PRACTICAL WATER ANIMATION USING PHYSICS AND IMAGE BASED METHODS

A Thesis
Presented to
The Academic Faculty

by

Huamin Wang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
December, 2009

PRACTICAL WATER ANIMATION USING PHYSICS AND IMAGE BASED METHODS

Approved by:

Professor Jarek Rossignac,
Committee Chair
College of Computing
Georgia Institute of Technology

Professor Greg Turk, Advisor
College of Computing
Georgia Institute of Technology

Professor Irfan Essa
College of Computing
Georgia Institute of Technology

Professor C. Karen Liu
College of Computing
Georgia Institute of Technology

Professor Peter J. Mucha
Department of Mathematics
The University of North Carolina at Chapel Hill

Date Approved: August 17th, 2009

Dedicated with love to my wife, Jingchao,

for her never-ending love, support, and gentle guidance throughout the years.

PREFACE

> *'**Determinism** is the philosophical proposition that every event, including human cognition and behaviour, decision and action, is causally determined by an unbroken chain of prior occurrences...'*
>
> - Peter Van Inwagen, in *An Essay on Free Will*

The world is as precise as a clock. When I was learning physics for the first time as a kid, I was fascinated by how accurately this world can be predicted by all kinds of physical laws. If you know how fast you throw an apple and where it leaves your hand, you can predict whether it will hit Isaac Newton's head. If you shine a laser pen towards mirrors and lenses, the optics provides you the whole light path so that you may defeat the entire Roman fleet, as Archimedes did. I could not stop imagining that one day everything can be predicted in this world, given all of physical laws. Obviously I was not the first one with this wild idea. Actually, philosophers already had a name for it: *determinism*. Soon I learned that *quantum mechanics* is another phenomenon existing in this world, even although I still blindly refuse to accept it just like Albert Einstein did. Luckily this is a thesis for a degree in computer science, not in physics.

Thus I turned myself to another question: given a supercomputer that can duplicate everything into a virtual world, can we really distinguish the virtual world from the real world? I was not surprised at all when I saw the movie *The Matrix* raising the same question that I had. Maybe there is no answer to this question, but at least, computer scientists can make such a scenario real.

Moving the real world into the virtual world is not so straightforward as writing down a collection of physical laws. They are far too simplified to cover all of the

aspects in real world mechanics. The real world is also greatly beyond our imagination in terms of its detail and complexity. Even a pebble has an irregular shape, texture details, and complicated light transport properties. Last but not least, an infinitely powerful supercomputer only exists in scientific fictions. Computational and storage resource of a real-world computer always restricts our choices when simulating the real world. All of these issues come together to make this problem challenging, yet fascinating to me.

This dissertation attempts to handle one particular phenomenon in the real world: liquid. I will first present several physically based algorithms to simulate small-scale liquid animation realistically and efficiently. A novel liquid surface modeling approach will then be proposed to combine water appearance observed in the real world together with a fluid dynamics model, that is, combining image based reconstruction techniques with physically based simulation techniques. This approach avoids most existing difficulties in conventional fluid modeling methods, and provides a promising prospective for future research in the area of modeling natural phenomena in computer graphics.

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my PhD advisor, Prof. Greg Turk, for his continuous support through my five year's PhD life, for his patience, motivation, enthusiasm, and immense knowledge. His encouragement helped me step away from those tough times and he is always there ready to help his students with all his effort. I cannot imagine I could have a better advisor and mentor than him. Thank you so much, Greg!

This dissertation is also simply impossible without all of my thesis committee members. I would like to thank Prof. Jarek Rossignac, Prof. Irfan Essa, Prof. C. Karen Liu and Prof. Peter J. Mucha for their encouragement, insightful suggestions, and sometimes tough questions. They helped me think deeper into these research problems and made this thesis work truly more valuable.

I would regret my doctoral years if I did not join the computer graphics group at Georgia Tech and meet my colleagues and labmates, Chris Wojtan, Mark Calson, Brooks Van Horn III, Yuting Ye, Sumit Jain, Vivek Kwatra, Howard Zhou, Jie Sun, Pei Yin, Jiaxin Wu, Ping Wang, Kai Ni, Byungmoon Kim, Brian Whited... The list is too long to be completed here and please forgive me for omitting your names, my friends. I will never forget those wonderful research ideas inspired from discussion with you, those sleepless nights spent together before a paper deadline, and all the helps I received when I was in trouble. My graduate school experience would not be so colorful simply without you.

My gratitude also goes to NVIDIA corporation, Adobe System Incorporated, Microsoft Research and Microsoft Research Asia for supporting my research and allowing me to obtain more research experience in research labs. In particular, I thank Prof.

Ruigang Yang, Dr. Gavin Miller, Dr. Hugues Hoppe and Dr. Kun Zhou for offering great opportunities to work with them, to broaden my knowledge, and to solve various exciting research problems.

Finally, I thank my wife, my parents for supporting me during this exciting, but stressful period. I am proud of being in this family and I am always working with all my efforts to let you feel proud of me.

Huamin Wang

Atlanta, Georgia

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Generating natural phenomena in a virtual world has a number of practical applications. Thanks to the rich and complicated details in the real world, the goal of realistically and efficiently reproducing natural phenomena in a digital world is well known as an open problem for graphics researchers.

In this dissertation, three different issues in modeling liquid animations have been addressed. First, a virtual surface method and its 3D numerical solution is proposed to account for surface tension effects and their interactions with solid surfaces in physically based fluid simulation. This allows us to generate various surface tension behaviors in small scale liquid, such as water drops, bubbles, water streamlets and capillary action.

The second issue that is addressed is how to make small scale fluid simulation more efficient. The solution proposed in this dissertation is a general shallow wave equation model, extended from the original shallow wave equations. By simplifying 3D incompressible fluid dynamics into a 2D problem under the shallow wave assumption, small scale liquid can be stably and efficiently simulated over arbitrarily curved surfaces, using implicit numerical schemes.

The third contribution is a novel hybrid framework for animating liquid that combines image based reconstruction techniques with physically based fluid simulation. While image based reconstruction cannot correctly generate fluid animations alone frame by frame due to depth reconstruction noise and errors, physically based simulation is used as a refinement tool to enforce incompressible fluid dynamics over the initial reconstruction by propagating shape information back and forth in space and time. In this way, water animations can be realistically and faithfully generated

from images without error accumulation or stability issues as in physically based simulation. The whole process is also more efficient than physically based simulation, because physics is only used to achieve temporal coherence rather than to generate the whole animation.

CHAPTER I

INTRODUCTION

1.1 Background

Generating natural phenomena in a virtual world has a number of practical applications: digital visual effects in computer games and movies, simulation and training systems such as virtual surgery, visualization of engineering and architecture designs, and visualization for scientific research. Due to the rich and complicated details in the real world, the goal of realistically and efficiently reproducing natural phenomena in a digital world is well known as an open problem. In this dissertation, I present novel algorithms to efficiently and realistically model one particular phenomenon in the real world: liquid. Proposed algorithms include pure physically based simulation algorithms and a hybrid modeling method that combines image based reconstruction techniques with physically based simulation. Here we will briefly examine two groups of state-of-the-art techniques for modeling natural phenomena: physically based simulation and image/video based reconstruction.

Physically based simulation techniques attempt to simulate phenomena in a digital world by following physical laws. For example, clouds in the sky can be created procedurally from gas simulation, and a burning furnace can be modeled by simulating the combustion process. Existing simulation techniques are able to produce a number of natural phenomenon animations, including rigid objects [5], hair [86], cloth [11], viscoelastic objects [117], fire [74], smoke [26] and water [96, 30, 24]. Because these techniques try to faithfully follow physical laws frame by frame, visually plausible results can be generated without a sudden violation of physics. However, they suffer

from error accumulation, including volume loss and detail loss. In addition, their results usually appear to be less realistic because all of the details in the real world may not be completely reproduced by simulation. Most simulation techniques require intense computational resources, especially for solving complicated object collisions and fluid dynamics. Researchers have invented various methods to accelerate physically based simulation, for example, using parallel processing units in graphics hardware. Nevertheless, physically based simulation that is both efficient and detailed is difficult to achieve.

The intuition behind *image based* reconstruction techniques is to replicate observations of the real world in a virtual world. The observation is usually done using multiple cameras that capture image sequences over time at specified locations in the real world. Captured images are then used to construct virtual geometric models. In contrast to data acquisition techniques that use optical or laser range scanners, a multi-camera system is easier and more affordable to configure even for outdoor data capture tasks. Using this basic idea, image based techniques have been successfully developed to capture a number of natural phenomena, including flowers [81], trees [101], hair [113], and cloth [114, 9]. Since the input comes directly from the real world, these techniques are realistic in details and comparably more efficient than simulation approaches. Unfortunately, image based methods are limited in several respects. First, occlusion is difficult to handle since the method needs to fill in what has not been observed. Second, most image based reconstruction methods are limited to certain cases: some of them assume that there exists dense and detailed texture over the object surface; some can only handle Lambertian surfaces, or surfaces with known BRDFs; and some rely on shape priors for the reconstruction process. Last but not least, maintaining temporal coherence and physical correctness is extremely challenging when reconstructing animation from image sequences.

An ideal algorithm to generate natural phenomena should have the following properties:

Efficiency: The algorithm should be sufficiently fast so that users can view and interact with the virtual environment in real time (> 15 fps).

Fidelity: Its result should be as realistic as the real world, following all of the physical laws.

Generality: The algorithm should be flexible enough to handle a wide variety of cases. For example, an ideal liquid animation simulator should be able to simulate all kinds of liquid behaviors, from large ocean waves to capillary action, with various visco-plastic properties.

Coupling: It should be easy to combine the simulation method with those of other natural phenomena. For example, a liquid simulator should also handle interactions with other material, such as rigid objects, elastic objects, thin shell objects or other types of fluid.

It is my belief that modeling natural phenomena will be an increasingly important research problem in computer graphics, and future research will be guided by these four key properties listed above.

1.2 The Problem

The specific problem that I am investigating in this dissertation is how to model liquid animation for graphics applications.

In recent years, considerable progress has been made in physically based fluid simulations [96, 30, 24] by adapting simulation techniques from computational fluid dynamics. These techniques are computationally expensive, typically involving hours or days of computing time in order to guarantee stability and accuracy in the results. Moreover, they are still lacking in realistic details compared with real liquid, and they are difficult to control and to interact with [64, 25].

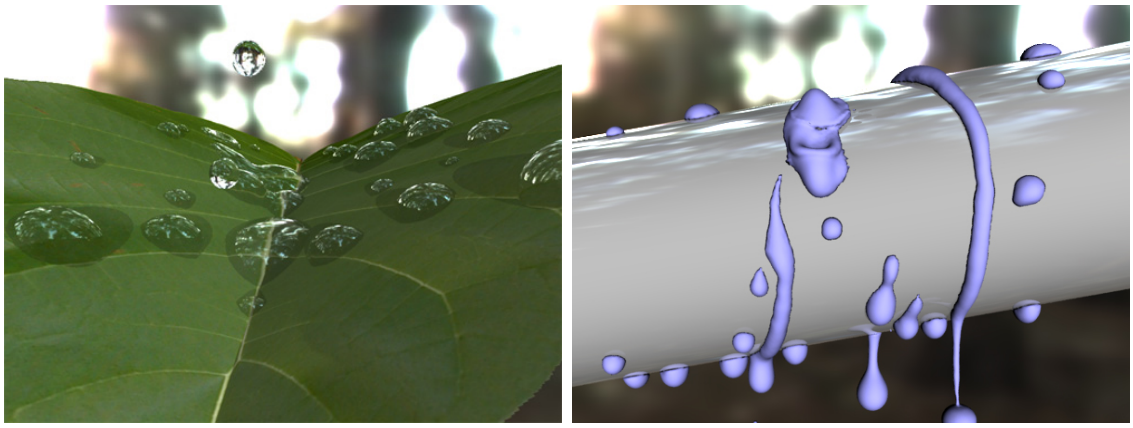


Figure 1: Water examples generated by the virtual surface method.

Solely using image based reconstruction techniques for liquid animation is difficult due to liquid's complex nature. Liquid can be both reflective and refractive, with few texture details. Liquid surface topology also changes dramatically even in a simple scenario, thus making it impossible to rely on any topological assumptions. As far as we know, no existing image based reconstruction algorithms are capable of reconstructing general liquid animation. Recent approaches proposed by Morris and Kutulakos [70] and Hullin et al. [45] successfully reconstructed transparent objects such as vases and glass by tracing the light transport using structured scanning, but they are not suitable for reconstructing complex time-varying liquid scenes that exhibit frequent occlusions.

In this dissertation, I will address three related problems in modeling liquid animation. First, I will study how to make conventional physically based fluid simulation even more realistic by incorporating surface tension effects and their interactions with solid surfaces. This approach results in important behavior in small scale liquid animation such as water drops, bubbles, water streamlets and capillary action. The key idea of this proposed algorithm is the use of a *virtual surface* at the contact front where water, air and solid meet. The virtual surface is an auxiliary surface used to find correct surface tension forces at the contact front. It is defined to form a specific angle with the solid surface, called *the contact angle*, in order to reflect different solid

Figure 2: Water drops simulated by the general shallow wave equation model.

hydrophobicity properties. For example, the virtual surface for a hydrophobic solid is defined by a large contact angle, so that the strong surface tension force will push the water drop into a round shape. On the other hand, the virtual surface for a hydrophilic solid is defined by a small contact angle, so the water drop will be flattened since the weak surface tension force cannot compete against gravity. Even for the same material, the advancing contact angle at the contact front where water moves forward is usually larger than the receding contact angle when water moves backward, causing water streamlets to leave long tails behind themselves. Two such examples are shown in Figure 1.

The second issue that my work addresses is how to make small scale liquid simulation faster. My solution is a general shallow wave equation model, extended from the original shallow wave equations. Like shallow wave equations, the general shallow wave equation model is simplified from incompressible fluid dynamics under the shallow wave assumption. It can also handle simulation of water flowing on arbitrarily curved surfaces. The surface tension effect and the virtual surface method have successfully been incorporated into this general shallow wave model. The whole system can then be efficiently solved using an implicit numerical solver and further accelerated using graphics hardware's parallel processing power. This system can efficiently simulate various small scale liquid animations under the shallow wave assumption

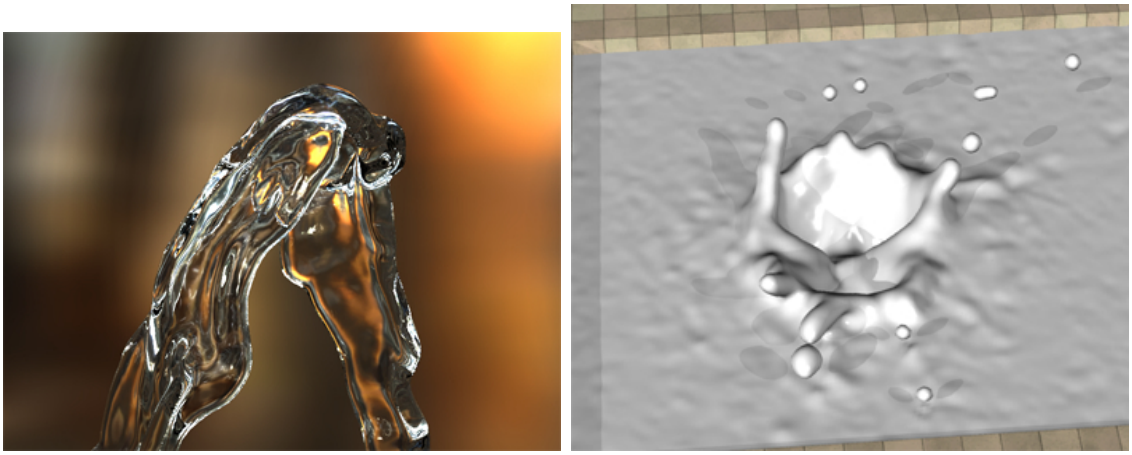


Figure 3: Water phenomena generated by a hybrid method.

in real time, providing the possibility to display, create and control liquid shapes interactively, shown in Figure 2.

My third contribution is a new hybrid method for animating liquid by combining image based reconstruction with physically based fluid simulation. Although image based reconstruction techniques cannot correctly generate liquid animations alone frame by frame, I believe that sufficient information exists in a series of frames by propagating information back and forth in time using physically based simulation. In other words, this approach is an image based reconstruction method that is constrained, guided and elaborated by physically based simulation. Similar ideas have been seen in recent image based reconstruction techniques to model other natural phenomena, such as trees [101], hair [113], and cloth [9].

The proposed hybrid reconstruction algorithm works as follows. A calibrated, synchronized multi-camera system is first used to capture videos of moving water, such as a water fountain, from several known locations. Silhouettes, salient point correspondences and other spatial features are then extracted from images to constrain the liquid surface reconstruction process. We assume that a perfectly reconstructed result should be spatio-temporally coherent. Unfortunately, initial results from pure image based reconstruction alone is usually far from satisfactory, due to occlusions,

correspondence errors and other issues. We classify all errors into two groups: *false positives*, when the reconstruction process creates a liquid region that does not actually exist in the real scene, and *false negatives*, in which real water is omitted in the result. Physically based simulation is then introduced to refine the whole shape sequence and to address false the problems of positives and false negatives, after propagating shape information from neighboring frames. For example, if a water drop suddenly appears in a single frame, it is likely to be false positive and it will be removed as long as it is not observed elsewhere. The final result is expected to match with the image input and to satisfy fluid dynamics as much as possible. The result can also be considered as a 4D hyper-surface that is continuous in both space and time, and can be rendered from any view point at any time instant in a new virtual environment with different lighting conditions.

In general, the result from this hybrid method is more visually plausible than that from image based reconstruction alone since it has been refined by physical laws. Compared with physically based fluid simulation, the hybrid method is free from stability and accuracy issues since physically based simulation is only used as an additional constraint. The experiment shows that the new method can realistically and efficiently generate liquid animations from video sequences, including pouring water, splashing water and a small water fountain.

In order to avoid issues in complicated topological changes, I use the level set method as the surface representation and shape evolution tool in most of the proposed methods. Many other recent techniques in physically based fluid simulation also make use of the level set method. Details about the level set method can be found in Section 2.4.

The rest of this thesis is organized as follows. In Chapter 2, we will examine how a typical fluid solver generates liquid animation. In Chapter 3, we will see how to model surface tension effects with a virtual surface method in a basic fluid simulator

for small scale liquid effects, as shown in Figure 1. Small scale liquid simulation, such as the one in Figure 2, can be simplified using the general shallow wave equation and accelerated by GPU's parallel processing power, as discussed in Chapter 4. A hybrid method for liquid animation will be introduced next in Chapter 5. Figure 3 shows two examples from this new method. Finally, conclusions of all the proposed methods will be given in Chapter 6.

CHAPTER II

FLUID SIMULATION BY EULERIAN APPROACHES

Physically based fluid simulation is a broad area that has been well studied in computational physics before it was introduced into the computer graphics community. In this chapter, we will discuss the popular Eulerian approach for solving viscous incompressible fluid dynamics. In contract to Lagrangian approaches that track shape elements directly, Eulerian approaches are a set of numerical methods that handle and manipulate shapes implicitly on a fixed grid without shape parametrization. Most liquid simulation techniques are comprised of two steps: first, update the velocity field by solving viscous incompressible fluid dynamics, and then evolve the liquid surface based on the given velocity field.

2.1 Viscous Incompressible Fluid Dynamics

The viscous incompressible flow of low-speed Newtonian fluid can be modeled by the Navier-Stokes equations, named after Claude-Louis Navier and George Gabriel Stokes. They are derived by applying Newton's second law to fluid motion, assuming that the fluid stress is the sum of a diffusing viscous term and a pressure term:

$$\begin{aligned} \vec{V}' &= -(\vec{V}\cdot\nabla)\vec{V} + \nu\nabla(\nabla\vec{V})/\rho - \nabla P/\rho + \vec{F}/\rho\,, \\ \nabla\cdot\vec{V} &= 0\,, \end{aligned} \tag{1}$$

in which $\vec{V}$ is the velocity field, ν is the viscosity coefficient, ρ is the liquid density, P is the pressure field, and $\vec{F}$ is the external force. It describes the low-speed dynamics of most liquid and gas that we observe in everyday life.

Here we will focus on conventional simulation techniques using Eulerian approaches with volumetric representations. This representation is insensitive to topological

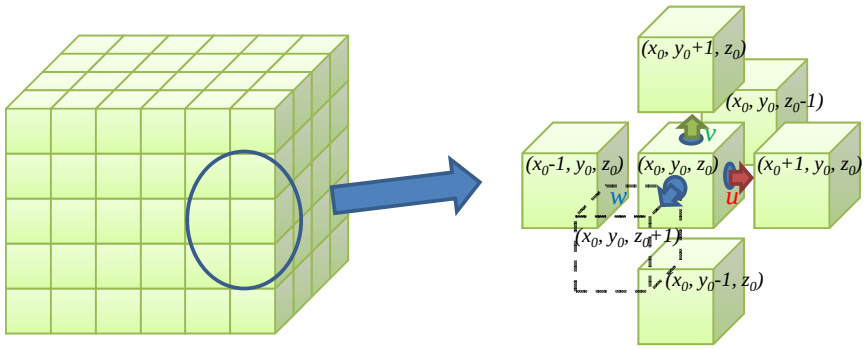


Figure 4: The grid representation.

changes compared with mesh representations. It is also easier to delimitate surface shapes, when compared with particle representations. In order to generate the next frame in a liquid animation, a typical liquid simulator first updates the liquid flow by the Navier-Stokes equations, and then evolves the liquid surface according to the calculated flow over time. We will first study how to discretize the computational space in a cartesian grid in Section 2.2, then examine a numercial solver for the Navier-Stokes equations by the method of characteristics in Section 2.3, and finally discuss different surface tracking algorithms based on volumetric representations in Section 2.4, including the level set method and the volume-of-fluid method.

2.2 Discretization

A 3D regular grid that uniformly discretizes the computational space is suitable for finite difference methods. Continuous function is sampled at the center of each grid cell as Figure 4 shows. During simulation, each grid cell is assigned with a signed distance value for implicit surface representation and the pressure value for pressure solving. In order to facilitate the pressure projection step, a staggered Marker-and-Cell (MAC) formulation is used for the velocity field, in which velocity components u, v and w (for coordinate axis x, y, and z respectively) are defined on cell faces connecting two grid cells as shown in Figure 4.

The continuous Navier-Stokes equations can be discretized using a finite difference

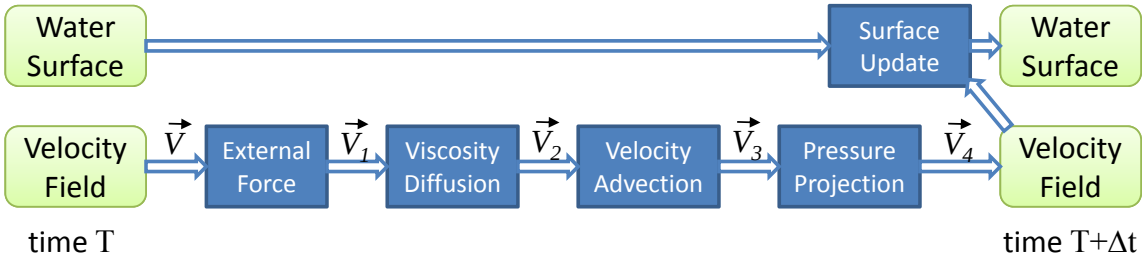


Figure 5: The pipeline.

scheme over the regular grid. For example, the gradient of a function ϕ can be calculated using central differencing scheme:

$$\begin{aligned}
\phi_x|_{x_0,y_0,z_0} &= (\phi(x_0+h,y_0,z_0) - \phi(x_0-h,y_0,z_0))/(2h) \\
\phi_y|_{x_0,y_0,z_0} &= (\phi(x_0,y_0+h,z_0) - \phi(x_0,y_0-h,z_0))/(2h) \\
\phi_z|_{x_0,y_0,z_0} &= (\phi(x_0,y_0,z_0+h) - \phi(x_0,y_0,z_0-h))/(2h)
\end{aligned} \tag{2}$$

in which h is the cell size. The Laplacian operator can be calculated as:

$$\begin{aligned}
\Delta\ \phi|_{x_0,y_0,z_0} = (\phi(x+h,y,z) + \phi(x-h,y,z) + \phi(x,y+h,z) + \phi(x,y-h,z) \\
+\phi(x,y,z+h) + \phi(x,y,z-h) - 6\phi(x,y,z))/h^2
\end{aligned} \tag{3}$$

2.3 Solving the Liquid Flow

By dividing the Navier-Stokes equations into a set of sub-problems using the *method of characteristics*, the liquid flow can be solved in four separate steps as shown in Figure 5. The first step adds external forces to the initial velocity field $\vec{V}_0$, including gravity and any user interaction forces. When an external force $\vec{F}$ is nearly constant such as gravity, a first-order forward Euler method is sufficient:

$$\begin{aligned}
\vec{V}' &= \vec{F}/\rho \\
\vec{V}_1 &= \vec{V}_0 + \Delta t \cdot \vec{F}/\rho
\end{aligned} \tag{4}$$

The diffusion step, corresponding to the viscosity term, damps the velocity field by diminishing velocity difference according to Equation 5:

$$\vec{V}' = \nu\nabla(\nabla\vec{V})/\rho = \nu\nabla^2\vec{V}/\rho \tag{5}$$

The first order forward Euler method can also be used if the viscosity coefficient ν is small:

$$\vec{V_2} = \vec{V_1} + \Delta t \nu \nabla^2 \vec{V_1}/\rho \tag{6}$$

If ν is large, however, an implicit Euler method is preferred to avoid stability issues by solving a sparse linear system with unknowns $\vec{V_2}$:

$$(\mathbf{I} - \frac{\Delta t \nu}{\rho} \nabla^2)\vec{V_2} = \vec{V_1} \tag{7}$$

in which $\mathbf{I}$ is the identity matrix.

The advection step means the velocity field should move forward with itself. A semi-Lagrangian method proposed in [96] solves the advection by tracing particles backward along the velocity field within a time interval Δt to obtain the old velocity:

$$\vec{V_3}(\vec{x}) = \vec{V_2}(p(\vec{x}, -\Delta t)) \tag{8}$$

where $p(\vec{x}, -\Delta t)$ is the location of the particle traced back along $\vec{V_2}$ from $\vec{x}$ within a time interval Δt.

The next step is used to maintain the fluid's incompressibility by projecting $\vec{V_3}$ back into the divergence-free space, also called *the projection method*. A scalar pressure field P is first introduced as an intermediate correction term:

$$\Delta P = \nabla \cdot \vec{V_3} \tag{9}$$

then the velocity field is corrected by the pressure gradient:

$$\vec{V_4} = \vec{V_3} - \nabla P \tag{10}$$

where P does not depend on Δt or ρ, assuming that ρ is constant. Equation 9 is a Poisson equation, which can be discretized as:

$$\begin{aligned}
&p(x_0 - h, y_0, z_0) + p(x_0, y_0 - h, z_0) + p(x_0, y_0, z_0 - h) \\
&p(x_0 + h, y_0, z_0) + p(x_0, y_0 + h, z_0) + p(x_0, y_0, z_0 + h) - 6p(x_0, y_0, z_0) = \\
&(u(x_0 + h/2, y_0, z_0) + v(x_0, y_0 + h/2, z_0) + w(x_0, y_0, z_0 + h/2) \\
&-u(x_0 - h/2, y_0, z_0) - v(x_0, y_0 - h/2, z_0) - w(x_0, y_0, z_0 - h/2))h
\end{aligned} \tag{11}$$

The pressure projection step is more computationally expensive than the other steps, and is commonly found to be the bottleneck of a whole liquid solver. Details about solving sparse linear systems including the Poisson equation can be found in many numerical algorithms literatures, such as [80]. After the pressure projection step, the velocity field becomes divergence-free and can be used to evolve the liquid surface.

2.4 Surface Evolution

Given the updated velocity field, we will discuss two common methods for evolving the water surfacesurface evolution when it is represented by a volumetric data structure: the level set method and the variational approach.

2.4.1 The level set method

In the level set method, a surface is defined as the zero level set of an implicit signed distance function, or called *the level set function*. The level set function measures the Euclidean distance from each grid cell to the liquid surface. A positive distance means the grid cell is outside of the surface; otherwise, the cell is inside. The liquid surface motion from the velocity field is then governed by the following partial differential equation [88, 78]:

$$\phi_t = -\vec{V} \cdot \nabla \phi \tag{12}$$

called *the level set equation*, which is an example of the Hamilton-Jacobi equation. Finite difference schemes can be used to estimate all the derivatives, including upwind differencing,

$$\phi_x|_{x_0,y_0,z_0} = \begin{cases} (\phi(x_0+h,y_0,z_0) - \phi(x_0,y_0,z_0))/h; & u(x_0,y_0,z_0) < 0 \\ (\phi(x_0,y_0,z_0) - \phi(x_0-h,y_0,z_0))/h; & u(x_0,y_0,z_0) \geq 0 \end{cases} \tag{13}$$

and Hamilton-Jacobi weighted ENO (HJ WENO) [77, 50], a preferable method in most of our experiments for better accuracy and volume preservation.

After surface motion, ϕ may no longer be a valid distance function even in the most continuous case, so the function needs to be re-initialized back to a signed distance

field. This can be formulated as a partial differential equation evolving over time till a steady state is reached:

$$\phi_t + |\nabla\phi| = 1 \tag{14}$$

At the steady state, $\phi_t = 0$ so the equation becomes $|\nabla\phi| = 1$. Since the interface should always be maintained, this equation will be solved separately for regions outside of the surface and regions inside of the surface, with boundary conditions at the interface. In practice, a fast algorithm can approximate this solution by propagating distance values from the surface in the surface normal direction only once, popularly referred to as the *fast marching method* [87, 106].

The first-order forward Euler method can be used to calculate the temporal integral for Equation 12. Higher-order explicit schemes, such as Runge-Kutta methods, can also be used in order to achieve more accuracy.

2.4.2 The particle level set method

Volume loss is a well known problem existing in the level set method, due to numerical dissipation in finite difference schemes. As a remedy, a hybrid method was proposed by Enright et al. [24] that uses particles together with a level set method.

In this method, volumeless particles are initialized around the interface and traced along the velocity field. They are then used to correct the signed distance function by enforcing its original distance to the interface. By doing this, the particle level set method can take benefits from an Eulerian method, while reduce volume loss thanks to a Lagrangian method. When the surface is under severe stretching or tearing, some surface regions may not have sufficient particles for good surface correction. In this case, particles will be reseeded to make sure each grid cell around the surface has at least a certain number of particles, 64 particles in [24] and 32 in our experiments.

2.4.3 The variational approach

each grid cell stores a density value ρ in a variational approach of Mullen et al. [71], evolving all of the iso-surfaces using an external velocity field by exchanging mass among grid cells. The density value ρ is limited in $[0, 1]$, with the surface defined as the iso-surface $\rho = 0.5$. The advection equation for the density field is defined as:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0 \tag{15}$$

By weakly assuming continuity in each grid cell C_i and applying the divergence theorem, we obtain:

$$\frac{\partial}{\partial t} \int_{C_i} \rho dV + \int_{\partial C_i} (\rho \vec{V} \cdot \vec{n}) dA = 0 \tag{16}$$

The first part of Equation 16 is an integral over the grid cell volume and the second part is an integral over the grid cell boundary. By spatial discretization, Equation 16 can be intuitively expressed as exchanging volumes between neighboring grid cells:

$$\frac{d\rho_i}{dt} = -\frac{1}{h^3} \sum_{j \in N(C_i)} f_{i \to j} \tag{17}$$

where $N(C_i)$ is the neighborhood of the grid cell C_i and $f_{i \to j}$ is the flux from grid cell C_i to C_j. The flux can be computed using an upwind piecewise-constant (PWC) approximation:

$$f_{i \to j} = \begin{cases} u_{i \to j} \rho_i, & u_{i \to j} > 0 \\ u_{i \to j} \rho_j, & u_{i \to j} \leq 0 \end{cases} \tag{18}$$

or any higher order upwinding schemes such as WENO-5 [51].

