may seem more natural choices. Just moving the volume fractions to the right-hand side leads to instability as the cell-centered volume fractions, needed to multiply $\rho_{\max}$ and ρ_c, may disagree somewhat with the face-centered volume fractions. Moving the densities into the matrix does not cause instabilities, but leads to fewer expansion steps and less lively motion. If the densities are moved to the matrix, moving the face fractions to the right-hand side has little effect.

The intuition behind the UIC solve is less straightforward than standard incompressible solves. In a traditional incompressible solve, the solver is prohibited from allowing divergence in any cell that has been labeled liquid. However, with unilateral incompressibility, the solver is free to relabel liquid cells as air by setting the pressure to zero and allowing negative divergence. The presence of zero pressures and negative divergence in the liquid prohibits the sort of long-distance pressure gradients that allow fluid to slosh back and forth. Instead, the fluid quickly settles and comes to rest. Intuitively, our formulation encourages the solver to consider cells labeled liquid as liquid by making them look full, as in an incompressible solve. However, this formulation is more sensitive to spatial oscillations in the density field that naturally occur from numerical errors in particle advection. These oscillations can lead to popping and even explosions, especially when timesteps are large relative to the grid spacing.

The fix, as detailed by Narain and colleagues [51], is a *density correction*—essentially an additional linear complementarity problem that operates on positions rather than velocities to instantaneously adjust the density to satisfy the

constraint that $\rho \leq \rho_{\max}$. The matrix in this solve sets Δt to 1 and removes the last term from $\mathbf{b}$ (i.e., setting the velocity to zero). After solving for pseudo-pressures, $\mathbf{y}$, in each cell, we compute the gradient on each face, $\mathbf{x} = \nabla \mathbf{y}$, and update the density of each cell using the divergence of $\mathbf{x}$,

$$\rho_c += \nabla \cdot \mathbf{x}. \tag{3.9}$$

We also apply averaged corrections to the face densities, ρ_f, and store the face-centered $\mathbf{x}$ for application as an instantaneous offset to particle positions during advection (immediately before time integration). Note that sign errors are especially easy to make.

3.2 Mass-full FLIP

Unilateral incompressibility and mass-full *fluid implicit particle* (FLIP) methods are very well-suited to each other. Unilateral incompressibility, like bilateral incompressibility, requires the solution of a global system every timestep. This solve is easily performed on mass-full FLIP's background grid. Moreover, we require density estimates at various locations on the grid. These are easily obtained by rasterizing particle mass onto the grid and dividing by volume. Furthermore, mass-full FLIP, like SPH, automatically conserves mass by conserving the number of particles—a very attractive feature. Finally, FLIP's advection scheme results in very low dissipation, perfect for our target of large-scale splashing liquids.

Given the popularity of FLIP in computer graphics, we only briefly describe the method (for details please see [76, 10]). The anatomy of a timestep in our system is as follows:

1. Rasterize particle velocities and masses onto the grid
2. Apply external forces (gravity)
3. Solve unilateral incompressibility and apply pressure gradient (see Section 3.1)

4. Extrapolate velocity field
5. Update particle velocities
6. Advect particles through velocity field

3.2.1 Particle Rasterization

In the first step, we rasterize particle velocities and masses onto the grid. Velocity and mass contributions from each particle are accumulated onto each face; just mass is contributed to cell centers. We use the standard trilinear weighting kernel. After all particles have contributed, velocities are normalized and masses are converted to densities by dividing by the volume of a cell. Specifically,

$$m_f = m_p \sum_p T(\mathbf{x}_p - \mathbf{x}_f), \tag{3.10}$$

$$u_f = \frac{m_p}{m_f} \sum_p u_p T(\mathbf{x}_p - \mathbf{x}_f), \tag{3.11}$$

$$\rho_f = \frac{m_f}{h^3}, \tag{3.12}$$

where m_f is the mass of a face, m_p is the mass of a particle, $T(\cdot)$ is the trilinear interpolation kernel, $\mathbf{x}_p$ is the particle position, $\mathbf{x}_f$ is the position of the center of a face, u_p and u_f are the u-components of the velocity of the particle and face, respectively (similar equations exist for the v- and w- components), ρ_f is the density at the face center, and h is the grid spacing. Note we do not need to explicitly store both m_f and ρ_f.

Because we divide by density in Equation (3.8), any cell that has mass less than a threshold is treated as air and not included in the unilateral incompressibility solve.

3.2.2 Velocity Extrapolation

We perform a simple velocity extrapolation algorithm that is similar in spirit to fast sweeping, but does not make use of levelset values. We first mark each

grid face that received any density. We then sweep over the grid several times. In each sweep, a face that has marked neighboring faces is assigned an average of the neighboring velocities and is itself marked. This approach is less accurate than building a levelset and applying fast sweeping, but is much faster.

3.2.3 Update Particle Velocity

As is commonly done, we use a combination of *particle in cell* (PIC) and FLIP to update the particle velocities. That is, we combine the trilinear interpolated velocity from the grid (PIC) with the change in grid velocity (FLIP).

3.2.4 Particle Advection

We use the second-order trapezoidal rule or first-order Euler integration for advection. In the later case, the extrapolation step may be skipped. Particle paths are clipped against domain boundaries and the levelset representation of obstacles. In the later case, we try stepping along the gradient of the levelset to place the particle on the obstacle surface. If this fails to converge (or places the particle outside the domain boundary), we perform several bisection steps along the particle's initial path to find a position with a positive (outside the obstacle), but small, levelset value.

3.3 Obstacle Handing

Our handling of obstacles is the most novel component of our approach. We sample the obstacles with particles, just as we do the liquid. We then rasterize obstacle particles onto the grid using the trilinear interpolation weights we also use for the liquid particles. In effect, we are blurring the boundaries of the obstacles in the same way that we blur the boundaries of the liquid. More specifically, to compute $\rho_{\max}$ for a cell, we approximate the integral

$$\rho_{\max} = \rho_f \frac{\iiint T(\mathbf{x} - \mathbf{x}_c) f(\mathbf{x}) dV}{\iiint T(\mathbf{x} - \mathbf{x}_c) dV} \tag{3.13}$$

where ρ_f is the default density of the fluid (1000 kg/m^3 for water), $T(\cdot)$ is the trilinear interpolation basis centered at the cell, $\mathbf{x}$ is a dummy variable for integration, $\mathbf{x}_c$ is the position of the cell center, and f is a *characteristic* or *indicator* function that is 0 inside the obstacle and 1 outside. The integral for the face volume fractions, v_f, which represent the maximum amount of liquid that can rasterize to a face, are computed similarly by leaving out the scaling by ρ_f. In practice, we uniformly sample the obstacles to evaluate the integral. This approach ensures that the volume computations agree with the rasterization stencil used by the fluid particles. If this were not the case, for example with the "box filter," a cell could be marked as containing zero fluid volume and yet, a liquid particle would be able to rasterize to it.

As our unilateral incompressibility solve does not allow negative pressures, there is no "suction" along obstacles and we automatically obtain wall-separating boundary conditions as cells next to obstacles "expand." That is, the pressure solve allows the fluid velocity field to point out of the obstacle, but not into it once the maximum density is reached. Additionally, as noted above, during advection, we ensure that particles are outside obstacle levelsets.

It is the novel combination of blurred obstacle boundaries, unilateral incompressibility, variational volume fractions and particle collision detection that allows our method to handle obstacles seamlessly—with no special effort we get wall-separation. These benefits do come at a cost, however. We sacrifice sharp boundary conditions and sharp interfaces. This sacrifice is justified in the context of large-scale single-phase liquid simulation, but would prove too great for multiphase flow or in cases where the boundary layer plays a key role.

3.4 Results and Discussion

We include two comparisons between our approach and an incompressible FLIP solver. Like the solver detailed above, our incompressible FLIP uses the second-order trapezoidal rule, which produces somewhat smoother results than the commonly used midpoint method, and our simple velocity extrapolation.

One difference is that while our particles will contribute mass to nearby cells due to the trilinear filter, our incompressible solver uses a "box filter" to determine which cells are labeled "liquid." In our experience, this diminished volume gain and led to better results from particle skinning. Our incompressible solver also uses traditional nonblurred obstacles.

In the first comparison, we set up two fountains (see Figure 3.1). Our approach allows the liquid to separate and expand in a natural manner, while the artificial surface tension of the incompressible solver causes the fountain to oscillate and collapse on itself.

The second comparison is a dam break with an obstacle (see Figure 3.2). In this example, our approach creates a large splash as the liquid passes over the obstacle, while the incompressible approach flows over the obstacle with almost no noticeable splash. This example also demonstrates incompressible FLIP's tendency toward volume gain as a single particle can force a cell to be labeled liquid. Our additional examples (see Figures 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11, and 3.12) demonstrate more large-scale, splashy fluid effects. Our simulations were performed with a variety of grid resolutions and domain scales, see Table 3.1 and Table 3.2 for details and timing results.

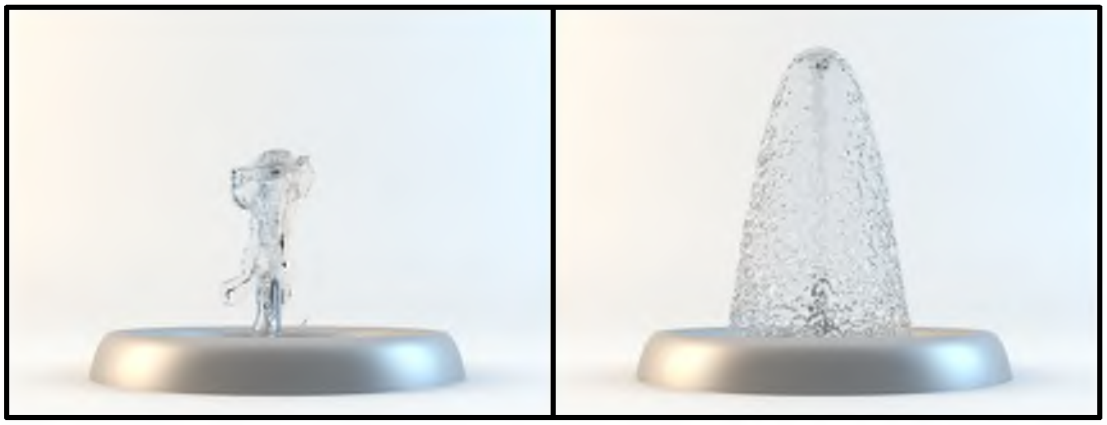

Figure 3.1. The artificial surface tension from forcing incompressibility leads to seeming small-scale behavior (left). Unilateral incompressibility allows the fluid to separate and leads to the impression of a larger-scale fountain (right).

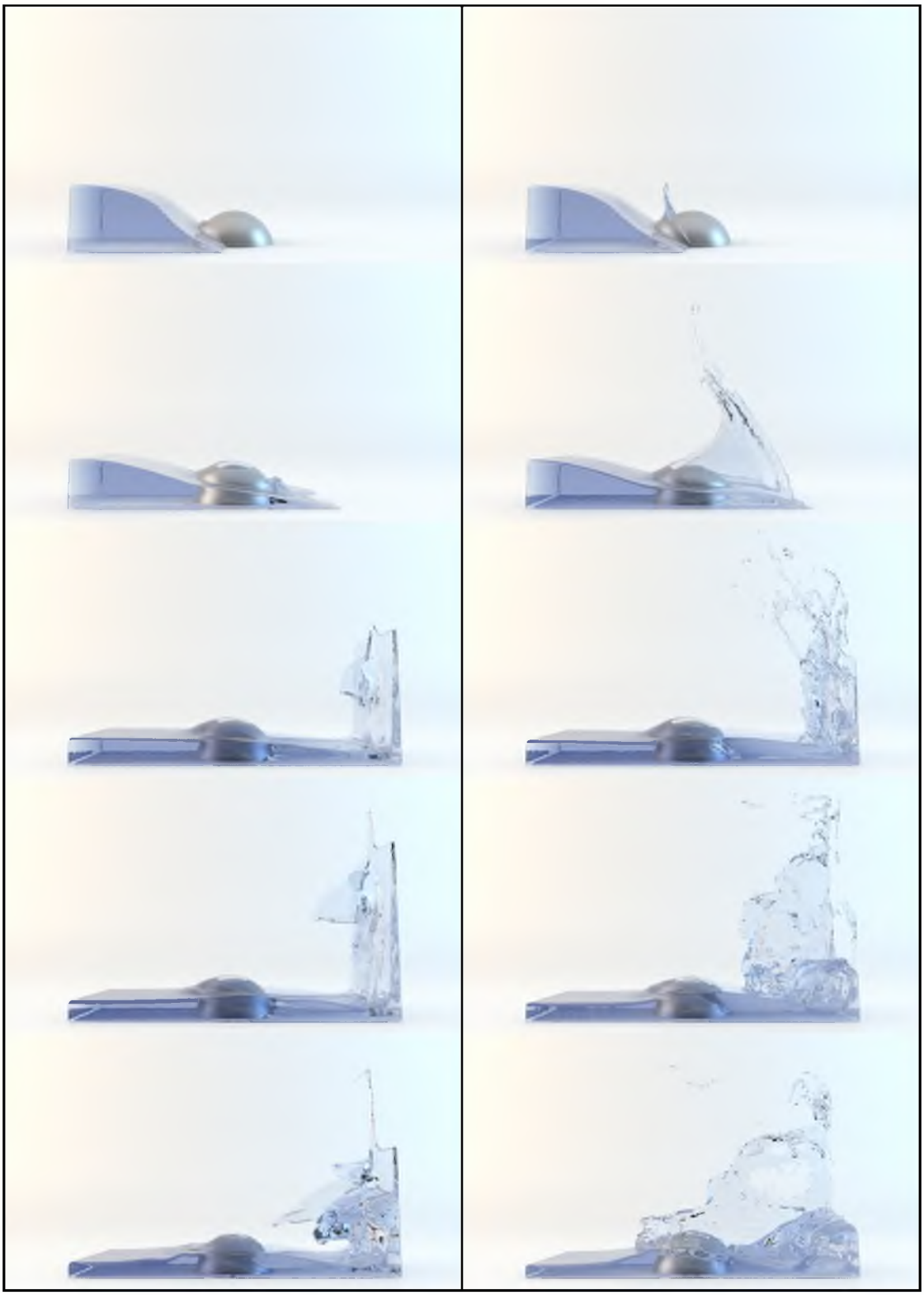

Figure 3.2. A dam breaking over an obstacle. Left Column: Incompressible FLIP. Right Column: Our Method.

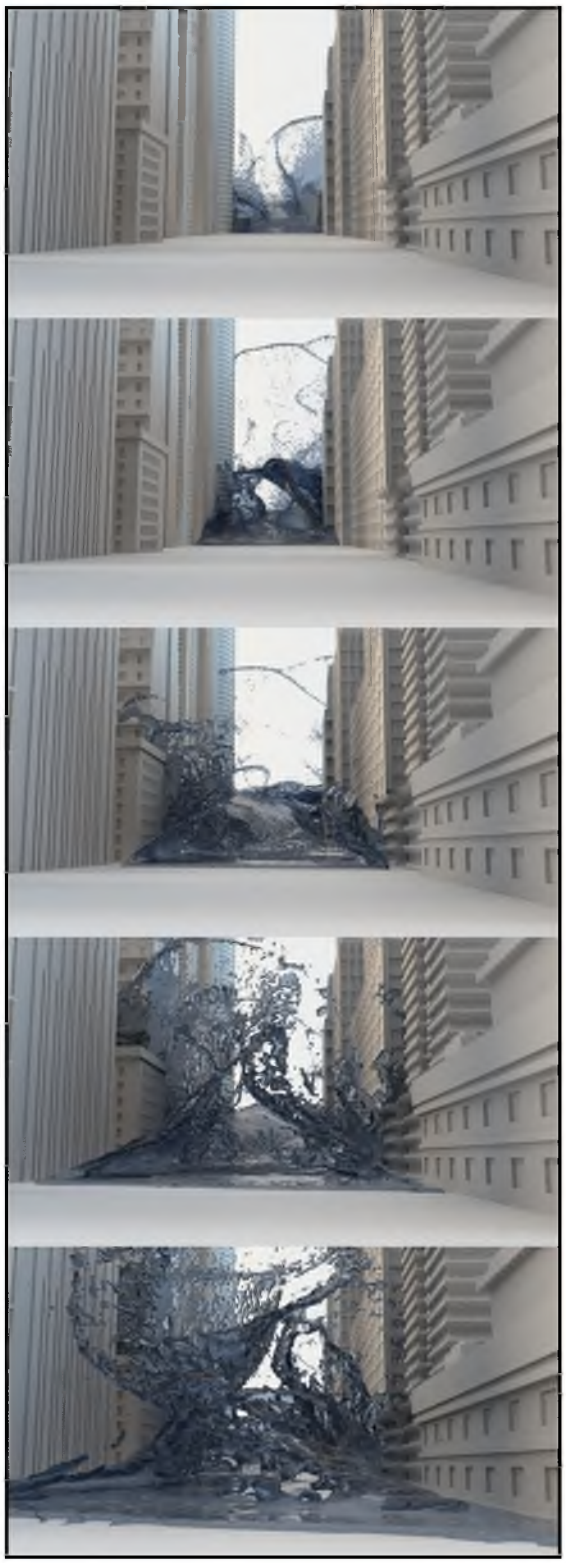

Figure 3.3. A sequence of images showing a city being flooded by a tidal wave.

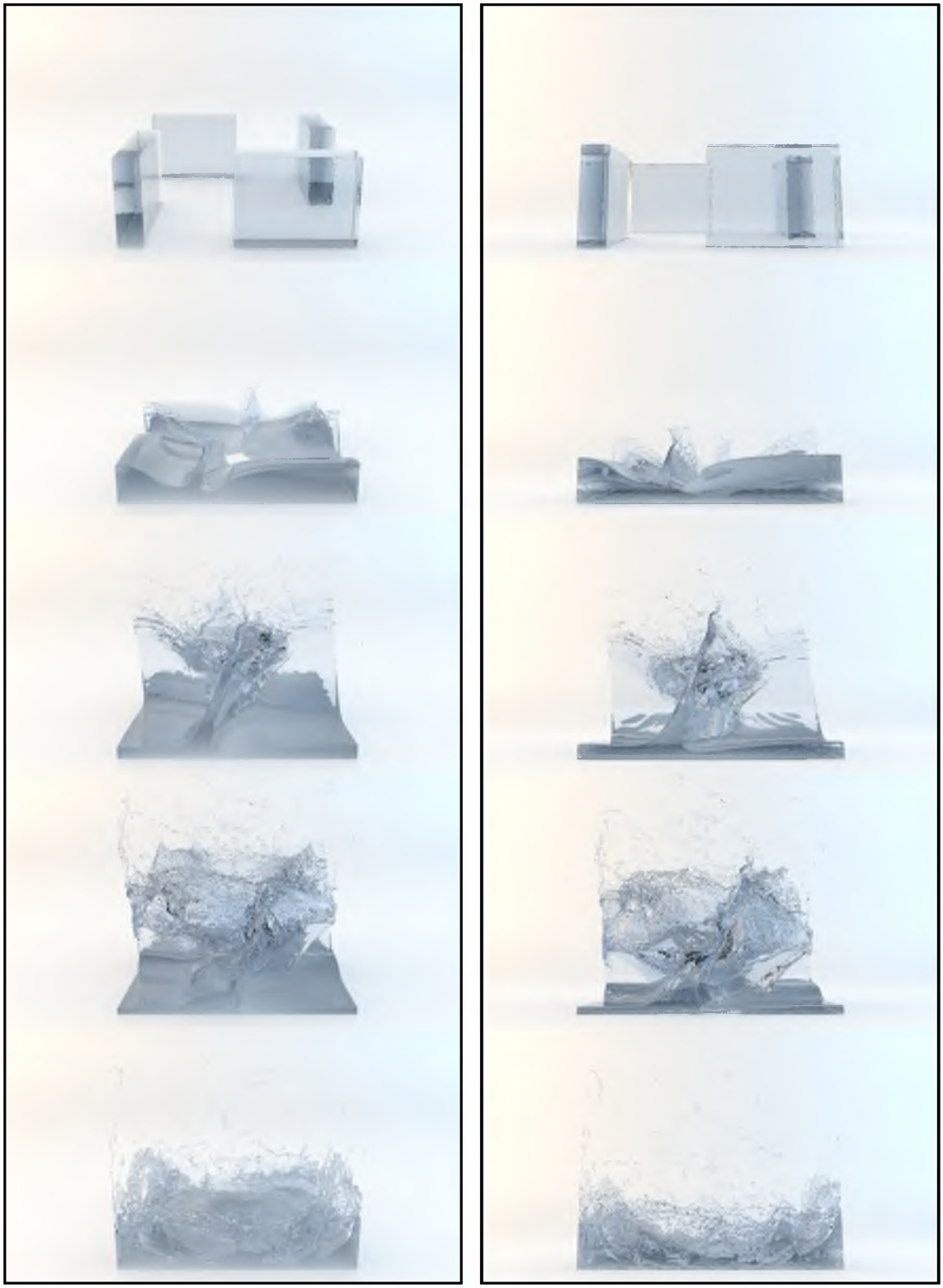

Figure 3.4. A quadruple dam break creates a large splash in the center of the scene. Left: Top view. Right: Side view.

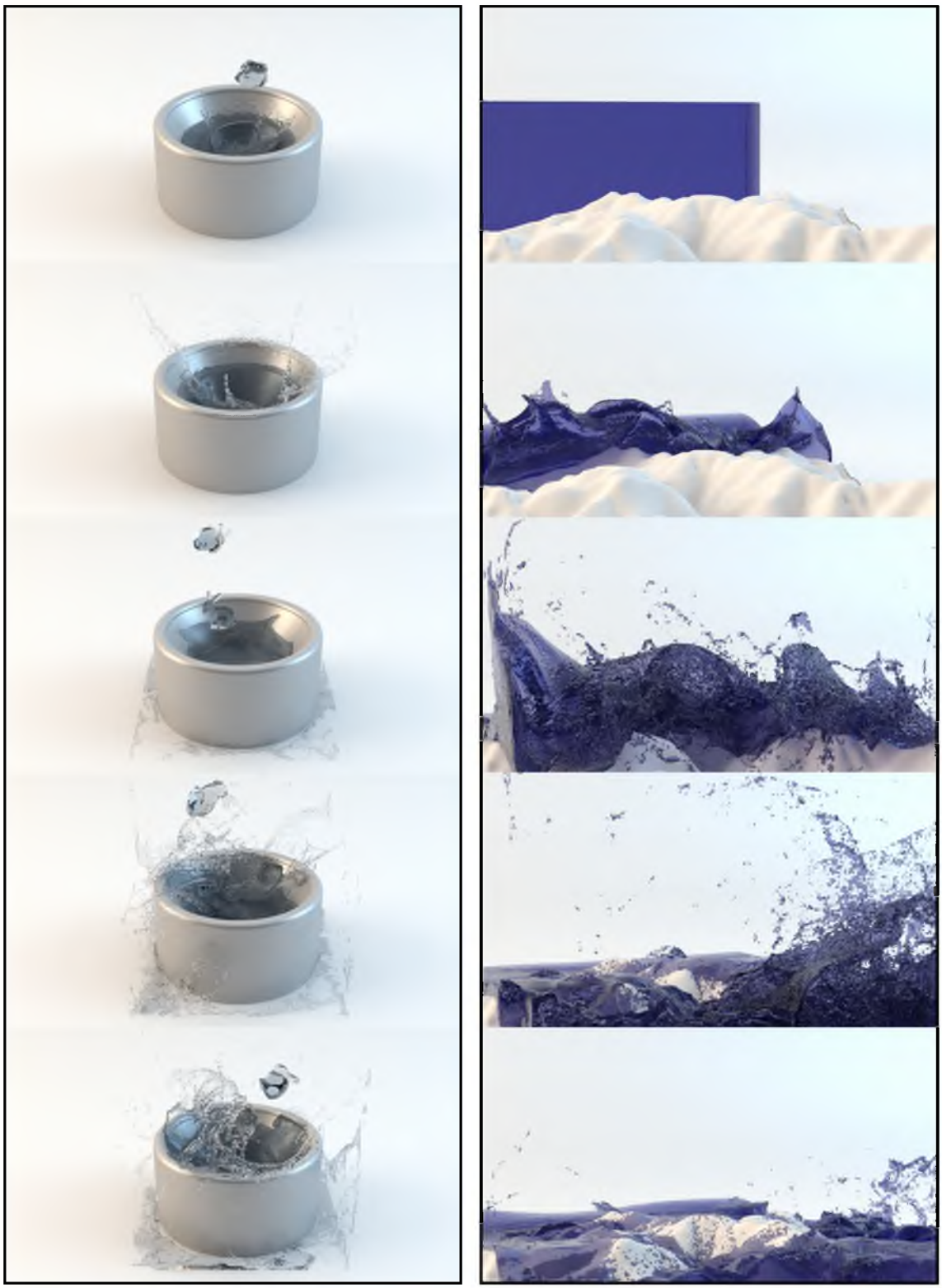

Figure 3.5. Left: Several liquid objects fall into a circular pool of water. Right: A large dam breaks over uneven terrain.

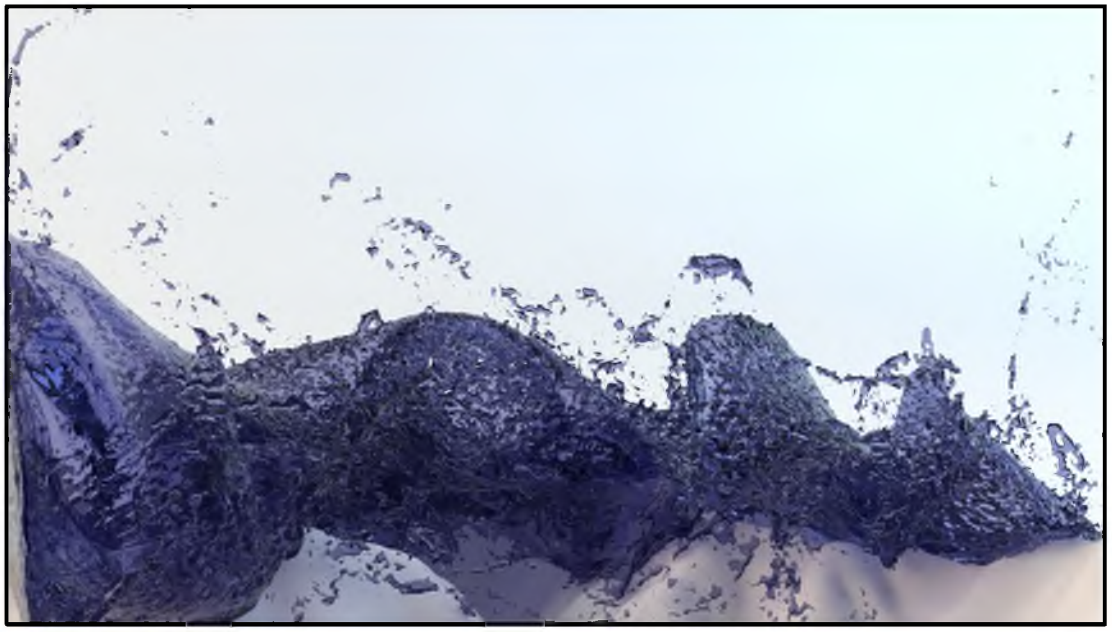

Figure 3.6. A closer view as a large dam breaks over uneven terrain.

Figure 3.7. Two streams of liquid collide.

Figure 3.8. Several liquid objects fall into a circular pool of water.

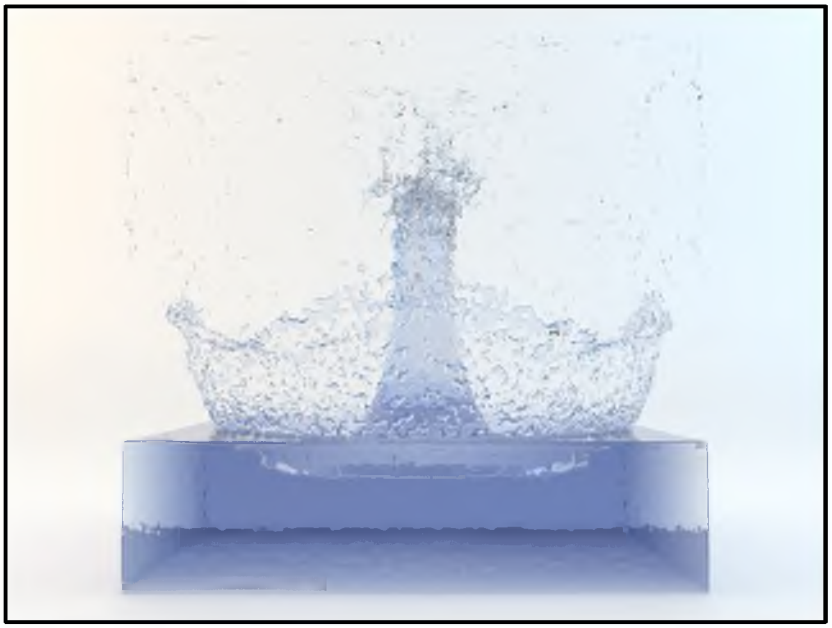

Figure 3.9. A series of underwater explosions cause large-scale splashing.