Compared with the level set method, the variational method does not have the volume loss problem since the mass is always preserved among grid cells. However, the density field becomes more and more diffused as time evolves, because velocity and density values are propagated further from the interface. A sharpening step is usually necessary to make the interface reasonably sharp by redistributing the density field. A mass re-injection step may further be used to explicitly maintain the volume after the sharpening step [71].

CHAPTER III

WATER DROPS ON SURFACES

3.1 Introduction

Surface tension and its effects on the interface between solids and water plays an important role in small scale fluid, such phenomena include water drops beading and flowing on a glass window, capillary action, and small water rivulets.

Real liquid that come into contact with a solid object forms a characteristic angle with the surface of the object known as the *contact angle* [19]. The contact angle for so-called *hydrophobic* surfaces causes water to bead up, while a *hydrophilic* surface allows a drop of water to spread out. We will use the term *affinity* to describe the hydrophobicity or hydrophilicity of a surface. The affinity between water and a surface affects not only the behavior of a static drop of water, but also greatly influences the motion of a moving drop. The fluid-solid interaction can also be seen to affect the behavior of drop merging and splitting and the motion of water rivulets.

The contribution of this chapter is the *virtual surface* method, which allows us to simulate small-scale behaviors of fluids with contact angle effects. Given a stable contact angle, this method estimates the appropriate surface tension at the contact line between the solid surface and liquid surface. It then implicitly constructs a virtual surface penetrating into the solid surface, replacing the original liquid-solid surface by the virtual surface, and estimates surface tension forces using this newly created surface. Unlike some other models that are focused on modeling axisymmetric water drops, this method can handle arbitrary 3D liquid shapes by using implicit signed distance functions to represent all surfaces. When the solid surface is sufficiently smooth, our virtual surface method accurately approximates the true surface tensions.

Water (or any other liquid) is effectively defined here by its viscosity and its surface tension against the air. The dominant factor that distinguishes the water motion between drops of the same size is the affinity between water and the solid material, and this can be quantified by the stable contact angle between the liquid-air and liquid-surface interfaces. In addition to the behavior for drops near equilibrium, we also treat the cases where the water is moving. We use a simple dynamic contact angle model for capillary solid coupling in terms of three contact angles: the receding contact angle, the wet advancing contact angle and the dry advancing contact angle. Our results indicate this model is sufficient for simulating many small-scale fluid motions.

Our fluid solver represents and updates the liquid surface using the particle level set method. Compared with other approaches such as molecular particle dynamics or adaptive Lagrangian meshing, the level set distance function can efficiently simulate a drop's internal fluid dynamics and can easily handle drop breakup and merging. Since the liquid volume only occupies a small portion of the whole domain in most small-scale liquid simulations, we use a sparse, piecewise representation of the grid in the fluid solver to save both computation time and memory.

3.2 Related Work

Previous work related to this problem can be found from three different areas: synthesizing water drop motion in graphics, computational fluid dynamics and its application in liquid simulations, and research on surface tension in physics.

In graphics, most previous water drop systems provide various ways to model water drops efficiently but remain incapable of capturing some of the physical drop motions that we observe in the real world. Dorsey et al. [20] used a particle system to synthesize drops and their effects on weathering appearance textures for large solid models, assuming that each drop's deformation is too small to be noticeable. Kaneda

et al. [54, 55, 53] used a particle system to simulate water drops flowing on a flat surface. Flowing water drops are modeled in Fournier et al. [32] by a mass-spring system with surface tension and volume conservation constraints. Though the mass-spring system allowed various efficient simulations, it has difficulty in handling the drop separating and merging processes, especially when many drops interact in a large scene. Yu et al. [120] successfully modeled static droplet shapes on flat surfaces using a metaball representation, and Tong et al. [102] later presented a volume-preserving approach to model water flows using metaballs, but neither considered the surface tension effects on the moving interface. Generally speaking, the above methods do not consider interaction between the fluid dynamics internal to the water drops and the surface tension at the liquid interfaces, making it relatively difficult to simulate a wide range of drop deformation and motion realistically and accurately.

Computational fluid dynamics has been successfully and practically applied to simulate fluid animation in graphics since Foster and Metaxas [31]. Shortly after that, the stable fluid method was introduced by Stam [96], in which the semi-Lagrangian method is used to handle liquid velocity advection. In a series of papers, Enright, Fedkiw and Foster [30, 24] used the level set method to evolve liquid surfaces so that more complex liquid motions can be simulated. They further showed how to combine the level set method with particles (the so-called *particle level set method*) to reduce volume loss and increase the surface accuracy. For a large viscosity, the time step must be extremely small according to the CFL condition when one solves the viscosity term using explicit schemes. Stam [96] showed the viscosity term can be solved with larger time steps using the implicit Euler method, assuming a uniform viscosity distribution. Recently Losasso et al. [62] demonstrated the use of an octree structure for surface evolution instead of a regular grid so that more surface details can be maintained. Surface tensions in [62] are used as first order Dirichlet pressure boundary conditions on air boundary cells by estimating mean curvatures from the surface's signed distance

function. A second order pressure boundary condition scheme was presented in [23].

In physics, chemistry and material science, researchers have performed numerous experiments to understand the liquid-solid interfacial tension and developed various simulation techniques for treating the liquid-solid interactions. Korlie [59] simulated a liquid drop on a flat solid surface using quasi-molecular particles. Feng et al. [29] studied the drop impact and flattening process using Lagrangian meshing with the finite element method. Bussman et al. [12] developed a volume tracking algorithm for the volume-of-fluid method, and they successfully simulated single drop splashing and impact on curved shapes in their later work, treating the contact angle as an immediate boundary condition. Healy [40] used the 2D level set method and enforced the contact angle by modifying the liquid-air surface immediately to simulate an axisymmetric drop impact on a flat surface. Zhao et al. [123] demonstrated drop falling and depositing effect using a variational level set evolution equation obtained by minimizing the surface tension energy. Sussman et al. [100] first proposed the virtual surface idea in 2D for flat solid surfaces to constrain contact angles under the level set framework. Renardy et al. [82] later implemented the same idea for the volume-of-fluid method, and their algorithm was also limited to flat solid surfaces in 2D. To our knowledge, there are no previously published methods to model 3D interfacial tensions for arbitrarily curved solid surfaces.

3.3 Physical Background

Surface tension (*interfacial tension*) is an important factor in small-scale liquid simulations. It is caused by unbalanced molecular cohesive forces in the interfacial region where two phases meet (liquid-air, liquid-solid or solid-air). There are two ways to analyze the surface tension's influence on the liquid motion. One is to use the surface

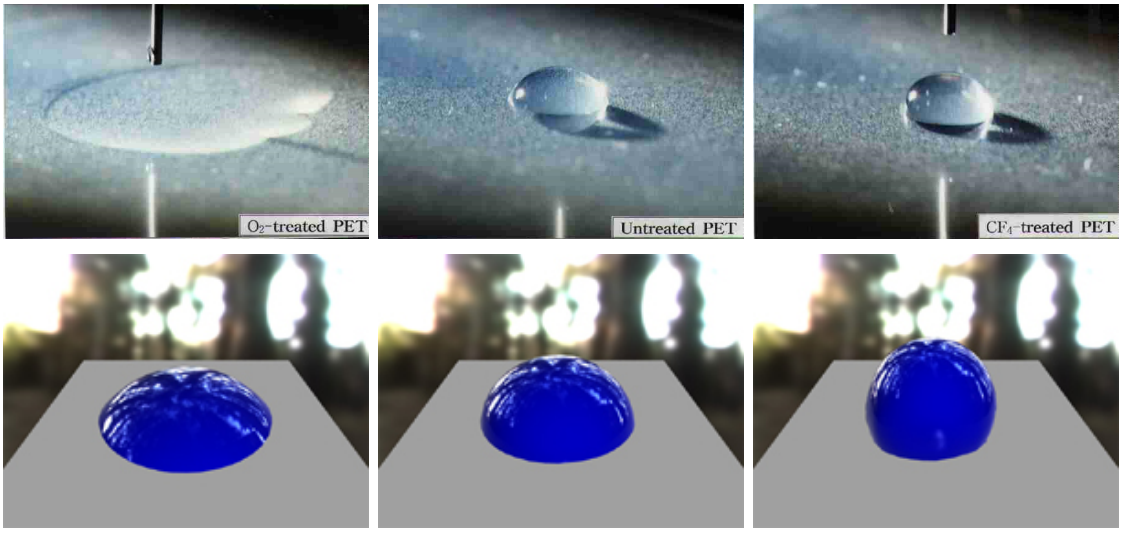


Figure 6: The top row shows the photographs of water drops in different shapes by treating the surface with different material. From left to right are: hydrophilic, normal and hydrophobic. The bottom row shows simulated stable drops sitting on the ground with different stable contact angles θ_s. Due to gravity, the actual angle between the drop surface and the ground is slightly different from θ_s.

tension force imposed onto the liquid surface directly in the incompressible Navier-Stokes equations (Equation 19),

$$\begin{aligned} \vec{V}' = -(\vec{V}\cdot\nabla)\vec{V} + \nu\nabla(\nabla\vec{V})/\rho - \nabla P/\rho + (\vec{F} - \gamma\kappa\cdot\vec{N})/\rho\,, \\ \nabla\cdot\vec{V} = 0\,, \end{aligned} \tag{19}$$

where $\vec{V}$ is the velocity field, ν is the viscosity coefficient, ρ is the liquid density, κ is the surface mean curvature, $\vec{N}$ is the liquid surface normal vector, $\vec{F}$ is the external force and γ is the surface tension coefficient. Surface tension can be similarly represented in terms of the pressure difference across the surface, according to Laplace's law,

$$\Delta P_{surf} = \gamma\cdot\kappa \tag{20}$$

where ΔP_{surf} is the pressure difference across the liquid surface. Both representations describe surface tension as being linearly dependent with respect to the surface mean curvature κ.

There are three different interfaces upon which surface tensions act on the contact front where a liquid surface meets a solid object: liquid-air, liquid-solid, and solid-air.

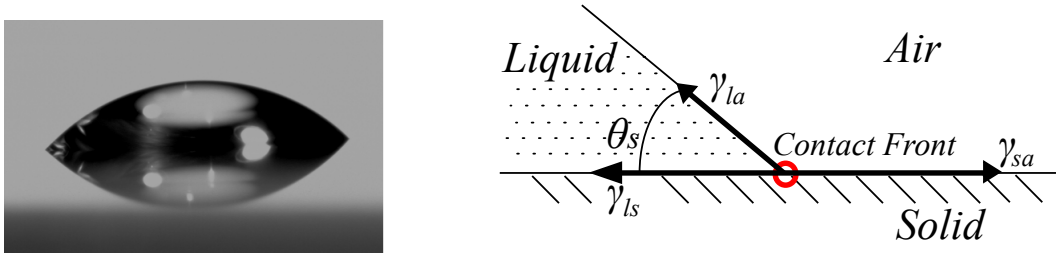


Figure 7: A photograph of a water drop in equilibrium and the illustration of its contact front.

According to Young's relation (Figure 7) [19], if the contact line at the intersection of these three interfaces reaches equilibrium with no external forces, the surface tensions will satisfy:

$$\gamma_{sa} - (\gamma_{la} \cos\theta_s + \gamma_{ls}) = 0 \tag{21}$$

where $\theta_s (0 < \theta_s < \pi)$ is the stable contact angle, and γ_{ls}, γ_{sa} and $\gamma = \gamma_{la}$ are interfacial tension coefficients for the liquid-solid, solid-air and liquid-air surfaces, respectively. Since it is difficult to measure surface tension directly, the stable contact angle is a common term used to quantify the affinity between a liquid and solid material. When θ_s is small (say, close to zero), the solid surface is said to be *hydrophilic*, and the liquid surface tends to spread flat. The solid surface is called *hydrophobic* if θ_s is large (close to π), and the liquid tends to bead up on the surface. The top row in Figure 6 shows the photographs of water drops in different shapes due to different solid surface property.

When external body forces act on the liquid, the actual equilibrium contact angle between the solid surface and liquid surface may slightly differ from θ_s. For example, the observed stable contact angle can be smaller than θ_s when a drop sits on a table, as the curvature at the contact line becomes slightly positive to hold the pressure due to gravity.

The characteristics of the liquid motion depend greatly on the liquid scale. Water moving in a large tank will behave in an entirely different manner than a water droplet that is flowing on a table. A number of different dimensionless numbers are

Table 1: Some symbols used in this chapter.

Symbols	Definition
γ	The surface tension
κ	The mean curvature $\nabla(\nabla\phi/\|\nabla\phi\|)$
ϕ	The signed distance function to some surface
Ω_s	The solid surface
Ω_l	The liquid surface with ϕ_l, or simply ϕ
Ω_{la}	The liquid air surface with ϕ_{la}
Ω_{ls}	The liquid solid surface with ϕ_{ls}
Ω_v	The virtual surface with ϕ_v
Ω_{new}	The new surface with ϕ_{new}

commonly used to characterize the relative scales of different forces. For instance, the Bond number is defined as the ratio between typical gravitational and surface tension forces, the Weber number describes the ratio between inertial and surface tension forces, and the Capillary number describes the ratio of viscous and surface tension forces. For our purposes here, it is sufficient to note that the capillary length of the liquid-air interface for water under gravitational acceleration is $\sqrt{\gamma/(\rho g)} \approx 4\,\text{mm}$. At scales orders of magnitude larger than this, surface tension effects are difficult to discern; but flows on the scales of a few capillary lengths (centimeters) typically have important surface tension and contact angle effects.

3.4 Virtual Surface Extrapolation

Our virtual surface approach makes use of the signed distance field ϕ that represents the liquid-air interface in order to simulate contact angle effects. The virtual surface can accurately capture the effects of each surface tension force on the contact line through a series of steps. The virtual surface is extended into the solid at the desired stable contact angle that would balance the surface tension forces on the contact line as in Young's relation (Equation 21). If the actual surface is not at the desired contact angle, the resulting kink has a non-zero curvature. The appearance of this curvature

in the pressure boundary condition of the projection step yields the desired forces.

Our method modifies the original liquid surface Ω_l around each contact front cell independently so that the curvatures calculated from the modified surface Ω_{new} correctly take all interfacial tensions into account. The full liquid surface Ω_l is the union of the liquid-air surface Ω_{la} and liquid-solid surface Ω_{ls}, with the full surface defined implicitly in the distance field ϕ. By modifying ϕ, we replace Ω_{ls} by a virtual surface Ω_v. We then estimate the mean curvature on this new surface and use the curvature as the surface tension pressure (Equation 20). Without the virtual surface method, estimating the surface tension is the same as using a stable contact angle π, which would greatly limit the range of phenomena that could be simulated.

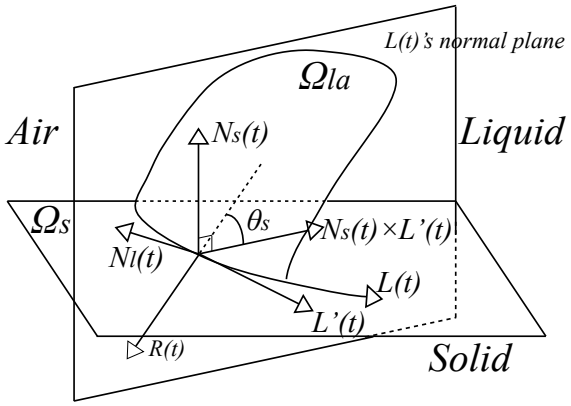
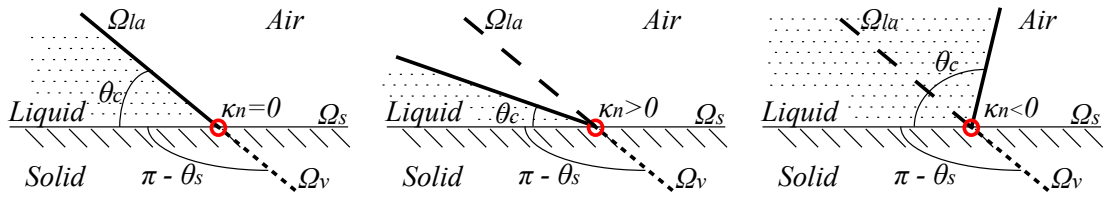


Figure 8: The solid surface and liquid air surface

We will now define the virtual surface. Let $L(t)$ be the curved contact line between the solid surface Ω_s and the liquid surface Ω_l (Figure 8). Let $N_s(t)$ be the surface normal of the solid and let $N_l(t)$ be the liquid surface normal on $L(t)$. The value t is a unit arc length parameter ($|L'(t)| = 1$), and t is chosen so that as t increases, the position $L(t)$ rotates counter-clockwise around the normal $N_s(t)$. By definition, $N_s(t)$ and $N_l(t)$ define a plane that is normal to the contact line $L(t)$. The angle between $N_s(t)$ and $N_l(t)$ defines the contact angle between Ω_s and Ω_l. Our virtual surface begins along $L(t)$ and extends down into the solid at a specific angle dictated by the



(a) Stable front: $\theta_c = \theta_s$ (b) Receding front: $\theta_c < \theta_s$ (c) Advancing front: $\theta_c > \theta_s$

Figure 9: The contact front in 2D.

solid-fluid affinity. Specifically, the virtual surface $V(s,t)$ is defined to be

$$\begin{aligned} V(s,t) = &\quad L(t) + sR(t) \quad (for \;\; s > 0)\,, \\ R(t) = &\quad -\sin\theta_s \cdot N_s(t) - \cos\theta_s \cdot (N_s(t) \times L'(t))\,. \end{aligned} \tag{22}$$

For a given t_0, $V(s,t_0)$ is geometrically a ray with an origin at $L(t_0)$ and that points in the direction $R(t_0)$, and this ray has an angle of $\frac{\pi}{2} + \theta_s$ with $N_s(t)$ in $L(t)$'s normal plane.

Let us examine the virtual surface in $L(t)$'s normal plane (Figure 9a). If the current angle θ_c equals the stable contact angle θ_s, then the contact line should be stable in the normal plane by definition. This is justified in the virtual surface method because the curvature κ_n in the normal plane is zero when N_s and N_l coincide. When θ_c is not equal to θ_s, using the mean curvature calculated by the virtual surface method provides a new way to estimate dynamic surface tensions on the contact line. Figure 9b shows the receding case ($\theta_c < \theta_s$) and Figure 9c shows the advancing case ($\theta_c > \theta_s$).

The input ϕ_l (or simply ϕ) to the virtual surface method is the original distance function of the liquid surface Ω_l, including Ω_{la} and Ω_{ls}. For open surfaces, such as Ω_{la} and Ω_{ls}, we define Ω in a region if for any point P, its closest point is not on Ω's boundary. We then define the full closed liquid surface through ϕ as the minimum defined value of ϕ_{la} (the distance to Ω_{la}) and ϕ_{ls} (the distance to Ω_{ls}). This signed distance will then be modified to create a new distance function ϕ_{new} in which Ω_{ls} is replaced by the virtual surface Ω_v. All surfaces are represented by signed distance

functions with no explicit formulations here.

In the remainder of this section, we first show how the virtual surface method works on contact front cells ($\phi = 0$) in 2D and 3D for a flat solid surface. We then describe how the method is naturally extended to general boundary cells and curved solid surfaces. We also assume the stable contact angle θ_s is unique in this section. In Section 3.5, we will discuss how to choose the contact angle according to different situations, including moving fluid surfaces and pre-wetted solid surfaces.

3.4.1 Modifying Surfaces in 2D

In 2D, the contact front is a single point, and the virtual surface is simply a ray extending from this contact point into the solid. We create the virtual surface by modifying the distance field values of ϕ to accurately reflect the distance between locations inside the solid and the closest point on the virtual surface. We then merge the virtual surface with the liquid-air surface, which is obtained from the original liquid surface.

For a given contact point, we locally modify the liquid surface in a small stencil of grid cells of ϕ, and these updated values will later be used to estimate curvatures. Our first-order curvature estimation scheme only requires the stencil size to be three nodes in each dimension. Our method can be easily extended, however, to handle larger stencil boxes when higher order curvatures are demanded.

Let C be a 3×3 stencil box centered at the contact front as shown in Figure 10. Without loss of generality, we assume the center $C_{0,0}$ is at the contact front ($\phi(0,0) = 0$), $C_{-1,0}$ is in water ($\phi(-1,0) < 0$) and $C_{1,0}$ is in air ($\phi(1,0) > 0$). In order to compute the new distance function ϕ_{new}, the first step is to calculate the virtual surface's distance function ϕ_v from each node on the Y=0 and Y=-1 planes inside of the solid. Let ψ be the distance to the contact point from each node on the Y=0 plane, then by definition we know $\psi(0,0) = 0$, $\psi(-1,0) = -h$ and $\psi(1,0) = h$ where

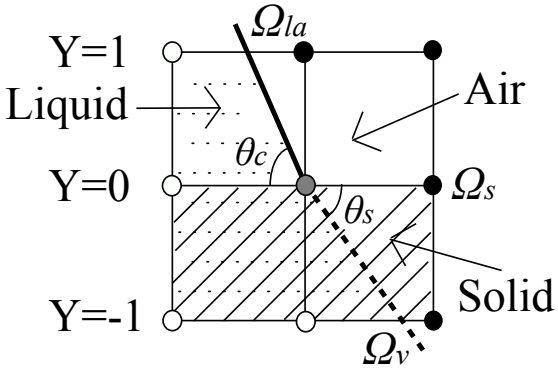


Figure 10: The 2D stencil box. The empty dots are water nodes, and the solid dots are air nodes. The Y=0 plane is the solid surface and the Y direction is the solid surface normal. The solid line is the liquid-air surface and the dashed line is the virtual surface.

h is the node size. For nodes on the Y=0 plane, ϕ_v is:

$$\phi_v(x,0) = \begin{cases} \psi(x,0)\sin\theta_s & x\cos\theta_s > 0 \\ \frac{x}{|x|}h & otherwise \end{cases} \tag{23}$$

For nodes on Y=-1 plane, ϕ_v is:

$$\phi_v(x,-1) = \begin{cases} [\psi(x,0) - h\cos\theta_s]\sin\theta_s & x\cos\theta_s + \sin\theta_s > 0 \\ \frac{x}{|x|}(\psi^2(x,0) + h^2)^{1/2} & otherwise \end{cases} \tag{24}$$

Given the virtual surface's distance function ϕ_v defined above for all nodes Y$\leq$ 0, we combine ϕ_v with the liquid-air distance function ϕ_{la} defined for all nodes Y$\geq$ 0 to form a new distance function ϕ_{new}. We first determine ϕ_{new} for Ω_{new}'s boundary nodes on each side, then estimate ϕ_{new} for the rest of the nodes using the fast marching algorithm from the boundary nodes. For ϕ_v's boundary nodes on the Y=-1 plane, $|\phi_v|$ is less than h, which means Ω_v is definitely closer than Ω_{la}. Therefore, for those nodes, the virtual surface's boundary nodes on the Y=-1 plane are also the new surface's boundary nodes, and $\phi_{new} = \phi_v$. Similarly, the liquid surface's boundary nodes on the Y=1 plane are also the new surface's boundary nodes: $\phi_{new} = \phi_{la} = \phi_l$, since the liquid-solid surface is beneath the solid surface Ω_s : Y=0. We finally determine the new surface's boundary nodes on the Y=0 plane. $C_{0,0}$ is definitely a boundary node

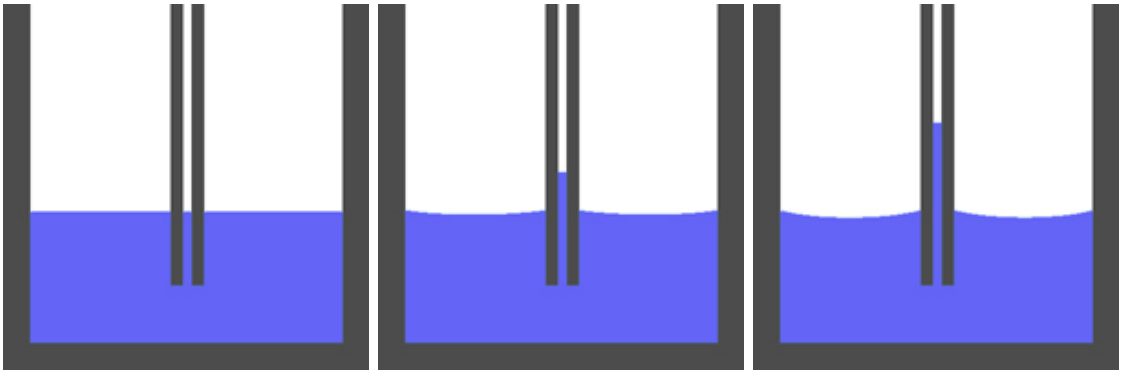


Figure 11: 2D capillary action. Solid surfaces are all hydrophilic.

for the new surface, with $\phi_{new} = 0$ by definition. For other two nodes, if they are air boundary nodes, we determine their values as:

$$\phi_{new} = \begin{cases} \phi_{la} \\ \phi_v \end{cases} = \begin{cases} \phi_l & |\phi_{la}| < |\phi_v| \\ \phi_v & otherwise \end{cases} \tag{25}$$

We can determine the values at air boundary nodes because the liquid-solid surface is a thin surface beneath the solid plane, so ϕ_{la} is still equal to ϕ_l for air-side nodes. Meanwhile, water boundary nodes are so close to the liquid-solid surface that their ϕ_l may in fact be equal to ϕ_{ls}.

ϕ_v and ϕ_l can be further used to fix any numerical errors that occur when estimating the new distances by the fast marching method. If a node is on the Y=0 or the Y=-1 plane, we bound its value ϕ_{new} by ϕ_v. If an air node is on the Y=0 or the Y=1 plane, we bound its value ϕ_{new} by ϕ_l. Again, we do not consider ϕ_l for any water nodes for the same reason described before: ϕ_l may not be equal to ϕ_{la}, but equal to ϕ_{ls}. Figure 11 shows a simulation of 2D capillary action using the 2D virtual surface method. The small contact angle causes a column of water to be drawn up into the thin tube. Also note the bending of the lower water surface.

3.4.2 Modifying Surfaces in 3D

The virtual surface method in 3D is similar to that in 2D: first calculate the distance function ϕ_v to the virtual surface, then merge it with the liquid-air surface by

calculating ϕ_{new} on the new surface's boundary nodes. The stencil box C is now a $3 \times 3 \times 3$ cube that is centered on the contact line ($\phi_l(0, 0, 0) = 0$), and the Y-axis is the constant solid normal direction.

The major difference for 3D is that the contact front in 3D is a curve on the solid surface plane, which causes new difficulty in determining the virtual surface's distance function ϕ_v. Fortunately we can show that if the contact line is sufficiently smooth and if the stable contact angle θ_s is not extreme (small $|\cos\theta_s|$), this 3D virtual surface's shortest distance problem can be reduced to the 2D ϕ_v's shortest distance problem in $L(t)$'s normal plane. The solution to this 2D case has already been given in Equation 23 and 24. We include the proof to a supporting claim in Appendix A for interested readers. According to the 2D solution, it is not necessary to know the exact position of the closest point $L(t_0)$ on $L(t)$. We do, however, need to know the shortest distance ψ from any node on the Y=0 plane to the contact line $L(t)$. Here we will show how to recover ψ from the original liquid surface's distance function ϕ_l.

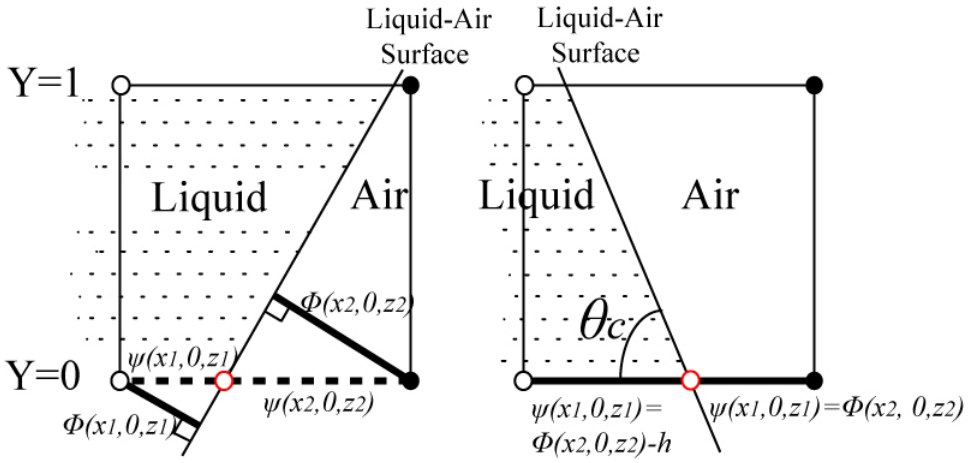


Figure 12: Two possible cases in estimating the distance ψ to the contact line.

Let $C_{x1,0,z1}$ and $C_{x2,0,z2}$ be two neighboring boundary nodes of the liquid surface such that $\phi_l(x1, 0, z1) < 0$ and $\phi_l(x2, 0, z2) > 0$. Also let $C_{x1,1,z1}$ and $C_{x2,1,z2}$ be two nodes above them on the Y=1 plane. There are two possible cases for estimating ψ

according to different liquid contact fronts (Figure 12). In the left case, when the current angle $\theta_c > \pi/2$, assuming that the liquid surface is sufficiently smooth, then $\frac{\psi(x1,0,z1)}{\psi(x2,0,z2)} = \frac{\phi_l(x1,0,z1)}{\phi_l(x2,0,z2)}$, and therefore,

$$\begin{aligned} \psi(x1,0,z1) &= \frac{\phi_l(x1,0,z1)}{|\phi_l(x1,0,z1)-\phi_l(x2,0,z2)|}h \\ \psi(x2,0,z2) &= \frac{\phi_l(x2,0,z2)}{|\phi_l(x1,0,z1)-\phi_l(x2,0,z2)|}h \end{aligned} \tag{26}$$

in which h is the node size. In the right case, when $\theta_c < \pi/2$, we have $\psi(x1,0,z1) = \phi_l(x2,0,z2) - h$ and $\psi(x2,0,z2) = \phi_l(x2,0,z2)$. We classify the boundary pair to be the left case if $\phi_l(x2,0,z2) > \phi_l(x2,1,z2)$; otherwise the right case applies.

For each boundary pair, ψ is calculated as given above. If one node is included in more than one boundary pair, the distance is chosen to be the one with the smallest absolute value. After determining ψ for all boundary nodes, we use the 2D fast marching method to estimate ψ for the rest of the nodes on the Y=0 plane.

Once we have calculated the virtual surface distance function ϕ_v, we determine the new surface's boundary nodes in a similar manner to the 2D case. Boundary nodes on the Y=$\pm$ 1 planes can be immediately determined by the virtual surface and liquid surface's boundary nodes. For air boundary nodes on the Y=0 plane, the new distance is chosen using Equation 25. The water boundary nodes are ignored for the same reason as before. After that, ϕ_{new} for non-boundary nodes are estimated using the 3D fast marching method, and they are further corrected using the known shortest distances to the virtual surface and to the liquid surface respectively, again similar to the 2D case. Figure 6 shows the shapes of drops with different stable contact angles simulated using a physically based fluid solver.

3.4.3 Virtual Surfaces on Curved Solid Shapes

So far we have described how to modify the surface near the contact front so that the surface tension can be estimated on a flat solid surface. It results in maintaining a characteristic stable contact angle. We will now discuss contact angles on curved

solid surfaces.

For each air boundary cell in a level set grid, surface tension is calculated as the boundary condition for the pressure projection step. Since those air boundary cells may not be exactly on the contact front, we cannot apply the virtual surface modification directly. One straightforward way to resolve this would be to first find the closest contact point to the air boundary cell, then use the surface tension estimated at this position as a Dirichlet boundary condition. Unfortunately, it is quite difficult to form a second order or higher boundary condition scheme for those closest points if they are not aligned with grid cells at all. Further, finding the closest contact point depends heavily on estimating the surface's signed distance gradient correctly, which is relatively unstable especially on highly curved surfaces.

Instead, we use a simple first order boundary condition scheme in our method. We choose the stencil box to be centered at the air boundary cell and the stencil node size to be the same as the grid cell size. We determine the stencil box's coordinate systems by surface normals: The stencil's Y axis is the solid normal direction $Y = N_s$ and the stencil's X axis is the orthogonalized liquid normal direction $X = N_l - (N_s \cdot N_f)N_s$. Then we sample the liquid distance function for each stencil node using trilinear interpolation, and subtract $\phi(0,0,0)$ from each stencil node's distance value. The actual liquid surface represented by the stencil box is the liquid surface's iso-contour where the distance equals $\phi_l(0,0,0)$. We modify this iso-surface represented in the stencil box and estimate the surface tension for that air boundary cell even if it is not immediately on the contact front. We then calculate the surface mean curvature for the center node $C_{0,0,0}$ in the stencil box by Equation 27 from [78]:

$$\begin{aligned}\kappa = (&\phi_x^2\phi_{yy} - 2\phi_x\phi_y\phi_{xy} + \phi_y^2\phi_{xx} + \phi_x^2\phi_{zz} - 2\phi_x\phi_z\phi_{xz} + \phi_z^2\phi_{xx} \\ &+\phi_y^2\phi_{zz} - 2\phi_y\phi_z\phi_{yz} + \phi_z^2\phi_{yy})/|\nabla\phi|^3\end{aligned} \tag{27}$$

where the first and second derivatives of ϕ are estimated using second order finite difference formulae.

Since the stencil's X direction is the orthogonalized liquid normal direction and we have $\phi_l(0,0,0) = 0$, node $C_{0,-1,0}$ should be inside of the liquid surface iso-contour. However, this may not be true in some cases, such as when the liquid normal directions have not been accurately estimated. In that case, we bound $\phi_l(0,-1,0)$ to be always less than some maximum value $\epsilon = -10^{-6}$ so that the virtual surface construction will not fail in case of missing the contact line when no nodes on the Y=0 plane satisfies $\phi_l < 0$.

Although the distance to the virtual surface that is calculated by this method is not exact, the value is still a good approximation to the actual shortest distance, given a sufficiently smooth solid surface or a small stencil size. Our experiments show that the virtual surface method can estimate surface tensions robustly and accurately.

3.5 Dynamic Contact Angle Model

In the real world, a unique stable contact angle is not sufficient to model fluid drop movements on solid surfaces. For instance, the phenomenon called *contact angle hysteresis* [84], where a tiny drop is suspended on a vertical plane, cannot be modeled with a single stable contact angle. This phenomenon requires a dynamic contact angle model, and can indeed be captured using only two stable contact angles: a receding (minimum) stable contact angle θ_s^r and an advancing (maximum) stable contact angle θ_s^a. Any angle between θ_s^r and θ_s^a can be a valid stable contact angle before the contact front starts to move.

Because it is sufficient to capture the effect of hysteresis, we use a simple dynamic model with two stable contact angles set by the contact front velocity in the liquid surface's normal direction. In the advancing case when the velocity is moving into previously dry regions, we use the advancing contact angle θ_s^a; otherwise, we use the receding contact angle θ_s^r. If the contact line is static (the velocity is below some threshold), we first calculate both boundary pressures P_r and P_a using θ_s^r and θ_s^a,

respectively. Since we assume $\theta_s^r \leq \theta_s^a$, $P_r \leq P_a$ and we then choose the actual pressure to be:

$$P = \begin{cases} 0, & if \quad P_r \cdot P_a < 0 \\ P_a, & if \quad P_a < 0 \\ P_r, & if \quad P_r > 0 \end{cases} \tag{28}$$

The actual values of the receding and advancing stable contact angles depend on the properties of both the liquid and the solid. The contact angles can also depend on the wetness of the solid surface. If the solid surface is already wet, liquid remaining on the surface can help subsequent drops move more freely on the surface. For such wettable solid surfaces, we maintain a *wetting history map* for the grid in order to indicate which regions have already been wetted. We then use a wetted advancing contact angle θ_{s-w}^a smaller than the dry advancing contact angle θ_{s-d}^a. We do not discriminate between the wet and the dry cases for the receding contact angle since the receding angle moves into a wetted region in both cases.

3.6 The Sparse Grid Representations

In order to simulate complex drop interactions, the grid domains that we use can become significantly larger than those in other liquid simulations. A typical grid domain in our experiments can contain $400 \times 400 \times 400$ grid cells. Fortunately, the liquid volume only occupies a small portion of the whole domain space. We use a sparse grid representation in which the domain is first subdivided into $8 \times 8 \times 8$ box regions. If the region contains any liquid or if it is close to the liquid surface, we activate this region and allocate memory for it. Otherwise, the region is inactive and no computation time or memory is used for the region.

Figure 13: The circled drop follows a previous drop's path.

3.7 Applications and Results

We have integrated the virtual surface method into our fluid solver and we have simulated several different small-scale liquid motion scenarios. Typically each simulation takes 5 to 8 days to simulate on one Pentium Xeon 2.8GHZ Workstation. Even though our algorithm works efficiently, our simulations are still relatively time consuming since the computational domain space is huge.

For the sake of completeness, we discuss the simulation parameters used for our simulations. For simplicity, we take constant time steps updating velocities every $2 \cdot 10^{-4}$ second. The liquid in each of our simulations here is taken to be water, as defined by its physical properties: the surface tension between the water-air interface is $\gamma = 73\,\mathrm{g/s^2}$ (at room temperature) and the viscosity is $\nu = 0.01\,\mathrm{cm^2/s}$. We apply no-slip conditions on the solid surface. The only external force used here is the acceleration due to gravity $g = 980\,\mathrm{cm/s^2}$. The typical drop size in our simulation is from 2mm to 6mm. We use a second order Runge-Kutta scheme to trace particles for both the semi-Lagrangian method in the velocity advection step and for the particle level set method. Our Poisson solver uses the preconditioned conjugate gradient method with a modified incomplete Cholesky decomposition preconditioner. Since

the water drop's velocity varies greatly and the volume loss is severe only on high-velocity surfaces, in the particle level set method, we choose the particle number for each grid cell according its velocity magnitude with a maximum of 32 particles per cell. Using the particle level set method dramatically reduces volume loss during simulations.

When the grid cell size is not sufficiently small, surface tension estimations for small drops are less accurate and may cause instability in drop motions. Fortunately, surface tensions on small drops can be ignored since their visual effects are hardly noticeable. In our experiment, we did not use surface tension if the water drop only contains 27 grid cells or less.

We have created several animated scenes based on our method of simulating fluid with interfacial tension. The first simulation considers flat window panes with varying surface properties, showing how the solid surface property and randomly added drops can influence the water drop's flowing paths as shown in Figure 14. In the beginning, water drops are identically distributed on each pane. The left pane has $\theta_s^a = 90^\circ$ and $\theta_s^r = 60^\circ$. The middle pane is more hydrophilic in the wake of the falling drop, with $\theta_s^a = 90^\circ$ and $\theta_s^r = 30^\circ$. In addition, we use a maximum receding surface tension bound to enhance the hysteresis effect. The right pane has similar contact angles as the left pane except that it also uses the wetting history and $\theta_{s-w}^a = 60^\circ$. Compared with those on the left pane, the drops on the middle pane leave longer trails because the receding contact angle is small and the receding surface tension is limited. On the right pane, the solid surface becomes wet after water drops flow on it, so that a water drop is likely to follow the previous drop's path (Figure 13). These three panes' affinities to water are similar to those of plastic, glass and marble, respectively.

The second and third simulations show a pipe and a bunny, with water dripping onto these surfaces from a height of roughly 0.1 meter. The surface of the pipe is represented analytically, and a distance field is used to represent the bunny. For

these solid surfaces, $\theta_s^a = 90°$ and $\theta_s^r = 60°$. The tube is tilted at an angle of 10° from horizontal. Notice the behavior of drops on the tube's bottom and the bunny's ears. Surface tension holds a drop from leaving the solid surface until enough water accumulates so that the drop becomes sufficiently large (Figure 15 and 16). Often when the drop leaves the surface, tiny satellite drops are formed when the thin connecting strand of water snaps.

The leaf in the fourth simulation is comprised of two planes spanning an angle of 120°, and the leaf axis is tilted at an angle of 15°. Figure 17 shows a sequence of a drop hitting the leaf and merging with its neighbor. Notice that drops flatten when they first hit the leaf, but then bead up due to the hydrophobic nature of the leaf. Also note the manner in which separate water drops flow to the middle of the leaf and join to form larger and longer rivulets (Figure19).

To generate the rendered images, we construct triangle meshes for the liquid surface using the marching cubes algorithm. Images were synthesized using our rendering program based on the physically-based ray tracer (pbrt) [79]. The environment maps are high dynamic range images from Paul Debevec's Light Probe Image Gallery. The leaf texture image is from Mayang's texture library (http://www.mayang.com/textures).

Figure 14: An example shows three window panels with different solid properties.

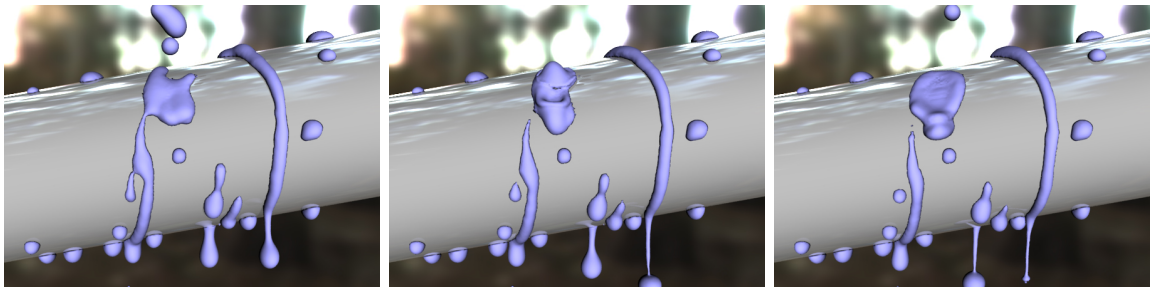


Figure 15: Water drops on a pipe.

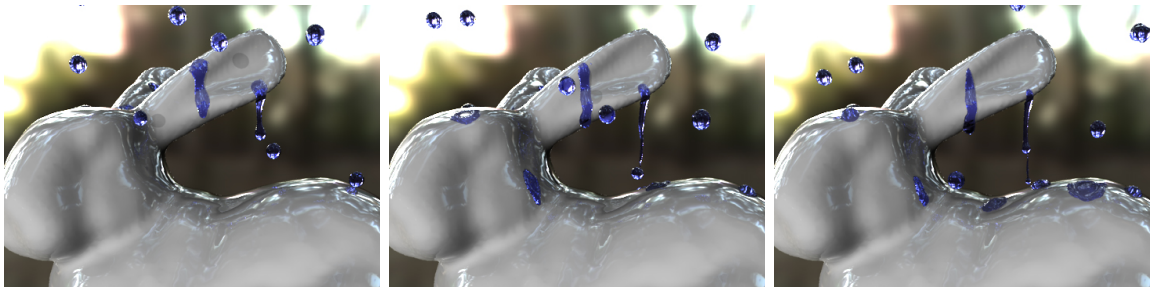


Figure 16: Water dripping off a bunny's ear.

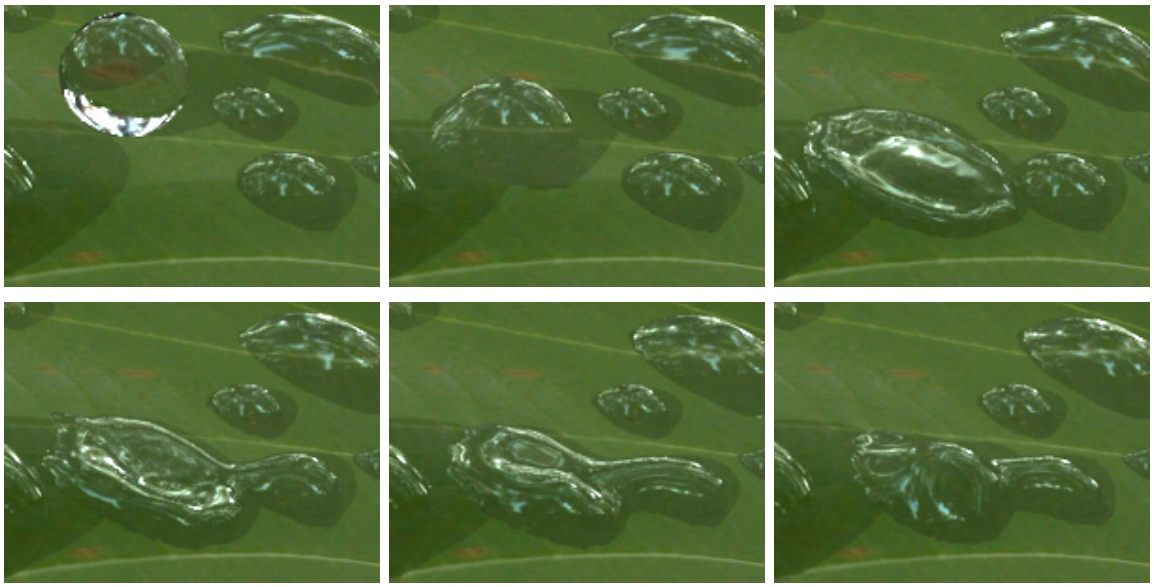


Figure 17: Drop impacts on a leaf. A flattened drop appears in the upper-right image. Time advances left to right, then top to down.

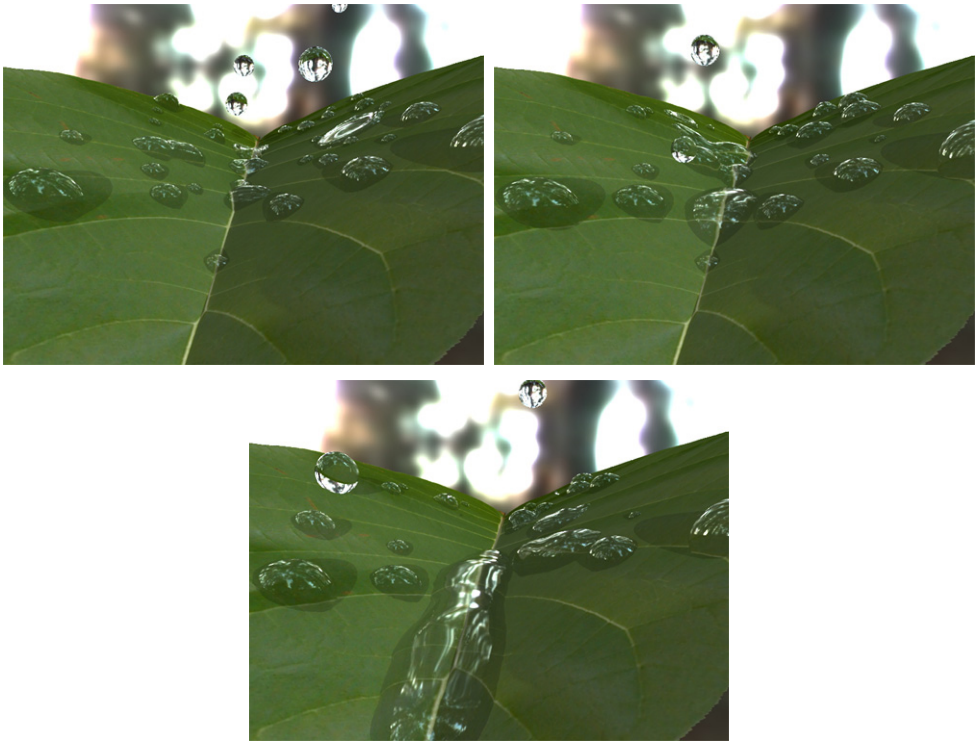


Figure 18: Drops on a leaf. These simulations show the formation of long rivulets.

Figure 19: Drops on a bunny. These simulations show the formation of long rivulets.

CHAPTER IV

GENERAL SHALLOW WAVE EQUATIONS

4.1 Introduction

Due to the complexity of the 3D Navier-Stokes equations, water behavior varies significantly under different circumstances. Researchers have invented many numerical methods to generate various realistic liquid effects by solving the Navier-Stokes equations. After our previous work successfully simulated small scale fluid behaviors, our main interest turned to making these simulations more efficient using a shallow wave assumption. This allows us to efficiently simulate examples such as toy boats floating in the bathtub and water drops streaming down a glass.

Compared with other representations such as particles, tetrahedral meshes or grid structures, a height field is more suitable to represent liquid under the shallow wave assumption. A height-field-based system is easier to implement and its computational space increases only quadratically with spatial resolution. Also, restricting the Navier-Stokes equations to 2D makes it possible to apply implicit numerical schemes, which often means more stability and higher efficiency. However, previous height-field-based techniques only supported a limited range of effects. In particular, surface tension was neglected, which diminished the accuracy of small-scale liquid simulations.

In order to remove these limitations, our first contribution is a general height-field-based system (Section 4.3) that solves the new General Shallow Wave Equations (GSWE), which are extended from the traditional shallow wave equations. This system builds height columns along surface normals rather than in the absolute gravity direction, as the left picture in Figure 20 shows. Forces in our system include gravity (Section 4.4) and surface tension (Section 4.5), and they are solved by implicit

schemes that are significantly more stable and efficient than explicit schemes. We further present a two-way liquid/rigid body coupling method in Section 4.6 for floating and drifting effects. Finally in Section 4.7, we demonstrate how this system can be optimized to run in real time on both CPUs and GPUs, which allows users to control and design fluid shapes interactively.

This system is most suitable for simulating low-speed fluid dynamics of small-scale incompressible 3D water flows on surfaces, which is expensive to generate with particle systems or grid systems. It can also provide an efficient way to predict other fluid behaviors before applying more accurate but time-consuming techniques. While each different system is capable of efficiently generating some specific effects, a comprehensive system that combines height fields with particles or grids as in [76, 49] should be able to capture more aspects of water animations in the future. We believe that the work presented in this chapter will be an important sub-component of such a system.

4.2 Related Work

Water animation is notoriously difficult and time-consuming to produce because solving the 3D Navier-Stokes equations requires a large computational domain. The computational domain increases cubically in 3D when water is represented either by particle systems [68, 72] or 3D volumetric systems [31, 96, 30, 24], not to mention the extra computational cost required by the CFL condition. As expected, a natural solution to improve simulation performance is to reduce the computational space. Instead of using cartesian grids in 3D, Losasso et al [62], Houston et al. [44] and Nielsen and Museth [75] proposed to use non-uniform grids such as octree structures or cells with various lengths. Feldman et al.[27], Klingner et al. [58] and Elcott et al. [22] considered solving fluid dynamics on unstructured tetrahedral meshes rather than grid structures. Treuille et al. [103] used Principal Component Analysis (PCA)

to further reduce a static computational domain into a principal component space. However, extending this method to handle free surface water animation is difficult.

Since the computational domain of a height-field-based system increases only quadratically with the spatial resolution, height fields were expected to be a more efficient representation after being introduced into the graphics community by the work of Kass and Miller [56] for rapid fluid simulations. This technique was later augmented with semi-Lagrangian velocity advection by Layton and van de Panne [60]. Instead of using shallow wave equations, Chen and da Vitoria Lobo [15] proposed to only solve pressure projection in 2D while keeping other simulation steps in 3D. Recently, Irving et al. [49] demonstrated how to combine a height field representation with a grid structure in order to simulate some non-height-field behaviors such as overturning and splashing.

In order to simulate 2D water flows on curved surfaces, Stam [97] proposed the use of quadrilateral meshes from Catmull-Clark subdivision. Flow simulation directly on manifold triangle meshes was demonstrated by Shi and Yu [91] and Elcott et al. [22] using discrete differential geometry (DDG) operators. Kim et al. [57] further adopted the BFECC method to reduce numerical dissipation in surface flows. Our method also simulates water flows on meshes directly, and we allow surfaces with arbitrary topology.

Considerable research has been done recently on grid systems and particle systems to model coupling between water and other objects, including rigid bodies, cloth and thin shells and other fluids. For height-field-based systems, one-way fluid/rigid body coupling methods were proposed by O'Brien and Hodgins [76] and Chen and da Vitoria Lobo [15] to simulate waves caused by moving objects. To our knowledge, however, two-way coupling algorithms for height fields have not yet been demonstrated.

Other topics related to this work include water drop simulation [32, 73], water wave simulation [42, 65], fluid surface tension [43, 16, 95, 112], and fluid control [104,

64, 25, 92].

4.3 General Shallow Wave Equations

Our goal in this section is to extend traditional Shallow Wave Equations (SWE) [56] to our new General Shallow Wave Equations (GSWE). As their names imply, SWE and GSWE are both based on the shallow wave assumption: the wave velocity is low and the wave height variation is small. The original shallow water equations can be derived from the Navier-Stokes equations according to the method of Saint-Venant [109]. However, the height field in SWE is built in the absolute gravity direction and the only external force is the gravity force in the horizontal direction. While a SWE system can be greatly simplified because of these, it suffers from the following limitations. Firstly, solid surfaces could not be too steep otherwise water drops will not be properly represented, as shown by the upper left picture in Figure 20. Secondly, it is not clear how to incorporate other forces, such as surface tension forces and user control forces, nor how to develop implicit schemes for arbitrary forces. Interaction between height-field-based water and the environment is also difficult to model in a physically-based manner.

The height field in GSWE is constructed along surface normals rather than in the absolute gravity direction as Figure 20 shows. In order to avoid self-intersection among water columns when the surface is fully detailed with small bumps, we use averaged surface normals from a low-resolution surface and represent the difference between the original surface and the low-resolution surface as a terrain height field $b(x)$ in the local background. The water height field is then defined as a function $h(x,t)$ of the surface position x and time t. Horizontal water velocity is $\vec{u}(x,t)$, and vertical velocity is implicitly given as $\partial h/\partial t$. Since our system allows arbitrary external forces, we first separate a 3D force $\vec{f}_{ext}$ into a 1D pressure component P_{ext}

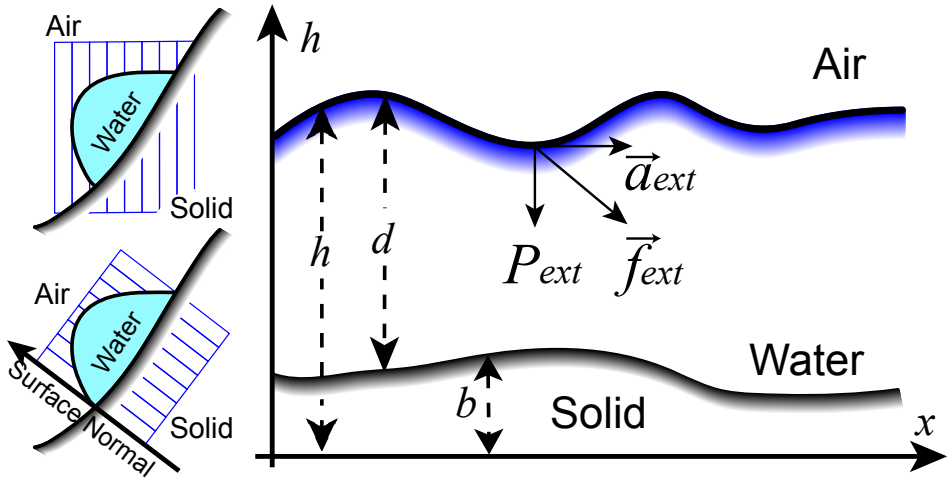


Figure 20: Left: The height field is built along surface normals rather than in the absolute gravity direction. Right: External forces are separated into pressure and acceleration components.

and a 2D acceleration component $\vec{a}_{ext}$ as the right picture in Figure 20 shows. Although these two components act on the water based on different mechanisms, they both take effect by changing the water's horizontal velocity $\vec{u}$. The pressure component, including air pressure, surface tension pressure and vertical gravity pressure, squeezes water and causes horizontal movement due to pressure difference. On the other hand, the acceleration component, including user control force and horizontal gravity acceleration, acts on the horizontal velocity immediately. By restricting the 3D Navier-Stokes equations to 2D surfaces, we can formulate the non-viscid General Shallow Wave Equations as follows:

$$\vec{u}_t = -(\vec{u} \cdot \nabla)\vec{u} - \nabla P_{ext}/\rho + \vec{a}_{ext} \tag{29}$$

$$h_t + \nabla \cdot (h - b)\vec{u} = 0 \tag{30}$$

in which ρ is the water density. Equation 29 updates the horizontal velocity due to both P_{ext} and $\vec{a}_{ext}$. Equation 30 updates the height field and maintains the incompressibility implicitly. Equation 30 can be reorganized into:

$$h_t + (h - b)\nabla \cdot \vec{u} + \vec{u} \cdot \nabla h = 0 \tag{31}$$

Since we are targeting at slow water, it is safe to ignore the velocity advection term $(\vec{u} \cdot \nabla)\vec{u}$ in Equation 29. According to method of characteristics, we separate the dynamics and update the height field twice, once according to the horizontal velocity field and once according to the height field itself (in the vertical direction only). The height field advection by the velocity field is first solved using an explicit solver:

$$h_t + \vec{u} \cdot \nabla h = 0 \tag{32}$$

The rest of Equation 29 and 31 are simplified by differentiating Equation 31 with respect to t and differentiating Equation 29 with respect to the spatial dimension. We then eliminate cross-derivatives:

$$\frac{\partial^2 h}{\partial t^2} = \frac{d\Delta P_{ext}}{\rho} - d\nabla \cdot (\vec{a}_{ext}) \tag{33}$$

When forces vary slowly through time such as user control forces, we treat them as if they are temporally invariant so that Equation 33 can be solved by explicit methods at each time step. However, two common natural forces, the gravity force and the surface tension force, are quite sensitive to height field changes as time evolves. If we insist on using explicit solvers in these cases, the system would require significantly smaller time steps according to the CFL condition in order to avoid instability. Therefore, we develop implicit schemes for both of these forces in Sections 4.4 and 4.5 respectively. They are combined and solved together in a single matrix system:

$$(\mathbf{A_g} + \mathbf{A_s} + \mathbf{I} - \mathbf{I_c})h^t = b - b_c \tag{34}$$

in which h^t is the unknown height field at time t, given all known height fields $h^{t-1}, h^{t-2}, ...$ previously calculated at time $t-1, t-2, ...$, respectively. Matrices $\mathbf{A_g}$ and $\mathbf{A_s}$ are formulated from implicit schemes for gravity and surface tension force. $\mathbf{I_c}$ and b_c are the coupling matrix and vector, which will be discussed in Section 4.6. b is a prospective height field augmented with artificial viscosity effects by a factor τ:

$$b = h^{t-1} + (1-\tau)(h^{t-1} - h^{t-2}) \tag{35}$$

After the height field has been solved, we finally update the velocity field using Equation 29 and apply some artificial surface friction by a damping factor $h(x)/(h(x)+d)$. When d is zero, there is no friction and when d goes larger, the friction effect becomes more obvious.

4.3.1 Spatial Discretization

Because the height field is built along solid surface normals, spatial discretization in our system depends on solid surfaces rather than absolute 3D space. Flat surfaces can be easily discretized into regular grids and then applied with finite difference schemes to formulate the GSWE matrix system.

For curved surfaces, we discretize the surface by a set of particles, instead of parameterizing the surface into quadrilateral grids. A particle repulsion algorithm is used over particles to approximate uniform distribution on the surface. A height column will then be built at each particle as Figure 21 shows.

In each iteration of the particle repulsion algorithm, six closest particles are found for each particle $\mathrm{X_i}$ and they become $\mathrm{X_i}$'s neighborhood $\mathrm{U_i} = \{\mathrm{X_{i_1}}, \mathrm{X_{i_2}}, ..., \mathrm{X_{i_6}}\}$. We use six neighbors because this is the average number of neighbors once the repulsion algorithm converges. We find that this approach produces similar results as other methods, such as repulsion of all points within a fixed radius. Next, we arrange all the neighbors in counter-clockwise order. The repulsion force applied on $\mathrm{X_i}$ is then calculated as:

$$\begin{aligned} f(\mathrm{X_i}) &= a \sum_j w_{i,j}(\mathrm{X_j} - \mathrm{X_i}) \\ w_{i,j} &= \cot \alpha_{i,j} + \cot \beta_{i,j} \end{aligned} \tag{36}$$

The force magnitude factor a is usually between 0.01 and 0.2 in our experiments. $w_{i,j}$ is a non-zero weight factor only when $\mathrm{X_j} \in \mathrm{U_i}$ as in [34], and $\alpha_{i,j}$ and $\beta_{i,j}$ are two angles facing toward the same edge $(\mathrm{X_i}, \mathrm{X_j})$ as Figure 21 shows. The advantage of calculating repulsion forces using Equation 36 instead of polynomials in [107] or Gaussian falloffs in [41] is to facilitate the surface tension scheme, as will be discussed

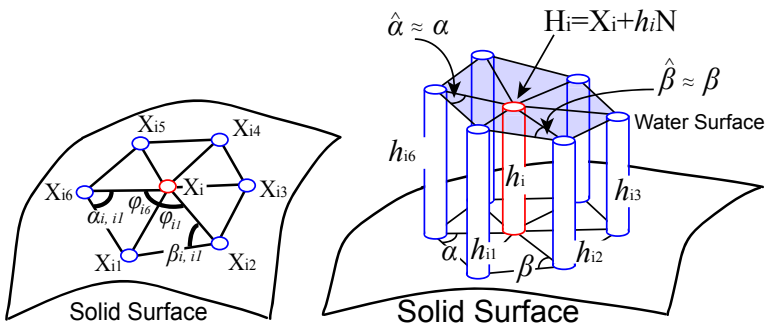


Figure 21: Particles and their height columns are constructed on a solid surface.

later in Section 4.5. In order to make $w_{i,j}$ symmetric, we add X_i into X_j's neighborhood once we know $X_j \in U_i$, even though X_i may not be necessarily in U_j. After

Figure 22: BUNNY: The left picture illustrates spatial discretization and the cell connectivity on the bunny model. Colored cells are water cells and black cells are boundary cells. The right picture shows a water drop changes its path after the bunny model was rotated 90°.

the repulsion process has converged, we construct the particle connectivity from the nearest neighbors and then we calculate the differential geometry operators directly from the neighborhood according to [34]. In this way, we do not need to reconstruct manifold triangle meshes as required in [91, 22].

4.3.2 Boundary Conditions

When simulating a large body of water with no dry areas, either Dirichlet boundary conditions or Neumann boundary conditions can be used on the boundary for different wave reflection effects.

When simulating water drops that are sparsely distributed on surfaces, we do not define the height field in dry regions in order to save memory and computational cost. These dry regions dynamically change over time and need to be updated as long as

water flows. Naively, any cell with non-positive height value should be treated as in the dry region. In practice, we exempt boundary cells (even if they have non-positive values) immediately neighboring to water cells from the dry region so that an accurate contact front can be continuously interpolated. The left picture in Figure 22 illustrates boundary cells in black on the bunny model. Boundary cells only exchange water with neighboring water cells when we solve GSWE. We apply boundary conditions between boundary cells and water cells, where the actual boundary is specified. Neumann conditions can prevent water from moving forward, while Dirichlet conditions allow water to move freely.

After the height field has been updated, we undefine boundary cells if they are no longer next to water cells. If a boundary cell becomes a water cell, its neighboring cells may become boundary cells and they will be updated. Boundary oscillation may occur if new boundary cells are not initialized properly with right height values. Similar to the fast marching algorithm [88, 78], our method initializes new boundary cells with C^1 continuity, which is sufficient in most cases as our experiment shows.