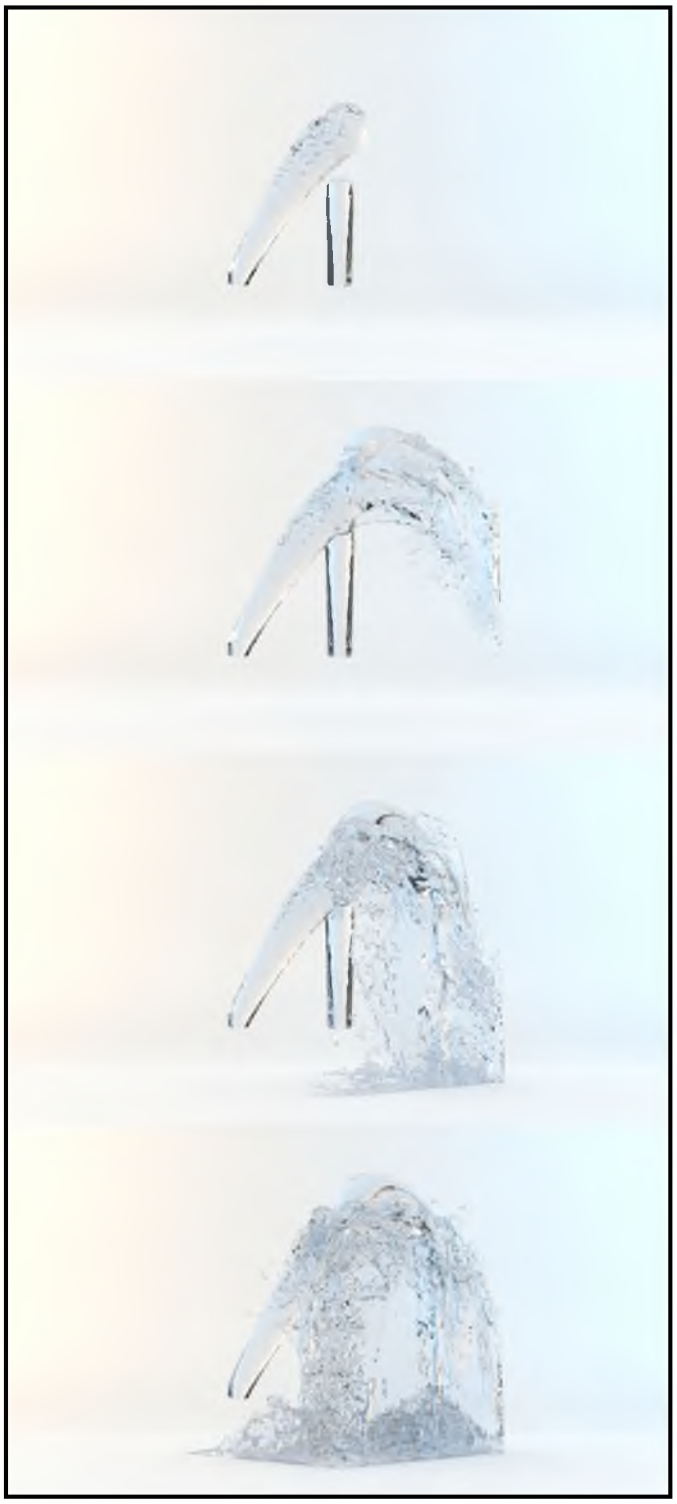

Figure 3.10. A sequence of images from an animation of two streams colliding.

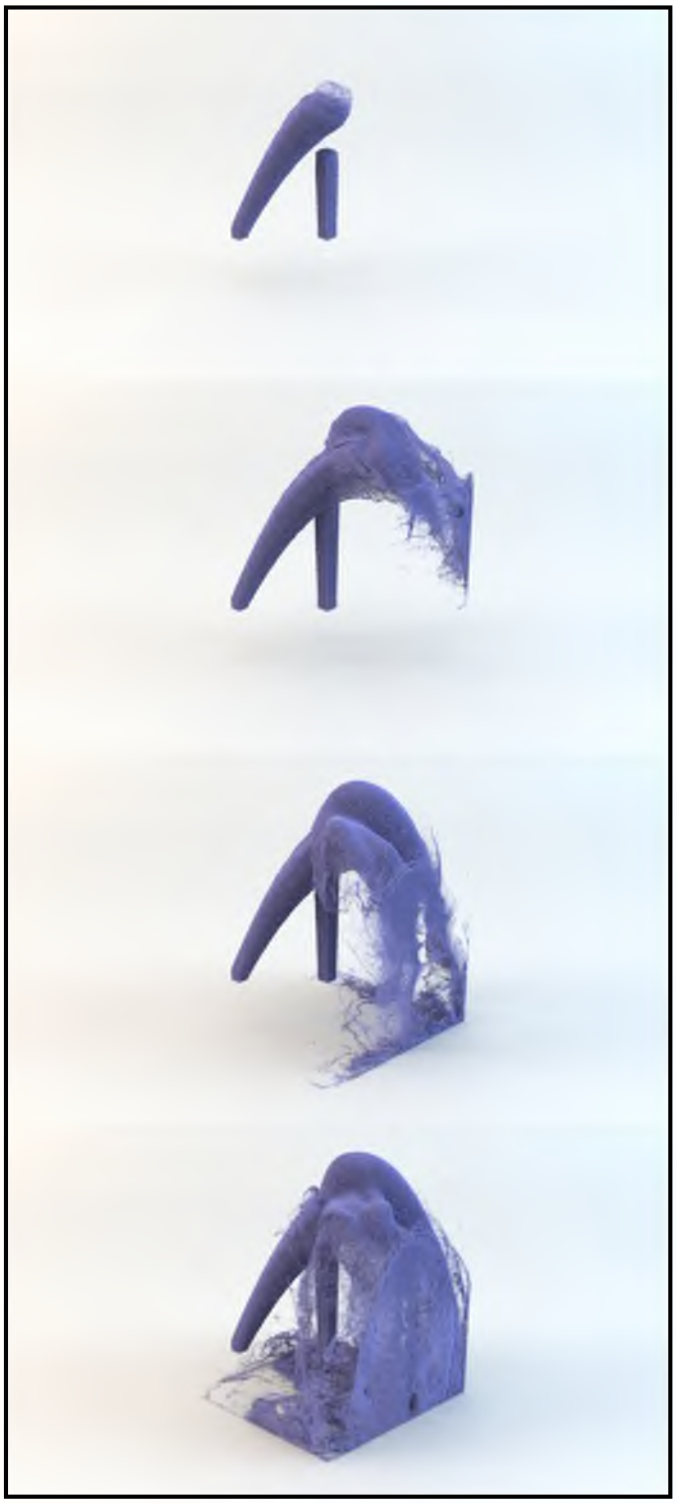

Figure 3.11. A sequence of images from an animation of two streams colliding visualized as particles.

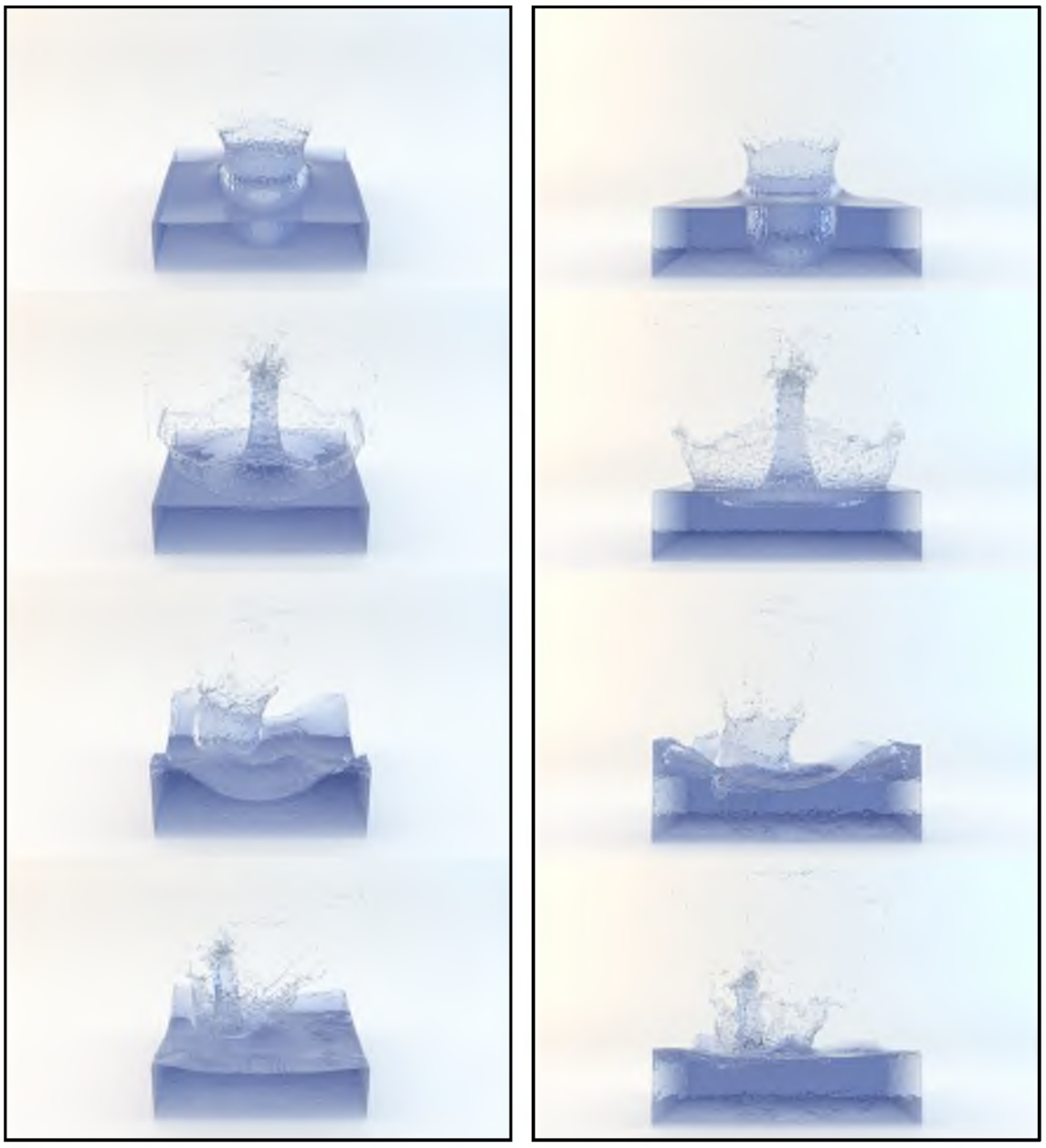

Figure 3.12. A sequence of images from an animation of several underwater explosions rendered from different viewpoints.

Table 3.1. Grid scale, resolutions, and particle counts for all examples in this chapter.

Figure	Grid spacing (h)	Domain size	Particle count
Figure 3.3	2	250x80x80	16,307,596
Figure 3.1 (UIC)	0.05	50x100x50	104,988
Figure 3.2 (UIC)	0.1	80x60x40	1,539,000
Figure 3.4	0.1	100x100x100	5,508,794
Figure 3.8	0.1	100x120x100	2,060,247
Figure 3.6	0.125	208x112x160	6,937,747
Figure 3.9	0.0625	80x80x80	3,430,944

Table 3.2. Timing results for all examples in this chapter. Timing results are given in average seconds for one 30 Hz frame.

Figure	Total time	Solve time	Extrapolate velocity	Particle advection	Particle rasterization
Figure 3.3	75.6	35.3	1.5	23.5	14.4
Figure 3.1 (UIC)	7.8	5.1	0.8	0.7	0.7
Figure 3.2 (UIC)	21.7	5.2	0.5	9	6.8
Figure 3.4	118.5	54.2	2.7	30.3	29.9
Figure 3.8	81.6	52.1	4.6	12.4	10.2
Figure 3.6	556	470	12.6	39.1	28.7
Figure 3.9	20	11.3	0.3	4.2	4

3.5 Limitations

A primary limitation of our approach is that there is no representation of the liquid surface, forcing us to rely on particle skinning approaches—generating surfaces from animated particle data, which can lead to artifacts, especially at low resolutions and with uniform or overly randomized samplings. Interestingly, while other researchers using FLIP have generally favored lower numbers of particles per cell, we found that increasing the number of particles per cell was a very effective strategy for achieving higher resolution animations without requiring the solution of larger LCPs. An alternative to particle skinning

would be to couple our particles to an explicit mesh surface as done by Yu and colleagues [74], though this would not allow for the frame-level parallelism of our particle skinning approach. Another disadvantage of our approach is the necessity for two LCP solves, both of which are more expensive than the simple Poisson solve in traditional incompressibility. We would also like to improve the computational efficiency of our approach. Many software design decisions were made favoring ease of debugging and experimentation over efficiency, so we believe there is much room for improvement.

One, perhaps subtle, disadvantage of assigning mass to the particles is that it makes reseeding very difficult. Placing too many particles in a cell will result in unwanted expansion, while too few will result in collapse. It took several iterations to produce the example in Figure 3.1, which required solving for the number of particles to add based on the fountain velocity and simulation timestep. Finally, we note that our approach essentially blurs both the liquid surface and obstacles. While this is acceptable for large-scale, splashy behavior, in many contexts, sharp boundaries are essential to compute the desired behavior [56].

CHAPTER 4

A POINT-BASED METHOD FOR ANIMATING ELASTOPLASTIC SOLIDS

We describe a point-based approach for animating elastoplastic materials. Our primary contribution is a simple method for computing the deformation gradient for each particle in the simulation. The deformation gradient is computed for each particle by finding the affine transformation that best approximates the motion of neighboring particles over a single timestep. These transformations are then composed to compute the total deformation gradient that describes the deformation around a particle over the course of the simulation. Given the deformation gradient, we can apply arbitrary constitutive models and compute the resulting elastic forces.

4.1 Computing the Deformation Gradient

In this section, we describe our method for computing the deformation gradient and the consequent elastic forces. We focus on the modifications we made to the open-source SPH simulator released by Adams *et al.* [1]. For additional details on SPH simulation, we refer the reader to that paper and its references.