4.4 Implicit Gravity Scheme

Since the height field is built along surface normals rather than in the gravity direction, we separate the gravity into two components (vertical and horizontal) as discussed in Section 4.3:

$$\begin{aligned} P_g(\mathrm{X_i}) &= \rho g_i h_i \\ \vec{a}_g(\mathrm{X_i}) &= \vec{g} - g_i \mathrm{N_i} \end{aligned} \tag{37}$$

in which g_i is the vertical gravity acceleration at $\mathrm{X_i}$: $g_i = -\vec{g} \cdot \mathrm{N_i}$. We calculate $\vec{a}_g$ once the solid surface and $\mathrm{X_i}$'s location is fixed, because g_i at $\mathrm{X_i}$ is constant over time. On the other hand, P_g depends on the height field and requires an implicit scheme for more stability.

This implicit gravity scheme is extended from [56] by applying different gravity acceleration g_i at each position $\mathrm{X_i}$. For a 1D height field $h = \{h_1, h_2, ..., h_n\}$ from X_1

to X_n, treating P_g as P_{ext} in Equation 33 and differentiating h_i^t using finite difference schemes in the spatial domain gives an implicit matrix $\mathbf{A_g}$ for solving the matrix system (Equation 34):

$$\begin{aligned} a_{i,i} &= \tfrac{1}{4\Delta x^2}[(d_i + d_{i+1})(g_i + g_{i+1}) + (d_i + d_{i-1})(g_i + g_{i-1})] \\ a_{i,i+1} &= a_{i+1,i} = -\tfrac{1}{4\Delta x^2}(d_i + d_{i+1})(g_i + g_{i+1}) \end{aligned} \tag{38}$$

Similarly, the matrix for curved surfaces by the differential geometry scheme is:

$$\begin{aligned} a_{i,i} &= \tfrac{1}{8A}\sum_j w_{i,j}(d_i + d_j)(g_i + g_j) \\ a_{i,j} &= a_{j,i} = -\tfrac{1}{8A} w_{i,j}(d_i + d_j)(g_i + g_j) \end{aligned} \tag{39}$$

The particle surface area A is assumed to be uniform for all particles, after the solid surface has been uniformly sampled by the particle repulsion algorithm.

4.5 Implicit Surface Tension Scheme

According to Laplace's law, surface tension pressure P_{surf} is related to surface mean curvature κ, which can be estimated using the Laplace-Beltrami operator in [34]:

$$\begin{aligned} P_{surf}(\mathrm{H_i}) &= \gamma \cdot \kappa_i \\ \kappa_i &= \tfrac{1}{2A}\left|\sum_j \hat{w}_{i,j}(\mathrm{H_j} - \mathrm{H_i})\right| \\ \mathrm{H_i} &= \mathrm{X_i} + h_i \mathrm{N_i} \end{aligned} \tag{40}$$

in which γ is the surface tension coefficient, and $\mathrm{H_i}$ is the water surface extended along surface normal $\mathrm{N_i}$ at $\mathrm{X_i}$ as Figure 21 shows. We assume that $\mathrm{N_i}$ is locally constant after the solid surface has been densely sampled. We also assume that the weight factor $\hat{w}_{i,j}$ on water surfaces is close to the weight factor $w_{i,j}$ on solid surfaces under the shallow wave assumption. Since the repulsion force in Equation 36 goes to zero after sampling, Equation 40 is simplified to:

$$\begin{aligned} \kappa_i &= \tfrac{1}{2A}\Big(\Big|\sum_j w_{i,j}(\mathrm{X_j} - \mathrm{X_i}) + \sum_j w_{i,j}(h_j \mathrm{N_j} - h_i \mathrm{N_i})\Big| \\ &= \tfrac{|\mathrm{N_i}|}{2A}\sum_j w_{i,j}(h_j - h_i) = \tfrac{1}{2A}\sum_j w_{i,j}(h_j - h_i) \end{aligned} \tag{41}$$

The Laplace-Beltrami operator on a curved surface is a generalized version of the Laplacian on a flat surface. This operator is naturally normalized since the sum of all coefficients for a particular node X_i is zero from Equation 41.

One way to formulate an implicit scheme for surface tension from Equation 40 and 41 is to discretize h as unknowns completely. For flat surfaces, the Laplacian of the height field gives surface tension pressure, which is again processed by a Laplacian operator in Equation 33. The matrix $\mathbf{A_s}$ for flat surfaces will then have a cell kernel that takes the form:

$$\begin{pmatrix} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & 20 & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{pmatrix} \tag{42}$$

The more complicated matrix formula for curved surfaces is skipped here. Although these complete implicit schemes are numerically stable, these matrices are complicated, asymmetric and dense, because a Laplacian-of-Laplacian operator spans beyond the 1-ring neighborhood. Fortunately we notice from Equation 42 that cells beyond the 1-ring neighborhood have much less influence than those in the 1-ring neighborhood. Therefore, we propose an incomplete implicit scheme by only formulating implicit correlations within the 1-ring neighborhood after taking P_{surf}'s effect (Equation 40) into account (Equation 33):

$$\begin{aligned} a_{i,i} &= \tfrac{\gamma}{4\rho A^2} \sum_j w_{i,j}^2 (d_i + d_j) \\ a_{i,j} &= -\tfrac{\gamma}{4\rho A^2} w_{i,j}^2 (d_i + d_j) \end{aligned} \tag{43}$$

and redundant correlations b_s will be added to vector b as:

$$b_s(i) = -\frac{\gamma}{8\rho A^2}\left\{B_i^{t-1} + 2\sum_j w_{i,j}^2(d_i + d_j)(h_j^{t-1} - h_i^{t-1})\right\}$$
$$B_i^{t-1} = \sum_j w_{i,j}(d_i + d_j)(K_j^{t-1} - K_i^{t-1}) \tag{44}$$
$$K_i^{t-1} = \sum_j w_{i,j}(h_j^{t-1} - h_i^{t-1})$$

We implemented and tested both implicit schemes, and we did not notice any significant benefits from using the complete scheme. Since the incomplete scheme is more straightforward and efficient, we choose the incomplete surface tension scheme for all of the following experiments if we do not explicitly say otherwise.

4.5.1 Drops: Surface Tension and Contact Angles

There are two important factors that contribute to the shape and motion of water drops: surface tension forces at the air/water interface, and the effect of the hydrophilicity of the surface on which drops are forming. To account for surface tension forces at the air/water interface, we apply surface tension pressures to both the water cells and the boundary cells as discussed in Section 4.3.2. To account for hydrophilicity, we use the virtual surface method proposed in the previous chapter to produce water drops with various contact conditions. Such contact conditions allow us to simulate hydrophobic surfaces on which drops will bead up, and hydrophilic surfaces on which drops are flattened. Although the actual boundary (also called the contact front) is between boundary cells and water cells, our experiments showed that it is safe to simply assume boundary cells as exact contact fronts and apply boundary surface tension pressures only to them. Since surface tension is estimated as weighted difference as in Equation 41, the virtual surface is interpreted as height difference between boundary cell h_b and virtual surface cell h_v as Figure 23a shows:

$$\Delta h_v = h_b - h_v = \Delta x \tan\theta_c \tag{45}$$

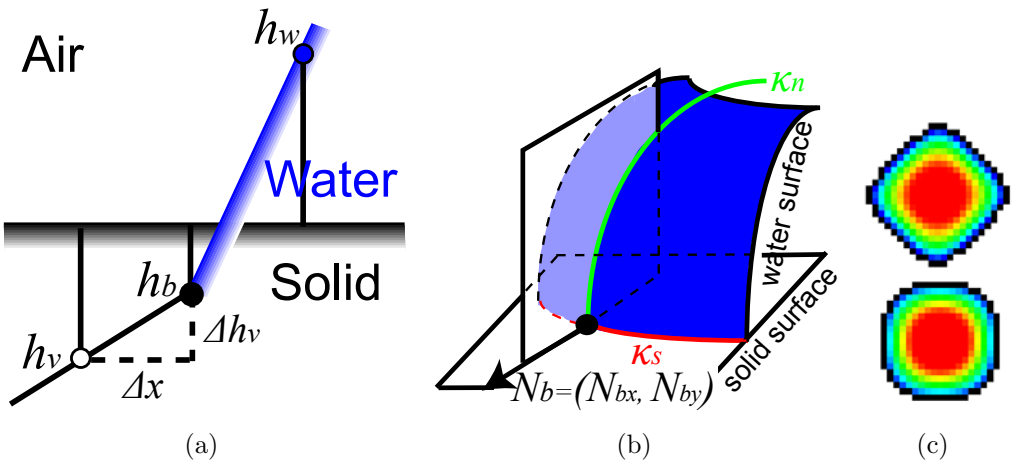
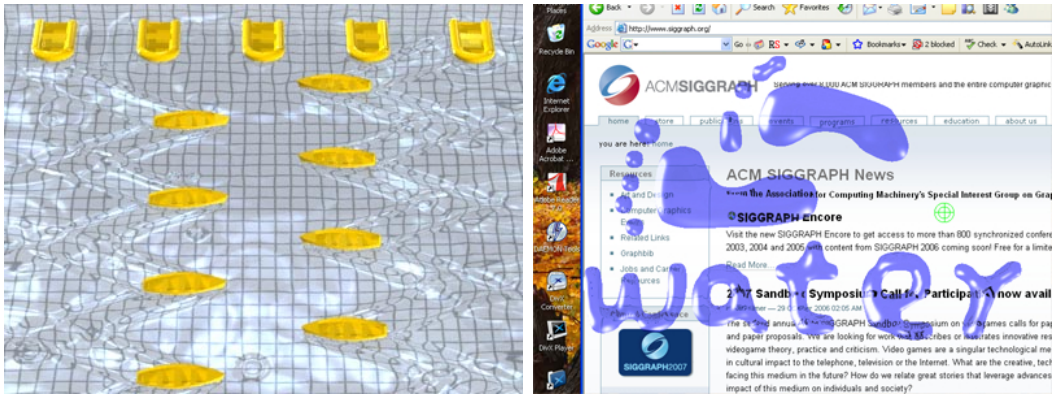


Figure 23: Illustrations of the virtual surface method. Part (a) gives the virtual surface in the 1D case. Part (b) shows the contact line normal N_b and two assumed principal components κ_n and κ_s. Part (c) shows water drop shapes without (top) and with (bottom) the angular factor α.

in which θ_c is the contact angle. For the 2D regular grid, we incorporate an angular factor α to count for the fact that the boundary normal is not aligned with the grid axis:

$$\Delta h_v = \alpha \Delta x \tan\theta_c$$
$$\alpha = \begin{cases} N_{bx} + N_{by}, & w_l + w_r + w_u + w_d = 1 \\ (w_l + w_r)N_{bx} + (w_u + w_d)N_{by}, & otherwise \end{cases} \tag{46}$$

in which w_l, w_r, w_u and w_d are 1 if left, right, up or down neighbor cells are in defined regions, otherwise 0. N_{bx} and N_{by} are the boundary's surface normal coordinates as shown in Figure 23b. Figure 23c shows that a water drop can correctly bead up into a circle by using the angular factor, otherwise it might become a diamond shape. On curved surfaces, since the connectivity between height columns could be arbitrary, instead of finding a proper angular factor, we sum the boundary mean curvature from two assumed principal components κ_n and κ_s as Figure 23b shows. The depth component κ_n in the normal plane modulates how steep the surface is, which can be estimated similarly to the 1D case. The angular component κ_s in the horizontal plane



(a) BOATS: Floating boats interact with water waves.

(b) INTERACTIVE: Fluid control by external accelerations from a user's cursor.

Figure 24: Examples.

measures the contact line curvature as:

$$\kappa_s = \sigma \sum_i \begin{cases} \varphi_i, & h_i - h_b > 0 \text{ and } h_{i+1} - h_b > 0 \\ \frac{(h_i - h_b)\varphi_i}{h_i - h_{i+1}}, & h_i - h_b > 0 \text{ and } h_{i+1} - h_b < 0 \\ \frac{(h_{i+1} - h_b)\varphi_i}{h_{i+1} - h_i}, & h_i - h_b < 0 \text{ and } h_{i+1} - h_b > 0 \end{cases} \tag{47}$$

in which σ is a scalar factor to modulate κ_s's magnitude, and φ_i is the angle between two neighboring edges as shown in Figure 21. We maintain a wetting history map from water paths and choose the contact angle based on the surface wetness.

4.6 Two-Way Fluid/Rigid Body Coupling

In this section, we describe our approach to coupling of fluid and rigid bodies. Our approach is straightforward, and it may be similar to methods that are incorporated in some games. Because we have not seen this method described in the literature, we include a complete description here. We assume the rigid object in our two-way water/rigid body coupling algorithm is small and floating on water, and our method cannot handle completely immersed objects so far. The coupling algorithm is separated into two steps: in Step 1, the height field is updated due to the effects of the rigid object; and in Step 2, the state of the rigid object is updated based on the effects of water pressure. Because the water surface is represented by a height field,

we assume that the bottom of each rigid object can also be approximated by a height field.

When a rigid object first contacts the height field in some cells, those cells no longer move freely. Instead, they strictly follow the rigid object by squeezing extra water into free neighboring cells. Therefore, we need to first recognize these constrained cells in Step 1. Given the height field h^{t-1}, h^{t-2} at time $t-1$ and $t-2$ respectively, and the height map of the rigid object's bottom s^t at time t, we can immediately recognize a constrained water cell i from $s_i^t < h_i^{t-1}$. However, $s_i^t \geq h_i^{t-1}$ is not a suitable criterion to determine when water cells lose contact with the rigid object, such as when an immersed object is bouncing away from water. A more sophisticated criterion will be given later after we discuss the Step 2.

Once the constrained water cells have been identified, their water heights are constrained to be the same as the solid heights: $h_i^t = s_i^t$. Since water cannot move freely in a constrained column, an unknown internal pressure $\tilde{P}$ will be formed to push redundant water into neighboring free columns. We interpret internal pressure as virtual gravity pressure and parameterize it using a virtual water height $\tilde{h_i}$ such that $\tilde{P_i} = \rho g_i \tilde{h_i}$. While h_i^t has already been known as s_i^t for a constrained cell, $\tilde{h_i}^t$ is still unknown. If we replace h_i^t by $\tilde{h_i}^t$ as an unknown for each constrained cell in the matrix system of Equation 33 and apply the implicit gravity scheme as usual, we can obtain exactly the same gravity matrix $\mathbf{A_g}$ as without coupling. The only difference is a partial identity coupling matrix $\mathbf{I_c}$ and a coupling vector b_c. Each of them is padded with zeros except for constrained cell i as:

$$\begin{aligned} \mathbf{I_c}(i,i) &= 1 \\ b_c(i) &= s_i^t \end{aligned} \tag{48}$$

In Step 2, we update the rigid object's velocity and position based on the effects of water pressure. According to Newton's law, the internal pressure at each constrained cell not only squeezes water, but also repels the rigid object at the same

time. Therefore, we use the internal pressure to estimate the floating force $\vec{f_i}$ on the solid object:

$$\vec{f_i} = (\rho g_i(\tilde{h}_i - h_i) - P_{surf}) \cdot \Delta area_i \cdot (-\vec{\mathcal{N}}_{\mathrm{i}}) \tag{49}$$

where P_{surf} is the surface tension pressure, $\Delta area_i$ is the rigid object area within the column, and $\vec{\mathcal{N}}_{\mathrm{i}}$ is the rigid surface normal. Since $\rho g_i(\tilde{h}_i - h_i) - P_{surf}$ in Equation 49 is supposed to be always positive for constrained cells, we use this as our actual criterion to recognize whether a column of water loses contact with the rigid object.

The state of each rigid object is then updated by integrating all floating forces according to standard rigid body dynamics. Interested readers are referred to SIGGRAPH course notes [116] for more details about rigid body equations of motion. After the rigid body states have been updated, we recalculate the rigid height map s^{t+1} and return to Step 1. We use the depth buffer in graphics hardware to accelerate the calculation of the rigid object's height map when the flat surface below the water is discretized by a regular grid. Note that all our examples of fluid/rigid body coupling were performed on regular grids. We think this method will also perform well for irregular grids if a suitable method for calculating the rigid object's height map is used.

4.7 Interactive Fluid Control

Since our system has little restriction on external forces, we can interactively control fluid shapes by specifying various external control forces. We provide two techniques to implement such external control forces $\vec{f_{ext}}$.

The first method uses external pressures calculated from distance maps to a target fluid shape. For example, the pressure field in the SCA example (Figure 5) uses the Euclidean distance. To put it another way, if we express the control pressure field as a terrain height field, control shapes are terrain valleys and water will flow from peaks to valleys as expected. Because distance maps are C^1 continuous in most places,

Table 2: Simulation statistics. When available, we list simulation speeds for both CPU (C) and GPU (G) solvers (rightmost column).

Name	Type	Resolution	Surf. Tension g (m/s^2)	Viscosity γ (N/m)	Speed τ (fps)
BUNNY	Surface	160,000	0.073	0.5	C: 11.0
GLASS	Surface	160,000	0.073	0.5	C: 4.1
POOL	Grid	400×400	0.073	0.3	C: 4.3
WINDOW	Grid	400×400	0.073	0.3	G: 9.2, C: 11.0
SPHERES	Coupling	400×400	0	0	G: 1.2, C: 0.4
DRIFT	Coupling	$1,200 \times 250$	0	0	G: 2.0, C: 0.8
BOATS	Coupling	400×400	0	0	G: 3.3, C: 1.3
INTERACT.	Accel.	400×400	0.073	0.3	C: 20
SCA	Pressure	400×400	0.073	0.3	G: 4.2, C: 3.0

those control forces are smooth and continuous in most places as well. This method is comparatively stable and robust. It runs faster when the target shape is static, since reconstructing the distance map (by a fast marching method in our practice) can be expensive.

Instead of constructing a pressure field, the second method modifies the fluid velocity directly by external accelerations. We create these accelerations from cursor motions, similar to some digital painting tools. We also clamp the magnitude of these synthetic accelerations in order to prevent instability caused by arbitrary user inputs. Figure 24b is a screenshot captured from our interactive shape design system. This method is relatively more efficient and more straightforward to implement.

4.8 Matrix Solver and Data Structures

We take advantage of the surface mesh connectivity to store the sparse GSWE matrices. Diagonal elements are stored at vertices, and off-diagonal elements are stored at mesh edges between each vertex and its 1-ring neighbors. We use directional edges when the matrix is asymmetric. We also create *hyper* edges that form connections between vertices and their 2-ring neighborhoods in order to store the matrix required

by the complete surface tension scheme.

To solve the symmetric positive definite matrix system defined by the implicit gravity scheme and the incomplete surface tension scheme, we use the preconditioned conjugate gradient method with either a Jacobi preconditioner or a modified incomplete Cholesky decomposition preconditioner. We also implemented the preconditioned Bi-Conjugate Gradient Stabilized method (Bi-CGSTAB) with modified incomplete LU decomposition to test the complete surface tension scheme.

The 2D regular grid matrix solver is implemented on both CPUs and GPUs (with a Jacobi preconditioner), as Table 4.7 shows. Details about GPU implementation can be found in [8]. The extension to a GPU matrix solver for curved surfaces should be straightforward, but remains as future work.

4.9 Results and Discussion

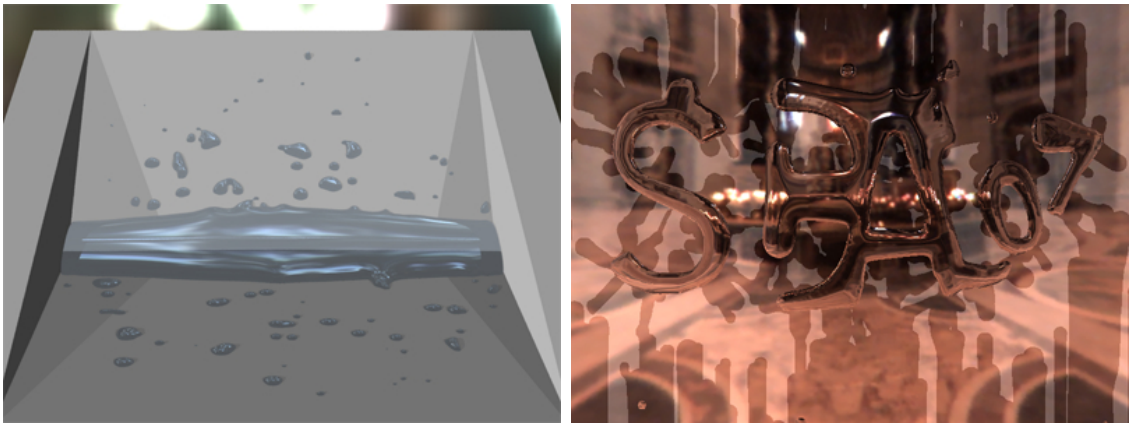
Our system was implemented using all the techniques described in the previous sections. We tested a variety of applications using this system on a Dell^{TM} XPS 700 PC with Intel Core2Extreme 2.92Ghz CPU and Dual Geforce 7950 graphics card. Test statistics are listed in Table 4.7. We use real-world units and parameters, including the water density ρ, which is $1.0 \times 10^3 kg/m^3$ at $3.98°C$. The frame rate for each example is measured in the worst case with maximum simulation burden. Table 4.7 shows that the GPU matrix solver runs faster on large bodies of water, such as the coupling examples. On the other hand, the CPU solver can better take advantage of sparse water, like the examples with water drops.

Our result shows multiple small scale water effects can be efficiently simulated using this general shallow wave model, including flowing water drops in the WINDOW and POOL example shown in Figure 27 and 25a, controlled water drops to formulate specific shapes show in Figure 24b and 25b, gravity-capillary waves in Figure 26, and solid-fluid coupling in Figure 28.

Compared with particle systems or grid systems that usually require several minutes to generate a single frame at similar resolution, most of our applications run at interactive rates and some of them run in real time. Compared with other height field systems, our system has a slightly higher computational cost for several reasons. Firstly, the overhead to handle a general system is small but not negligible. Secondly, more iterations are needed to solve the matrix system with a surface tension matrix $\mathbf{A_s}$, especially when the surface tension coefficient γ is large. The two-way coupling algorithm introduces more iterations too, since constrained cells cause the whole matrix to be less diagonally dominant. This is clearly shown in Table 4.7 from the SPHERES example and the DRIFT example. Although SPHERES uses lower resolution, the DRIFT example runs twice as fast since it involves less constrained cells than SPHERES.

Interestingly, when gravity is in the same direction as the surface normal, the gravity pressure component g_i in Equation 37 becomes negative, which causes a less diagonally dominant matrix. Eventually with sufficiently strong gravity, the matrix is failed to be positive definite and its solution becomes unpredictable. Since the influence of $\mathbf{A_g}$ on the matrix system depends on the water depth, ideally we can remove redundant water from the height field in order to keep the system solvable. This scenario corresponds to water drops dripping from the bottom of a solid surface, after the amount of accumulated water goes above some threshold.

The WAVE example shown in Figure 26 illustrates water waves due to a raised column of water. Our result matches with dispersion relations in that waves under both gravity and surface tension will disperse faster than waver under gravity only. It should be noted that shallow wave assumption implies that the wave length should be much larger than the water depth, which means the system theoretically cannot exactly represent Kelvin waves in deep ocean or capillary waves with wave lengths $< 1.7cm$. Fortunately, we noticed from our examples that when the water depth is close



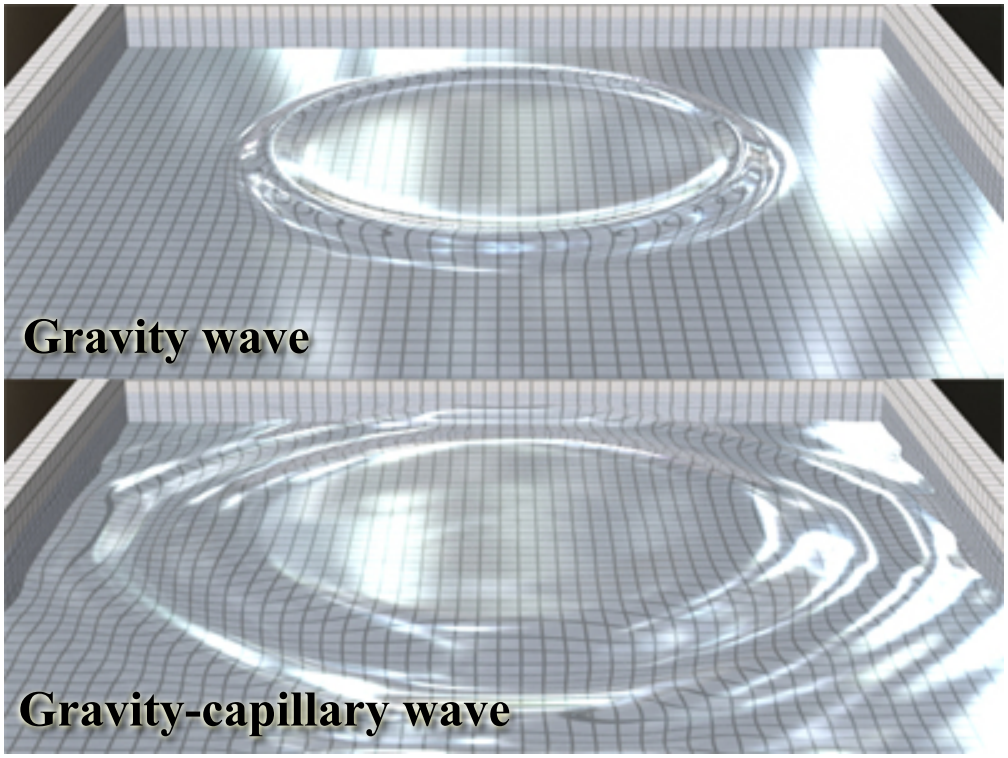
(a) POOL (b) SCA

Figure 25: Water drops are efficiently simulated using general shallow wave equations, and they can be controlled by user input to formulate specific target shapes.

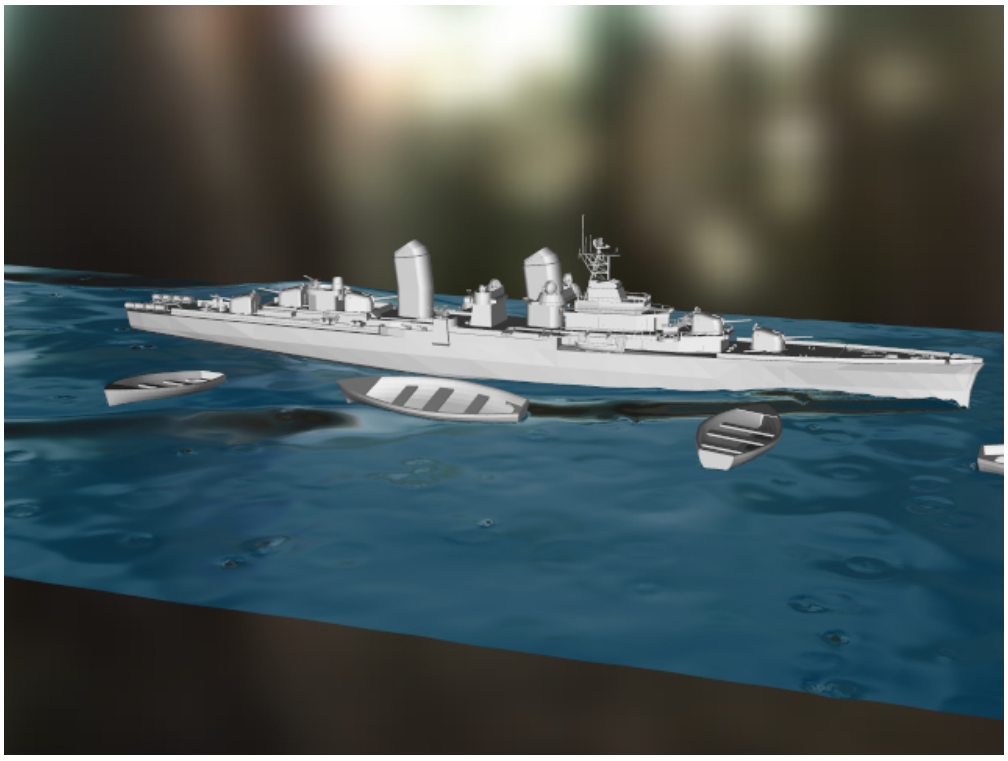
to the wave length, shallow waves still provides a reasonable and fast approximation to gravity-capillary waves for graphics applications, like ripples on a small pool.

One limitation of our GSWE approach is that water columns may intersect if the surface is concave. We eliminate high -frequency curved regions by using averaged normals and the background height field $b(x)$, but this cannot entirely remove all concavities. We have not come across problems from concavities in our examples, but this issue may arise for some geometric configurations.

We provided GPU rendering with single reflection and refraction during online simulation for all of our applications. Some examples were supported by offline GPU rendering with multi-sampling to remove aliasing artifacts. We also used an offline photorealistic ray tracer to produce high-quality videos for some examples.



(a) WAVE



(b) DRIFT

Figure 26: The difference between gravity wave and gravity-capillary wave is shown by the WAVE example. This is used to simulate a toy battleship sailing in a water tub (DRIFT).

Figure 27: WINDOW: Water drops are flowing on the window panel with a white waterproof boundary "water".

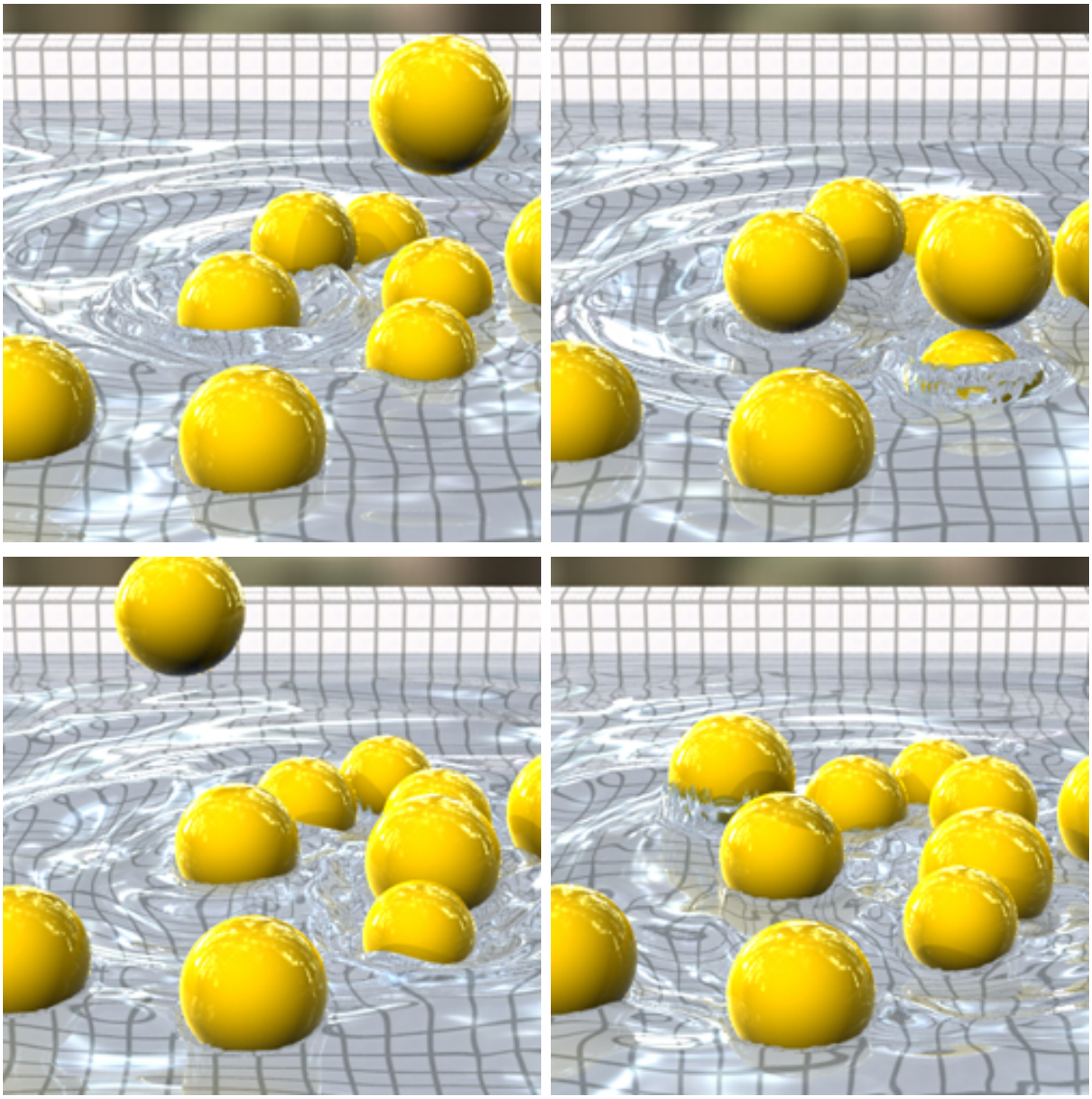


Figure 28: SPHERES: Spheres cause waves and splashes in the height field. They have different volumes but identical mass.