Our goal is to compute elastic forces in a point-based simulation. In order to do so, we must first compute the deformation in the vicinity of each particle, p_i. We first consider how to compute the deformation around p_i over a single timestep. Let $\mathbf{x}_i$ be the position of p_i at the beginning of the timestep and $\mathbf{y}_i$ be the location of p_i at the end of the timestep. If p_j are the neighbors of p_i, we seek

the transformation matrix **F**, such that

$$\sum_{j=1}^{n} \left\| \mathbf{F}(\mathbf{x}_j - \mathbf{x}_i) - (\mathbf{y}_j - \mathbf{y}_i) \right\|^2 \tag{4.1}$$

is minimized. If we let

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_n \end{pmatrix} \tag{4.2}$$

and

$$\mathbf{Y} = \begin{pmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \mathbf{y}_3 & \cdots & \mathbf{y}_n \end{pmatrix}, \tag{4.3}$$

where the individual $\mathbf{x}_i$ and $\mathbf{y}_i$ are column vectors; then if the deformation can be represented with an affine transformation, we have

$$\mathbf{FX} = \mathbf{Y}. \tag{4.4}$$

Taking the transpose of both sides and multiplying both sides by **X** we obtain the normal equations,

$$\mathbf{X}\mathbf{X}^T\mathbf{F}^T = \mathbf{X}\mathbf{Y}^T. \tag{4.5}$$

Solving for **F**, we have

$$\mathbf{F} = \left(\left(\mathbf{X}\mathbf{X}^T \right)^{-1} \mathbf{X}\mathbf{Y}^T \right)^T. \tag{4.6}$$

This solution gives us the best linear transformation for the neighborhood around p_i in a least-squares sense. However, we want nearer particles to have greater influence, so we multiply the columns of **X** and **Y** by a weighting kernel. Our implementation uses the *poly6* kernel (the default smoothing kernel given by Müller *et al.* [48]),

$$W_{\text{poly6}}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} \left(h^2 - r^2\right)^3 & 0 \le r \le h \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

4.2 Composing Deformation Gradients

This approach gives us the deformation over a single timestep. However, this deformation is a linear transformation and transformations compose through

multiplication. Thus, to compute the deformation over some time interval, we break the interval into a series of k timesteps, estimate the deformation over each timestep, and compute

$$\mathbf{F} = \prod_{i=1}^{k} \mathbf{F}_i. \tag{4.8}$$

We note that only a single $\mathbf{F}$, representing the total elastic deformation, need be stored for each particle. The individual $\mathbf{F}_i$ are computed during the associated timestep, but not stored.

4.3 Constitutive Model

Once we have the deformation gradient, we can apply any constitutive model we like, compute strain, stress, and elastic forces, and move the simulator forward. In our implementation, we diagonalize $\mathbf{F}$ into $\mathbf{U}\hat{\mathbf{F}}\mathbf{V}^{\mathbf{T}}$ using a singular value decomposition [35] and apply the multiplicative plasticity model described by Bargteil *et al.* [3] to obtain the elastic deformation, $\hat{\mathbf{F}}_e$. We then compute the diagonalized stress as in Irving *et al.* [35],

$$\hat{\mathbf{P}} = 2\mu_e(\hat{\mathbf{F}}_e - \mathbf{I}) + \lambda \mathrm{Tr}\left(\hat{\mathbf{F}}_e - \mathbf{I}\right)\mathbf{I}. \tag{4.9}$$

Following Solenthaler *et al.* [64] and accounting for our diagonalized deformation gradient and stress, the elastic force p_i exerts on p_j is

$$\mathbf{f}_{ij} = -2v_i v_j \mathbf{U}\hat{\mathbf{F}}_e\hat{\mathbf{P}}\mathbf{V}^T\mathbf{d}_{ij} \tag{4.10}$$

where v_i and v_j are the volumes of particles p_i and p_j and

$$\mathbf{d}_{ij} = \nabla W(\mathbf{F}_e^{-1}(\mathbf{y}_j - \mathbf{y}_i), h). \tag{4.11}$$

Note that the vector from y_i to y_j is back projected to the reference space before applying the weighting kernel. We use the weighting kernel developed specifically for elastic forces by Solenthaler *et al.* [64],

$$W(\mathbf{r}, h) = \begin{cases} c\frac{2h}{\pi}\cos\left(\frac{(r+h)\pi}{2h}\right) + c\frac{2h}{\pi} & 0 \le r \le h \\ 0 & \text{otherwise,} \end{cases} \tag{4.12}$$

where

$$c = \frac{\pi}{8h^4\left(\frac{\pi}{3} - \frac{8}{\pi} + \frac{16}{\pi^2}\right)}. \tag{4.13}$$

In order to ensure conservation of momentum, $\mathbf{f}_{ij}$ and $\mathbf{f}_{ji}$ are averaged and equal and opposite forces are applied to the particles. We note that because we have diagonalized $\mathbf{F}$ and $\mathbf{P}$, the forces computed in Equation (4.10) are rotationally invariant.

4.4 Results and Discussion

Our implementation adds the elastic forces described in this chapter to the open source SPH simulator by Adams and colleagues [1]. The resulting system may be thought of as a "unified SPH" simulator and is capable of simulating liquids and solids as well as materials that demonstrate properties of both liquids and solids. In fact, many of our examples included SPH pressure forces as well as elastic forces, as we found that pressure forces provided additional stability. We refer the interested reader to the paper by Adams and colleagues [1] and the associated source code for details such as time integration (symplectic forward Euler), neighborhood selection (the 30 nearest neighbors within a given radius), etc.

Figures 4.1-4.7 demonstrate our method's ability to handle a wide range of materials. Figure 4.2 shows an example with a modified version of our plasticity model that divides the flow rate by the magnitude of the stress, so that the material flows more easily under small stresses. Figure 4.4 demonstrates a hyper-elastic material where the eigenvalues of $\mathbf{F}$ are squared before computing the stress. Figure 4.6 compares one of our simulations with real-world footage of bread dough and Figure 4.7 demonstrates the effects of varying our material parameters. Figure 4.8 shows a comparison of our approach with an approach that stores and compares to a reference shape and then removes plastic deformation before computing elastic forces and an additive approach that computes Green's strain at every timestep and adds it to the total elastic strain. As is expected, storing the

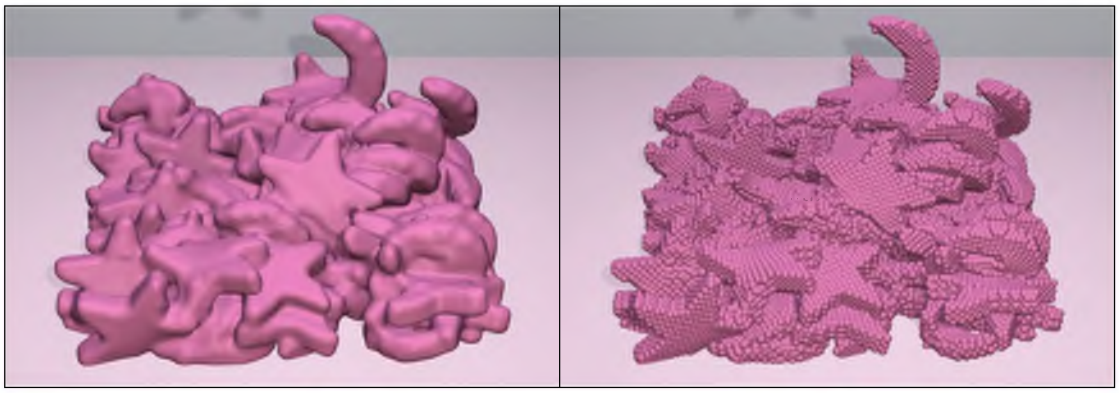

Figure 4.1. Two different shapes form a pile on the ground. The right image shows the simulation particles.

Figure 4.2. An armadillo demonstrates non-Newtonian behavior similar to a cornstarch solution—resisting large stresses, it initially bounces on the ground, but when the stress is reduced it flows readily.

reference configuration works very well for largely elastic bodies, but under large plastic flow, the simulation becomes unstable. Conversely, an additive model of elastic deformation works well enough when most of the deformation is plastic, but fails to return to the rest shape when the deformation is primarily elastic.

Table 4.1 summarizes our computation times. All results were obtained on a single core of a Xeon E5410 (2.33 Ghz), with 16 GB of memory available. Profiling has shown that in the example in Figure 4.3, 14% of the computation time was spent in our elasticity code. Half of this time was spent performing eigendecompositions. This total cost is roughly twice the cost of surface tension forces, which we did not use in our examples. We note that our examples were run with very conservative timesteps—some of our examples ran successfully with 10x larger timesteps.

Generating visually appealing, time-coherent surfaces for particle-based simulations remains a difficult problem that is beyond the scope of this disserta-

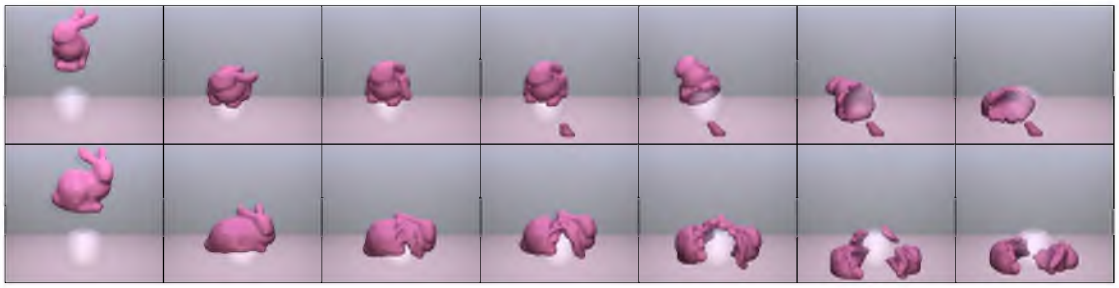

Figure 4.3. An elastoplastic bunny falls on a sphere.

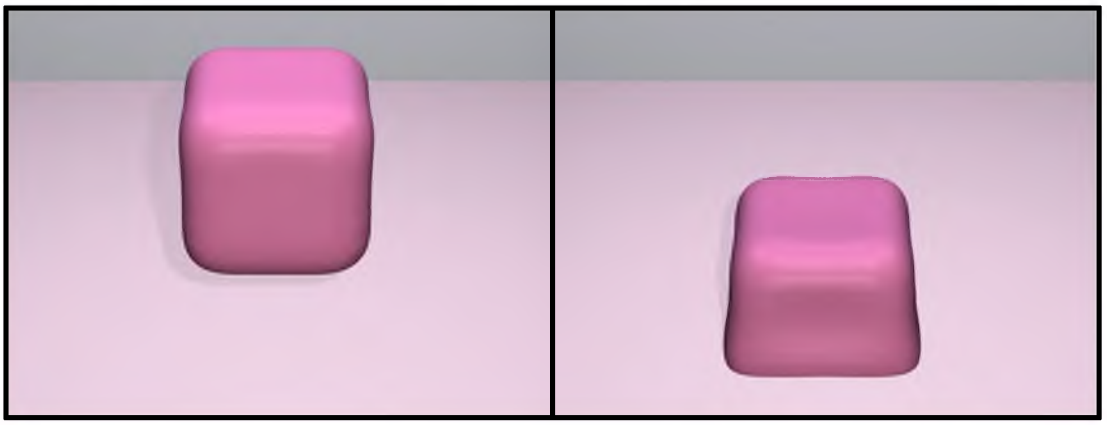

Figure 4.4. Hyperelastic boxes dropped on the ground. The left cube is quite stiff, the right cube is softer.

Figure 4.5. Three cylinders with different material properties fall on the ground.

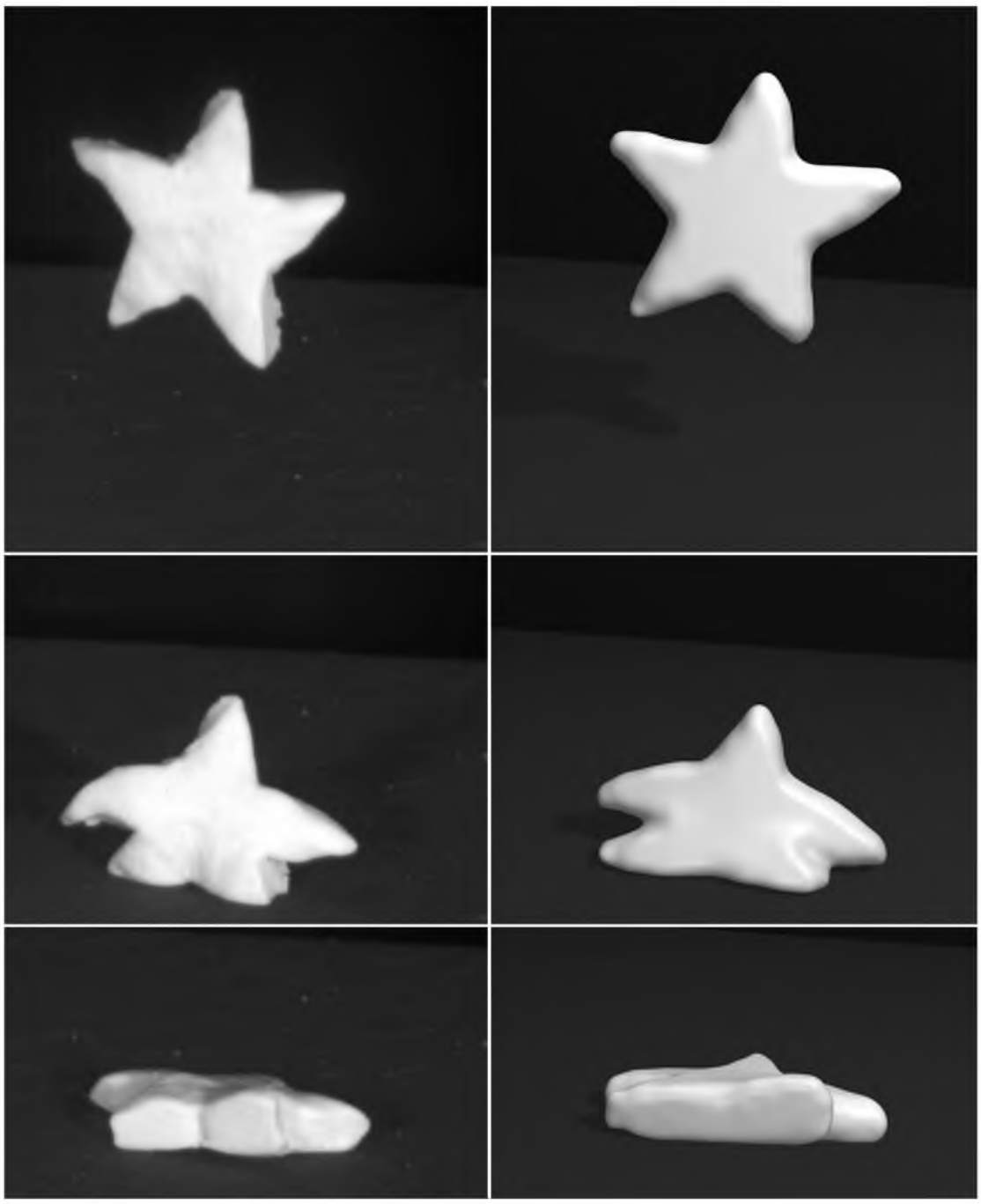

Figure 4.6. Real-world footage of bread dough shaped like a star (left) is compared to a simulation (right).

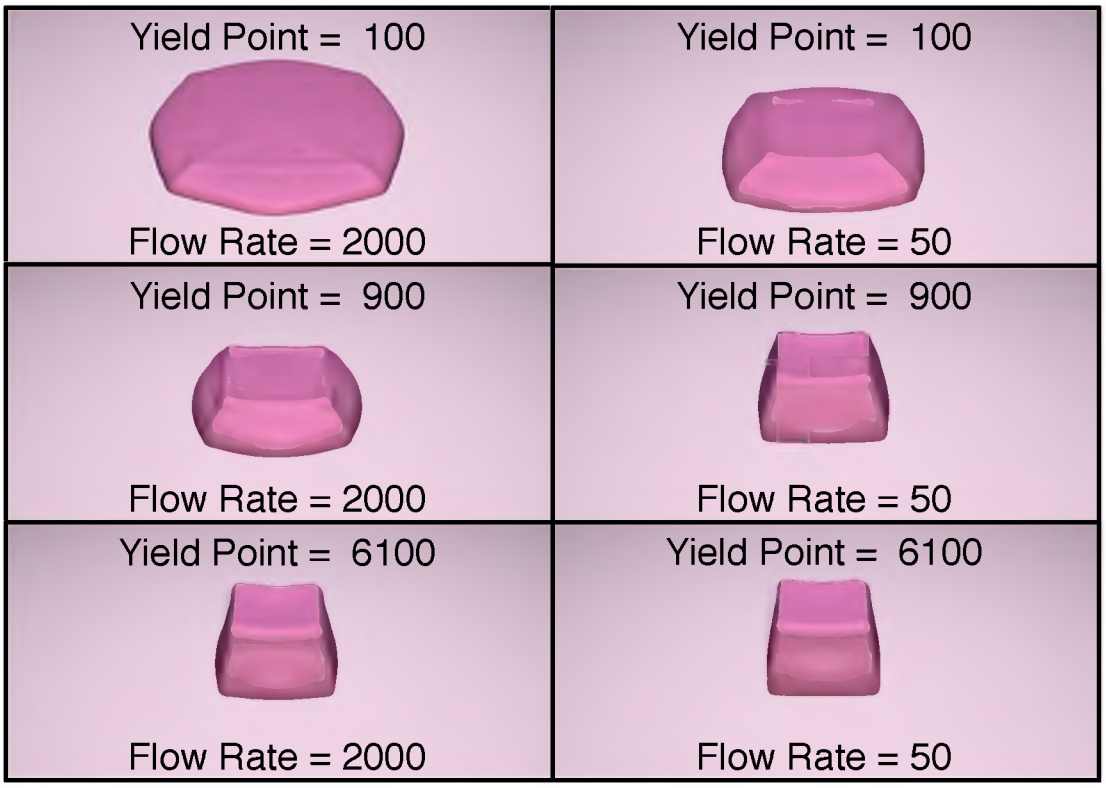

Figure 4.7. We demonstrate the effects of our plastic material parameters by dropping a box on the ground.

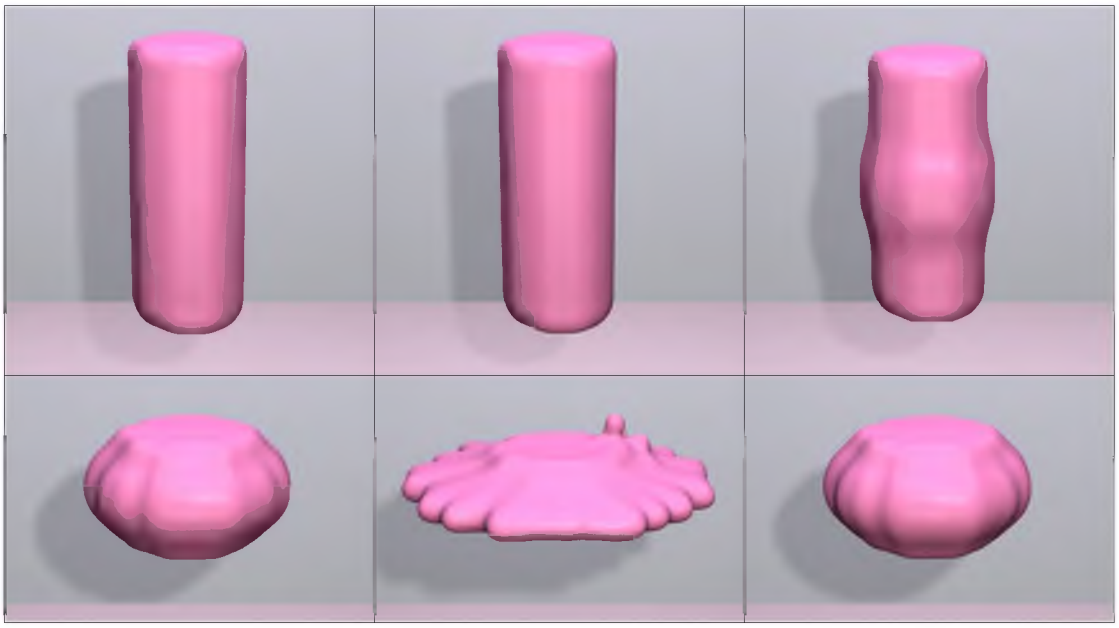

Figure 4.8. Final frames in a comparison of our method (left) against a method that uses a rest configuration (middle) and a method with an additive strain model (right). The top row is an elastic material, the bottom row is a very plastic material. The simulation consists of applying and then releasing an analytic compression force that increases away from the center of the object. The lower middle image is the last frame before the simulation became unstable.

Table 4.1. Timing results for the examples in this chapter.

Figure	Δt (ms)	Particles	Sec/Frame
Fig. 4.2	0.5	52316	96.4888
Fig. 4.3	0.1	40556	379.065
Fig. 4.5 (left)	0.1	16770	138.257
Fig. 4.5 (center)	0.1	16770	137.909
Fig. 4.5 (right)	0.1	16770	141.591
Fig. 4.4 (left)	0.1	665	4.76452
Fig. 4.4 (right)	0.1	665	4.88767
Fig. 4.7 (top, left)	0.1	12152	102.545
Fig. 4.7 (center, left)	0.1	12152	86.9948
Fig. 4.7 (bottom, left)	0.1	12152	94.4097
Fig. 4.7 (top, right)	0.1	12152	96.135
Fig. 4.7 (center, right)	0.1	12152	87.1509
Fig. 4.7 (bottom, right)	0.1	12152	94.3632

tion. Our results were generated with a variation of the method described by Williams [72].

One of the paramount concerns in any computer graphics simulator is stability and ours is no exception. One of the sources of error and instability in our approach is the estimation of the deformation gradient. In particular, if a particle does not have enough neighbors or the distribution of particles is degenerate, $\mathbf{X}\mathbf{X}^T$ will be ill-conditioned. To address this problem, we do not update the deformation gradient if a particle has less than a set number of neighbors (6 in our implementation), if $\mathbf{X}\mathbf{X}^T$ is ill-conditioned, or if the update would cause any of the eigenvalues of $\mathbf{F}$ to be less than or equal to zero. We also note that plastic flow tends to improve stability by bringing $\mathbf{F}$ towards the identity. Consequently, relaxing the constraint that plastic deformation be volume preserving in cases when $\mathbf{F}$ encodes large volume changes further improves stability. When the method does fail, it tends to be in areas around sharp features, where a particle's neighbors subtend a small solid angle, or in areas where topological changes are occurring. Addressing these issues is an important area of future work.

CHAPTER 5

ENHANCEMENTS TO MODEL-REDUCED FLUID SIMULATION

We present several enhancements to model-reduced fluid simulation that allow improved simulation bases and two-way solid-fluid coupling. Specifically, we present a basis enrichment scheme that allows us to combine data-driven or artistically derived bases with more general analytic bases derived from Laplacian Eigenfunctions. We handle two-way solid-fluid coupling in a time-splitting fashion; we alternately timestep the fluid and rigid body simulators, while taking into account the effects of the fluid on the rigid bodies and vice versa. We employ the vortex panel method to handle solid-fluid coupling and use dynamic pressure to compute the effect of the fluid on rigid bodies.

5.1 Reduced Fluid Simulation

In this section, we will first briefly review the mechanics of reduced fluid simulation, then in following sections, we introduce our basis enrichment scheme, and finally present our approach for two-way solid-fluid coupling.

The basic mechanics for reduced fluid simulation were introduced by Treuille and colleagues [69]. We begin with the incompressible Navier-Stokes equations which describe the motion of a viscous fluid,

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \nabla^2 \mathbf{u} + \nabla p + \mathbf{f}_e \tag{5.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{5.2}$$

where $\mathbf{u}$ is the velocity, ν is the viscosity parameter, p is the pressure, and $\mathbf{f}_e$ are the external forces. The goal of reduced simulation is to reduce the dimensionality of $\mathbf{u}$ through Galerkin projection onto a low-dimensional basis,

$$\tilde{\mathbf{r}} = \mathbf{B}^T\mathbf{u} \tag{5.3}$$

where, $\tilde{\mathbf{r}} \in \mathbb{R}^r$ represents the reduced coefficients and $\mathbf{B}$ is the basis represented as a matrix with r columns, each representing a basis function.

A typical fluid simulation in computer graphics employs *operator splitting* breaking the simulation into several individual steps: advection, applying external forces, applying viscosity, and projection onto a divergence-free field. To perform reduced fluid simulations, we must address each of these steps.

Fortunately, because we only include divergence-free fields in our basis, we can only represent divergence-free fields, removing the need for the expensive projection step. External forces are easily handled by Galerkin projection onto the basis. Specifically, given external forces, $\mathbf{f}_e$, we compute reduced forces

$$\tilde{\mathbf{f}}_e = \mathbf{B}^T\mathbf{f}_e. \tag{5.4}$$

These are simply scaled and added to the reduced velocity coefficients,

$$\tilde{\mathbf{r}} := \tilde{\mathbf{r}} + s\tilde{\mathbf{f}}_e, \tag{5.5}$$

for some scaling factor s that accounts for density, grid-spacing, and timestep.

The diffusion term is also easily handled. Being a linear operator, the discretization of the diffusion operator $\nabla^2\mathbf{u}$ can be represented as a matrix $\mathbf{D}$. Projecting into the subspace, we get the reduced diffusion matrix

$$\tilde{\mathbf{D}} = \mathbf{B}^T\mathbf{D}\mathbf{B}, \tag{5.6}$$

which is precomputed for a given domain.

The nonlinear advection operator, $-(\mathbf{u}\cdot\nabla)\mathbf{u}$, is more complicated. The nonlinearities preclude it from being written as a single reduced matrix. Instead, a reduced advection matrix for each basis function can be precomputed and then

at runtime combined into the final reduced advection operator. The discretization of the advection operator for a given velocity field, $\mathbf{u}$, can be expressed as a matrix, $\mathbf{A_u}$. This matrix, when applied to a field, $\mathbf{v}$, (i.e., $\mathbf{A_u v}$) has the effect of advecting $\mathbf{v}$ through $\mathbf{u}$.

Thus, we precompute, for each basis function or *mode*, $\mathbf{b}_i$, in the basis $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_r]$ a matrix, $\mathbf{A_{b}}_i$, that represents advection through the velocity field $\mathbf{b}_i$. Each of these matrices can be reduced

$$\tilde{\mathbf{A}}_{\mathbf{b}_i} = \mathbf{B}^T \mathbf{A}_{\mathbf{b}_i} \mathbf{B}, \tag{5.7}$$

during precomputation. During simulation, the reduced advection matrix is computed by summing all mode advection matrices weighted by their corresponding reduced state coefficient

$$\tilde{\mathbf{A}} = \sum_i \tilde{\mathbf{A}}_{\mathbf{b}_i} r_i. \tag{5.8}$$

Viscosity and advection can be combined into a single update from time t to $t + \Delta t$ and can be written as:

$$\tilde{\mathbf{r}}^{t+\Delta t} = \left(e^{\Delta t(\nu \tilde{\mathbf{D}} + \tilde{\mathbf{A}})}\right) \tilde{\mathbf{r}}^t. \tag{5.9}$$

This matrix-vector product is computed efficiently using an iterative Taylor approximation [71].

We note that while the reduced simulation can proceed without the notion of a *grid*, for collecting training data and visualization purposes, a grid is useful. In our system, we explicitly use the grid for solid-fluid coupling.

5.2 Basis Enrichment

The divergence-free bases used in reduced fluid simulations have been constructed in either of two ways. The first method involves running a training simulation and then extracting a reduced basis using a Singular Value Decomposition (SVD). This process is accomplished by concatenating velocity-field snapshots of a high-resolution fluid simulation into a matrix, computing the

SVD, and then selecting r singular vectors [69]. A basis generated in this way can capture motion similar to the training data very well in the least squares sense; however, it suffers from a number of problems. Arbitrary motion during runtime can be problematic as the basis may not generalize well to motion outside of the training simulation, e.g., using a training simulation where an obstacle generates flow in one half of the domain for a runtime where the obstacle moves to the other half. To minimize problems from over fitting, a significant amount of simulation data has to be precomputed. Additionally, it can be difficult for artists to know what kinds of training simulations to run in order to generate a suitable basis, not to mention the large amount of precomputation space and time needed.

The second method involves creating a basis using an analytic approach, for example choosing Eigenfunctions of the Laplacian operator. For a few simple domains, these bases can be computed in closed form. In more general domains, the Eigenfunctions of the discrete Laplacian operator are computed using an Eigendecomposition [18]. In simple domains like a box, the advection operators can be computed analytically and because the modes are only loosely coupled, the resulting matrices are sparse. The Eigenfunction modes work well for gross flow and do not suffer from over-fitting, but detailed flow can require an impractically large number of modes.

To give artists control over generating a basis, we provide a velocity drawing tool. After the velocity has been drawn, it is projected onto a divergence-free field and the artist can timestep the simulation to generate the desired velocity field. This process allows an artist to create different flow effects, such as vortices or laminar flow paths, with minimal training data. Alternatively, artists can simply interact with the simulation to generate training data. We will now describe how to combine different bases; a similar approach has been used in the context of reduced bases for direct to indirect radiance transfer [41].