CHAPTER V

PHYSICALLY-GUIDED LIQUID MODELING FROM VIDEOS

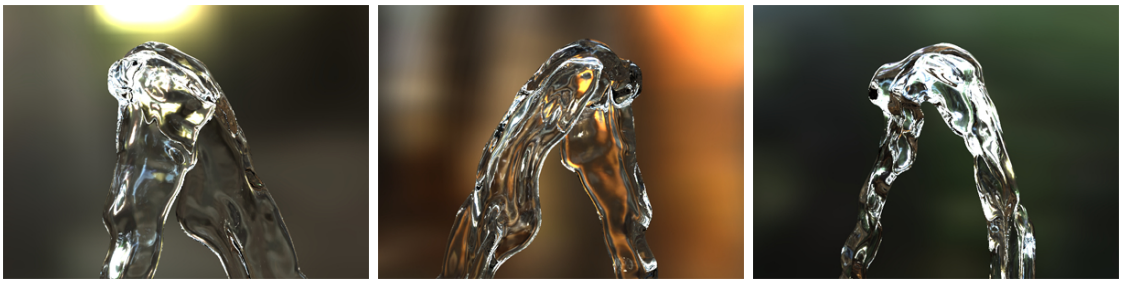


Figure 29: A synthetic rendering of a 3D model that was reconstructed from video of a fountain. These are three static views of the same instant in time.

5.1 Introduction

In recent years, modeling complex real world objects and scenes using cameras has been an active research topic in both graphics and vision. Exemplary work in this broad topic includes reconstructing flower models [81], tree models [101], hair [113], urban buildings [94, 118], human motion [124, 18], and cloth [114, 9]. The typical approach is to use one or more cameras to capture different views, from which the 3D shape information of the scene can be estimated by matching feature points. User interaction is often required to refine the initial 3D shape to create high-quality models. Missing from the list of objects that have been successfully reconstructed from video is water. Water's complex shape, frequent occlusions, and generally non-Lambertian appearance cause even the best feature matching methods to yield poor depth maps. Its dynamic nature and complex topological changes over time make human refinement too tedious for most applications.

In computer graphics, a common technique to produce water animation is physically-based fluid simulation, which is based on simulating fluid dynamics from the initial state of a fluid scene. While realistic water animation can be generated by various numerical simulation approaches, these methods can suffer from numerical errors that accumulate over time, including volume loss and loss of surface details. The computational cost is another issue in physically based fluid simulation, since the governing partial differential equations are expensive to solve and the time steps need to be sufficiently small to maintain stability and accuracy.

In this chapter we present the idea of combining physically-based simulation with image-based reconstruction to model dynamic water from video. That is, we adapt physically-based methods as a correction tool to refine the water surface that is initially generated from matching feature points. In order to enforce temporal coherence, we develop a 3D flow estimation method to approximate the velocity flow among reconstructed shapes in neighboring frames. The surface optimization method then removes redundant errors, applies physically based constraints such as volume preservation and spatio-temporal smoothness, and completes the shape sequence by filling in missing liquid regions. In this way, the final dynamic water model matches the fidelity of the real world and the results are physically sound, even though fluid dynamics may not be strictly enforced in certain cases. Since fluid dynamics is only used as a constraint rather than the target function to derive the entire surface sequence, this process is efficient and should be easy to accelerate using graphics hardware.

Incorporating the physical properties of fluid provides strong constraints on the possible water surface shape. This means the quality and coverage requirement for the initial 3D shape is significantly reduced. This allows us to generate plausible 3D water surface models even when the scene is observed by just one stereo camera, that is, when more than 50% of the surface is occluded. A single-depth-view solution is much easier to set up and use than the typical requirement of a surrounding array of

cameras.

To help with spatial feature matching, we add white paint to the water to avoid refraction, and we use a light projector to place a static random texture pattern onto the moving fluid surfaces. The equipment used for our capturing system is shown in Figure 31. It should be noted that our surface optimization scheme is not tied to any particular depth acquisition method, nor does it require appearance-based feature tracking over time, and this makes the use of active range sensing methods possible. The choice of using a stereo camera in the existing system is due to two considerations. First, since this technique can be potentially used with any capturing and reconstruction system, it is interesting to test its performance in a tough case when less surface information is provided. Second, one part of our ultimate goal along this line of research is to reconstruct large, outdoor fluid phenomena, in which case a hand-held stereo camera is much more practical than a surrounding multi-camera system.

We view this new approach for creating liquid models as an alternative to creating liquid animation through direct physically based simulation. As with other camera-based data capturing methods, our approach has the benefit of capturing the nuances and details in liquid that may be difficult to achieve using simulation alone. With a complete 3D model, the captured water shape can be re-rendered seamlessly with other graphics models. Though not demonstrated in this thesis, our framework should allow artists to design and modify a coarse initial shape in order to create stylized animations. Therefore we believe that our method may have applications in feature-film special effects and in video game content creation.

5.2 Related Work

In this section, we will briefly review previous efforts for generating water animations, including volumetric reconstruction methods and physically based simulation

approaches. Then we will discuss existing similar spatio-temporal reconstruction techniques that incorporate heuristics, such as local rigid motion and topological consistency. Since this proposed hybrid work is also related to 3D volumetric metamorphosis algorithms, we will compare these techniques and illustrate the role that physics plays in our hybrid method and its effects, which is not incorporated in traditional shape metamorphosis algorithms.

5.2.1 Fluid Modeling and Reconstruction

Although reconstructing and modeling fluid scenes from the real world is known to be a challenging problem, researchers have attempted to use various capturing equipment and techniques, and these efforts have been successful for certain fluid scenarios.

Using a specialized capturing system that includes a laser, a mirror galvanometer, a cylindrical lens and a high-speed camera, Hawkins et al. [38] demonstrated a technique for capturing time-varying volumetric data of participating media such as smoke by rapidly sweeping a laser sheet through the volume. At 5000 frames per second and 200 frames for each volume, this allows the volume to be scanned at 25 Hz in their work. Instead of scanning through the volume, Gu et al. [35] used a projector to cast a sparse set of structured light patterns over the volume and reconstructed the volume by decoding the integral measurements between the volume density and compressive light patterns. By simplifying participating media volume as a connected, semi-transparent surface, Hasinoff and Kutulakos [37] reconstructed dynamic semi-transparent scenes from a small set of simultaneous views (even only two views). Ihrke et al. [47, 48] and Trifonov et al. [105] used eight views and 72 - 360 views, respectively, for recovering flames and smoke, as well as transparent objects. Morris and Kutulaskos [70] and Hullin et al. [45] successfully reconstructed static transparent objects, such as a vase or a glass, by tracing light transport under structured scanning. A time-varying height-field surface can also be reconstructed by

searching refractive disparity as proposed by Morris and Kutulakos [69] if each light is refracted only once. Liquid thickness can be measured when liquid is dyed with fluorescent chemical as demonstrated by Ihrke et al. [46], in which case the liquid surface is calculated as a minimum solution to a photo-consistency-based error measure using the Euler-Lagrangian formulation. Atcheson et al. [4] used Schlieren tomography to capture fluids with time-varying refraction index values, such as heated smoke.

As far as we know, no existing volumetric reconstruction techniques considers this problem as a 4D spato-temporal problem. In other words, the fluid shape in each volume was independently reconstructed as a single 3D spatial problem. In fact, the use of our hybrid method proposed in this chapter is not limited to any particular capturing or reconstruction setup. This method can be used to handle spatio-temporal data generated from other reconstruction techniques, assuming that the underlying physical laws are known. It is capable of automatically removing inconsistency and noise, which often exist in reconstructed data.

5.2.2 Physically Based Fluid Simulation

Numerous researchers have used physically based numerical simulation to animate fluids. Using incompressible fluid dynamics as the governing equations and formulating fluid animation as a numerical simulation problem, Foster and Metaxas [31] generated realistic water animations using computational fluid dynamics techniques. Shortly after that, Stam [96] proposed the stable fluid method that uses a semi-Lagrangian method for handling fluid velocity advection. In a series of papers, Enright, Fedkiw and Foster [30, 24] used the level set method and particles to evolve liquid surfaces for more complex liquid motions. Losasso et al. [62] demonstrated the use of an octree structure for speed and memory efficiency. In addition to volumetric representations, water animation can also be simulated using particle systems [68, 72] or tetrahedral meshes. Klingner et al. [58] and Bargteil et al. [6] used tetrahedra meshes and a finite

element method to simulate water, so more surface details can be maintained.

When solving fluid dynamics as an initial boundary value problem, numerical errors, including volume loss and detail loss, are often found in physically based fluid simulation. Another problem in this area is how to match user-specified fluid shapes at particular time instants. McNamara et al. [64] and Fattal and Lischinski [25] studied how to constrain fluid simulation by creating artificial control forces. This allowed them to generate target-driven animation effects similar to ours, through optimization over physically based simulation. In contrast to their work, we only use physics as a heuristic constraint to improve spatio-temporal volumetric data, rather than governing equations in numerical simulation. This makes our system more efficient than physically based fluid simulation, and it avoids large error accumulation, which is a difficult problem in physically based fluid simulation.

5.2.3 Spatial-Temporal Reconstruction

Although static surface reconstruction has been an active research problem in both graphics and vision for decades, reconstructing dynamic scenes in a spatio-temporal manner has not been well studied until recently. The greatest challenge in 4D space-time reconstruction is how to introduce extra shape information over time to correct errors and complete occluded regions in initial reconstruction results. Additional shape information can either be provided through comparing feature correspondences between frames [63, 36, 114, 9], or between each frame and a surface prior [1, 2, 122, 3]. Under the assumption that the surface topology does not change over time, these techniques have successfully modeled cloth animations, human faces and body motions. Unfortunately, this fixed-topology assumption is not valid for modeling liquid, since the topology of a liquid surface changes frequently and dramatically in most liquid animations.

Assuming that the deformation can be approximated locally in space and time as

linear rigid motion under dense spatio-temporal sampling, Wand et al. [110] and Mitra et al. [66] formulated the reconstruction problem as a 4D hypersurface optimization problem. Sharf et al. [89] extended this idea to surfaces represented by volumetric data and incorporated other constraints including both spatio-temporal continuity and incompressibility, as our method does. However, they required sufficient temporal sampling to make sure that the surface moves less than one grid cell in each time step, and they cannot handle more challenging situations when the input surface sequence contains outliers and many missing parts. In our system, we do not put any limits on the capturing rate or the grid resolution, so the liquid surface can move significantly faster in each time step (up to roughly 10 grid cells).

5.2.4 Volumetric Metamorphosis

Given two shapes that are represented by volumetric data, a volumetric metamorphosis algorithm tries to find a smooth transition from one shape to another in a volumetric form. A number of existing volumetric metamorphosis algorithms [10, 61] are 3D extensions of 2D image metamorphosis algorithms and they require user specified feature correspondences as input. Each feature correspondence, such as a point or a rectangular block, determines local deformation from one shape to another. The overall deformation is then linearly interpolated from local deformation based on distance weights. When two shapes are well aligned, intermediate shapes can be interpolated by a variational approach [108] even without feature correspondences. This variational interpolation uses a generalization of thin-plate interpolation and this simply yields a single implicit function for surface reconstruction from contours. Volumetric metamorphosis can also be done by linear interpolation in the frequency domain [39], however it is impossible to recover the spatial flow nor further apply any physically based constraints from there.

How to automatically find feature correspondences is an important question in

the shape metamorphosis community. Various methods, including ball map [115], ortho map [14], and an implicit approach [17], can be used to find correspondences in the XOR region between two registered shapes. Without registration, the method proposed by Sederberg and Greenwood [85] finds correspondences that minimize the bending energy for shape morphing, but they can only handle closed loops in 2D. Zhang et al. [121] further extended this idea to handle meshes in 3D.

As far as we know, there is no other morphing algorithm that is easy to combine with physically based constraints, including spatio-temporal continuity and incompressibility, especially when a long sequence of shapes are to be used for interpolation. Figure 41 shows the difference between the result of an ideal shape morphing algorithm by minimizing the deformation energy and the result of our surface optimization method in a rotational example. Our method without physically based constraints can also be considered as a shape morphing algorithm with automatic feature matching.

5.2.5 Other related Techniques

Researchers in other domains have used various techniques to capture the behavior of fluids. The fluid imaging community regularly makes use of the Particle Imaging Velocimetry (PIV) method to capture flow fields from the real world by tracking macroscopic particles mixed in the fluid. This method creates fluid velocity values in the *interior* of bodies of water, but the approach cannot be used to reconstruct the geometry of the water surface due to the difficulty in maintaining the distribution of those particles in fluid. More details of the PIV method can be found in [33].

Bhat et al. [7] studied temporal continuity in liquid videos by tracing textured 2D particles over video sequence, in order to synthesize new liquid videos. This work was done completely in the image domain and no 3D models were reconstructed.

To obtain a complete space+time model from dynamic scenes, a camera array

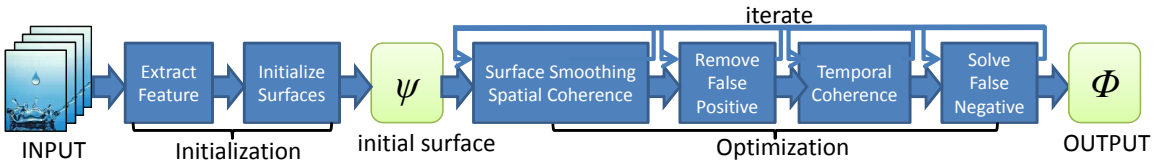


Figure 30: The entire fluid modeling system pipeline.

system is usually deployed to capture objects from different views (e.g., [52, 93, 125]). Recently, using a sparse camera array has become an active research topic (e.g., [111, 67, 89]. We push this trend to the limit by using only one pair of cameras with a narrow baseline. We show that by incorporating physically based constraints, the amount of input data needed for complete 4D modeling can be dramatically reduced.

5.3 Overview of the Fluid Reconstruction Method

Given a video sequence captured by a synchronized, calibrated stereo camera system, the goal of our hybrid water modeling technique is to efficiently reconstruct realistic 3D fluid animation that is physically plausible. Our framework consists of two stages as shown in Figure 30. Let $\{I_t\}$ and $\{J_t\}$ be video sequences captured by stereo camera at time $t \in [0, T]$. In the first stage, an initial shape sequence $\Psi = \{\psi_t\}$ is assembled by reconstructing each frame independently using depth from stereo. Ψ is then optimized in the second stage to generate a shape sequence $\Phi = \{\phi_t\}$ that satisfies spatio-temporal coherence.

Spatial coherence means that Φ should still be similar to the captured data input Ψ, while temporal coherence means that Φ should satisfy the behavior of fluid as much as possible. Mathematically, this is interpreted as the minimum solution to the following energy functional:

$$\mathbb{E}(\Phi) = \sum_{t=0}^{T} (\mathbb{E}_d(\phi_t, \psi_t) + \mathbb{E}_s(\phi_t)) + \sum_{t=0}^{T-1} \mathbb{E}_n(\phi_t, \phi_{t+1}) \tag{50}$$

Here $\mathbb{E}_d(\phi_t, \psi_t)$ calculates the similarity between ϕ_t and ψ_t, and $\mathbb{E}_n(\phi_t, \phi_{t+1})$ measures

how closely Φ satisfies physically based constraints in local. Besides spatio-temporal coherence, a spatial smoothness term $\mathbb{E}_s$ is also introduced to provide surface smoothness, which is a common observation in real fluid scenes due to surface tension. If we treat the solution to Equation 50 as a smooth hyper-surface in 4D, the combination of the spatial smoothness term $\mathbb{E}_s$ and the temporal smoothness term $\mathbb{E}_n$ can then be considered as spatio-temporal smoothness in 4D. To solve Equation 50, each energy will be treated as an independent sub-problem and solved separately in multiple iterations to obtain a final water animation that reaches a minimum of the sum of all energies (as illustrated in Figure 30). Our experiment indicates that five optimization iterations are sufficient for most cases.

We choose implicit signed distance functions to represent the 3D liquid surface geometry for two reasons. First, signed distance functions are neutral to frequent topological changes that occur in the evolution of liquid surfaces. Second, the 3D flow estimation method can easily take advantage of this representation for comparing similarity between iso-surfaces (Section 5.6.1.1). Our signed distance functions are discretely defined on a regular grid for simplicity.

5.4 Surface Initialization

We use a single pair of calibrated cameras for scene acquisition. This consists of two synchronized high-speed greyscale Dragonfly Express cameras. The water is contained in a small round tub, and the water is dyed with white paint, making it opaque to allow light patterns to be projected onto its surface. A LCD projector is positioned directly behind the cameras and it projects a static random pattern to provide artificial spatial features on the water surface. The projector position is chosen to minimize self shadows, which would cause spatial features to be lost. We typically capture the scene at a resolution of 640×480 at 200fps. The equipment setup is shown in Figure 31.

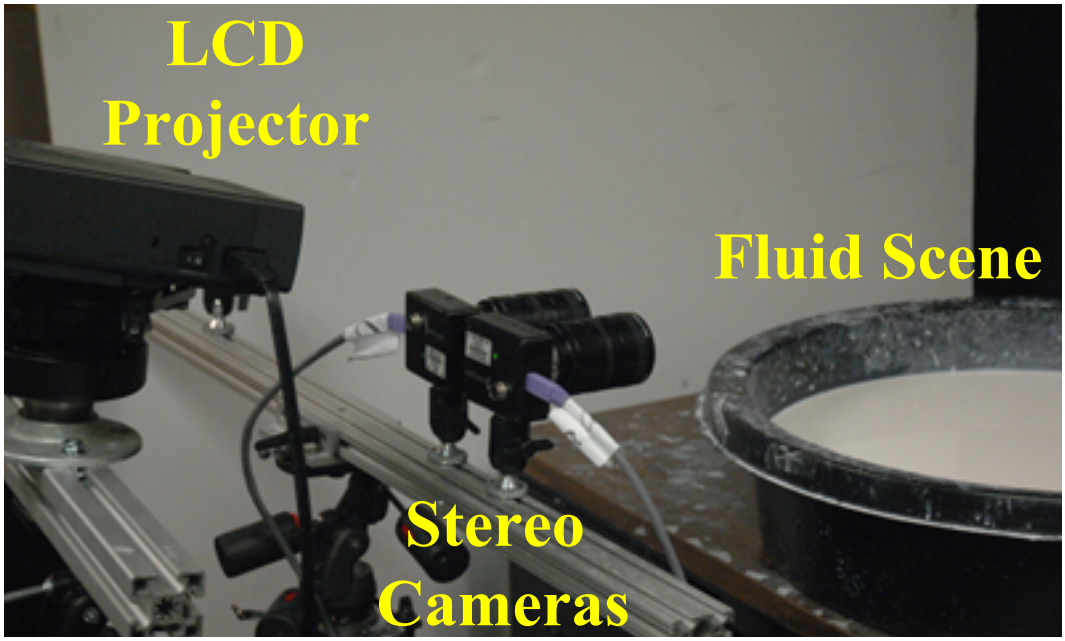


Figure 31: The capturing setup.

5.4.1 Depth Extraction

Captured stereo sequences $\{I_t\}$ and $\{J_t\}$ are first rectified according to epipolar geometry so that any 3D point will be projected onto the same horizontal line in I_t and J_t. The depth of pixel p on I_t is determined from its feature correspondence p' on J_t. Since I_t and J_t are rectified, p' must be on the same horizonal line as p. The problem of stereo matching is then to find the *disparity*, which is defined as the difference of horizontal coordinates between p and p'. Stereo matching is a well studied problem in computer vision. We use the method proposed in [99], which is based on belief propagation. Let $D_p = \arg\min_x M_p(x)$ be the disparity for each pixel p on I_t, where $M_p(x)$ is a similarity measurement for each displacement candidate x, the following energy functional describes the whole function set $\{M_p : p \in I_t\}$ as its minimum solution:

$$\sum_{p \in I_t} \left(E_0(M_p, C_p) + \sum_{q \in N_p} E_1(M_p, M_q) \right) \tag{51}$$

q is one of four pixel neighbors to p. E_0 defines the difference between M_p and the initial cost C_p, given by pixel neighborhood similarity (usually measured by Sum of

Squared Differences (SSD)), and E_1 defines the disparity continuity among neighboring pixels.

Basically, the purpose of using belief propagation is to propagate messages among neighbor pixels. Both M_p and C_p are discretized as vectors in pixel resolution and are updated iteratively by linearly blending the received messages from pixel neighborhood. The final depth for each pixel is given by the disparity that has the minimum cost. Implementation details of this algorithm can be found in [99, 28]. In addition, we use a sub-pixel interpolation scheme proposed in [119] to smooth the depth map and to avoid aliasing artifacts that are caused by disparity vector discretization.

Shadow regions and regions occluded in another camera view often contain depth errors due to missing spatial correspondences. Those regions are near depth boundaries and they usually belong to the further (occluded) surfaces. Our solution is a two-pass algorithm. In the first pass, a left/right check in [21] is carried out over depth maps from both cameras to identify regions with inconsistent disparity values. Then in the second pass, the belief propagation algorithm estimates the depth map as usual without the pixels from the problem regions. After that, their depth values are assigned from already calculated regions through Laplacian smoothing. This two-pass algorithm works best when a problem region is encircled by good regions and it fails when the problem region is close to shape silhouettes. Placing the projector directly behind the cameras helps reduce shadow regions as discussed before, however, this cannot completely avoid shadows.

Figure 32 shows the depth map from a fountain example. Although we tried to make the liquid as opaque as possible by saturating it with white paint, some light scattering still occurred beneath the liquid surface. This effect blurred the light pattern and caused noise in the depth map, which is more obvious after the initial surface reconstruction, as shown in Figure 33 (top). For the same reason, thinner water regions became more transparent, which made the stereo matching process

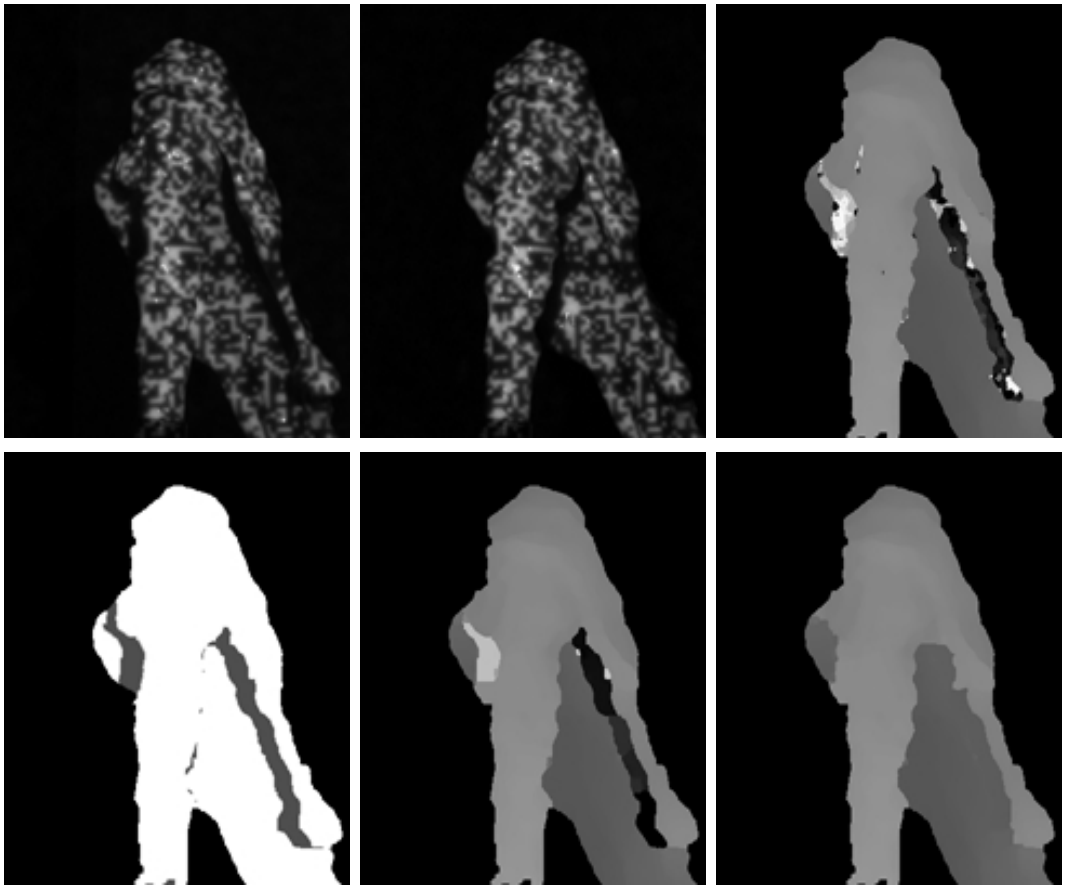


Figure 32: The first two images in the top row are captured images from stereo camera after rectification. A random texture pattern is projected on the water surface. The top right image is the noisy depth result without belief propagation. The bottom row from left to right shows problem regions (in gray) recognized by the two-pass algorithm, and the depth result before and after the two-pass algorithm.

more difficult. The stereo matching algorithm also failed when water regions became smaller, because of the transparency issues and due to resolution limits in the light pattern. For instance, tiny water drops in the pouring example were usually omitted from the depth map.

5.4.2 Surface Initialization Heuristics

The depth map created in Section 5.4.1 only covers part of the real liquid surface, in fact, less than 50%. Since the optimization method in Section 5.6 requires a roughly complete surface sequence as input, we use a heuristic method for creating an initial

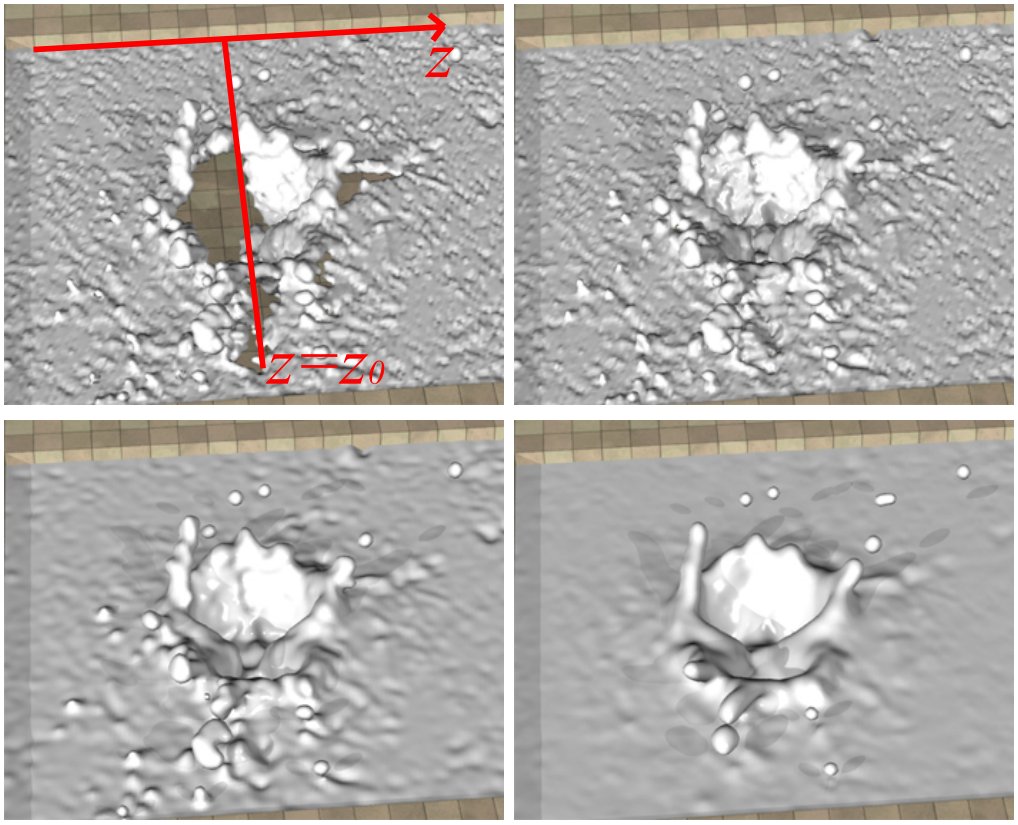


Figure 33: The initialization result for the splash example. From left to right and top to bottom are the partial surface Γ_t from the depth map, the surface after initialization, the surface after initialization and smoothing, and the final reconstructed surface, respectively.

guess of the surface. All surfaces at this point and later on are represented as signed distance functions on a regular volumetric grid.

Using Γ_t, the partial surface defined according to the depth map at time t, an initial surface ψ_t is created by the union of all spheres centered at Γ_t with radius r:

$$\psi_t(\vec{x}) = \min_{y \in \Gamma_t}(\|\vec{x} - \vec{y}\| - r) \tag{52}$$

This method is effective when water behaves like a thin shell or film, as the pouring example shown in Figure 43. It is not sufficient when large water regions are missing due to occlusion, therefore, we use some simple heuristics in our experiments to approximate the missing regions. For instance, the splash example of Figure 44 is nearly symmetric to a vertical plane $z = z_0$. We first fill in the water volume beneath

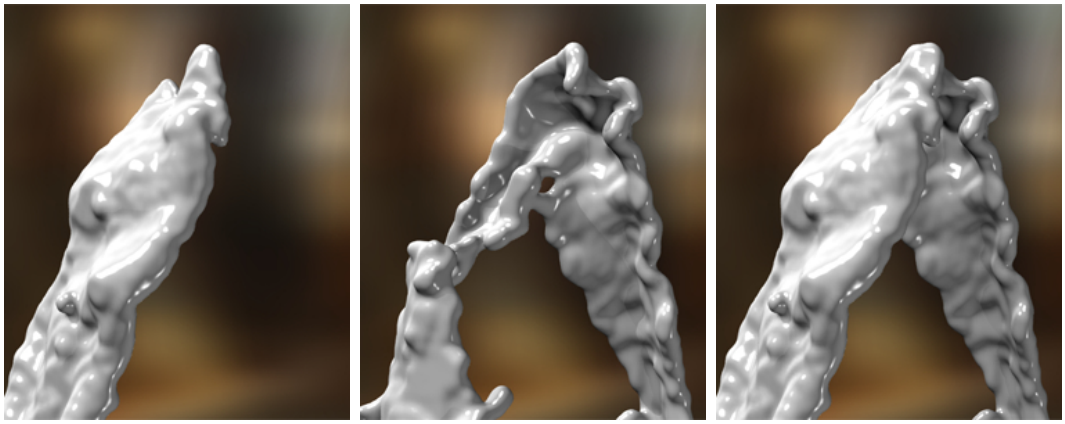


Figure 34: While each surface reconstructed solely in the front view (left) or back view (middle) cannot cover the whole surface, a reasonable initial surface can be generated by assembling two views together, even if they were captured at different times.

the visible surface as a height field. Let $\vec{y}$ be the lowest point in Γ_t on the same vertical line as a grid cell $\vec{x}$:

$$\vec{y} = \arg\min \vec{y}_y \qquad (\vec{y} \in \Gamma_t, \quad (\vec{y} - \vec{x}) \parallel Y\ axis\ ) \tag{53}$$

For any $\vec{x}$ whose $\vec{y}$ exists, its signed distance value is simply:

$$\psi_t(\vec{x}) = \vec{y}_y - \vec{x}_y \tag{54}$$

If $\vec{x}$ doesn't have $\vec{y}$ but its symmetric counterpart $\vec{x}' = \vec{x} + (2z_0 - \vec{x}_y)(0, 1, 0)^T$ to the plane has a $\vec{y}'$ defined, the signed distance can then be duplicated from its counterpart as: $\psi_t(\vec{x}) = \psi_t(\vec{x}')$. Finally, distance values for the rest of the undefined grid cells are interpolated from their defined horizontal neighbors to complete the surface ψ_t. An example is shown in Figure 33.