To exploit any sparsity that might exist in the Laplacian Eigenfunctions, we would like to keep this basis intact when including the data-driven, artist-generated modes. Thus, given a Laplacian Eigenfunction basis, $\mathbf{E}$, and velocity

fields generated by an artist, $\mathbf{D} = [\mathbf{d}_1 \dots \mathbf{d}_N]$, where each column is a user-generated velocity field scaled to unit length, we would like to construct a combined basis that keeps the structure of $\mathbf{E}$ intact. First, the SVD of $\mathbf{D} = \mathbf{USV}^T$ is computed and the left singular vectors, $\mathbf{U}$, with corresponding singular values greater than zero are retained. $\mathbf{U}$ is then deflated against the basis,

$$\mathbf{U}_d = \mathbf{U} - \mathbf{EE}^T\mathbf{U}, \tag{5.10}$$

where the columns of $\mathbf{U}_d$ now contain the parts of the velocity fields, $\mathbf{U}$, that could not be represented by the basis, $\mathbf{E}$. The columns of matrix $\mathbf{U}_d$ are now orthogonal to the columns of $\mathbf{E}$ but may no longer be orthogonal to each other, i.e., $\mathbf{U}_d^T\mathbf{U}_d$ may not be the identity. To generate a basis that spans the same subspace, we simply compute the SVD of $\mathbf{U}_d$ and retain the singular vectors corresponding to non-zero singular values,[1] resulting in an orthonormal basis $\mathbf{R}$. Concatenation of $\mathbf{E}$ and $\mathbf{R}$ forms an orthogonal basis, perfectly valid for reduced fluid simulation. From now on, we therefore assume that $\mathbf{B}$ is the concatenated matrix $[\mathbf{E}|\mathbf{R}]$.

We would also like the ability to specifically activate the artist-generated modes during runtime. If one wishes to directly excite an artist-created mode during runtime, the projection of those modes into $\mathbf{B}$ can be precomputed. At runtime the resulting coefficients can be added to the reduced state. No projection is necessary during runtime.

5.3 Two-way Solid-fluid Coupling

We use the reduced fluid simulation engine described in Section 5.1 and Box2D [8] for rigid body simulation. To couple them, we use a time splitting technique and alternately timestep each simulator while taking into account the effects of the fluid on the rigid bodies and vice versa.

---

[1] While $\mathbf{U}$ is full rank, if there is a large overlap between $\mathbf{U}$ and $\mathbf{E}$, deflation will result in a rank deficient matrix $\mathbf{U}_d$ (with zero singular values).

5.3.1 Solid-to-Fluid Coupling

To account for the effect of rigid bodies on the fluid flow, we adopt a vortex panel method [17, 53]. This approach has two advantages over previous work. First, obstacles are not limited to a finite range of spatial influence. In fact, they have global influence, though the fall-off is quite fast. Second, we avoid the substantial precomputation of sampling the object's effect at various positions and orientations in the domain. Our only precomputation involves inverting matrices. Finally, we note that our approach generalizes beyond reduced fluid simulation and could be used in other contexts, such as smoothed particle hydrodynamics, Eulerian, or semi-Lagrangian methods.

In two dimensions, objects are discretized into M piecewise linear segments called panels. In our system, the panel lengths are chosen to be on the order of the fluid simulation's grid spacing. The panels are then used both as quadrature points and as vorticity sources that cancel flow normal to the obstacle.

The velocity, $\mathbf{u} = (u, v)$, generated by a panel at a point $\mathbf{x}$ in the local coordinate system of the panel, is given by

$$u = \frac{\gamma\beta}{2\pi}, \quad v = \frac{\gamma}{2\pi} ln \frac{d_o + \epsilon}{d_e + \epsilon}, \tag{5.11}$$

where γ is the *panel strength*, β is the angle subtended by the panel from the point $\mathbf{x}$, d_o, and d_e are the distances from $\mathbf{x}$ to the origin and end of the panel, respectively, and ϵ is a small constant to avoid division by zero (see Figure 5.1).

To cancel the flow normal to an object, we must consider the interactions between all the panels of the object. To do so, we compute a coupling matrix $\mathbf{P} \in \mathbb{R}^{M \times M}$ that encodes the influence of the strength of panel i on the velocity at panel j. Specifically, let $\bar{\mathbf{u}}_{ij}$ be the velocity induced at the midpoint of panel j by panel i when panel i has unit strength (i.e., $\gamma_i = 1$). Then the P_{ji} is given by

$$P_{ji} = -\bar{\mathbf{u}}_{ij} \cdot \mathbf{n}_j, \tag{5.12}$$

where $\mathbf{n}_j$ is the normal vector of the j-th panel.

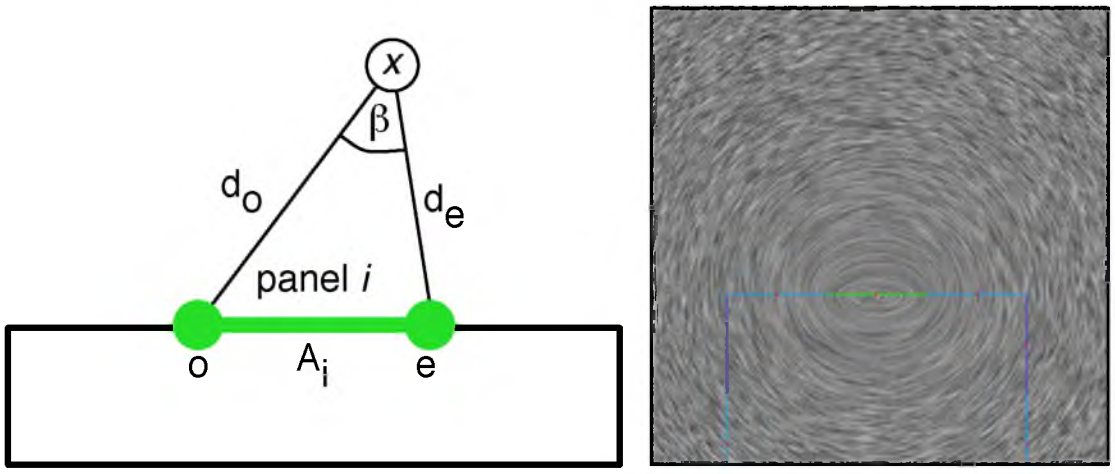

Figure 5.1. A vortex panel. Left: Panel coordinate system. Right: Velocity field induced by the panel.

Given **P** and a velocity field, **u**, to cancel the flow normal to the obstacle, we must solve the linear system,

$$\mathbf{P}\gamma = \mathbf{b} \tag{5.13}$$

where γ is the panel strength vector, and **b** is a vector encoding the violation of the boundary condition. Specifically,

$$b_i = A_i\left(\mathbf{u}_f - \mathbf{u}_o\right)\cdot \mathbf{n}_i \tag{5.14}$$

where b_i is the violation at panel i, A_i is the panel area, $\mathbf{u}_f$ is the fluid velocity evaluated at the midpoint of the panel, and $\mathbf{u}_o$ is the velocity of the object. This approach corresponds to a 1-point quadrature rule. Of course, higher order methods could be used.

As described, the $M \times M$ panel coupling matrix **P** is singular and an additional constraint must be added in order to obtain a unique solution. We add the constraint that there is zero circulation around the boundary, i.e.,

$$\sum_i^M A_i\gamma_i = 0. \tag{5.15}$$

This constraint is encoded by adding a row to the panel matrix containing the panel lengths and a zero to the end of **b**. The panel matrix is computed in object

space, allowing for rigid body transformations without modification. $\mathbf{P}$ can be inverted during precomputation; at runtime, panel strengths are computed with a single matrix-vector product.

Some distributions of panels are problematic when objects contain symmetries. For example, a square with two panels per side is unable to cancel the normal velocities induced from rigid body rotation. In such cases, it suffices to use an odd number of panels per side.

5.3.2 Multiple Bodies

Thus far, we have described how to handle a single object. To handle multiple objects, we must account for their interaction. Ideally, we would compute a single coupling matrix encoding the interactions of all panels in the system. However, this would require solving a new and much larger linear system every step, removing the ability to precompute an inverse [12]. Instead, we employ a fixed point iteration approach that takes advantage of the precomputed inverse panel matrices. First, the panel strengths of each object are computed to satisfy the boundary conditions of the reduced velocity field, i.e., for all objects i we compute

$$\gamma_i = \mathbf{P}_i^{-1}\mathbf{b}_i. \tag{5.16}$$

We then iteratively solve for panel strengths that additionally satisfy object-object interactions.

Each iteration, for each object i in our simulation:

1. Compute $\mathbf{b}_i^{obj}$, which is the boundary violation induced by all other objects.
2. Store the previously computed panel strengths.
3. Solve for the new panel strengths,

$$\gamma_i = \mathbf{P}_i^{-1}(\mathbf{b}_i^{orig} + \mathbf{b}_i^{obj}). \tag{5.17}$$

4. Compute the norm of the difference in panel strengths.

Iterations are performed until the panel strengths converge, or a user-specified tolerance or iteration limit is reached. This scheme, which falls into the class of Schwarz alternating methods [68], is guaranteed to converge to a unique solution for second-order PDE's. Golas *et al.* [27] successfully demonstrate an alternating method to couple Eulerian grids with vortex particle methods.

This alternating scheme may fail due to the singularities that occur when evaluating the velocity very near a panel. Velocities evaluated too close to a panel should not be relied upon and instead another approach should be taken, such as interpolating from reliable positions [31].

5.3.3 Domain Boundaries

When an object approaches the domain boundary, the velocity field induced by its vortex panels will not generally respect the solid wall boundary conditions (see Figure 5.2). For simple domains with closed form Laplacian Eigenfunctions, we employ the *method of images*—used in electrostatics to handle wall boundary conditions—to accurately and efficiently enforce the wall boundary conditions. To do so, objects that violate the solid wall boundary conditions above an error threshold are reflected across the solid wall. The resulting combined velocity field will only have tangential components along the solid wall. The velocities induced from the reflected panels are evaluated only at positions that fall inside the domain. This approach will correctly satisfy the domain wall boundary conditions by canceling the normal components of the velocity induced by the original object. This method is similar in spirit to Long *et al.* [40] who used the reflection properties of the discrete sine/cosine transform to handle solid wall domain boundaries.

5.3.4 Feedback

The resulting velocity field is a combination of the reduced fluid velocity, $\mathbf{u}_r$, and a panel velocity field, $\mathbf{u}_p$, where

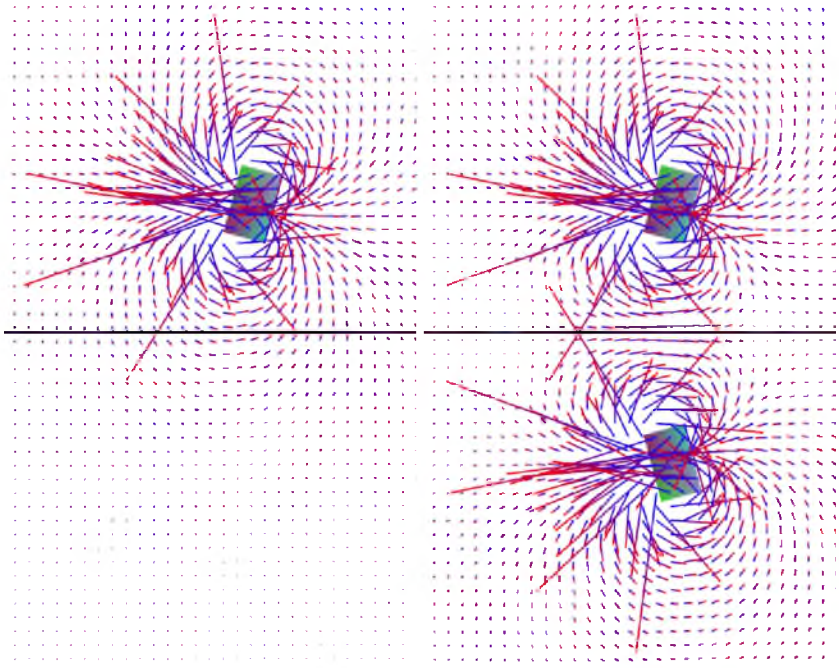

Figure 5.2. Domain boundary comparison. Left: A visualization of the velocity field of an object near a domain boundary. Note that, along the black line, the velocities point into and out of the domain. Right: After the addition of a mirrored object below the black line, there is no flow across the domain boundary.

$$\mathbf{u}_p = \sum_i \mathbf{u}_i \tag{5.18}$$

and $\mathbf{u}_i$ is the velocity field induced by panel i. $\mathbf{u}_p$ can be evaluated at any specific point in space through evaluation of Equation (5.11) and a straightforward summation. For example, to advect a tracer particle, we can combine the reduced velocity, reconstructed in the neighborhood of the particle, with the velocity evaluated from the panels.

However, the panel strengths have no memory and must be recomputed from scratch each timestep. Thus, we need to feed their contribution back into the reduced fluid simulation to preserve momentum. This step can be accomplished by iterating over the panels and summing their contribution to the background grid. The resulting velocity field, $\mathbf{u}_p$, is then be projected into the reduced space and added to the reduced coefficients. However, naïvely evaluating Equation (5.11) at every background grid velocity sample is

computationally expensive and can be especially wasteful if there are large errors when $\mathbf{u}_p$ is projected into the reduced basis.

Instead, we approximate the contributions of panels to distant background grid samples using a quadtree data structure. Specifically, we build a quadtree over the background grid where the root corresponds to the entire domain and the leaf nodes correspond to disjoint subgrids. In our examples, the maximum size of a leaf node subgrid is 4×4, corresponding to 4 u and 4 v velocity samples. We use a precomputed error metric to determine how deep to descend the quadtree when evaluating $\mathbf{u}_i$. To precompute this error metric, we consider a unit strength vortex panel and evaluate $\mathbf{u}_i$ at the center of the quadtree node, $\mathbf{c}$, and additional sample points inside the quadtree node, $\mathbf{s}_j$. Then, the maximum error induced by using a constant approximation of $\mathbf{u}_i$ for the quadtree node is

$$\max_j \|\mathbf{u}_i(\mathbf{c}) - \mathbf{u}_j(\mathbf{s}_j)\|. \tag{5.19}$$

We compute these error samples for quadtree nodes at a number of distances and directions from the panel and store the maximum error incurred at a given level of the quadtree for a given distance.

At runtime, when computing the contribution of $\mathbf{u}_i$ to $\mathbf{u}_p$, which is stored on the background grid, we use these precomputed values to determine the error induced by approximating the velocities using the center of a quadtree node. If the error is below a threshold, the panel velocity is evaluated at the center of the quadtree node and this value is added to all the background velocity values covered by the quadtree node; otherwise, we descend the tree.

When using our quadtree acceleration, we still must project $\mathbf{u}_p$ onto the reduced basis. Note that some details of the velocity field will be lost in this projection and, in particular, the reduced velocity field may not respect obstacles boundaries. However, before this feedback, the velocity field $\mathbf{u}_r + \mathbf{u}_p$ does satisfy the boundary conditions and can be evaluated exactly at any point in space in time linear in the number of panels and the number of reduced coefficients. This velocity should be used for, e.g., advecting tracer particles.

5.3.5 Fluid-to-Solid Coupling

We incorporate fluid to solid coupling by computing the dynamic pressure on the boundary of the rigid body. From the dynamic pressure, we compute the force, which is then added to the rigid body simulation. The dynamic pressure, sometimes called the velocity pressure, is

$$q = \frac{1}{2}\rho \mathbf{u}^T \mathbf{u}, \tag{5.20}$$

where ρ is the density of the fluid, and $\mathbf{u}$ is the fluid velocity. For each panel, we have already computed the difference in relative velocity between the obstacle and fluid when solving for the panel strengths. From that velocity, we compute the dynamic pressure q at panel centers and then multiply by the panel area to get forces [59], which are normal to the panels. Specifically, the force on panel i is

$$\mathbf{f} = A_i q \mathbf{n}_i, \tag{5.21}$$

which is then applied to the rigid body at the panel centers.

Buoyancy forces can optionally be included with

$$\mathbf{f}_i = -\rho A_i h_i g \mathbf{n}_i, \tag{5.22}$$

where h_i is the depth of the panel center and g is the scalar gravitational constant. The minus sign is to signify that the force is in the direction opposite the surface normal of the panel.

Both forces integrate over surfaces and require that objects are closed.

5.4 Results and Discussion

In our first example, we have a single data-driven mode with 63 Eigenmodes. The artist input and the input after it has been projected to be divergence-free is shown in Figure 5.3. The Eigenmodes poorly capture this "jet," but represent gross flow well, while our enhanced basis captures both the gross flow and the jet well; see Figure 5.4.

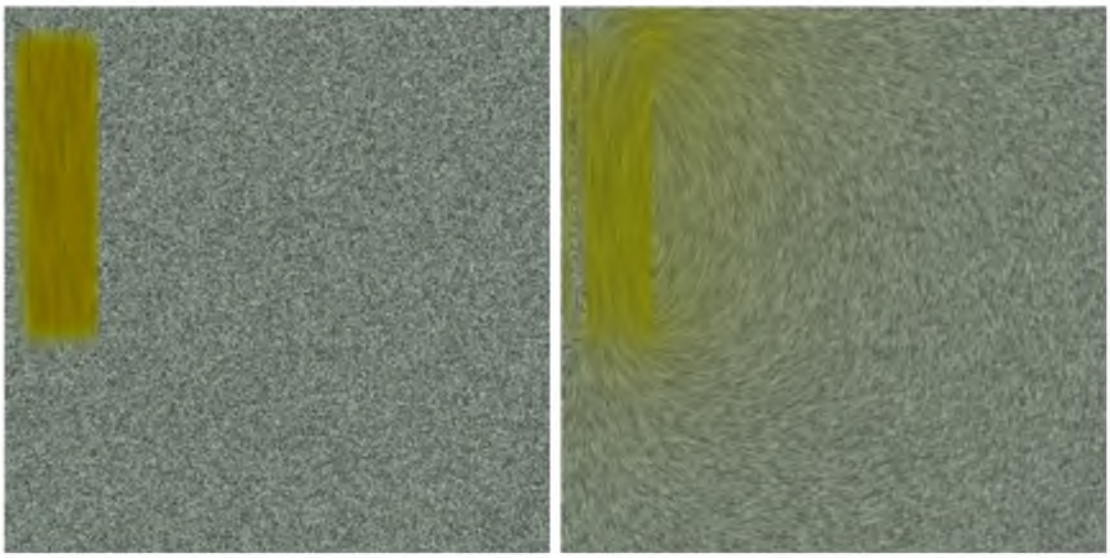

Figure 5.3. Artist input comparison. Left: Line integral convolution (LIC) is used to visualize the input from the artist. Right: The input from the artist after it has been projected to be divergence free.

Our second example uses a 128x128 grid with 67 Eigenmodes and contains two pairs of falling objects; each pair has one object above the other. After being released, the objects above catch up to the objects below, closing the gap between them. The objects that start out above draft off of the objects below, allowing them to fall faster through the fluid, demonstrating the effects of solid-fluid coupling and object-object interaction; see Figure 5.5.

Finally, we have combined both our basis enhancement and two-way coupling into a simple 2D game; see Figure 5.6. The game uses 73 Eigenmodes and there are 15 objects with a total of 147 panels. Timing results in Table 5.1 show that the naive approach of computing feedback from the panel velocities to the reduced simulation dominates timing, taking $41ms$ in this example. By using our quadtree feedback approach, we can reduce the time spent computing feedback by increasing the error threshold of the approximation. In practice, this error threshold can be quite large because this error is hidden by errors made when projecting the resulting velocity field into the reduced basis.

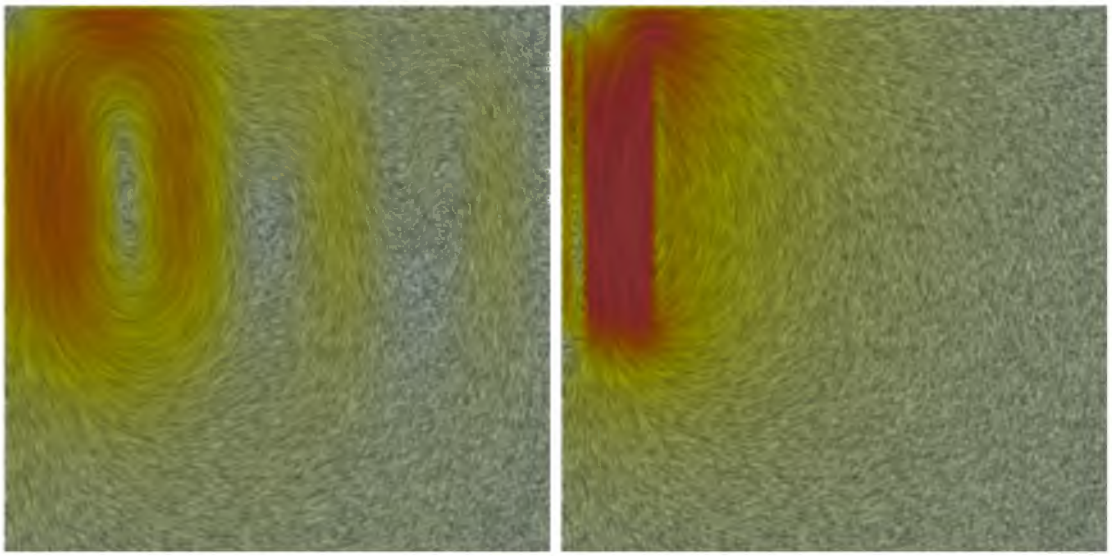

Figure 5.4. Basis comparison. Left: Only Eigenmodes. Right: A data driven mode with Eigenmodes. When exciting the jet with high intensity, the induced flow is not well represented using only the Eigenmodes.

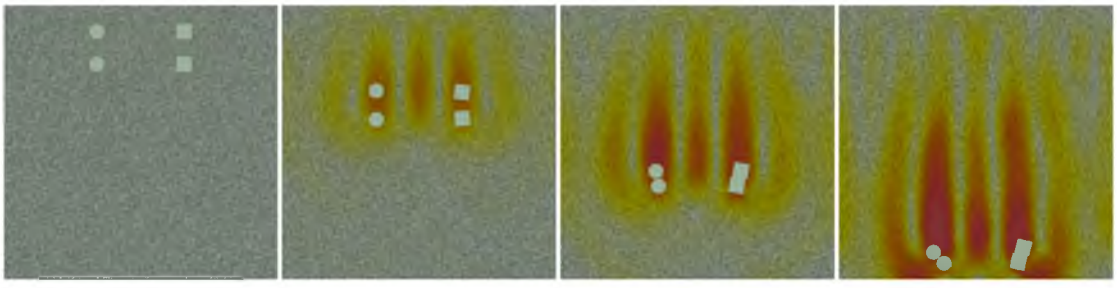

Figure 5.5. Drafting example: Objects above draft off of and catch up to the objects below. This example demonstrates solid-fluid coupling and object-object interactions.

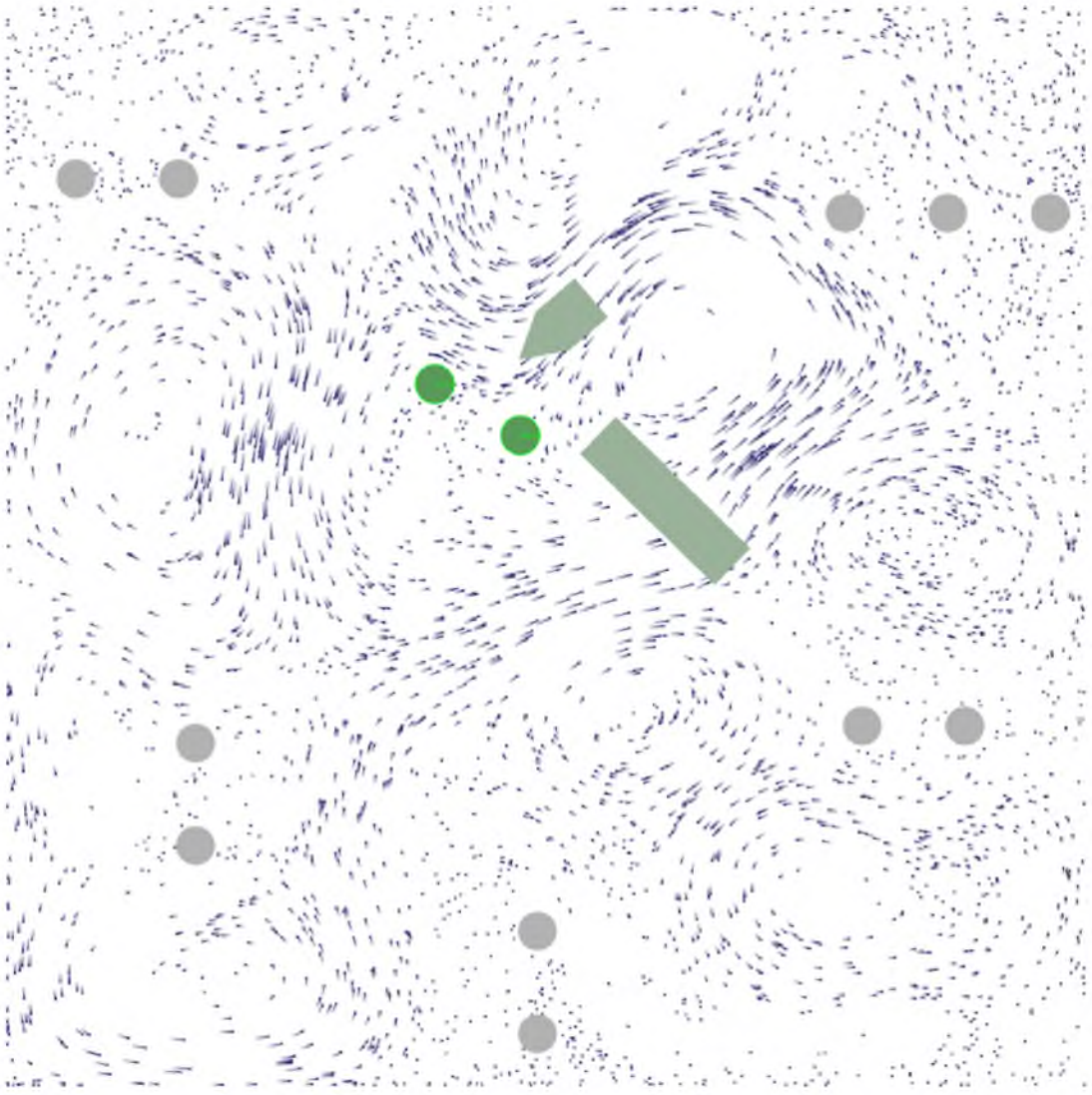

Figure 5.6. An image from a game using our system.

Table 5.1. Timings in *ms* for game scene with 73 modes on a 65x65 staggered grid.

Description	Time (*ms*)
Advect	0.744
Diffuse	0.00536
Panel Solves	5.626
Panel Feedback Naive	41.175
Panel Feedback Quadtree	5.825

CHAPTER 6

CONCLUSIONS

This dissertation considered three problems in physics-based animation: large-scale splashing liquids, elastoplastic material simulation, and dimensionality reduction techniques for fluid simulation. We demonstrated that our novel combination of unilateral incompressibility, mass-full FLIP, and blurred boundaries provides a very effective simulation strategy for large-scale splashing liquids. By avoiding the artificial surface tension of traditional incompressibility, our approach is able to simulate liquids that mix freely with the surrounding air, while also avoiding the oscillations present in smoothed particle hydrodynamics. The particle-based mass-full FLIP is well-suited to splashes and the thin-sheets they create and do not suffer from mass loss or gain. Our blurred boundaries unify the liquid and obstacle representations and work with the unilateral incompressibility to allow liquid to detach from obstacles. Overall, we believe our approach offers a number of advantages over the state-of-the-art for animating large-scale, splashy liquids. In future work, we would like to experiment with alternative LCP solvers, such as the multigrid method of Chentanez and Müller [13] and interior point methods.

Additionally, we demonstrated that our point-based approach for animating elastoplastic materials is well-suited to simulating materials that experience large plastic deformations. It is also capable of simulating rather stiff elastic materials, though some drift is inevitable. Unfortunately, our approach is not well-suited to the large elastic deformations exhibited by soft objects. In such cases, the deformation gradient becomes ill-conditioned and our method breaks

down. Recently, Jones *et al.* [36] addressed this limitation and are able to handle large elastic deformations by including a rest configuration. Other researchers extended our method to handle larger timesteps and improved stability by using an implicit integrator, which we suggested was a promising area of future work [75]. Other interesting areas of future work include addressing topological changes in a physically based manner (currently, topological changes occur when particle neighborhoods change), and methods for resampling/adaptive sampling. The last direction is particularly interesting as it may improve stability as well as provide performance benefits.