Space-Time Merging When a water scene exhibits temporal repetitiveness rather than spatial repetitiveness, similar ideas can be applied to reconstruct the initial shape ψ_t using video sequences captured at different times from different viewpoints. For example, a single pair of cameras is unable to simultaneously capture both the downstream (front) and the upstream (back) surfaces in our fountain case in Figure 34

due to occlusions. Our solution is to capture each of these streams in turn, that is, first capture the upstream view, then move the camera around to capture the downstream view. While each surface immediately generated from video cannot cover the whole liquid surface, they can be stitched together assuming that temporal repetitiveness provides a similar dynamic appearance over time. In practice, we first recover the relative camera pose of each view using static points (fiducials) in the scene. From the pose information, different sequences can then be aligned. It should be noted that this alignment does not require high-accuracy, and even slightly moving points can be used as fiducials. If the original sequences are sufficiently long, we choose two optimal sub-sequences from the original input such that the overall difference between them are minimized after spatial alignment. The shape difference is measured as the Sum of Squared Differences (SSD) over an intersection region between two signed distance functions. Although the resulting surface ψ_t may not correspond exactly to a real water scene, our experiment shows that this approach produces visually plausible initial results.

5.5 Surface Smoothing and Spatial Coherence

Real liquid surfaces are smooth due to the effect of surface tension, especially for water at small scales. To provide similar results in our method, we use a surface smoothing scheme each time the surface sequence is updated.

The mean curvature flow can provide a surface smoothing effect, and this can be achieved using the following level set formulation:

$$\phi_s = \kappa \cdot |\nabla \phi| \tag{55}$$

in which $s \rightarrow \infty$, standing for a steady solution. κ is the surface mean curvature. The mean curvature flow tries to minimize the surface area by enforcing the mean curvature uniformly distributed over the whole surface, so that the tendency of this

method is to evolve a surface toward a spherical shape. Such a level-set mean curvature evolution is straightforward to implement, but it performs excessive smoothing and undermines the fidelity to the original geometry (our spatial constraints). In fact, although the final goal of the surface tension effect is similar to that of the mean curvature flow over time, the surface tension distribution at a particular time instant should not be the steady state of the mean curvature flow. Therefore, we choose the smoothing scheme proposed by Schneider and Kobbelt [83] under the following target PDE instead:

$$\Delta_B \kappa(\vec{x}) = 0 \tag{56}$$

Δ_B is the Laplace-Beltrami operator on a curved surface. Assuming that κ along the surface normal on iso-surfaces is locally linear, Δ_B can be approximated simply by a Laplacian operator in 3D, and the solution is an evolution equation in the fourth order:

$$\phi_s = \Delta\kappa \cdot |\nabla\phi| \tag{57}$$

Intuitively, this will produce a liquid surface with C^2 continuous mean curvature, leading to a C^2 surface tension field over the surface. This fourth-order equation is better at preserving spatial constraints, and it also preserves volumes according to the divergence theorem. We solve Equation 57 iteratively using a first-order forward Euler method. Instead of calculating $\Delta\kappa$ directly using a four-order scheme, we first calculate κ for each grid cell close to the surface boundary, then calculate $\Delta\kappa$, both by central differencing:

$$\Delta\kappa = \nabla \cdot \nabla\kappa = \kappa_{xx} + \kappa_{yy} + \kappa_{zz} \tag{58}$$

In practice, we combine the surface smoothing step together with a fidelity term for spatial coherence (penalizing differences from the input geometry):

$$\phi_s = \alpha(\psi - \phi) + \Delta\kappa \cdot |\nabla\phi| \tag{59}$$

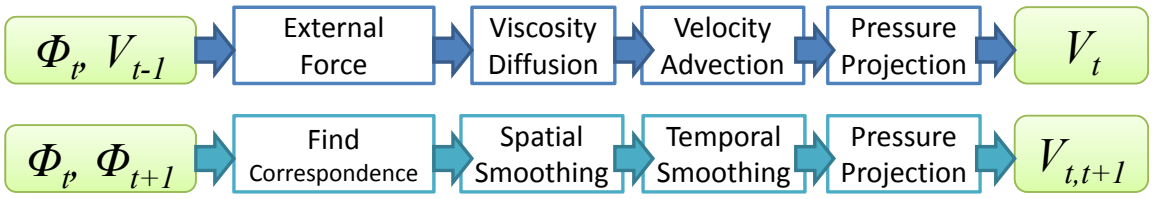


Figure 35: The velocity field estimation scheme for physically based fluid simulation (top) and the 3D flow estimation method (bottom).

α is a coefficient balancing between fidelity and smoothing strength, ranging from 0 to 0.1. For the data sets we have processed, only 5 to 10 iteration steps are needed to produce good smoothing results with preservation of most of the spatial features.

5.6 Temporal Coherence

In this section we will discuss the surface optimization approach for enforcing temporal coherence, corresponding to $\mathbb{E}_n$ in Equation 50. The basic idea is to iteratively update the liquid surface at each frame based on the prediction from its neighboring frames, as shown in Section 5.6.3, similar to a Laplacian smoothing algorithm. In order to achieve this, we first present a 3D temporal flow estimation method in Section 5.6.1, which takes incompressibility into account. When local minimum happens due to two types of errors, *false-positives* and *false-negatives*, two extra steps will be used to address these issues in Section 5.6.2 and 5.6.4, respectively. False-positives are errors when certain surface components fail to find their temporal correspondences in adjacent frames, since they do not actually exist in the real fluid scene. Meanwhile, false-negatives are defined as missing surface components in the reconstruction result, preventing other frames from finding temporal correspondences.

5.6.1 3D Temporal Flow Estimation

Given a fluid surface ϕ_t at time t and a velocity field $\vec{v}_{t-1,t}$ from the previous time step, physically based fluid simulation first calculates the new velocity flow $\vec{v}_{t,t+1}$ by solving fluid dynamics equations, then evolves ϕ_{t+1} from ϕ_t using $\vec{v}_{t,t+1}$. Different

from physically based fluid simulation, both ϕ_t and ϕ_{t+1} are given as input to our problem, and our goal in this section is to find the corresponding velocity field $\vec{v}_{t,t+1}$ that is most likely to cause the surface advection between ϕ_t and ϕ_{t+1}.

The dynamic behavior of water can be described by incompressible viscous fluid dynamics according to the Navier-Stokes equations. Most fluid simulation methods in graphics use operator splitting (sometimes called the *method of characteristics*) to calculate the separate terms of the Navier-Stokes equations. In such cases, a fluid solver first applies external forces on $\vec{v}_{t-1,t}$, then advects the velocity flow by itself, damps the velocity flow by viscosity diffusion, and finally projects the velocity field back to a divergence-free space to maintain incompressibility. Figure 35 top shows these steps.

If we treat ϕ_t and ϕ_{t+1} as knowns and $\vec{v}_{t,t+1}$ as unknowns, a solution $\vec{v}_{t,t+1}$ is expected to satisfy following properties:

1) surface advection: $\vec{v}_{t,t+1}$ evolves ϕ_t to ϕ_{t+1}.

2) Spatial continuity: $\vec{v}_{t,t+1}$ is continuous in space, due to the viscosity effect.

3) temporal continuity: $\vec{v}_{t-1,t}$ is continuously related to $\vec{v}_{t,t+1}$ and $\vec{v}_{t+1,t+2}$, due to velocity advection.

4) Incompressibility: $\vec{v}_{t-1,t}$ is divergence-free.

These properties are formulated as constraints to $\vec{v}_{t,t+1}$ as shown in Figure 35 bottom, and the goal of this method is to find $\vec{v}_{t,t+1}$ that can satisfy those constraints as much as possible. A $\vec{v}_{t,t+1}$ exactly satisfying all constraints may not exist sometimes due to internal conflictions in input shapes. For example, surface advection and incompressibility cannot be both satisfied if ϕ_t and ϕ_{t+1} do not have the same volume. In this case, the solution balances between all of these constraints and the method iteratively relaxes each constraint until a balance is reached. The initial correspondence search step (Section 5.6.1.1) first finds a velocity field that can satisfy the surface advection constraint based on a signed-distance-similarity measure. The

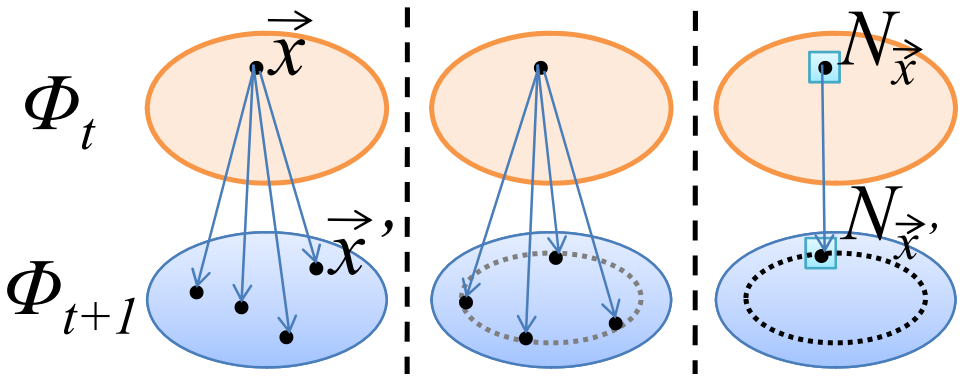


Figure 36: Three correspondence methods. For each point $\vec{x}$ in water ($\phi_t(\vec{x}) < 0$), the left scheme matches $\vec{x}$ with any point $\vec{x}'$ in water $\phi_{t+1}(\vec{x}')$. The middle scheme matches $\vec{x}$ with $\vec{x}'$ so that their signed distance values are the same $\phi_t(\vec{x}) = \phi_{t+1}(\vec{x}')$. The right scheme looks for $\vec{x}'$ so that their neighborhoods are similar, and this is the method that we use.

initial result is then smoothed in space and time, and projected back to be divergence-free to enforce incompressibility, as a fluid solver does. Our flow estimation method is related to the classical optical flow problem in computer vision, in which image intensity similarity and spatial continuity are two constraints that are to be satisfied. In addition to spatial continuity, however, our method also considers temporal continuity and incompressibility.

5.6.1.1 Correspondence Search

Given two liquid surfaces ϕ_t and ϕ_{t+1} at time t and $t+1$ respectively, the velocity estimation method first tries to find a correspondence $\vec{x}'$ in ϕ_{t+1} for each grid cell $\vec{x}$ within or around water in ϕ_t. The velocity field $\vec{v}_{t,t+1}$ at $\vec{x}$ is then implied as $(\vec{x}'-\vec{x})/\Delta t$. To find $\vec{x}'$ under the surface advection constraint, a conservative approach is to accept any water grid cell $\vec{x}'$ in ϕ_{t+1}: $\phi_{t+1}(\vec{x}') < 0$ ($\phi_t(\vec{x}) < 0$), as illustrated in Figure 36 left. Because each $\vec{x}$ has more than one acceptable $\vec{x}'$ using this method, other constraints must be used to remove the ambiguity and the whole process has to be performed over several iterations. An alternative approach is to treat the signed distance field as an image, and then use a classical optical flow algorithm to find

correspondences based on $\vec{x}$'s signed distance value: $\phi_{t+1}(\vec{x}') = \phi_t(\vec{x})$. This method successfully restricts the ambiguities by only searching correspondences over $\vec{x}$'s iso-surface in ϕ_{t+1}, as shown in Figure 36 middle. However, this approach still requires 10 to 20 iterations in order to get plausible results, as our experiment shows. To further reduce ambiguities and accelerate the convergence, we use a local $3 \times 3 \times 3$ neighborhood window $N_{\vec{x}}$ centered at $\vec{x}$ aligned with grid as the shape feature at $\vec{x}$. In this way, we look for $\vec{x}'$ such that $\vec{x}$'s neighborhood and $\vec{x}'$'s neighborhood are similar to each other:

$$\begin{aligned} \vec{x}' = \arg\min \sum_{\vec{y} \in N_{\vec{x}}} w(\vec{y}) \left(\phi_{t+1}(\vec{y} + \vec{x}' - \vec{x}) - \phi_t(\vec{y})\right)^2 \\ \vec{v}_{t,t+1}(\vec{x}) = (\vec{x}' - \vec{x})/\Delta t \end{aligned} \tag{60}$$

$w(\vec{y})$ is a weight function for the neighborhood, typically a Gaussian falloff. We calculate $\vec{x}'$ for all water grid cells in all frames, which gives us a sequence of initial forward velocity fields $\{\vec{v}_{t,t+1}\}$ with only surface advection considered.

In order to process the whole surface optimization algorithm, we also need the backward velocity field $\vec{v}_{t+1,t}$, which evolves the shape sequence in the backward direction. One way to do this is to convert the forward velocity field $\vec{v}_{t,t+1}$ into a backward velocity field $\vec{v}_{t+1,t}$ by inverting and resampling $\vec{v}_{t,t+1}$. However, a resampling process causes blurry effects over $\vec{v}_{t+1,t}$ and the inverse of $\vec{v}_{t,t+1}$ is not necessarily a valid velocity field for $\vec{v}_{t+1,t}$ in many cases. It is also possible to explicitly calculate the backward velocity field sequence $\{\vec{v}_{t+1,t}\}$ like the forward sequence, but this will introduce extra storage and computational cost.

Our approach is to estimate the mean velocity field $\vec{v}_t$ from time instant $t-1$ to $t+1$ by using errors in both directions simultaneously. $\vec{v}_t$ can also be treated as an approximation to the instant velocity field at time t. Under constant acceleration, this approximation is exact. How to approximate $\vec{v}_{t,t+1}$ and $\vec{v}_{t,t-1}$ using $\vec{v}_t$ will be discussed in details in Section 5.6.3. This method is formulated mathematically as

follows:

$$\begin{aligned}
&\vec{v}_t(\vec{x}) = \arg\min D(\vec{x}, \vec{v}) \\
&= \arg\min \sum_{\vec{y} \in N_{\vec{x}}} w(\vec{y}) \left(\left(\phi_{t+1}(\vec{y}') - \phi_t(\vec{y})\right)^2 + \left(\phi_{t-1}(\vec{y}'') - \phi_t(\vec{y})\right)^2 \right) \\
&\vec{y}' = \vec{y} + \vec{v}_t(\vec{x})\Delta t \\
&\vec{y}'' = \vec{y} - \vec{v}_t(\vec{x})\Delta t
\end{aligned} \tag{61}$$

in which $\vec{y}'$ and $\vec{y}''$ are $\vec{y}$'s correspondence location in ϕ_{t+1} and ϕ_{t-1} respectively. Estimating the mean velocity field $\vec{v}_t$ not only gives us a way to approximate the velocity fields $\vec{v}_{t,t+1}$ and $\vec{v}_{t,t-1}$ in both directions, and it also helps remove ambiguities in searching correspondences because the local temporal continuity of the velocity field at $\vec{x}$ has been implicitly considered.

To solve Equation 61, we differentiate $D(\vec{x}, \vec{v})$ with respect to $\vec{v}_t(\vec{x})$, therefore:

$$\sum_{\vec{y} \in N} w(\vec{y}) \begin{pmatrix} +\left(\phi_{t+1}(\vec{y}') - \phi_t(\vec{y})\right) \nabla\phi_{t+1}(\vec{y}') \\ -\left(\phi_{t-1}(\vec{y}'') - \phi_t(\vec{y})\right) \nabla\phi_{t-1}(\vec{y}'') \end{pmatrix} = 0 \tag{62}$$

$\nabla\phi$ is close to 1 by the definition of a signed distance function, therefore, Equation 62 can be solved iteratively using the Newton-Ralphson method. $\vec{v}$ can be initialized with zeros, several candidate seeds from heuristics, or user input if available. In each iteration, a new velocity flow $\vec{v}_{\text{new}}(\vec{x})$ is calculated from the old one $\vec{v}_{\text{old}}(\vec{x})$:

$$\sum_{\vec{y} \in N} w(\vec{y}) \begin{pmatrix} +\left(\phi_{t+1}(\vec{y}'_{\text{new}}) - \phi_t(\vec{y})\right) \nabla\phi_{t+1}(\vec{y}'_{\text{old}}) \\ -\left(\phi_{t-1}(\vec{y}''_{\text{new}}) - \phi_t(\vec{y})\right) \nabla\phi_{t-1}(\vec{y}''_{\text{old}}) \end{pmatrix} = 0 \tag{63}$$

$\vec{y}'_{\text{new}}, \vec{y}''_{\text{new}}$ and $\vec{y}'_{\text{old}}, \vec{y}''_{\text{old}}$ are $\vec{y}$'s correspondences calculated using new and old velocity flows respectively. We further linearize $\phi_{t+1}(\vec{y}'_{\text{new}})$ and $\phi_{t-1}(\vec{y}''_{\text{new}})$ as:

$$\begin{aligned}
\phi_{t+1}(\vec{y}'_{\text{new}}) &\approx \phi_{t+1}(\vec{y}'_{\text{old}}) + \nabla\phi^T_{t+1}(\vec{y}'_{\text{old}})(\vec{v}_{\text{new}} - \vec{v}_{\text{old}})\Delta t \\
\phi_{t-1}(\vec{y}''_{\text{new}}) &\approx \phi_{t-1}(\vec{y}''_{\text{old}}) - \nabla\phi^T_{t+1}(\vec{y}''_{\text{old}})(\vec{v}_{\text{new}} - \vec{v}_{\text{old}})\Delta t
\end{aligned} \tag{64}$$

Combining Equation 63 with 64 gives us a linear system with $\vec{v}_{\text{new}}(\vec{x})$ as unknowns. We terminate the iteration process if maximum iteration number is reached or if

$|\vec{v}_{\text{new}} - \vec{v}_{\text{old}}|$ drops below certain threshold (0.05 of a grid cell size in our experiments). We also limit $|\vec{v}_{\text{new}} - \vec{v}_{\text{old}}|$ in each iteration by an upper bound for smoother convergence. We limit the search space within a region around the initial guess to prevent large velocity changes in case the problem is ill defined. This method is similar to the classic Kanade-Lucas-Tomasi (KLT) feature tracker [90] with translational motion only, except that our problem is defined in 3D and the search procedure considers both forward and backward directions.

If the correspondence search in one direction goes beyond the spatio-temporal boundary, we assume that any configuration is possible outside of the 4D space-time grid, so the corresponding error in Equation 61 and correction in 63 are simply ignored.

Since the correspondences are found completely based on shape deformation, the result becomes highly sensitive to surface motion when the motion is small, as shown in Figure 37f. Another problem called the aperture problem in the KLT feature tracker also exists in this method, which brings ambiguities into the velocity field for a flat liquid surface. When the change in the signed distance function does not reflect the actual surface motion, for example when two water drops merge or split (Figure 38), this estimation method fails to work well according to Equation 61 around the contact region.

Although the initial goal of applying physically based constraints on the velocity field is to augment the temporal coherence process with physical plausibility, we found that it also provides a solution to some of the above problems by propagating velocity information from well-defined regions to ill-defined regions. To accelerate the propagation into the less-deformed regions (shown in Figure 37), we scale the update $|\vec{v}_{\text{new}} - \vec{v}_{\text{old}}|$ by the error metric in Equation 61 so that the propagation process goes faster.

5.6.1.2 Error Measure

We use the error metric $D(\vec{x}, \vec{v})$ in Equation 61 to determine whether $\vec{x}$'s correspondence exists in the neighboring frame. If D at time t is below some tolerance ϵ, we categorize $\vec{x}$ at time t as Type-I, which means it has correspondences in both ϕ_{t-1} and ϕ_{t+1}. Otherwise, the method will try to locate correspondences only in ϕ_{t-1} or ϕ_{t+1}, with a tolerance $\epsilon/2$. Searching in one direction can simply be done by removing the error and correction contribution from the opposite direction in Equation 61 and 63. Grid cells belong to type-II if a correspondence exists in ϕ_{t+1}, or type-III, if a correspondence exists in ϕ_{t-1}. The rest of the grid cells will be type-O, which means that no correspondence has been found in either ϕ_{t-1} or ϕ_{t+1}.

5.6.1.3 Spatial Smoothing

Our spatial smoothing step mimics the viscosity effect for spatial continuity, and the process is similar to an explicit Laplacian smoothing solver:

$$\vec{v}_t(\vec{x}) = \sum s(\vec{y})\vec{v}_t(\vec{y}) \tag{65}$$

$\vec{y}$ is a water grid cell within an immediate grid neighborhood of $\vec{x}$, including $\vec{x}$ and its six neighbors. $s(\vec{y})$, the spatial smoothing kernel covering $\vec{p}$ and its six immediate neighbors, and is typically defined as:

$$s(\vec{y}) = 1 - 6\beta \quad (\vec{y} = \vec{x}), \qquad s(\vec{y}) = \beta \quad (otherwise) \tag{66}$$

β is specified by user according to their expectation of the velocity spatial smoothness. Although this step looks similar to the explicit viscosity solver in a physically based fluid simulation, it only provides a similar effect to viscosity diffusion and it is not based on actually fluid dynamics since only a static velocity field is involved.

To generate large viscosity effects, some fluid simulation algorithms [96, 13] formulate viscosity diffusion implicitly as a sparse linear system. Instead, our method

simply applies the smoothing process multiple times to produce a visually plausible large viscosity effect.

5.6.1.4 Temporal Smoothing

In the real world, liquid inertia provides a continuous velocity field sequence by propagating the velocity along the liquid motion, and this is called velocity advection in physically based fluid simulation. The temporal smoothing step mimics this effect by using a Laplacian smoothing algorithm:

$$\vec{v}_t(\vec{x}) = (1-2\gamma)\vec{v}_t(\vec{x}) + \gamma(\vec{v}_{t-1}(\vec{x} - \vec{v}_t(\vec{x})\Delta t) + \vec{v}_{t+1}(\vec{x} + \vec{v}_t(\vec{x})\Delta t)) \tag{67}$$

γ is typically chosen from $[0, 0.3]$ in our experiments. This step can be applied multiple times to achieve a large smoothing effect, as the spatial smoothing step does. In practice, we prefer to use larger γ and fewer iterations (one to three) to avoid excessive blurring effects in the temporal domain. For example, small water splashes may occur even before a water drop hits the water surface in the splash example (Figure 37), because the real splash can be propagated backward too far if many iterations are used.

5.6.1.5 Pressure Projection

While the velocity is defined at each grid cell for the convenience of estimation and optimization, it is not straightforward to couple velocity with pressure, which is also defined at the grid cell. Our solution is to first interpolate the velocity flow to a staggered grid formulation by averaging, then apply pressure projection, and convert the velocity back to the grid cell center. Details about the pressure projection in a staggered Mark-and-Cell (MAC) grid can be found in [96]. We use Dirichlet boundary condition with zero outer pressure when water bodies are completely within the space-time grid. Water bodies expanding beyond the space-time grid are ignored in this step since their volumes outside of the grid are unknown.

After the pressure projection step, the error values are recalculated and the grid cells are re-categorized. This category information will later be used to determine whether a missing correspondence is caused by false-positives or false-negatives.

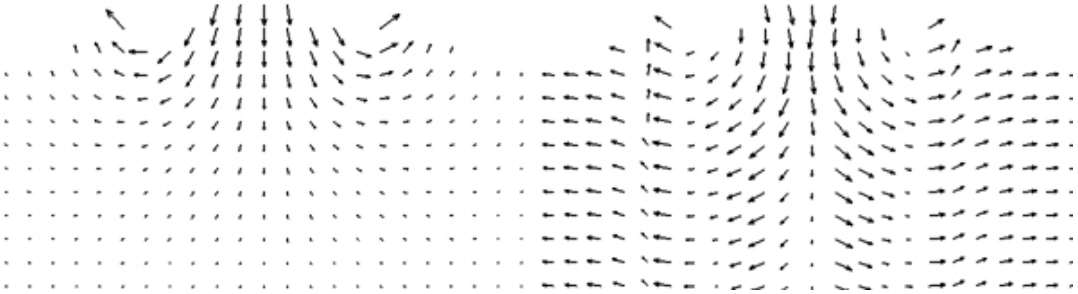
5.6.1.6 Discussion

Given a time-varying shape sequence, the solution to a velocity field that evolves one shape into another is not unique. In general, arbitrary vortices can be added to plausible solutions as long as each vortex does not affect the surface shape. The velocity estimation method described above finds a velocity field with few vortices in most cases, even though it fails if the signed distance function changes dramatically, such as the contact region where two drops merge, as shown in Figure 38g. Such a solution may or may not be similar to the actual velocity field in the real world. On the other hand, it is theoretically impossible to recover the flow underneath the water surface if the flow does not affect the surface because we can only observe the surface. Fortunately, since the estimated velocity field is used for surface optimization, which only modifies the liquid surface, ambiguities in these solutions do not seem to have any noticeable effects in our results.

Apply physically based constraints improves the initial estimation result by propagating information from well-defined regions to ill-defined regions. The actual solution is a balance between the initial correspondence search and physically based constraints. When the initial correspondence has too many errors, the final result can be far away from the actual velocity field, and may lack physical plausibility. Fortunately, those regions are rare and small. Plausible results can still generated even if some correspondence errors exist, such as those caused by topological changes, as shown in Figure 38.

The side effect of physically based constraints is the excessive smoothing effect over the velocity field in both space and time. When reflected in the temporal coherence

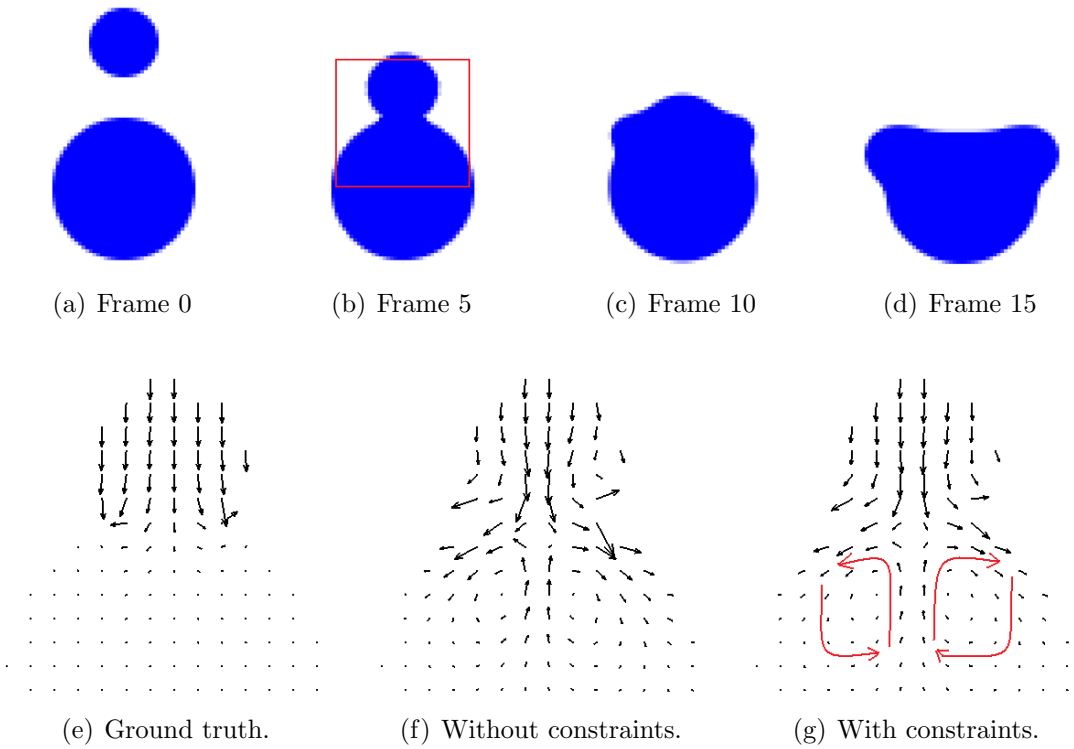
(a) Frame 9 (b) Frame 14 (c) Frame 19 (d) Frame 24



(e) Ground truth. (f) Without constraints.

(g) With constraints. (h) Difference between (e) and (g).

Figure 37: A synthetic 2D splash sequence (a, b, c, d) and the estimated velocity fields for Frame 19 (c). This example was simulated using a five-times smaller time step and the simulated velocity field is used as the ground truth in (e). The wave surface remains less changed before the splash wave is propagated, causing the region far away from the impact location to obtain higher velocity in (f) than the actual. This problem is addressed by applying physically based constraints. The result after applying the constraints is shown in (g) and the error magnitude is shown in (h).



(a) Frame 0 (b) Frame 5 (c) Frame 10 (d) Frame 15

(e) Ground truth. (f) Without constraints. (g) With constraints.

Figure 38: A synthetic 2D sequence (a, b, c, d) shows a small water drop merging with a larger drop under the surface tension. The velocity field is estimated for Frame 5 (b). From (e) to (g), the figure shows the ground truth of the velocity field from simulation, the estimation without constraints and with constraints. Applying physically based constraints can smooth out velocity errors due to the rapid changes in the sign distance function when two water drops merge. However, it cannot remove the vortex ambiguity show by the red arrows (g). The final result in (g) is different from the ground truth in (e) mostly around the merging point.

process, it creates overly smooth shapes and less dynamic behaviors in the temporal domain. Therefore, we use as few iterations as possible to make sure that velocity information is propagated while minimizing the extra smoothing effects.

5.6.2 False-Positive Removal

Since both false-positives and false-negatives can cause missing correspondences in neighboring frames, the first task is to determine the cause of a missing correspondence. We do this by counting how many consecutive frames a water region appears in, assuming that real water should exists in several consecutive frames. For example, when a water region only appears in frame t, or frame t and $t+1$, it is less likely

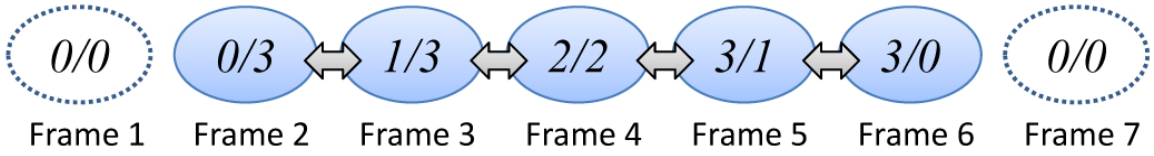


Figure 39: Forward/backward tags for a water drop that sequentially appears from frame 2 to 6.

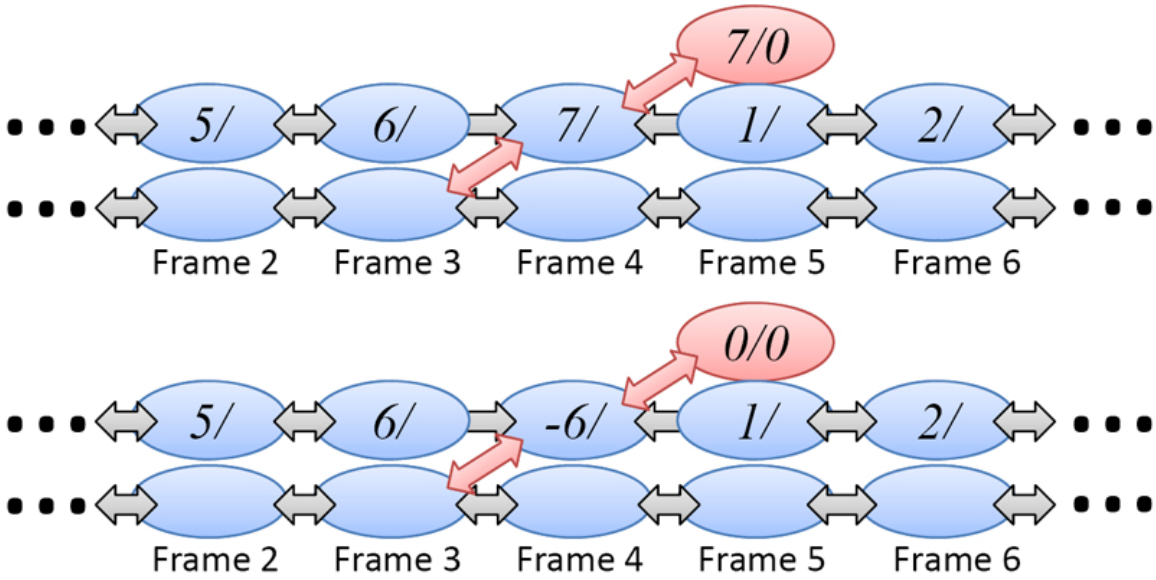


Figure 40: When two water drops (in blue) sequentially appear from frame 2 to 6 and an incorrect water drop (in red) suddenly appears in frame 5, their particular positions cause an ambiguity in the velocity flow at frame 4. Without the velocity consistency check at the top diagram, T_4^f will be erroneously propagated to the red water drop, preventing it from being removed. After the consistency check, this can be addressed as shown in the bottom diagram.

to be a water component in the real scene. Such false positives should be removed. However, if it exists in more frames, missing correspondences are more likely to be caused by false-negatives, so the component should be retained. This assumption fails in certain cases, for example, when a water drop is intermittently missing in a sequence, it will be treated as a false-positive even though it is more likely to be in the real scene. Fortunately, our experiments show that such cases are rare and both false-positives and false-negatives are usually distributed continuously over time.

To count how many times a water region appears in previous and successive frames, two count tags are assigned to each water grid cell: a forward tag T_t^f and a backward

Algorithm 1 The pseudo code for calculating forward tags.

```
for t = 0 to T do
  T_t^f = 0;
end for
for t = 0 to T do
  for each fluid cell x⃗ do
    if φ_t(x⃗) < band_width and type(x⃗, t) = I then
      if T_t^f(x⃗) < 0 then
        T_t^f(x⃗) = 0;
      end if
      T_t^f(x⃗) += 1;
      next = T_t^f(x⃗);
      x⃗' = x⃗ + v⃗_t(x⃗)Δt;
      if |v⃗_t(x⃗) − v⃗_{i+1}(x⃗')| > θ then
        next = −next;
      end if
      T_{t+1}^f(y⃗) = max(next, T_{t+1}^f(⌈x'_x⌉, ⌈x'_y⌉, ⌈x'_z⌉));
                  ......
      T_{t+1}^f(y⃗) = max(next, T_{t+1}^f(⌊x'_x⌋, ⌊x'_y⌋, ⌊x'_z⌋));
    end if
  end for
end for
```

tag T_t^b. Here we will explain how to set up the forward tags T_t^f by tracing particles from each grid cell. Backward tags are calculated in a similar fashion, except in the opposite direction. All T_t^f are initialized to be zero, and then processed forward from frame 0 to frame T. In each frame, every type-I water grid cell $\vec{x}$ first increases its T_t^f by 1, and propagates its value to all eight cells adjacent to $\vec{x}' = \vec{x} + \vec{v}_t(\vec{x})\Delta t$ in frame $t+1$:

$$T_{t+1}^f(\vec{y}) = \max(T_t^f(\vec{x}), T_{t+1}^f(\vec{y})), \quad \vec{y} \approx \vec{x}' \tag{68}$$

Figure 39 shows a single water drop example with both tags calculated. $T_t^f + T_t^b + 1$ is the count of the times that a water grid cell appears in previous and successive frames, and $T_t^f + T_t^b + 2$ is the number at the two ends. For each water grid cell $\vec{x}$ at time t, if $T_t^f + T_t^b$ is below some threshold k, it is considered to be a false-positive and will be removed by setting $\phi_t(\vec{x})$ to infinity. Theoretically, the trajectory of a free-falling water region existing in three consecutive frames can be predicted, assuming that the

acceleration is constant. In practice we choose k from the range of 3 to 7, in order to remove more false-positives and keep more confidence in the remaining water regions.

The ambiguity in velocity estimation may prevent false-positives from be removed if we are only counting correspondences, as shown in Figure 40 top. To solve this issue, we also compare $\vec{v}_t(\vec{x})$ with $\vec{v}_{t+1}(\vec{x}')$. If their difference is above certain threshold θ, $\vec{x}'$ will be considered to involve a velocity ambiguity, tagged as negative, and will lose its ability to propagate. One may ask whether the temporal smoothing step in the velocity estimation method can help the water region in frame 4 to choose the right velocity flow by exchanging velocity with its neighbors. Unfortunately this is hard to achieve since the velocity estimation method does not know which velocity is correct until the false-negatives and false-positives are recognized, and we have to defer such decisions until the false-positive removal step.

It should be noted that ideally $\vec{x}$'s correspondence at time $t+1$ should be calculated as $\vec{x}' = \vec{x} + \vec{v}_{t,t+1}(\vec{x})\Delta t$ instead of using the mean velocity $\vec{v}_t$. Since the difference between $\vec{v}_t$ and $\vec{v}_{t,t+1}$ does not affect the false-positive removal step much, we simply use $\vec{v}_t$ and we did not notice any difference in our experiment. The pseudo code for computing forward tags is given in Algorithm 1. The pouring example in Figure 43 shows redundant water regions that are removed as false-positives by this algorithm.

5.6.3 Optimization for Temporal Coherence

Given all the estimated velocity fields, we will apply optimization over $\mathbb{E}_n$ in Equation 50 to achieve temporal coherence. Let $\vec{x}$ be a type-I grid cell with both correspondences in the space-time grid, and let $\mathrm{P}^-(\phi_{t-1}, \vec{x}, \vec{v}_{t,t-1})$ and $\mathrm{P}^+(\phi_{t+1}, \vec{x}, \vec{v}_{t,t+1})$ be two functions that predict $\vec{x}$'s signed distance value according to neighboring frames ϕ_{t-1} and ϕ_{t+1} respectively. An ideal solution is to iteratively refine ϕ_t by interpolating its predicted values:

$$\phi_t(\vec{x}) = \omega(\mathrm{P}^-(\phi_{t-1}, \vec{x}, \vec{v}_{t,t-1}) + \mathrm{P}^+(\phi_{t+1}, \vec{x}, \vec{v}_{t,t+1})) + (1 - 2\omega)\phi_t(\vec{x}) \qquad (69)$$

in which ω is a relaxation coefficient between 0 and 0.5. A naive approach would be to replace $\vec{v}_{t,t+1}$ and $\vec{v}_{t,t-1}$ by the mean velocity field $\vec{v}_t$ as in the false-positive step, so:

$$\phi_t(\vec{x}) = \omega(\mathrm{P}^-(\phi_{t-1}, \vec{x}, \vec{v}_t) + \mathrm{P}^+(\phi_{t+1}, \vec{x}, \vec{v}_t)) + (1 - 2\omega)\phi_t(\vec{x}) \tag{70}$$

Equation 70 implies that $\vec{x}$ will be at the average position of its correspondences, assuming that $\vec{v}_{t,t+1} = \vec{v}_{t,t-1}$ in each iteration. Therefore, the tendency after applying Equation 70 in multiple iterations is to introduce blurry effects in dynamics by smearing the accelerations in the liquid motion.

Instead, we approximate $\vec{v}_{t,t+1}$ and $\vec{v}_{t,t-1}$ by mean velocity fields:

$$\begin{aligned} \vec{v}_{t,t+1}(\vec{x}) &= (\vec{v}_t(\vec{x}) + \vec{v}_{t+1}(\vec{x} + \vec{v}_t(\vec{x})\Delta t))/2 \\ \vec{v}_{t,t-1}(\vec{x}) &= -(\vec{v}_t(\vec{x}) + \vec{v}_{t-1}(\vec{x} - \vec{v}_t(\vec{x})\Delta t))/2 \end{aligned} \tag{71}$$

This approximation is proved to be exact when the acceleration is constant. To implement a prediction function, one may think of evolving ϕ_{t-1} and ϕ_{t+1} to time t using the given velocity flow using by the level set method. This implementation is limited by the CFL condition, and must be done in multiple sub-steps when the fluid surface moves much more than one grid cell at each time step. To avoid the computational cost incurred in subdividing the time steps, a simple alternative is to use the signed distance value of $\vec{x}$'s correspondence in both frames:

$$\mathrm{P}^+(\phi_{t+1}, \vec{x}, \vec{v}_{t,t+1}) = \phi_{t+1}(\vec{x} + \vec{v}_{t,t+1}(\vec{x})\Delta t) \tag{72}$$

similar to a simplified one-step semi-Lagrangian method. Unfortunately, this method suffers from volume lose (10% to 20% in each iteration as our experiment shows) because the required resampling process gradually smears out the surface sequence in each iteration. Instead, we use a non-interpolating semi-Lagrangian method described in [98] by combining it with the level set method, under the assumption that the velocity flow is locally uniform due to viscosity effects. This method first separates the offset vector $\vec{v}_{t,t+1}(\vec{x})\Delta t$ into a rounded-up integer component $\vec{o}_i$ and the remaining

floating component $\vec{o}_f$. The surface motion in the integral displacement is to shift signed distances in $\vec{x}$'s local neighborhood to $\vec{x} + \vec{o}_i$. The remaining surface motion corresponding to $\vec{o}_f$ is less than half of a grid cell in any axis, so the level set method can be safely used with less stability issues. This method is described as follows:

$$\begin{aligned} \mathrm{P}^+ &= \phi_{t+1}(\vec{x} + \vec{o}_i) - (-\vec{o}_f)\nabla\phi_{t+1}(\vec{x} + \vec{o}_i) \\ \vec{o} &= \vec{v}_{t,t+1}(\vec{x})\Delta t \\ \vec{o}_i &= \mathrm{Round}(\vec{o}); \quad \vec{o}_f = \vec{o} - \vec{o}_i \end{aligned} \tag{73}$$

When the surface motion is originally less than half of a grid cell ($\vec{o}_i = 0$), the method is simply reduced to the level set method. This hybrid method can successfully reduce volume loss to less than 5% over more than 5 optimization iterations in most of our experiments.

The above scheme is for type-I grid cells. If a grid cell $\vec{x}$ belongs to type-II or III, or if one of $\vec{x}$'s correspondence goes beyond the space-time grid, it is impossible to predict it from both directions. In these cases, we only consider the prediction in one direction. For example, the following equation calculates the update only from the backward direction:

$$\phi_t(\vec{x}) = \gamma\mathrm{P}^-(\phi_{t-1}, \vec{x}, \vec{v}_{t,t-1}) + (1-\gamma)\phi_t(\vec{x}) \tag{74}$$

5.6.4 False-Negative Completion

The final step in the system pipeline is to resolve false negatives by adding missing water regions back. Type-II and III grid cells can be safely assumed to be caused by false-negatives at this time. We first propagate their velocities to the neighboring frame with no correspondence by trilinear resampling. We then apply the one-way prediction tool provided in Equation 74 to grid cells with newly updated velocities to recover missing water regions. The false-negative completion process is executed more than once to allow water regions to propagate over multiple frames. After this, the

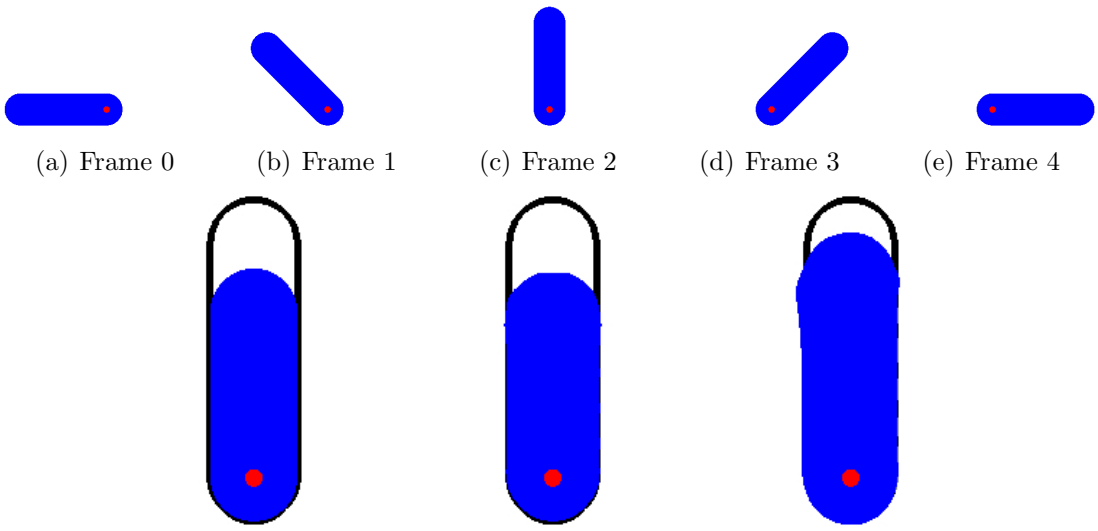
whole optimization procedure may be performed again to remove any spatio-temporal incoherence incurred by the false-negative completion step.

5.6.5 An extreme example

In order to demonstrate the effectiveness of this surface optimization algorithm, we tested it in 2D using an example shown in Figure 41. The blue rod rotates 180 degree around the red axis with a volume of 300 voxels in this example. Given the shape in Frame 1 and 3, it is impossible to know that the motion is rotational and the result of a shape metamorphosis algorithm that minimizes the deformation energy is shown in (f). Because the top of the rod in (f) is limited by the top of the shape in (b) and (d), the volume has only 233 voxels, 20% less than before. A surface optimization algorithm without physically based constraints can be considered to be a feature-based shape morphing algorithm, and it gives a similar result in (g), with 249 voxels. Applying physically based constraints gives the more plausible result in (h). This result is taller due to temporal coherence over all five frames in the surface optimization process. It also preserves more volume (283 voxels), with only 9% volume loss. The remaining volume loss is mostly due to the non-interpolating semi-Lagrangian method, especially because the velocity field in this rotation example does not quite satisfy the spatial smoothness assumption. The difference between (h) and the background truth (the black curve) is caused by the fact that the acceleration in a rotational motion is not constant. In practice, the deformation is usually much smaller than this one and the actual velocity field is more spatial continuous, making the temporal coherence algorithm more nearly volume preserving.

5.7 Results and Discussion

In this section we present results from several different real-world fluid examples. Please watch the accompanying video to see animations for each of these results. The scene in the pouring example shown in Figure 43 is made of free-falling streams poured



(a) Frame 0 (b) Frame 1 (c) Frame 2 (d) Frame 3 (e) Frame 4

(f) Shape Morphing. (g) Without constraints. (h) With constraints.

Figure 41: A testing example: a blue rod rotates around a red axis. The black curve in (f), (g) and (h) shows the silhouette of the rod in (c) as the ground truth. A shape morphing algorithm that minimizes the deformation energy generates the in-between shape in (f) from (b) and (d), which is smaller than the shape in (c). Using a velocity field without any constraints in the optimization process gives a similar shape in (g). With all the constraints, our algorithm produces the shape in (h).

out of a coffee cup. The animation is reconstructed at a resolution of $176 \times 120 \times 120$ for 155 frames. The splash example in Figure 44 shows the scene of a bottle lid dropping into an open region of water. The resolution of the 3D volume grid in this example is $158 \times 120 \times 163$ and includes 138 frames. Since it is difficult to discriminate the lid from the water surface, we modeled the lid as if it was part of the fluid surface. External acceleration and incompressibility are ignored for the large water body in the pool because it expands beyond the spatial grid and the pressure projection step cannot be easily resolved for it. The fountain example in Figure 1 (on the first page) was reconstructed from two video sequences captured at different times from two different viewpoints, containing the upstream and the downstream views of a small water fountain by placing stereo camera in front and back respectively. The resolution is $110 \times 150 \times 140$ and the result has 120 frames.

Table 3: Common variables and their values in our experiment.

Name	Definition	Value
α	a balancing coefficient between fidelity and shape smoothness in Equation 59	0.1
β	a parameter used in the velocity spatial smoothing kernel in Equation 66	1/7
γ	a parameter used in the velocity temporal smoothing kernel in Equation 67	$[0, 0.3]$
ω	a relaxation coefficient for temporal coherence by interpolation in Equation 69	0.3
ϵ	a tolerance bound in Error Measure, in Section 5.6.1.1	$[64, 256]$
k	a threshold to determine false positives by tag values in Section 5.6.2	[3, 7]
θ	a threshold to avoid velocity inconsistency when calculating tags in Section 5.6.2	0.2

All examples are calculated on an Intel Quad-Core 9550 workstation with 8G memory. The initial reconstruction and smoothing process in the first part of our algorithm typically took 10-30 minutes. Velocity estimation took 20-40 minutes and the rest of the surface optimization algorithm took 20-30 minutes. Five surface optimization iterations are usually sufficient to generate acceptable results. Overall, each frame takes two to three minutes to process on average. Compared with physically-based fluid simulation, this method is not limited to any CFL condition and the surface motion in each time step ranges between four to eight grid cell sizes on average. At the same resolution, a physically-based simulation would have to use smaller time steps to maintain stability and accuracy, causing significantly higher computational cost than this method, especially for rapid water motion.

Memory becomes an issue in our experiments since loading the entire 3D volume sequence with the velocity field and other data structure would require at least 6GB of memory, even for a small example. One possible solution would be to create a dynamic grid data structure that allocates memory only to grid cells that are close to the water surfaces. We choose to keep a simple grid data structure and use a large

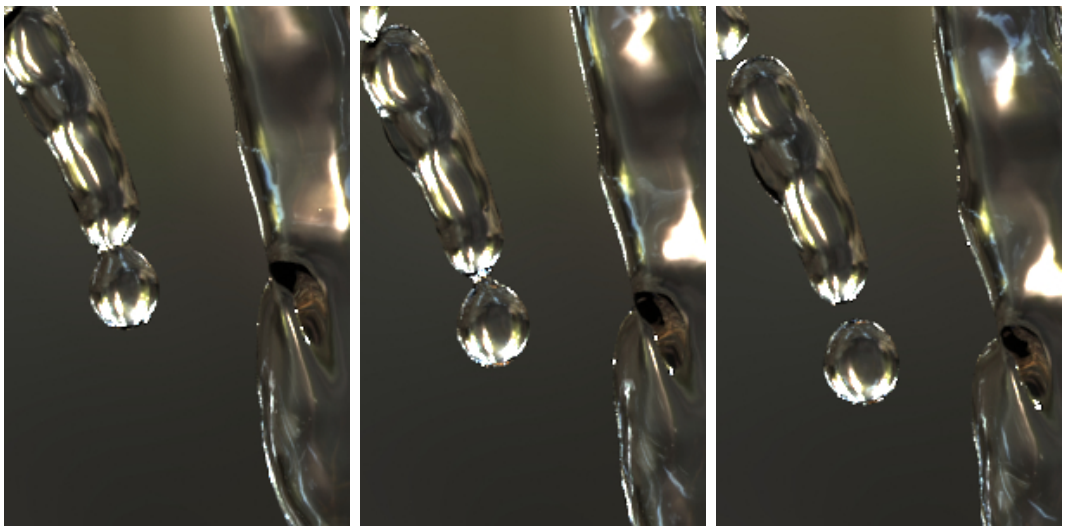


Figure 42: The middle shape is interpolated from the left shape and the right shape according to the estimated velocity field. In this way, the water drops break up continuously from the stream.

amount of virtual memory instead. Each iteration procedure requires at least one full memory swap to disk for each frame and contributes about 10% to 20% of the computational time.

The estimated velocity flow can be used to increase the temporal resolution of a sequence or to complete missing frames in case of data capture failure. Figure 42 shows that the shape in the middle frame is successfully interpolated from the left shape and the right shape. In addition, our algorithm can be considered to be a feature-based shape morphing method even without requiring shape alignment as pre-processing. When treating regions beyond the space-time grid as missing regions in the false-negative completion method in Section 5.6.4, our algorithm can also automatically extrapolate shapes in those regions in a spatio-temporal fashion as shown in Figure 45.

Our experiments show that our results can faithfully inherit the nuances and details of the fluids from regular video input. Our optimization algorithm converges in less than five iterations in most cases, therefore, it is safe from error accumulation over time. Although the overall solution may not be identical to the real world case,

for example, when a sudden splash is caused under strong acceleration, the result is a minimum to the whole energy functional (Equation 50) that closely satisfies spatio-temporal coherence and provides a visually plausible animation for graphics applications.

The existing system has several limitations due to the techniques we are currently using. Here we provide a list of those limitations. We plan to look for better solutions to these problems in the future.

Data Capture and Stereo Matching The existing capture system dyes the liquid with white paint and projects a random light pattern onto the liquid surface for better spatial feature extraction. Even when the liquid is highly saturated, light scattering effects can still occur beneath the liquid surface, causing small random noise in the reconstructed depth map. The liquid also becomes more transparent in small water drops and thin water films, making it impossible to reconstruct very fine details of the liquid surface. Shadows and occlusions are always difficult to handle in a stereo matching algorithm. The proposed two-pass algorithm gives a way to recognize those regions and then to propagate good depth information to them. However, this is only an approximation and it may or may not be the same as the real depth. Using more cameras can provide more visible surfaces and lessen the occlusion problem, but deploying and synchronizing multiple cameras makes the data capture process more difficult. In the future, we are interested in finding new capture devices or methods that are more practical for capture and reconstruction of general outdoor liquid effects.

Surface Initialization To complete missing regions in the liquid surface that are caused by camera occlusion, we use heuristic functions in the surface initialization process, including spatial repetitiveness and temporal repetitiveness. In a complicated liquid scene with severe occlusion problems, heuristic functions may no longer be

applicable. For instance, in the fountain example, the water flow appears to be too thin when viewed from a side angle because its thickness is unknown. Again, using multiple cameras can ameliorate this problem, but does not fully solve it. Under the single-stereo-camera paradigm, user interaction may be the only way to address this problem.

Velocity Estimation Generally speaking, the solution to a velocity field that evolves one shape into another shape is not unique. Given an initial guess, our velocity estimation algorithm is guaranteed to find a minimum solution in a gradient descent fashion. However, this solution may not be the optimal, especially when water regions merge or split, causing the distance field to change more rapidly than the actual surface motion. Applying physically based constraints helps resolve these issues by prorogating well-defined velocities into those problem regions. Extensively using physically based constraints will cause blurring effects in the velocity fields, so we prefer to use as few constraint iterations as possible. In a tough case when the initial velocity field has too many errors from the correspondence search, the estimation process becomes a tug-of-war between errors and blur.

Although this method does not require shape registration and alignment before correspondence search, it does require a good initial guess to reduce the searching time if two shapes are too different from each other. The searching space becomes larger when the liquid surface moves rapidly in a time step. At this time, the maximum offset our computation can handle is around 10 voxels. This method would fail if the liquid surface moves faster than this or if the frame rate drops.

Temporal Coherence Our temporal coherence algorithm is based on the assumption that the surface moves with constant acceleration. Therefore, the result can be different from real physics especially for large time steps when the acceleration

changes greatly, such as during rotation. To help with volume preservation, the non-interpolating semi-Lagrangian method prefers a spatially smooth velocity field. In a highly dynamic scene, this method may introduce more volume loss. Like the velocity estimation method, fewer iterations are preferable to avoid error accumulation, including excessive blurring artifacts and volume loss.

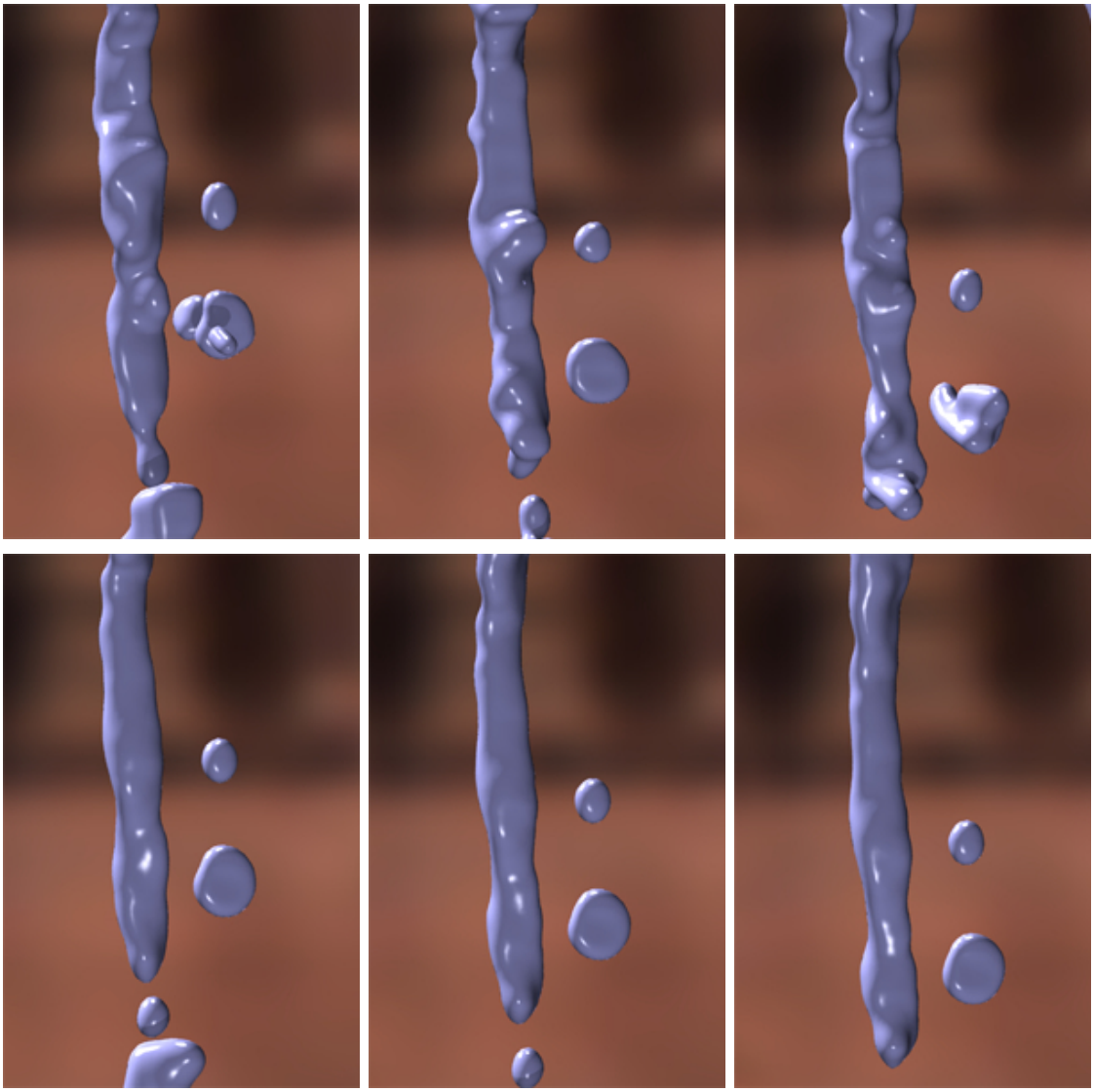


Figure 43: The pouring example: The top row shows the result without using any surface optimization. The bottom row show the result after applying our algorithm.

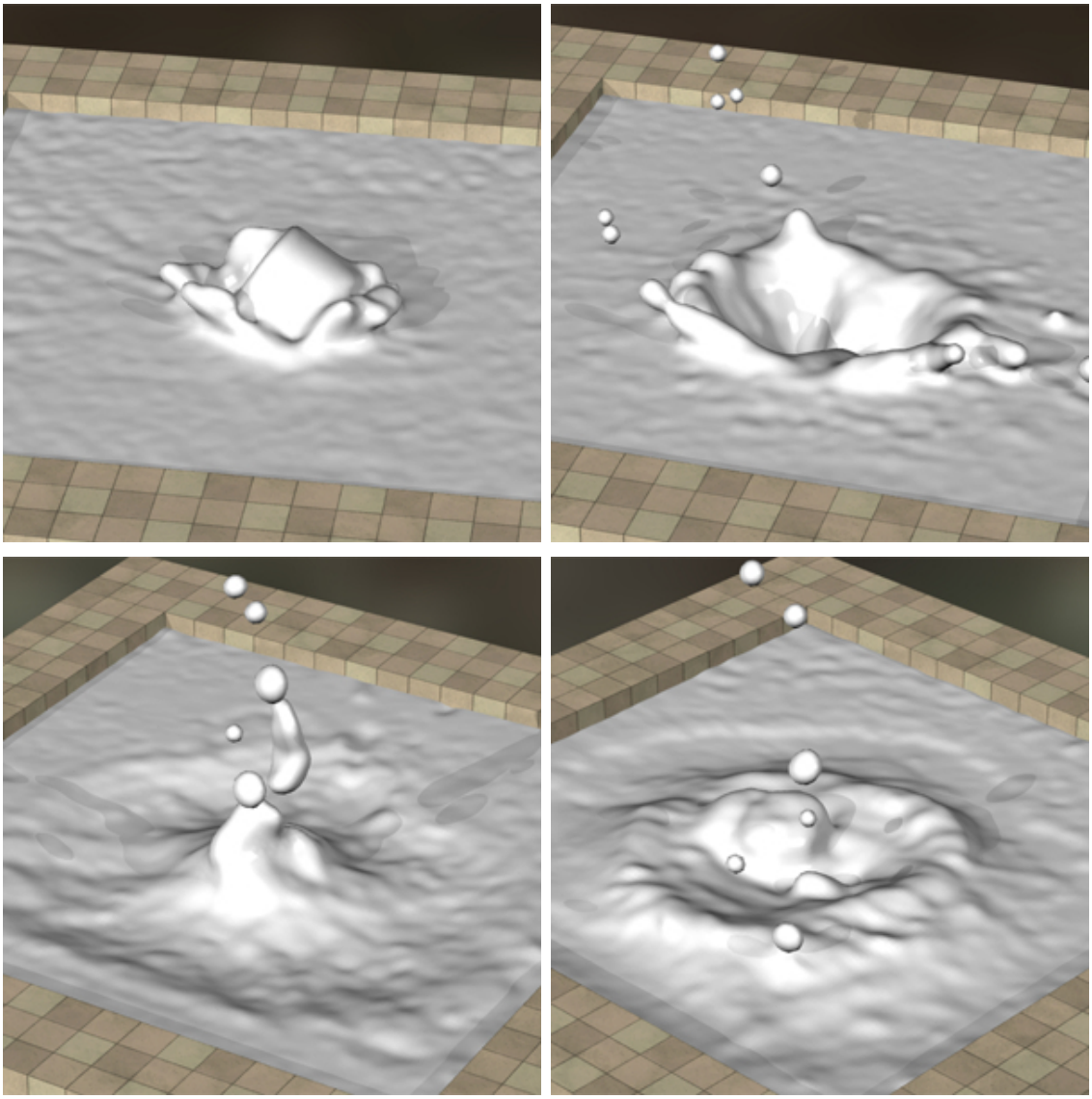


Figure 44: The splash example.

Figure 45: Water regions beyond the spatial grid (in red) can be easily generated by spatio-temporal extrapolation. We begin with data only in the small vertical region shown at the left, and we create a final sequence with a larger vertical component (right).

CHAPTER VI

CONCLUSION

Three water animation techniques are described in this thesis, targeting toward different aspects in natural phenomena modeling problems.

6.1 Water Drops on Surfaces

We have presented an algorithm to solve the capillary solid coupling problem by modeling surface tensions between the liquid and solid object. The virtual surface method replaces the liquid-solid surface by a virtual surface beneath the solid surface so that the estimated boundary pressure can represent all surface tensions on the contact front. We use a dynamic contact angle model to choose different stable contact angles according to the contact fronts velocity and surface wetness. Our results show that the algorithm is robust, accurate, and ready to be incorporated into level set fluid solvers.

Although we use a sparse grid representation for our simulations, the memory and the computing times used for simulating smallscale fluid phenomena are still large. We plan to concentrate much of our future effort on further reducing these computational costs. Related to this is the issue of maintaining fine details of the fluid surface. Since the virtual surface method depends on sampling the signed distance function represented by the grid cells, the grid domain needs to be sufficiently refined in order to keep surface details. An octree structure might be useful for representing cells on the contact front. Another possible way to recover the surface details might be to use the particles from the particle level set method, possibly by reconstructing a point set surface.

6.2 General Shallow Wave Equations

We have presented a new physically based framework to efficiently simulate small-scale 3D water flows on solid surfaces under shallow wave assumptions. This framework is governed by the General Shallow Wave Equations (GSWE) and it is based on a height field representation. We have developed several techniques within this framework including an implicit gravity scheme, an implicit surface tension scheme and two-way fluid/rigid body coupling. Our experiments show that our system is fast, stable, straightforward to implement on both CPUs and GPUs. The approach is flexible enough to produce multiple effects such as water waves, rivulets, water drop effects, fluid/rigid body coupling and interactive control and shape modeling.

Looking into the future, our short-term plan includes developing a GPU matrix solver for curved surfaces and incorporating our algorithm into an interactive media painting system. system. In the long term, we would like to study how to combine height field techniques with particle systems and grid systems so that non-height-field effects can be efficiently produced as well. Motion reduction of the height field provides another possibility to explore in the future.

6.3 Physically-Guided Liquid Surface Modeling from Videos

We have presented a hybrid framework to efficiently reconstruct realistic water animation from real water scenes by using video-based reconstruction techniques together with physically-based surface optimization. Using depth maps from stereo vision, the novelties of our framework include a surface initialization algorithm to create a rough initial guess of the surface sequence, a 3D flow estimation method to define temporal coherence, and a surface optimization method to remove false-positives, enforce temporal coherence and complete false-negatives.

Looking towards the future, a more robust acquisition system and algorithms that

do not require the dying of water is our first priority. Solving this will make it practical to capture the complex shapes and interactions of real water in our everyday environment, not just in a laboratory. Since our method does not require feature tracking in video, we are optimistic that some active sensing methods may be developed. In addition, we are also interested in finding a more memory efficient data structure for the grid representation and in creating a GPU implementation of our algorithm for acceleration. We speculate that there may be a way to more closely combine the physically-based optimization and video-based reconstruction. We are planning to explore other interesting fluid effects, including viscoelastic fluid and gas animation. Providing more flexible tools based on this algorithm is also an interesting topic that we plan to study in the future, and this will help artists to design specific water animation effects via editing the end visual images directly, rather than setting up initial conditions and various force controls.

More broadly, we believe this combination of reconstruction with simulation can be extended to model many dynamic real-world objects with the promise of *significantly* reducing the amount of captured samples. Ultimately we hope that the acquisition of 4D models can be as simple as sweeping a camera around – something currently limited to static scenes only. Adding simulation into the pipeline is, in our view, a very promising direction.

APPENDIX A

VALIDATION FOR THE VIRTUAL SURFACE METHOD

CLAIM 1: Given any point P and its closest point $L(t_0)$ on $L(t)$, let $V(s_0, t_0)$ be a point on Ω_v so that $P - V(s_0, t_0)$ is perpendicular to $R(t)$. If $1 + s\cos\theta_s|L''(t)| > 0$ and $N_s(t)$ is constant, $V(s_0, t_0)$ is a closest point to P.

PROOF: Let $SD(t)$ and $SD(s, t)$ be squared Euclidean distance functions from P to $L(t)$ and $V(s, t)$ respectively. $L(t_0)$ is the closest point to P, so: $SD'(t0) = -2[P - L(t_0)] \cdot L'(t_0) = 0$ and $SD''(t0) = -2[P - L(t_0)] \cdot L''(t_0) + 2 > 0$. Let $k(t)$ be a scalar function $1 + s\cos\theta_s|L''(t)|$, we see:

$$\partial V(s, t)/\partial t = L'(t) - s\cos\theta_s(N_s \times L''(t)) = k(t)L'(t)$$

$k(t) > 0$ for all t is satisfied if $L(t)$ is sufficiently smooth ($|L''(t)|$ is small) and if $|\cos\theta_s|$ is small enough. Geometrically, this means the virtual surface has no self-intersections so that $V(s_0, t_0)$ and $L(t_0)$ share the same tangent direction $L'(t)$. For $SD(s, t)$, we get:

$$\begin{aligned}
\tfrac{\partial SD(s_0,t_0)}{\partial s} &= -2[P - V(s_0, t_0)] \cdot R(t_0) && = 0 \\
\tfrac{\partial SD(s_0,t_0)}{\partial t} &= -2k(t_0)[P - V(s_0, t_0)] \cdot L'(t_0) && = 0 \\
\tfrac{\partial^2 SD(s_0,t_0)}{\partial s^2} &= 2|R(t_0)|^2 && > 0 \\
\tfrac{\partial^2 SD(s_0,t_0)}{\partial s\partial t} &= 2k(t_0)R(t_0) \cdot L'(t_0) && = 0
\end{aligned}$$

$$\begin{aligned}
\tfrac{\partial^2 SD(s_0,t_0)}{\partial t^2} &= -2[k(t_0)(P - V(s_0, t_0)) \cdot L'(t_0)]' \\
&= -2k(t_0)[(P - V(s_0, t_0)) \cdot L'(t_0)]' \\
&= k(t_0)(SD''(t_0) - 2s\sin\theta_s N_s \cdot L''(t_0) \\
&\quad -2s\cos\theta_s((N_s \times L'(t_0)) \cdot L''(t_0) - |L''(t_0)|)) \\
&= k(t_0)SD''(t_0) > 0
\end{aligned}$$

since $(N_s \times L'(t)) \cdot L''(t) = |L''(t)|$. So $V(s_0, t_0)$ is a local minimum of $SD(s,t)$.

CLAIM 2: If $L(t_0)$ is the closest contact point to $C_{x,0,z}$, then $L(t_0)$ is also the closest contact point to $C_{x,-1,z}$.

PROOF: The node $C_{x,-1,z}$'s position is: $C_{x,-1,z} = C_{x,0,z} - hN_s$, where N_s is the Y direction in this case. Let $SD_0(t)$ and $SD_{-1}(t)$ be the squared distances to two nodes respectively, $SD_{-1}(t)$ satisfies:

$$\begin{aligned} SD'_{-1}(t) = &\quad SD'_0(t) + hN_s \cdot L'(t_0) = 0 \\ SD''_{-1}(t) = &\quad -2(C_{x,-1,z} - L(t_0)) \cdot L''(t_0) + 2(L'(t_0)) \cdot L'(t_0) \\ = &\quad SD''_0(t_0) + 2hN_s \cdot L''(t_0) = SD''_0(t_0) > 0 \end{aligned}$$

Therefore, $L(t_0)$ is also the closest contact point to $C_{x,-1,z}$.

REFERENCES

[1] Allen, B., Curless, B., and Popović, Z., "Articulated body deformation from range scan data," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 612–619, 2002.

[2] Anguelov, D., Koller, D., Pang, H.-C., Srinivasan, P., and Thrun, S., "Recovering articulated object models from 3d range data," in *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, (Arlington, Virginia, United States), pp. 18–26, AUAI Press, 2004.

[3] Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., and Davis, J., "Scape: shape completion and animation of people," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 408–416, 2005.

[4] Atcheson, B., Ihrke, I., Heidrich, W., Tevs, A., Bradley, D., Magnor, M., and Seidel, H.-P., "Time-resolved 3d capture of non-stationary gas flows," in *Proc. of ACM SIGGRAPH Asia 2008*, vol. 27, 2008.

[5] Baraff, D., "Rigid body simulation," in *Physically Based Modeling: SIGGRAPH 2001 Course 25*, 2001.

[6] Bargteil, A. W., Wojtan, C., Hodgins, J. K., and Turk, G., "A finite element method for animating large viscoplastic flow," in *Proc. of ACM SIGGRAPH 2007*, vol. 26, 2007.

[7] Bhat, K. S., Seitz, S. M., Hodgins, J. K., and Khosla, P. K., "Flow-based video synthesis and editing," in *Proc. of ACM SIGGRAPH 2004*, pp. 360–363, 2004.

[8] Bolz, J., Farmer, I., Grinspun, E., and Schröder, P., "Sparse matrix solvers on the *GPU*: conjugate gradients and multigrid," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 917–924, 2003.

[9] Bradley, D., Popa, T., Sheffer, A., Heidrich, W., and Boubekeur, T., "Markerless garment capture," in *Proc. of ACM SIGGRAPH 2008*, vol. 27, 2008.

[10] Breen, D. E. and Whitaker, R. T., "A level-set approach for the metamorphosis of solid models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 2, pp. 173–192, 2001.

[11] Bridson, R., Marino, S., and Fedkiw, R., "Simulation of clothing with folds and wrinkles," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, (New York, NY, USA), p. 3, ACM, 2005.

[12] Bussmann, M., Mostaghimi, J., and Chandra, S., "On a three-dimensional volume tracking model of droplet impact," *Phys. Fluids*, vol. 11, p. 1406, 1999.

[13] Carlson, M., Mucha, P. J., Van Horn III, R. B., and Turk, G., "Melting and flowing," in *Proc. of SCA '02*, pp. 167–174, 2002.

[14] Chazel, F., Lieutier, A., and Rossignac, J., "Orthomap: Homeomorphism-guaranteeing normal-projection map between surfaces," no. GIT-GVU-04-28, 2004.

[15] Chen, J. X., da Vitoria Lobo, N., Hughes, C. E., and Moshell, J. M., "Real-time fluid simulation in a dynamic virtual environment," *IEEE Comput. Graph. Appl.*, vol. 17, no. 3, pp. 52–61, 1997.

[16] Cohen, J. M. and Molemaker, M. J., "Practical simulation of surface tension flows," in *SIGGRAPH Sketches*, p. 70, 2004.

[17] Cong, G. and Parvin, B., "A new regularized approach for contour morphing," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 1, pp. 458–463 vol.1, 2000.

[18] de Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.-P., and Thrun, S., "Performance capture from sparse multi-view video," in *Proc. of ACM SIGGRAPH '08*, (New York, NY, USA), pp. 1–10, 2008.

[19] de Gennes, P., "Wetting: Statics and dynamics," *Rev. Mod. Phys.*, vol. 57, no. 3, pp. 827–863, 1985.

[20] Dorsey, J., Pedersen, H. K., and Hanrahan, P., "Flow and changes in appearance," in *Proc. of ACM SIGGRAPH '96*, pp. 411–420, 1996.

[21] Egnal, G. and Wildes, R. P., "Detecting binocular half-occlusions: Empirical comparisons of five approaches," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 8, pp. 1127–1133, 2002.

[22] Elcott, S., Tong, Y., Kanso, E., Schröder, P., and Desbrun, M., "Stable, circulation-preserving, simplicial fluids," *ACM Trans. Graph.*, 2007.

[23] Enright, D., Nguyen, D., Gibou, F., and Fedkiw, R., "Using the particle level set method and a second order accurate pressure boundary condition for free surface flows," in *Proc. of the 4th ASME-JSME Joint Fluids Eng. Conf., FEDSM2003-45144*, pp. 1–6, 2003.

[24] Enright, D., Marschner, S., and Fedkiw, R., "Animation and rendering of complex water surfaces," in *Proc. of ACM SIGGRAPH '02*, pp. 736–744, 2002.

[25] Fattal, R. and Lischinski, D., "Target-driven smoke animation," in *Proc. of ACM SIGGRAPH 2004*, 2004.

[26] Fedkiw, R., Stam, J., and Jensen, H. W., "Visual simulation of smoke," in *Proc. of ACM SIGGRAPH 2001*, (New York, NY, USA), pp. 15–22, ACM, 2001.

[27] Feldman, B. E., OBrien, J. F., and Klingner, B. M., "Animating gases with hybrid meshes," in *Proc. of ACM SIGGRAPH '05*, 2005.

[28] Felzenszwalb, P. F. and Huttenlocher, D. P., "Efficient belief propagation for early vision," *Int. J. Comput. Vision*, vol. 70, no. 1, pp. 41–54, 2006.

[29] Feng, Z. G., Domaszewski, M., Montavon, G., and Coddet, C., "Finite element analysis of effect of substrate surface roughtness on liquid droplet impact and flattening process," *Journal of Thermal Spray Technology*, vol. 11, no. 1, pp. 62–68, 2002.

[30] Foster, N. and Fedkiw, R., "Practical animation of liquids," in *Proc. of SIGGRAPH '01*, pp. 23–30, 2001.

[31] Foster, N. and Metaxas, D., "Realistic animation of liquids," *Graph. Models Image Process.*, vol. 58, no. 5, 1996.

[32] Fournier, P., Habibi, A., and Poulin, P., "Simulating the flow of liquid droplets," in *Graphics Interface*, p. 133, June 1998.

[33] Grant, I., "Particle image velocimetry: a review," in *Proc. of the Institution of Mechanical Engineers*, vol. 211, p. 55C76, 1997.

[34] Grinspun, E., Schröder, P., and Desbrun, M., "Discrete differential geometry: An applied introduction." SIGGRAPH Course #14, 2005.

[35] Gu, J., Nayar, S. K., Grinspun, E., Belhumeur, P. N., and Ramamoorthi, R., "Compressive Structured Light for Recovering Inhomogeneous Participating Media," in *Europian Conference on Computer Vision (ECCV)*, Oct 2008.

[36] Guskov, I., Klibanov, S., and Bryant, B., "Trackable surfaces," in *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, (Aire-la-Ville, Switzerland, Switzerland), pp. 251–257, Eurographics Association, 2003.

[37] Hasinoff, S. W. and Kutulakos, K. N., "Photo-consistent reconstruction of semitransparent scenes by density-sheet decomposition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 5, pp. 870–885, 2007.

[38] Hawkins, T., Einarsson, P., and Debevec, P., "Acquisition of time-varying participating media," in *Proc. of ACM SIGGRAPH 2005*, 2005.

[39] He, T., Wang, S., and Kaufman, A., "Wavelet-based volume morphing," in *VIS '94: Proceedings of the conference on Visualization '94*, (Los Alamitos, CA, USA), pp. 85–92, IEEE Computer Society Press, 1994.

[40] Healy, W. M., *Modeling the Impact of a Liquid droplet on a Solid Surface*. PhD thesis, Georgia Institute of Technology, 1999.

[41] Heckbert, P., "Fast surface particle repulsion." SIGGRAPH '97: New Frontiers in Modeling and Texturing Course, pages 95–114, 1997.

[42] Hinsinger, D., Neyret, F., and Cani, M.-P., "Interactive animation of ocean waves," in *Proc. of SCA '02*, pp. 161–166, 2002.

[43] Hong, J.-M. and Kim, C.-H., "Animation of bubbles in liquid," *Comp. Graph. Forum (Eurographics Proceedings)*, vol. 22, no. 3, pp. 253–262, 2003.

[44] Houston, B., Nielsen, M. B., Batty, C., Nilsson, O., and Museth, K., "Hierarchical *RLE* level set: A compact and versatile deformable surface representation," *ACM Trans. Graph.*, vol. 25, no. 1, pp. 151–175, 2006.

[45] Hullin, M. B., Fuchs, M., Ihrke, I., Seidel, H.-P., and Lensch, H. P. A., "Fluorescent immersion range scanning," in *Proc. of ACM SIGGRAPH 2008*, 2008.

[46] Ihrke, I., Goldluecke, B., and Magnor, M., "Reconstructing the geometry of flowing water," in *ICCV '05*, (Washington, DC, USA), pp. 1055–1060, IEEE Computer Society, 2005.

[47] Ihrke, I. and Magnor, M., "Image-Based Tomographic Reconstruction of Flames," in *ACM Siggraph / Eurographics Symposium Proceedings, Symposium on Computer Animation*, pp. 367–375, 2004.

[48] Ihrke, I. and Magnor, M., "Adaptive grid optical tomography," *Graph. Models*, vol. 68, no. 5, pp. 484–495, 2006.

[49] Irving, G., Guendelman, E., Losasso, F., and Fedkiw, R., "Efficient simulation of large bodies of water by coupling two and three dimensional techniques," in *Proc. of ACM SIGGRAPH '06*, p. 38, 2006.

[50] Jiang, G.-S. and Peng, D., "Weighted eno schemes for hamilton jacobi equations," *SIAM J. Sci. Comput.*, vol. 21, pp. 2126–2143, 2000.

[51] Jiang, G.-S. and Shu, C.-W., "Efficient implementation of weighted eno schemes," vol. 126, no. 1, pp. 202–228, 1996.

[52] Kanade, T., Rander, P., Vedula, S., and Saito, H., "Virtualized reality: Digitizing a 3d time-varying event as is and in real time," in *Mixed Reality, Merging Real and Virtual Worlds*, pp. 41–57, 1999.

[53] Kaneda, K., Ikeda, S., and Yamashita, H., "Animation of water droplets moving down a surface," *The Journal of Visualization and Computer Animation*, vol. 10, no. 1, pp. 15–26, 1999.

[54] Kaneda, K., Kagawa, T., and Yamashita, H., "Animation of water droplets on a glass plate," in *Proceedings Computer Animation '93*, pp. 177–189, 1993.

[55] Kaneda, K., Zuyama, Y., Yamashita, H., and Nishita, T., "Animation of water droplet flow on curved surfaces," in *Proceedings PACIFIC GRAPHICS '96*, pp. 50–65, 1996.

[56] Kass, M. and Miller, G., "Rapid, stable fluid dynamics for computer graphics," in *Proc. of ACM SIGGRAPH '90*, pp. 49–57, 1990.

[57] Kim, B., Liu, Y., Llamas, I., and Rossignac, J., "Advections with significantly reduced dissipation and diffusion," *IEEE TVCG*, 2007.

[58] Klingner, B. M., Feldman, B. E., Chentanez, N., and O'Brien, J. F., "Fluid animation with dynamic meshes," in *Proc. of ACM SIGGRAPH 2006*, 2006.

[59] Korlie, M., "Particle modeling of liquid drop formation on a solid surface in 3-d," *Compute. and Math. with Appl.*, vol. 33, no. 9, pp. 97–114, 1997.

[60] Layton, A. T. and van de Panne, M., "A numerically efficient and stable algorithm for animating water waves," *The Visual Computer*, vol. 18, no. 1, pp. 41–53, 2002.

[61] Lerios, A., Garfinkle, C. D., and Levoy, M., "Feature-based volume metamorphosis," in *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 449–456, ACM, 1995.

[62] Losasso, F., Gibou, F., and Fedkiw, R., "Simulating water and smoke with an octree data structure," in *Proc. of SIGGRAPH '04*, 2004.

[63] Marschner, S. R., Guenter, B. K., and Raghupathy, S., "Modeling and rendering for realistic facial animation," in *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, (London, UK), pp. 231–242, Springer-Verlag, 2000.

[64] McNamara, A., Treuille, A., Popovic, Z., and Stam, J., "Fluid control using the adjoint method," in *Proc. of ACM SIGGRAPH 2004*, 2004.

[65] Mihalef, V., Metaxas, D., and Sussman, M., "Animation and control of breaking waves," in *Proc. of SCA '04*, pp. 315–324, 2004.

[66] Mitra, N. J., Flöry, S., Ovsjanikov, M., Gelfand, N., Guibas, L., and Pottmann, H., "Dynamic geometry registration," in *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, (Aire-la-Ville, Switzerland, Switzerland), pp. 173–182, Eurographics Association, 2007.

[67] Mitra, N. J., Flory, S., Ovsjanikov, M., Gelfand, N., Guibas, L., and Pottmann, H., "Dynamic geometry registration," in *Eurographics Symposium on Geometry Processing*, 2007.

[68] Monaghan, J. J., "Smoothed particle hydrodynamics," *Annual Review of Astronomy and Astrophysics*, vol. 30, pp. 543–574, 1992.

[69] Morris, N. J. W. and Kutulakos, K. N., "Dynamic refraction stereo," in *Proc. of International Conference on Computer Vision*, 2005.

[70] Morris, N. J. W. and Kutulakos, K. N., "Reconstructing the surface of inhomogeneous transparent scenes by scatter trace photography," in *Proc. of 11th Int. Conf. Computer Vision*, 2007.

[71] Mullen, P., McKenzie, A., Tong, Y., and Desbrun, M., "A variational approach to eulerian geometry processing," in *Proc. of ACM SIGGRAPH 2007*, 2007.

[72] Müller, M., Charypar, D., and Gross, M., "Particle-based fluid simulation for interactive applications," in *Proc. of SCA '03*, pp. 154–159, 2003.

[73] Neyret, F., Heiss, R., and Senegas, F., "Realistic rendering of an organ surface in real-time for laparoscopic surgery simulation," *the Visual Computer*, vol. 18, pp. 135–149, may 2002.

[74] Nguyen, D., Fedkiw, R., and Jensen, H. W., "Physically based modeling and animation of fire," in *Proc. of ACM SIGGRAPH 2002*, 2002.

[75] Nielsen, M. B. and Museth, K., "Dynamic tubular grid: An efficient data structure and algorithms for high resolution," *Journal of Scientific Computing*, vol. 26, no. 3, 2006.

[76] O'Brien, J. F. and Hodgins, J. K., "Dynamic simulation of splashing fluids," in *Computer Animation '95*, pp. 198–205, 1995.

[77] Osher, S. and Shu, C.-W., "High order essentially non-oscillatory schemes for hamilton-jacobi equations," *SIAM J. Numer. Anal.*, vol. 28, pp. 902–921, 1991.

[78] Osher, S. and Fedkiw, R., *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[79] Pharr, M. and Humphreys, G., *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 2004.

[80] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., "Numerical recipes in c: The art of scientific computing. second edition," 1992.

[81] Quan, L., Tan, P., Zeng, G., Yuan, L., Wang, J., and Kang, S. B., "Image-based plant modeling," in *Proc. of ACM SIGGRAPH 2006*, 2006.

[82] Renardy, M., Renardy, Y., and Li, J., "Numerical simulation of moving contact line problems using a volume-of-fluid method," *Journal of Computational Physics*, vol. 171, pp. 243–263, 2001.

[83] Schneider, R. and Kobbelt, L., "Geometric fairing of irregular meshes for freeform surface design," *Computer aided geometric design*, vol. 18, pp. 359–379, 2001.

[84] Schwartz, L. W. and Garoff, S., "Contact angle hysteresis on heterogeneous surfaces," *Langmuir*, vol. 1, no. 2, pp. 219–230, 1985.

[85] Sederberg, T. W. and Greenwood, E., "A physically based approach to 2–d shape blending," in *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 25–34, ACM, 1992.

[86] Selle, A., Lentine, M., and Fedkiw, R., "A mass spring model for hair simulation," in *Proc. of ACM SIGGRAPH 2008*, vol. 27, (New York, NY, USA), pp. 1–11, ACM, 2008.

[87] Sethian, J., "A fast marching level set method for monotonically advancing fronts," in *Proceedings Natl. Acad. Sci.*, vol. 93, pp. 1591–1595, 1996.

[88] Sethian, J., *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[89] Sharf, A., Alcantara, D. A., Lewiner, T., Greif, C., and Sheffer, A., "Space-time surface reconstruction using incompressible flow," in *Proc. of ACM SIGGRAPH Asia 2008*, vol. 27, pp. 1–10, 2008.

[90] Shi, J. and Tomasi, C., "Good features to track," in *Proc. of CVPR 1994*, pp. 593–600, 1994.

[91] Shi, L. and Yu, Y., "Inviscid and incompressible fluid simulation on triangle meshes," *Comput. Animat. Virtual Worlds*, vol. 15, no. 3-4, pp. 173–181, 2004.

[92] Shi, L. and Yu, Y., "Controllable smoke animation with guiding objects," *ACM Trans. Graph.*, vol. 24, no. 1, 2005.

[93] Simon, S. V., Baker, S., Seitz, S., and Kanade, T., "Shape and motion carving in 6d," in *Computer Vision and Pattern Recognition*, 2000.

[94] Sinha, S. N., Steedly, D., Szeliski, R., Agrawala, M., and Pollefeys, M., "Interactive 3d architectural modeling from unordered photo collections," *Proc. of SIGGRAPH Asia 2008*, vol. 27, 2008.

[95] Song, O.-Y., Shin, H., and Ko, H.-S., "Stable but nondissipative water," *ACM Trans. Graph.*, vol. 24, no. 1, 2005.

[96] Stam, J., "Stable fluids," in *Proc. of ACM SIGGRAPH '99*, pp. 121–128, 1999.

[97] Stam, J., "Flows on surfaces of arbitrary topology," in *Proc. of SIGGRAPH '02*, vol. 22, pp. 724–731, 2003.

[98] Staniforth, A. and Côté, J., "Semi-lagrangian integration schemes for atmospheric models," *Monthly Weather Review*, vol. 119, no. 9, p. 2206, 1991.

[99] Sun, J., Zheng, N.-N., and Shum, H.-Y., "Stereo matching using belief propagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 7, p. 787, 2003.

[100] Sussman, M. and Uto, S., "A computational study of the spreading. of oil underneath a sheet of ice," *CAM Report 98-32, University of California, Dept. of Math, Los Angeles*, 1998.

[101] Tan, P., Zeng, G., Wang, J., Kang, S. B., and Quan, L., "Image-based tree modeling," in *Proc. of ACM SIGGRAPH 2007*, 2007.

[102] Tong, R., Kaneda, K., and Yamashita, H., "A volume-preserving approach for modeling and animating water flows generated by metaballs," *The Visual Computer*, vol. 18, no. 8, pp. 469–480, 2002.

[103] Treuille, A., Lewis, A., and Popović, Z., "Model reduction for real-time fluids," *Proc. of ACM SIGGRAPH '06*, vol. 25, no. 3, pp. 826–834, 2006.

[104] Treuille, A., McNamara, A., Popović, Z., and Stam, J., "Keyframe control of smoke simulations," in *Proc. of SIGGRAPH '03*, vol. 22, pp. 716–723, 2003.

[105] Trifonov, B., Bradley, D., and Heidrich, W., "Tomographic reconstruction of transparent objects," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, (New York, NY, USA), p. 55, ACM, 2006.

[106] Tsitsiklis, J., "Efficient algorithms for globally optimal trajectories," *IEEE Trans. on Automatic Control*, vol. 40, pp. 1528–1538, 1995.

[107] Turk, G., "Generating textures on arbitrary surfaces using reaction-diffusion," in *Proc. of ACM SIGGRAPH '91*, pp. 289–298, 1991.

[108] Turk, G. and O'Brien, J. F., "Shape transformation using variational implicit functions," in *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 335–342, ACM Press/Addison-Wesley Publishing Co., 1999.

[109] Uijttewaal, W. S. and Jirka, G. H., *Shallow Flows: Research Presented at the International Symposium on Shallow Flows, Delft, Netherlands*. A.A. Balkema, 2004.

[110] Wand, M., Jenke, P., Huang, Q., Bokeloh, M., Guibas, L., and Schilling, A., "Reconstruction of deforming geometry from time-varying point clouds," in *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, (Aire-la-Ville, Switzerland, Switzerland), pp. 49–58, Eurographics Association, 2007.

[111] Wand, M., Jenke, P., Huang, Q., Bokeloh, M., Guibas, L., and Schilling, A., "Reconstruction of deforming geometry from time-varying point clouds," in *Eurographics Symposium on Geometry Processing*, 2007.

[112] Wang, H., Mucha, P. J., and Turk, G., "Water drops on surfaces," in *Proc. of ACM SIGGRAPH 2005*, 2005.

[113] Wei, Y., Ofek, E., Quan, L., and Shum, H.-Y., "Modeling hair from multiple views," in *Proc. of ACM SIGGRAPH 2005*, 2005.

[114] White, R., Crane, K., and Forsyth, D., "Capturing and animating occluded cloth," in *Proc. of ACM SIGGRAPH 2007*, 2007.

[115] Whited, B. and Rossignac, J., "B-morphs between b-compatible curves in the plane," in *Proceedings of 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, 2009.

[116] Witkin, A. and Baraff, D., "Physically based modeling: Principles and practice." Online SIGGRAPH Course notes, 1997.

[117] Wojtan, C. and Turk, G., "Fast viscoelastic behavior with thin features," in *Proc. of ACM SIGGRAPH 2008*, 2008.

[118] Xiao, J., Fang, T., Tan, P., Zhao, P., and Quan, L., "Image-based facade modeling," *Proc. of SIGGRAPH Asia 2008*, vol. 27, 2008.

[119] Yang, Q., Yang, R., Davis, J., and Nister, D., "Spatial-depth super resolution for range images," in *Proc. of CVPR 2007*, vol. 0, pp. 1–8, 2007.

[120] Yu, Y.-J., Jung, H.-Y., and Cho, H.-G., "A new water droplet model using metaball in the gravitational field," *Computer and Graphics*, vol. 23, pp. 213–222, 1999.

[121] Zhang, H., Sheffer, A., Cohen-Or, D., Zhou, Q., van Kaick, O., and Tagliasacchi, A., "Deformation-driven shape correspondence," 2008.

[122] Zhang, L., Snavely, N., Curless, B., and Seitz, S. M., "Spacetime faces: high resolution capture for modeling and animation," in *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, (New York, NY, USA), pp. 548–558, ACM, 2004.

[123] Zhao, H.-K., Merriman, B., Osher, S., and Wang, L., "Capturing the behavior of bubbles and drops using the variational level set method," *Computational Physics*, vol. 143, pp. 495–518, 1998.

[124] Zitnick, C. L., Kang, S. B., Uyttendaele, M., Winder, S., and Szeliski, R., "High-quality video view interpolation using a layered representation," in *Proc. of ACM SIGGRAPH '04*, (New York, NY, USA), pp. 600–608, 2004.

[125] Zitnick, C. L., Kang, S. B., Uyttendaele, M., Winder, S., and Szeliski, R., "High-quality video view interpolation using a layered representation.," *ACM Transactions on Graphics,*, vol. 23, no. 3, pp. 600–608, 2004.

INDEX

VITA

Huamin Wang was born in Lishui, a small town in Zhejiang Province, in the southeast part of China on May 19th, 1981, to Zhencai Wang and Yunmei Wang. Before he could walk and talk, the family moved to the city of Nanjing, and at the age of 7, the family moved again to the city of Hangzhou, where he began his school years. After completing his middle school work one year earlier than most other students at the high school attached to Zhejiang University, he enrolled at Zhejiang University as an undergraduate student in the mixed class, an honored program for both science and engineering students to achieve research capabilities. He graduated in June 2002 with a Bachelor of Engineering degree in Computer Science, and shortly after that, he started as a Master student at Stanford University, under the supervision of Prof. Leonidas J. Guibas and Prof. Ron Fedkiw. In 2005, he completed his work with a Master of Science degree and he became a PhD student in the computer graphics group, College of Computing at Georgia Institute of Technology. Working with his PhD advisor Prof. Greg Turk, he was able to pursue his broad research interests in computer graphics, computer vision, numerical methods and image processing algorithms related to graphics problems. In summer 2006, he worked as a research intern with Dr. Gavin Miller in the advanced Technology Lab at Adobe System Incorporated for faster water drop simulation, which was ended in one research publication and one patent *System and Method for Simulating Shallow Wave Effects on Arbitrary Surfaces*. In summer 2007, he worked as a research intern with Dr. Hugues Hoppe in Microsoft Research on image factorization problems. This work was published in proceedings of SIGGRAPH 2008 and a patent was also filed later based on it. He continued his work in Microsoft Research Asia with Dr. Kun Zhou, where he had

a great chance to meet with pioneer computer graphics researchers in China and to know the recent research progress in Asia. He started working on his thesis after he returned to Atlanta from China in early 2008 and this thesis was finally completed in Summer 2009, with all the support and help from his advisor, colleagues, friends and family.

Permanent Address:

Qingfeng Xincun 6, 1-402

Hangzhou, China, 310013

This thesis was typed by the author.