We believe our approach has a number of advantages over competing techniques. In particular, it does not require any rest configuration, no remeshing is needed, it can handle elastic and large plastic deformations in a unified framework and it is simple to implement and inexpensive to compute. While we have demonstrated our approach with a particle-based method, the general approach to computing the deformation gradient should be applicable in other simulation methods, such as Eulerian grid-based or finite element techniques. This work has garnered 26 citations in the last 3 years and inspired follow-on work.

Finally, we discussed several enhancements to dimensionally reduced fluid simulations: a basis enrichment scheme to mix data-driven and analytic modes, and a new approach to two-way solid-fluid coupling. Our enrichment scheme enables the combination of the generality of Eigenmodes with the context awareness and art directability of data-driven modes. Our approach to solid-fluid coupling combines vortex panel methods for solid-to-fluid coupling, dynamic pressure for fluid-to-solid coupling, the method of images to handle domain boundaries, and a quadtree-based method to accelerate the solid-to-fluid coupling. This approach enables robust coupling of dynamic objects to the dimensionally reduced simulation and requires no training data. In future work, we plan to extend the technique to 3D.

APPENDIX

PUBLICATIONS

Assembling Large Mosaics of Electron Microscope Images Using GPU,
Kannan Venkataraju, Mark Kim, Dan Gerszewski, James R. Anderson, Mary Hall,
In proceedings of Symposium on Application Accelerators in High Performance Computing 2009. Urbana, Illinois, July 2009

A Point-based Method for Animating Elastoplastic Solids,
Dan Gerszewski, Haimasree Bhattacharya, Adam W. Bargteil,
In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2009*

Physics-Inspired Upsampling for Cloth Simulation in Games,
Ladislav Kavan, Dan Gerszewski, Adam W. Bargteil, Peter-Pike Sloan,
In *ACM Transactions on Graphics (SIGGRAPH 2011), August 2011, Vol 30. No. 4.*

Physics-based Animation of Large-scale Splashing Liquids,
Dan Gerszewski, Adam W. Bargteil
In *ACM Transactions on Graphics (SIGGRAPH ASIA 2013), November 2013, Vol 32, No 6.*

Enhancements to Model-reduced Fluid Simulation,
Dan Gerszewski, Ladislav Kavan, Peter-Pike Sloan, Adam W. Bargteil,
Proceedings of ACM Motion in Games, Dublin, Ireland, November 2013.

Basis Enrichment and Solid-fluid Coupling for Model-reduced Fluid Simulation,
Dan Gerszewski, Ladislav Kavan, Peter-Pike Sloan, Adam W. Bargteil,
Computer Animation and Virtual Worlds, in submission January 2014

REFERENCES

[1] Adams, B., Pauly, M., Keiser, R., and Guibas, L. J. Adaptively sampled particle fluids. *ACM Trans. Graph. 26*, 3 (2007), 48.

[2] Alduán, I., and Otaduy, M. A. Sph granular flow with friction and cohesion. In *Proc. of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2011).

[3] Bargteil, A. W., Wojtan, C., Hodgins, J. K., and Turk, G. A finite element method for animating large viscoplastic flow. *ACM Trans. Graph. 26*, 3 (2007), 16.

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

[5] Becker, M., Ihmsen, M., and Teschner, M. Corotated sph for deformable solids. In *Proceedings of the Fifth Eurographics Conference on Natural Phenomena* (Aire-la-Ville, Switzerland, Switzerland, 2009), NPH'09, Eurographics Association, 27–34.

[6] Becker, M., and Teschner, M. Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, 209–217.

[7] Bodin, K., Lacoursiere, C., and Servin, M. Constraint fluids. *IEEE Transactions on Visualization and Computer Graphics 18*, 3 (Mar. 2012), 516–526.

[8] Box2D. Erin Catto, 2011.

[9] Brackbill, J. U., and Ruppel, H. M. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys. 65* (August 1986), 314–343.

[10] Bridson, R. *Fluid Simulation for Comuter Graphics*. A K Peters, 2008.

[11] Bridson, R., and Müller-Fischer, M. Fluid simulation: Siggraph 2007 course notes. In *ACM SIGGRAPH 2007 courses* (2007), 1–81.

[12] Brochu, T., Keeler, T., and Bridson, R. Linear-time smoke animation with vortex sheet meshes. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012), 87–95.

[13] Chentanez, N., and Müller, M. A multigrid fluid pressure solver handling separating solid boundary conditions. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, 83–90.

[14] Chentanez, N., and Müller, M. Real-time eulerian water simulation using a restricted tall cell grid. *ACM Trans. Graph. 30*, 4 (July 2011), 82:1–82:10.

[15] Chorin, A., and Marsden, J. *A Mathematical Introduction to Fluid Mechanics*. Springer, 2000.

[16] Clavet, S., Beaudoin, P., and Poulin, P. Particle-based viscoelastic fluid simulation. In *The Proccedings of the Symposium on Computer Animation* (2005), 219–228.

[17] Cottet, G.-H., and Koumoutsakos, P. *Vortex methods: Theory and practice*. Cambridge University Press, June 2000.

[18] De Witt, T., Lessig, C., and Fiume, E. Fluid simulation using laplacian eigenfunctions. *ACM Trans. Graph. 31*, 1 (Feb. 2012), 10:1–10:11.

[19] Dostál, Z. *Optimal Quadratic Programming Algorithms: With Applications to Variational Inequalities*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[20] Dostál, Z., and Schöberl, J. Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination. *Comput. Optim. Appl. 30*, 1 (Jan. 2005), 23–43.

[21] Enright, D. P., Marschner, S. R., and Fedkiw, R. P. Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21*, 3 (2002), 736–744.

[22] Fedkiw, R., Stam, J., and Jensen, H. W. Visual simulation of smoke. In *The Proceedings of ACM SIGGRAPH 2001* (2001), 15–22.

[23] Foster, N., and Fedkiw, R. Practical animation of liquids. In *The Proceedings of ACM SIGGRAPH 2001* (2001), 23–30.

[24] Foster, N., and Metaxas, D. Realistic animation of liquids. In *The Proceedings of Graphics Interface* (1996), 204–212.

[25] Goktekin, T. G., Bargteil, A. W., and O'Brien, J. F. A method for animating viscoelastic fluids. *ACM Trans. Graph. 23*, 3 (2004), 463–468.

[26] Goktekin, T. G., Reisch, J., Peachey, D., and Shah, A. An effects recipe for rolling a dough, cracking an egg and pouring a sauce. In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches* (New York, NY, USA, 2007), ACM, 67.

[27] Golas, A., Narain, R., Sewall, J., Krajcevski, P., Dubey, P., and Lin, M. Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 148:1–148:9.

[28] Gross, M., and Pfister, H. *Point-Based Graphics.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[29] Gupta, M., and Narasimhan, S. G. Legendre fluids: a unified framework for analytic reduced space modeling and rendering of participating media. In *Proc. of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), 17–25.

[30] Harlow, F., and Welch, J. Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *The Physics of Fluids 8* (1965), 2182–2189.

[31] Hess, J., and Smith, A. Calculation of non-lifting potential flow about arbitrary three-dimensional bodies. Technical Report E.S. 40622, Douglas Aircraft Division, 1962.

[32] Hieber, S. E., and Koumoutsakos, P. A Lagrangian particle method for the simulation of linear and nonlinear elastic models of soft tissue. *J. Comp. Phys. 227*, 21 (2008), 9195–9215.

[33] Holmes, P., Lumley, J., and Berkooz, G. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry.* Cambridge Monographs on Mechanics. Cambridge University Press, 1996.

[34] Irving, G. *Methods for the Physically Based Simulation of Solids and Fluids.* PhD thesis, Stanford University, 2007.

[35] Irving, G., Teran, J., and Fedkiw, R. Invertible finite elements for robust simulation of large deformation. In *The Proceedings of the Symposium on Computer Animation* (2004), 131–140.

[36] Jones, B., Ward, S., Jallepalli, A., Perenia, J., and Bargteil, A. Deformation embedding for point-based elastoplastic simulation. *ACM Trans. Graph.* (2014).

[37] Keiser, R., Adams, B., Gasser, D., Bazzi, P., Dutré, P., and Gross, M. A unified Lagrangian approach to solid-fluid animation. In *The Proceedings of the Symposium on Point-Based Graphics* (2005), 125–133.

[38] Kim, T., and Delaney, J. Subspace fluid re-simulation. *ACM Trans. Graph. 32*, 4 (July 2013), 62:1–62:9.

[39] Lentine, M., Zheng, W., and Fedkiw, R. A novel algorithm for incompressible flow using only a coarse grid projection. *ACM Trans. Graph. 29*, 4 (July 2010), 114:1–114:9.

[40] Long, B., and Reinhard, E. Real-time fluid simulation using discrete sine/cosine transforms. In *Proc. of the symposium on Interactive 3D graphics and games* (2009), 99–106.

[41] Loos, B. J., Antani, L., Mitchell, K., Nowrouzezahrai, D., Jarosz, W., and Sloan, P.-P. Modular radiance transfer. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 178:1–178:10.

[42] Losasso, F., Shinar, T., Selle, A., and Fedkiw, R. Multiple interacting liquids. *ACM Trans. Graph. 25*, 3 (2006), 812–819.

[43] Losasso, F., Talton, J., Kwatra, N., and Fedkiw, R. Two-way coupled sph and particle level set fluid simulation. In *IEEE TVCG* (2008), vol. 14.

[44] Lumley, J. The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation* (1967), 166178.

[45] Lumley, J. *Stochastic tools in turbulence*. Applied mathematics and mechanics. Academic Press, 1970.

[46] Macklin, M., and Müller, M. Position based fluids. *ACM Trans. Graph. 32*, 4 (July 2013), 104:1–104:12.

[47] McAdams, A., A., S., Ward, K., Sifakis, E., and Teran, J. Detail preserving continuum simulation of straight hair. *ACM Transactions on Graphics (SIGGRAPH Proceedings) 28(3)* (2009).

[48] Müller, M., Charypar, D., and Gross, M. Particle-based fluid simulation for interactive applications. In *The Proceedings of the Symposium on Computer Animation* (2003), 154–159.

[49] Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., and Alexa, M. Point based animation of elastic, plastic and melting objects. In *The Proceedings of the Symposium on Computer Animation* (2004), 141–151.

[50] Narain, R., Golas, A., Curtis, S., and Lin, M. Aggregate dynamics for dense crowd simulation. In *SIGGRAPH Asia '09* (2009), vol. 28, 122:1–122:8.

[51] Narain, R., Golas, A., and Lin, M. C. Free-flowing granular materials with two-way solid coupling. *ACM Transactions on Graphics (Proceedings of Siggraph Asia 2010)* (2010).

[52] O'Brien, J. F., Bargteil, A. W., and Hodgins, J. K. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph. 21*, 3 (2002), 291–294.

[53] Park, S. I., and Kim, M. J. Vortex fluid for gaseous phenomena. In *Proc. of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), 261–270.

[54] Pauly, M., Keiser, R., Adams, B., Dutré;, P., Gross, M., and Guibas, L. J. Meshless animation of fracturing solids. *ACM Trans. Graph. 24*, 3 (2005), 957–964.

[55] Pfaff, T., Thuerey, N., and Gross, M. Lagrangian vortex sheets for animating fluids. *ACM Trans. Graph. 31*, 4 (July 2012), 112:1–112:8.

[56] Pfaff, T., Thuerey, N., Selle, A., and Gross, M. Synthetic turbulence using artificial boundary layers. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 121:1–121:10.

[57] Raveendran, K., Wojtan, C., and Turk, G. Hybrid smoothed particle hydrodynamics. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, 33–42.

[58] Ruilova, A. Creating realistic cg honey. In *SIGGRAPH '07: ACM SIGGRAPH 2007 posters* (New York, NY, USA, 2007), ACM, 58.

[59] Saffman, P. *Vortex Dynamics*. Cambridge University Press, 1995.

[60] Schechter, H., and Bridson, R. Ghost sph for animating water. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012) 31*, 4 (2012).

[61] Simo, J., and Hughes, T. *Computational Inelasticity*. Springer-Verlag, 1998.

[62] Sirovich, L., and of Applied Mathematics, B. U. D. *Turbulence and the Dynamics of Coherent Structures*. Quarterly of applied mathematics. Brown University, Division of Applied Mathematics, 1987.

[63] Solenthaler, B., and Pajarola, R. Predictive-corrective incompressible sph. In *ACM SIGGRAPH 2009 papers* (2009), 40:1–40:6.

[64] Solenthaler, B., Schläfli, J., and Pajarola, R. A unified particle model for fluid-solid interactions. *Journal of Visualization and Computer Animation 18*, 1 (2007), 69–82.

[65] Stam, J. Stable fluids. In *The Proceedings of ACM SIGGRAPH* (1999), 121–128.

[66] Stanton, M., Sheng, Y., Wicke, M., Perazzi, F., Yuen, A., Narasimhan, S., and Treuille, A. Non-polynomial galerkin projection on deforming meshes. *ACM Trans. Graph. 32*, 4 (July 2013), 86:1–86:14.

[67] Terzopoulos, D., and Fleischer, K. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *The Proceedings of ACM SIGGRAPH* (1988), 269–278.

[68] Toselli, A., and Widlund, O. *Domain Decomposition Methods - Algorithms and Theory*. Springer, 2004.

[69] Treuille, A., Lewis, A., and Popović, Z. Model reduction for real-time fluids. *ACM Trans. Graph. 25*, 3 (July 2006), 826–834.

[70] Wicke, M., Ritchie, D., Klingner, B. M., Burke, S., Shewchuk, J. R., and O'Brien, J. F. Dynamic local remeshing for elastoplastic simulation. *ACM Transactions on Graphics 29*, 4 (July 2010), 49:1–11. Proceedings of ACM SIGGRAPH 2010, Los Angles, CA.

[71] Wicke, M., Stanton, M., and Treuille, A. Modular bases for fluid dynamics. *ACM Trans. Graph. 28*, 3 (July 2009), 39:1–39:8.

[72] Williams, B. Fluid surface reconstruction from particles. Master's thesis, University of British Columbia, 2008.

[73] Wojtan, C., and Turk, G. Fast viscoelastic behavior with thin features. *ACM Trans. Graph. 27*, 3 (2008), 1–8.

[74] Yu, J., Wojtan, C., Turk, G., and Yap, C. Explicit mesh surfaces for particle based fluids. *EUROGRAPHICS 2012 30* (2012), 41–48.

[75] Zhou, Y., Lun, Z., Kalogerakis, E., and Wang, R. Implicit Integration for Particle-based Simulation of Elasto-Plastic Solids. *Computer Graphics Forum 32*, 7 (2013), 215–223.

[76] Zhu, Y., and Bridson, R. Animating sand as a fluid. In *ACM SIGGRAPH 2005 papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM.