SIMULATION OF FLUIDS WITH REDUCED DIFFUSION, THIN LIQUID FILMS, VOLUME CONTROL, AND A MESH FILTER IN RATIONAL FORM

A Thesis
Presented to
The Academic Faculty

by

Byungmoon Kim

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
Computer Science

Georgia Institute of Technology
December 2006

SIMULATION OF FLUIDS WITH REDUCED DIFFUSION, THIN LIQUID FILMS, VOLUME CONTROL, AND A MESH FILTER IN RATIONAL FORM

Approved by:

Professor Jarek Rossignac,
Committee Chair
Computer Science
Georgia Institute of Technology

Professor Greg Turk
Computer Science
Georgia Institute of Technology

Professor Jarek Rossignac, Advisor
Computer Science
Georgia Institute of Technology

Professor Irfan Essa
Computer Science
Georgia Institute of Technology

Professor Yingjie Liu
School of Mathematics
Georgia Institute of Technology

Professor Xiangmin Jiao
Computer Science
Georgia Institute of Technology

Date Approved: 10 November 2006

ACKNOWLEDGEMENTS

This thesis could not have been completed without the support, guidance, and care of my advisor, Jarek Rossignac. I thank him for his patience when I was struggling with various topics during my research.

I would also like to express my gratitude for the rest of my committee members, Professor Yingjie Liu, Professor Irfan Essa, Professor Greg Turk, and Professor Xiangmin Jiao. Yingjie provided me with not only research directions but also warm encouragements and cares for my future career. Irfan and Greg assisted me with friendly advices and showed me how I can grow as a researcher in graphics area. While I was invited to their groups, I could observe how a leading research topics are found, studied, and evaluated by the graphics community. Xiangmin provided me with an important insight that can further strengthen a research topic examined in this thesis.

Thanks are also due to my masters' degree advisor Panagiotis Tsiotras who taught me the importance, value, and joy of theory and mathematics. I wish to thank Georgia Tech Professors who provided valuable classes. From the classes I took, I discovered a mountain of knowledge and the joy of learning, and strengthened my desire to learn, which will last for the rest of my life.

I would like to mention that, through out my study at Georgia Tech, I enjoyed studying and collaborating with my lab mates, Lorenzo, Ignacio, Justin, Jason, Brian, Jerry, James, Brooks, and other students in GVU lab.

Finally and most importantly, I thank my wife, Jin Young, for her love and effort in raising our two sons Justin Donghwan and Michael Jihwan. I am grateful to my parents, parent-in-laws, sisters and brothers for their lifelong supports during my study. To them, I dedicate this thesis.

This research was supported by the NSF under the ITR Digital clay grant 0121663 and the NSF DMS 0511815. I would like to express my appreciation for the sponsors.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

This thesis is concerned with the evolution of implicit or explicit surfaces.

The first part of this thesis addresses three problems in fluid simulation: advection, thin film, and the volume error. First, we show that the back and forth error compensation and correction (BFECC) method can significantly reduce the dissipation and diffusion. Second, thin film is hard to simulate since it has highly complex liquid/gas interface that requires high memory and computational costs. We address this difficulties by using cell centered octree grid to reduce memory cost and a multigrid method to reduce computational cost. Third, the volume loss is an undesired side effect of the level set method. The known solution to this problem is the particle level set method, which is expensive and has small but accumulating volume error. We provide a solution that is computationally effective and can prevent volume loss without accumulation.

The second part of this thesis is focused on filtering a triangle mesh to produce a mesh whose details are selectively reduced or amplified. We develop a mesh filter with a rational transfer function, which is a generalization from previously developed mesh filters. In addition, we show that the mesh filter parameters can be computed from the physical size of mesh feature.

CHAPTER I

INTRODUCTION

1.1 Overview

In most applications of computer graphics, objects and their evolution are modeled in terms of their bounding surfaces. Such surfaces may be represented in various ways. Many representation schemes approximate the surface with piecewise simple patches. The simplest form of such a patch is a triangle. Hence, the most popular representation in computer graphics is a triangle mesh, which represents the surface by a finite set of sample points, called vertices, and by a connectivity graph, often stored as some form of a triange/vertex incidence list. We view such models as explicit representations of the surface.

In contrast, one may chose to use an implicit representation, where the surface is defined as the zero-crossing of some implicit function. A popular implicit representation is used in the level set method, where the implicit function is an approximation of the signed distance to the desired surface and is defined by an interpolation of distance values sampled along a regular or irregular lattice.

This thesis is concerned with techniques that evolve these surfaces to alter their shape or to produce physically realistic animations. These two (explicit and implicit) surface representations offer different advantages and drawbacks for computing such evolutions.

Explicit representations can be easily evolved by updating the vertex locations. Such an evolution may be guided by measures of local curvature or other differential properties and may be applied precisely to the vertices. However, the proper evolution of an explicit surface model requires complex and costly processing to detect and prevent or to resolve self-intersections or to adjust the sampling frequency in stretched areas.

In contrast, the level set method represents the surface as zero contour, or zero level set,

of the scalar field. Therefore, the surface may be moved implicitly by simply adjusting the scalar field. Resampling and topological changes are implicit and need not be computed. If an explicit representation of a particular state of the surface (frame) is needed, it may be extracted by an iso-surface extraction process. However, it may prove difficult to evolve an implicit surface precisely, due to the limited sampling resolution of the lattice.

Those different properties of the two surface representations make them suitable for different applications.

To advantages of explicit meshes have made them popular in the graphic and animation community. Firstly, the frames of an explicit mesh representations may be quickly rendered, while the implicit representation are typically first converted into their explicit form through a slow iso-surface extraction process, and hence cannot be rendered in realtime. Secondly, an explicit mesh has low memory cost and provides a high accuracy for modeling the desired surface, while the implicit representation requires a significantly higher storage and offers less precision.

Among the numerous techniques for mesh processing tasks, an important one, studied in this thesis, is the filtering of the mesh, either to remove noise produced by an imperfect acquisition process or to smooth small details which may impede fast transfer or analysis.

In contrast, implicit level set representations are popular when one wants to track surfaces that undergoes topological changes or to simplify the topology of a surface. For example, a level set approach was proposed for surface editing [47]. Level set representations have also been used for shape segmentation and for fitting a surface to an unstructured point cloud [52, 60]. Since the topology of the final surface is initially not known, an evolution without topological constraints is needed. The level set method fits well to this requirement.

Finally, the level set representation is used heavily to track the boundary between fluid and air or between two different fluids in fluid simulations. Since these boundaries typically undergo significant topological changes during the simulation, the level sets representations are preferred by most researchers.

In this thesis, we use both of these implicit and explicit surface models. First, for the implicit model, we use the level set model to represent liquid/gas interface and even to represent the thin water film that separates two air bubbles in foam. Second, for the explicit model, we choose the triangle mesh model, and study the generalization of the linear mesh filter and the computation of filter coefficients.

The thesis is organized as follows. In the reminder of the introduction, we provide a brief overview of the problems that we have addressed and of our main contributions.

Then, in Chapter 2, we explore the advection problems in the fluid simulation. The first order semi-Lagrangian method has severe diffusion and dissipation. This leads to damped fluid motion, loss of smoke density, loss of texture detail, and volume loss. We show that BFECC method can reduce these phenomena significantly. This chapter is based on our previous publications [37, 38]

In Chapter 3, we investigate the simulation of thin liquid films, which is need to simulate bubbles. We propose to use cell centered octree grid and multi grid method to reduce memory and computation costs.

In Chapter 4, we study the volume loss problem, that comes with the level set advection method. We show that the volume error can be corrected by applying carefully computed divergence.

In Chapter 5, we explore the mesh filter that has a transfer function in rational form. In addition, we show that the coefficients of the filter can be computed from mesh feature size. This chapter is based on out previous works [39].

1.2 Advections with Significantly Reduced Dissipation and Diffusion

In computer graphics applications, such as fluid simulation and vector field visualization, various properties, including velocity vector components, smoke density, level set values, and texture or dye colors, must often be transported along some vector field. Those transportation problems, referred to as *advection* problems, can be performed on various grids

such as uniform and adaptive grids or triangulated surfaces. Here are five common uses of advection in computer graphics:

- *Velocity advection* transports the velocity field along the velocity itself. This step is required in all non-steady flow simulation based on the Navier-Stokes equation.
- *Smoke density advection* transports smoke along the velocity field.
- Sometimes, we may want to advect a colored image, which may be thought of as texture or colored smoke. We call this process *image advection*.
- When a level set method [51] is used to simulate a free surface or a two-phase flow such as a water surface simulation, the level set values must be transported as well. We refer to this process as *level set advection*.
- A vector field may be visualized by advecting dye on the vector field. We call this process *dye advection*.

Advection steps can be implemented by an upwind [53, 60] or a semi-Lagrangian [65] method. Due to its stability for large time steps, the latter is often preferred. These two methods can be implemented with various order of accuracies, for example, first, second, and higher-order accuracies. Despite their low accuracy, first-order methods are popular in computer graphics because of their simplicity. However, lack of accuracy in first-order methods results in a significant amount of numerical diffusion and dissipation. For example, in velocity advection, fluid motion is dampened significantly, which may remove small scale and even large scale motions. In smoke density advection, a premature dilution of smoke occurs, preventing simulation of weakly dissipative and diffusive smoke. In level set advection, significant volume loss takes place. In dye advection, diffusion causes blur and the dissipation produces dark patterns or even an early termination of the trajectory.

Researchers have proposed solution to each of these problems. For dye and texture advection, combining first-order advection and particles increased the accuracy [33]. For

level set advection, the particle level set method [17] produces little volume loss. For smoke advection, cubic interpolation reduces diffusion and dissipation [21]. For velocity advection, small scale motions can be maintained by adding vorticity [21, 59], by using particles [59, 54]. All of these solutions can be considered as problem-specific enhancements of the first-order advection. We notice that the FLIP method, introduced recently in [80], advects properties with particle, and therefore, may be applied to any advection achieving zero dissipation. However, in advecting dye or smoke from a source, new particles may have to be created as smoke or dye volume grows. In contrast, a purely Eulerian-based high-order advection method can reduce dissipation and diffusion significantly while taking advantage of the simplicity of the Eulerian grid.

There are many such high order methods for improving the accuracy in the advection steps, such as the WENO scheme [43, 32] and the CIP method [70, 64]. Generally speaking, higher order methods are more difficult to implement, in particular for non-uniform and adaptive meshes. Also because the solutions may contain singularities, special treatments are usually necessary. In [17], Enright et. al. made the particle level set method [18] more efficient and easier to implement by using a first order semi-Lagrangian method to compute the level set equation while propagating the particles with higher order methods, which produces high resolution near interface corners. Local mesh refinement near non-smooth regions of the solutions is an effective technique for improving the accuracy. However, it increases the implementation complexity, particularly when a sophisticated underlying numerical scheme is used. There is also a trade-off between the levels of adaptive mesh refinement and the formal order of accuracy of the underlying scheme. We propose to improve on all the issues addressed above and demonstrate the accuracy of our methods for a number of problems. The underlying scheme we use for computing the advections is the "back and forth error compensation and correction" algorithm (BFECC) [13] and [15]. When applied to the first order semi-Lagrangian Courant-Isaacson-Rees (CIR) scheme, this BFECC method has second order formal accuracy in both time and space. It essentially

calls the CIR scheme 3 times during each time step, and thus maintains the benefits of the CIR scheme such as the stability with large time steps, convenience for use in non-uniform meshes, and low cost in computation.

In [13, 15], the authors proposed BFECC as an alternative scheme for interface computations using the level set method and have tested it on the Zalesak's disk problem and simple interface movements with static or constant normal velocity fields on uniform meshes. We have found it beneficial to adapt the method to level set advection in fluid simulations that contain complicated dynamically varying velocity fields. It is also useful to further apply the method to other types of advections.

We apply BFECC to the various advection problems mentioned earlier and show that BFECC provides significant reduction of dissipation and diffusion for all these applications. The ideas presented here were introduced in a workshop paper [37]. Here, we provide a more detailed treatment of the proposed solution, and new examples of applications to dye advection, advections on a triangle mesh, and on an adaptive quad-tree mesh.

1.3 Simulation of Thin Liquid Films

In real fluids, we often observe a bubble rising to surfaces, floating around for a while, and then bursting or merging. In soap water, the bubble will last much longer than in pure water. When multiple bubbles are rising, the bubbles will interact with other bubbles and liquids. If a large number of bubbles are rising, and if they do not burst, they will stack, forming the *wet foam*. The water between those stacked bubbles will drain, leaving micrometer-thin films of liquid between bubbles. This is called the *dry foam*. The micrometer-thin film maintains its thickness due to the disjoining pressure, which is a result of various molecular interactions. The simulation of liquids with thin film is still a challenging open problem.

Bubbles with thin films have a very complex interface, which can lead to high memory and computation costs. We propose methods that can reduce those costs. First, to reduce memory cost, we collocate velocity, pressure, and level set variables at the octree center.

Second, to reduce the computational cost, we apply a multigrid method, which greatly reduces the computation time when the interface is extremely complex. In addition, thanks to the volume control discussed in section 4, volume of fluid is well preserved or controlled. Therefore, we do not use more expensive volume preserving method such as particle level set method.

In addition, we provide future directions to address two additional problems in the simulation of thin liquid films.

Thin liquid films occur in bubbles, or in liquid splash. Films thinner than grid resolution quickly ruptures, making bubbles burst earlier than desired, or thin films in splash simulation rupture earlier than desired. This premature rupture hampers simulations of bubbles or splash with thin liquid films. Therefore, preventing thin films from rupture and making them last longer is essential in the simulation of bubble and thin film. We propose an idea that can be used to increase the life span of thin liquid films.

Another difficulty in the simulation of a thin film is the computation of the surface tension. When the level set method is used to represent liquid and gas domains, surface tension is computed by differentiating the level set function, which is differentiable near the interface. However, in the thin film, the level set is locally singular and hence not differentiable. Curvature computed by differentiating the level set function near the thin film is noisy. When this noisy curvature is used to compute the surface tension, the thin film breaks quickly. In this thesis, we discuss a method to compute surface tension without using the level set gradient.

1.4 Controlling Fluid Volume

The fluid simulations using level set method are subject to volume error, which typically occurs as volume loss of bubbles or liquid drops. In particular, our bubble simulation suffered from volume loss. The known solution is using the particle level set method, which can reduce volume loss down to negligible amount in staggered grid. However, in a

very long simulation, the volume error may be visible. In non-staggered grids, the pressure projection step produces a vector field whose divergence is $O(\Delta x^2)$. This truncation error can cause additional volume error that does not exist in the staggered grid. In addition, at the wall boundary, velocity across the wall cannot be exactly enforced to be zero, making very slow flow across wall boundary. If one runs simulation for a long time, this error can be accumulated to a visible amount of volume gain or loss. Therefore, correction of these volume error is necessary.

Our solution is computing the volume error of each fluid region and then modifying the pressure projection step so that the projected velocity field has nonzero divergence. To the region that lost volume, we apply positive divergence to inflate that region. To the region that gained volume, we apply negative divergence to deflate that region. Thus, the volume error can be corrected by the divergence. However, a care must be taken when one compute the desired divergence. If the divergence is too small, the volume error will not be corrected. If the divergence is too large, the simulation may suffer artifacts and even become unstable. Therefore, the divergence must be computed carefully.

In computing the divergence, we first start from modeling the volume change equation using the divergence theorem. Although the resulting volume equation is nonlinear, we can design a nonlinear feedbacks that yield linear equation that is easy to analyze. From the time response of the linear equation, we design the gains that corrects the volume error quickly and stably. Finally, we validate this by several experiments.

1.5 Mesh Filter

Triangle meshes obtained using sensors such as stereo cameras, 3D laser scan, etc. are subject to high frequency noise. To reduce such noise, researchers have developed various mesh filter algorithms. Mesh filters in [72, 11, 78, 50] reduce eigen modes associated with high frequency spectrum of the Laplacian operator by using a transfer function [72]. One can design the transfer function to decay as a function of frequency to obtain a linear filter.

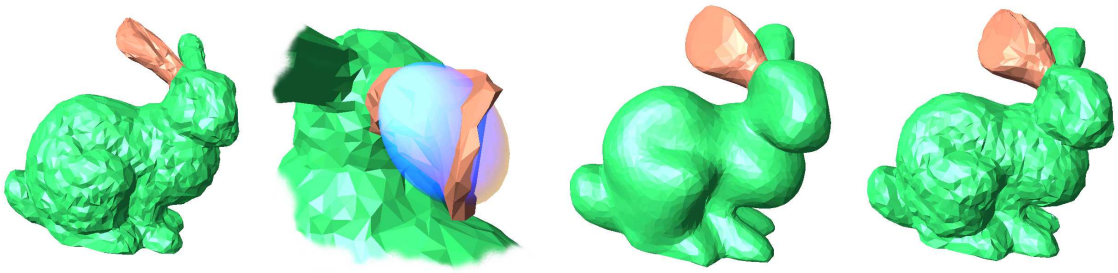

Figure 1: The user first selects a feature (first). The dimensions of the feature, shown by an approximating ellipsoid (second), are computed, whose frequencies are used to set the filter gains. In the next two images, band exaggeration filters are applied to grow the ear, while the higher frequency bumps may be smoothened out (third) or preserved (forth) by varying other filter parameters. The filter parameters are: $s_1 = 20, s_2 = 25, G_0 = 1, G_1 = 2$ and $G_\infty = 0$ (third), 0.9 (forth)

The first mesh filter using this idea had a polynomial transfer function [72] that does not require a matrix solver. Since the filtered vertex location can be written in an explicit form, this filter is called an explicit filter. The explicit mesh filter introduced by Taubin [72] has been subsequently generalized to a bandpass filter [71]. Implicit forms of mesh filters followed [11] and [78]. We propose a generalization of these frameworks that combine both explicit and implicit formulations into a more flexible second order filter. More importantly, we propose a filter, whose frequencies can be extracted automatically from the physical dimensions of a user-selected mesh feature. As illustrated in Fig. 1, the user selects a feature of the mesh (such as a nose, ear, or noise bump) by spraying and diffusing paint on it. The GeoFilter system that we have developed computes automatically an ellipsoid approximating selected feature. The dimensions of this ellipsoid guide the user in adjusting the filter parameters so as to achieve the desired result.

Filters are used profusely for audio or image signals that are regularly sampled over time or space. Since these domains are already Euclidean, regular samplings can be easily expressed in mathematical form. When constructing filters for such signals, one can directly use the property of the analog signal. For example, when the frequency greater than 10KHz needs to be attenuated, one can use the lowpass filter that has 10KHz as cutoff frequency.

In contrast, the quantitative relation between filter frequency and size of mesh feature has been overlooked in mesh filtering. Therefore, when one wants to attenuate a noisy pattern of size 0.1, one cannot directly use this number to compute the frequency of a mesh filter. In previous mesh filtering frameworks, such as [72, 78], the user must try different cutoff frequencies until the desired result is obtained.

Through out this thesis, we assume that the vertices of the *mesh* are discrete samples on an unknown *smooth surface*. To clarify the discussion, we distinguish between *discrete operators* defined on the triangle mesh and *continuous operators* defined on the associated smooth surface.

To understand the frequencies in a mesh filter, one may consider cutting the mesh into local patches and resampling the surface along a regular grid, so as to obtain a piecewise regular domain. One may apply the sampling theorem to this regular domain together with local coordinate chart, a homeomorphism between the patch and regular domain. This way, one may analyze mesh filters mathematically. However, this approach involves several delicate processes. Therefore, we propose to follow an alternative approach. First, we assume that the discrete triangle mesh is a close approximation of the smooth surface it samples. We further assume that the spectral properties of the smooth surface are preserved in the mesh. With this assumption, we can ignore the discretization effects. We have validated this idea with several examples in section 5.4 and Fig. 46. We show that amplification factors of our filter are very close to the ones predicted theoretically in various frequencies. We also note that the choice of discrete Laplacian operator is crucial for this assumption. We chose the formulation initially proposed in [55] with proper weights [50, 58] for computing the mean curvature normal vectors.

Previously proposed transfer functions [72, 78, 11, 50, 58] are limited to lowpass filtering. We propose to broaden their applications to other filtering effects such as exaggeration, which was explored in a multi-resolution framework in [28]. To achieve this extension, we propose to combine explicit [72, 50] and implicit [78, 11, 58] forms to obtain a rational

form. In our construction of a second order filter, we show that the resulting framework allows band pass, notch, band exaggeration with optional high frequency reduction and high pass filters as well as lowpass filters.

CHAPTER II

FLUID SIMULATION: REDUCED DISSIPATION

2.1 Fluid Simulation Overview

For practical purposes, water and many other fluids may be considered as incompressible. For example, water or an air flow that has the Mach number less than 0.3 have negligible compressibility effect [34]. Such incompressible fluids are governed by the incompressible Navier-Stokes equation

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u}\cdot\nabla\mathbf{u} + \nu\nabla\cdot(\nabla\mathbf{u}) - \frac{1}{\rho}\nabla P + \frac{\mathbf{f}}{\rho}, \tag{1}$$

and the continuity equation that represents incompressibility condition

$$\nabla\cdot\mathbf{u} = 0, \tag{2}$$

where $\mathbf{u} = [u\ v\ w]^T$ is the velocity of fluid, P is the pressure, $\mathbf{f}$ is the external forces such as surface tension force and gravity, and ρ is the density. The terms with ∇ and $\nabla\cdot$ operators are defined as

$$\mathbf{u}\cdot\nabla\mathbf{u} = \begin{bmatrix} u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \\ u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} \\ u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} \end{bmatrix},\ \nabla\cdot(\nabla\mathbf{u}) = \begin{bmatrix} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \\ \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \\ \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \end{bmatrix},\ \nabla P = \begin{bmatrix} \frac{\partial P}{\partial x} \\ \frac{\partial P}{\partial y} \\ \frac{\partial P}{\partial z} \end{bmatrix}. \tag{3}$$

Since the fluid is incompressible, (2) represents volume and mass conservation laws. By solving (1) and (2), we can produce fluid simulations. To solve these two equations, we sample velocity and pressure on Eulerian grids. With this discretization, solving (1) and (2) reduces down to updating velocity and pressure values at each time step using various numerical methods that approximate (1) and (2). We describe those numerical methods as follows.

The first method that we chose is the operator splitting [65] that solves terms in the right hand side of (1) one by one, i.e., we solve for one term to produce an intermediate velocity and then solve for another term using this intermediate velocity to produce another intermediate velocity. After all four terms are solved, we obtain the velocity field in the next time step. Four terms in the right hand side of (1) are solved by different methods. We explain them in more detail.

We first apply external force term $\frac{\mathbf{f}}{\rho}$ using an explicit time integration method. Let $\hat{\mathbf{u}}$ be the velocity obtained after this step. Then, $\hat{\mathbf{u}}$ is computed as

$$\hat{\mathbf{u}} = \mathbf{u}^n + \frac{\mathbf{f}}{\rho}. \tag{4}$$

The external forces used in this thesis is the gravity force and the surface tension force.

The next step is applying the diffusion term $\nu\nabla\cdot(\nabla\mathbf{u})$. We use the implicit time integration method. Let $\bar{\mathbf{u}}$ be the velocity obtained after this diffusion step. The implicit integration yields the following linear equation.

$$(\mathbf{I} + \nu\Delta t\nabla^2)\bar{\mathbf{u}} = \hat{\mathbf{u}}. \tag{5}$$

The velocity $\bar{\mathbf{u}}$ can be computed by solving this linear equation.

The next term is the velocity advection $-\mathbf{u}\cdot\nabla\mathbf{u}$. To solve this advection step, the upwind method or the CIR (Courant-Isaacson-Rees) method can be used. The upwind method discretizes the advection term by using derivatives in $\nabla\mathbf{u}$ computed from $\mathbf{u}$ values at the grid points in the $-\mathbf{u}$ direction. The upwind method is stable only if the time step is small enough [65]. In contrast, the CIR method does not discretize the advection equation. Instead, the CIR method first computes the current location $\mathbf{x}'$ that is transported to a grid point *xbb* in the next time step. The location $\mathbf{x}'$ is computed by $\mathbf{x}' = \mathbf{x} - \Delta t\bar{\mathbf{u}}$. Since $\mathbf{x}'$ is not a grid point, an interpolation is necessary to compute the $\mathbf{u}(xbb')$, which will be the velocity $\mathbf{u}(\mathbf{x})$ in the next time step, since $\mathbf{x}'$ will be transported to $\mathbf{x}$. Unlikely to the upwind method, the CIR method is stable regardless of the time step [65]. Therefore, we choose the CIR method.

Since this CIR method can be applied not only to the advection of velocity but also to advections of any other properties, and since our contribution in this chapter is improving CIR advection method for such generalized advections, we explain CIR advection method in the section 2.3.1 in more detail. As described in the section 2.3.1, the velocity advection is achieved by back tracing velocity field and interpolating velocity, i.e.,

$$\tilde{\mathbf{u}}(\mathbf{x}) = \bar{\mathbf{u}}(\mathbf{x} - \Delta t\bar{\mathbf{u}}), \tag{6}$$

where $\mathbf{x}$ is the location of a grid point, $\mathbf{x} - \Delta t\bar{\mathbf{u}}$ is the back traced point, and $\mathbf{u}(\mathbf{x} - \Delta t\bar{\mathbf{u}})$ is computed by trilinear interpolation using velocity values at the grid point near the back traced point $\mathbf{x} - \Delta t\bar{\mathbf{u}}$.

The final step is applying the pressure projection term $-\frac{1}{\rho}\nabla P$ and the incompressibility condition $\nabla \cdot \mathbf{u} = 0$. Since this is the last step, the velocity obtained after this step is the velocity in the next time step $\mathbf{u}^{n+1}$. For this final step, we use the Chorin's pressure projection method [8, 9]. We first take an explicit integration of the pressure term $-\frac{1}{\rho}\nabla P$ as

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}} - \frac{1}{\rho}\nabla P, \tag{7}$$

where P and $\mathbf{u}^{n+1}$ are unknown. Fortunately, the unknon $\mathbf{u}^{n+1}$ can be removed by using the continuity equation $\nabla \cdot \mathbf{u}^{n+1} = 0$. By taking divergence of (7), and then by applying the continuity equation, we obtain the following equation with the only unknown variable P.

$$\nabla \cdot \left(\frac{\Delta t}{\rho} \nabla P \right) = \nabla \cdot \tilde{\mathbf{u}} \tag{8}$$

The first order discretization of (8) is

$$\begin{aligned} \frac{\Delta t}{\Delta x^2} \left(\frac{P_{i-1,j,k} - P_{i,j,k}}{\rho_{i-\frac{1}{2},j,k}} + \frac{P_{i+1,j,k} - P_{i,j,k}}{\rho_{i+\frac{1}{2},j,k}} + \frac{P_{i,j-1,k} - P_{i,j,k}}{\rho_{i,j-\frac{1}{2},k}} + \frac{P_{i,j+1,k} - P_{i,j,k}}{\rho_{i,j+\frac{1}{2},k}} + \frac{P_{i,j,k-1} - P_{i,j,k}}{\rho_{i,j,k-\frac{1}{2}}} + \frac{P_{i,j,k+1} - P_{i,j,k}}{\rho_{i,j,k+\frac{1}{2}}} \right) \\ = \frac{1}{\Delta x} \left(\tilde{u}_{i+\frac{1}{2},j,k} - \tilde{u}_{i-\frac{1}{2},j,k} \; + \; \tilde{v}_{i,j+\frac{1}{2},k} - \tilde{v}_{i,j-\frac{1}{2},k} \; + \; \tilde{w}_{i,j,k+\frac{1}{2}} - \tilde{w}_{i,j,k-\frac{1}{2}} \right). \end{aligned} \tag{9}$$

We assume $\Delta x = \Delta y = \Delta z$ here and through the rest of the discussion. This first order approximation is identical to [64]. Higher-order formulations can be found in [1, 68]. If ρ is constant, we obtain the pressure projection $\frac{\Delta t}{\rho}\nabla^2 P = \nabla \cdot \tilde{\mathbf{u}}$ in [65].

After solving (9), we can use the pressure to compute $\mathbf{u}^{n+1}$ using (7).

2.2 Previous Works

In fluid simulation, earlier work such as [24] required small time step in order to prevent simulation variables from diverging. This problem was successfully remedied in [65] by introducing semi-Lagrangian advection and implicit solve for the viscosity term. The pressure projection is used to enforce incompressibility of the fluid. This solution is popular for the simulation of incompressible smoke [21] and for the more challenging free surface flows [23, 18].

Semi-Lagrangian velocity advection [65] produces a fair amount of dissipation, *i.e.*, the velocity dissipates quickly since the linear interpolation in the first-order semi-Lagrangian produces a large error. In [21], vorticity was added to generate a small scale fluid rolling motion. Recently in [59] and [54], vortex particles were used to transport vortices without loss. In [64], the authors addressed this built-in dissipation problem by increasing the advection accuracy. They adopted the constrained interpolation profile (CIP) [70] method, which increases the order of accuracy in space by introducing the derivatives of velocity to build a sub-cell velocity profile. A nice feature of this CIP method is that it is *local* in the sense that only the grid point values of one cell are used in order to update a point value. However, in this CIP method, all components of velocity and their partial derivatives should be advected, increasing the implementation complexity and computation time, especially in 3D. In addition, it is also worth noting that CIP has higher order accuracy in space only. Therefore, high order integration of characteristics is also necessary. In contrast, the BFECC method used here can be implemented more easily and exhibits second-order accuracy in both space and time and is *local* during each of its operational steps. By *local*, we mean that the interpolation is performed in one grid cell.

Song et al. [64] focused on applying CIP to generate more dynamic water surface behavior. We demonstrate that having less dissipative and diffusive advection provides

significant benefits in smoke simulations. This is illustrated in the middle five images of Fig. 9, where a large amount of dissipation makes the smoke look dark. In contrast, when BFECC is used, the smoke keeps its full brightness throughout the simulation, as shown in the last five images.

Fluid simulation on a curved surface domain has been the subject of several studies. Recently, [20] introduced the unstructured lattice Boltzmann model for fluid simulation on triangulated surfaces. For the advection of smoke, the first-order semi-Lagrangian advection is used on a flattened neighborhood. Stam [66] mapped the surface on a flat domain and then solved the Navier-Stokes equation. The advections still remain first-order accurate. Shi et al [61] proposed to perform a semi-Lagrangian advection directly on the triangular mesh without mapping it onto a flat domain. The accuracy is still first order and dissipation and diffusion cannot be smaller than the amount that already exists in the advection step. We show that BFECC can be easily applied to the advections on a triangulated domain.

For most simulations of liquid surfaces, we use the two-phase fluid model and variable density projection, both of which have been broadly studied in mathematics and fluid mechanics [69, 49, 29], and have been used in graphics applications [30, 31], where the authors simulated air bubbles rising and merging and in [70, 64], where splash style interactions between water surface and air are studied.

2.3 A CIR Advection Method and The BFECC Method

In this section, we first review the first order advection method known as CIR, and then review the BFECC method introduced in [13]. We demonstrate that the first order advection method such as CIR can be modified to the second order accurate BFECC method, with trivial amount of effort.

Since we want to apply it to various advections, we use φ to denote an advected quantity and reserve the symbol ϕ for the level set function. This φ can be the velocity components

u, v, w, smoke density, the RGB color of an image or a dye, or level set function ϕ. For a given velocity field $\mathbf{u}$, φ satisfies the advection equation

$$\varphi_t + \mathbf{u} \cdot \nabla \varphi = 0. \tag{10}$$

2.3.1 The CIR Advection Method

The advection is transporting a property along a vector field. Therefore, the property is constant when it is measured at a location that moves by the velocity field. One can see this from the advection equation. First, notice that when a point $\mathbf{x}$ moves by the velocity field $\mathbf{u}$, $\frac{d\mathbf{x}}{dt} = \mathbf{u}$, and therefore,

$$\begin{aligned} 0 =& \varphi_t + \mathbf{u} \cdot \nabla \varphi \\ =& \frac{\partial \varphi}{\partial t} + \frac{d\mathbf{x}}{dt} \cdot \nabla \varphi \\ =& \frac{dt}{dt}\frac{\partial \varphi}{\partial t} + \frac{dx}{dt}\frac{\partial \varphi}{\partial x} + \frac{dy}{dt}\frac{\partial \varphi}{\partial y} + \frac{dz}{dt}\frac{\partial \varphi}{\partial z} \\ =& \frac{1}{dt}\left(\frac{\partial \varphi}{\partial t}dt + \frac{\partial \varphi}{\partial x}dx + \frac{\partial \varphi}{\partial y}dy + \frac{\partial \varphi}{\partial z}dz\right) \\ =& \frac{d\varphi}{dt}. \end{aligned} \tag{11}$$

This implies that $\varphi(x, y, z, t)$ is constant, along the characteristic line, i.e., when $\dot{x} = u, \dot{y} = v$, and $\dot{z} = w$.

Now, suppose that $\mathbf{x}$ is a grid location and we want to compute the value of φ at $\mathbf{x}$ in the next time step. Since the value of φ is constant along the characteristic line, if we back trace the velocity field for the time Δt and let that location be $\mathbf{x}'$ the value of $\varphi(\mathbf{x}')$ should be the same as $\varphi(\mathbf{x})$ in the next time step. The semi-Lagrangian advection method uses this fact. Therefore, semi-Lagrangian advection includes back tracing the characteristic and then interpolating value at the back-traced location.

The CIR method uses linear characteristic and linear interpolation. Therefore, as illustrated in Fig. 2, $\mathbf{x}'$ is approximated by $\mathbf{x} - \Delta t\mathbf{u}$, and the value $\varphi(\mathbf{x} - \Delta t\mathbf{u})$ is computed by linearly interpolating φ at neighboring grid locations.

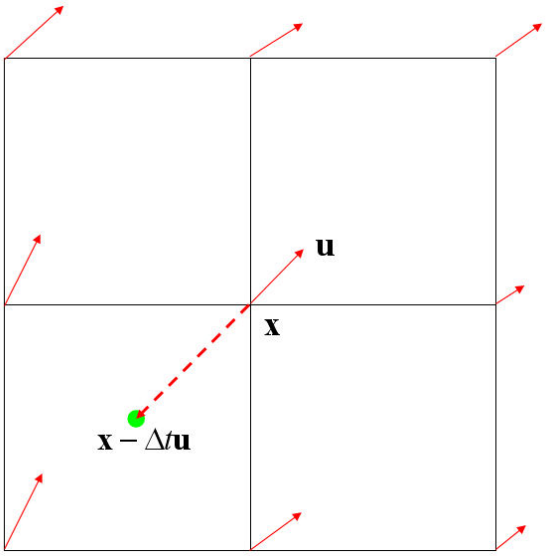

Figure 2: An illustration of the CIR advection method.

2.3.2 The BFECC Method

Once any first order advection such as CIR is implemented, one can easily extend it to the BFECC method. Let L be the first-order upwinding or CIR steps to integrate (10), *i.e.*,

$$\varphi^{n+1} = L(\mathbf{u}, \varphi^n). \tag{12}$$

The implementation of $L(\cdot,\cdot)$ can be found in [60, 53]. With this notation, the BFECC can be written as follows:

$$\begin{aligned} \varphi^{n+1} &= L\left(\mathbf{u}, \quad \varphi^n + \frac{1}{2}\left(\varphi^n - \bar{\varphi}\right)\right) \\ &\text{where } \bar{\varphi} = L\left(-\mathbf{u}, \; L\left(\mathbf{u}, \varphi^n\right)\right). \end{aligned} \tag{13}$$

As illustrated in Fig. 3, one may understand this method intuitively as follows. If the advection step $L(\cdot,\cdot)$ is exact, the first two forward and backward steps return the original value, *i.e.*, $\varphi^n = \bar{\varphi}$. However, this is not the case due to the error in advection operation L. Suppose L contains an error e. Then the first two forward and backward steps will produce the error $2e$, *i.e.*, $\bar{\varphi} = \phi^n + 2e$. Therefore, the error can be computed as $e = -\frac{1}{2}(\varphi^n - \bar{\varphi})$. We subtract this error e before performing the final forward advection step.

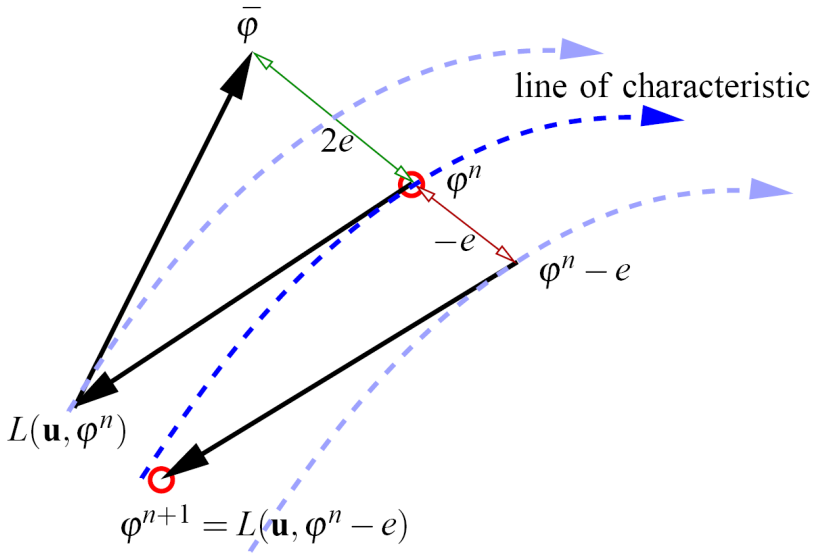

Figure 3: Sketch of BFECC method implemented using CIR advections. The error is revealed by using forward/backward advected values and then the error is compensated before the final advection. Notice that this is only a sketch and should not be interpreted geometrically. For example, e is not a vector.

Then the equation (13) becomes $\varphi^{n+1} = L(\mathbf{u}, \varphi^n - e)$. This step will add an additional e, which will be canceled by the subtracted amount $-e$. This method is proven to be second-order accurate in both space and time [13, 14].

2.3.3 Implementation of BFECC

In this section, we provide the pseudo-code for the BFECC method to demonstrate the simplicity of the BFECC implementation. Let the function First-Order-Step($u, v, w, \varphi^n, \varphi^{n+1}$) implement $L(\cdot,\cdot)$, *i.e.*, an upwind or a semi-Lagrangian integration of the scalar field φ. u, v, and w are velocity components, φ^n and φ^{n+1} are values at n^{th} and $n+1^{\text{th}}$ time steps, respectively. Then BFECC is implemented as follows:

First-Order-Step($u, v, w, \varphi^n, \varphi^*$)

First-Order-Step($-u, -v, -w, \varphi^*, \bar{\varphi}$)

$\varphi^* := \varphi^n + (\varphi^n - \bar{\varphi})/2$

First-Order-Step($u, v, w, \varphi^*, \varphi^{n+1}$)

The variavles φ^* and $\bar{\varphi}$ are intermediate variables.

2.4 Applications of BFECC for Various Advections

2.4.1 BFECC for Velocity Advection

We can use (13) to implement the velocity advection step in solving the Navier-Stokes Equation. In this case, φ becomes u, v, and w. We show that BFECC can reduce the damping in the first-order semi-Lagrangian implementation of velocity advection, which is a well-known drawback of semi-Lagrangian advection [65].

For a multi-phase flow, this BFECC needs to be turned off near the interface to prevent velocities of different fluids with different densities from mixing. We turn BFECC off, *i.e.*, use the first-order semi-Lagrangian, for the grid points where $|\phi| < 5\Delta x$. We also turn it off near the boundary. Notice that reducing velocity dissipation is important not only near the interface, but also in the entire fluid domain. In other words, turning BFECC off near the interface has little effect since it is still turned on in most of the fluid domain.

As shown in Fig. 5, applying BFECC adds details as well as large scale fluctuations in the smoke motion. Notice that these details and large-scale fluctuations cannot be obtained from the vorticity confinement and vortex particle methods [21, 59], which add only small scale rolling motions. We also performed the same simulation in a coarser grid of 100×40 in Fig. 6. In this case, the flow did not fluctuate at all around obstacles with first-order semi-Lagrangian advection. However, when BFECC was added, the flow fluctuated as it did in the refined grid. We conclude that BFECC can create physically realistic fluctuations

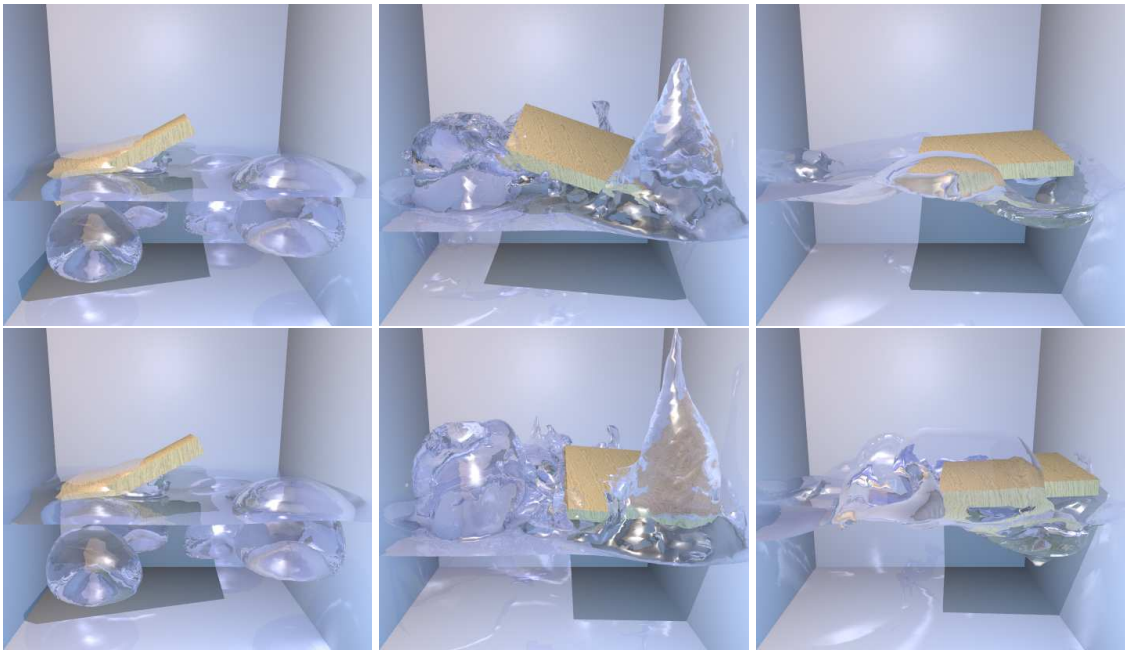

Figure 4: In the bottom row, a highly dynamic behavior of water interaction with air, air bubbles, and a solid is made possible by the two-phase formulation and the BFECC-based reduction of the dissipation in the velocity advection step. In the top row, the BFECC is turned off and the splash is reduced.

even in a coarse grid.

In the simulation of splashing liquid surfaces, BFECC creates higher splashes as shown in Fig. 4, thanks to the reduced velocity damping effect.

Velocity advection can be important also when rigid bodies are involved as well. In Fig. 7, the cup does not tumble due to the velocity dissipation in the first-order semi-Lagrangian method, while the cup does correctly tumble when BFECC is applied to the velocity advection step.

2.4.2 BFECC for Smoke Density and Image Advection

We also apply BFECC to the advection of smoke density for smoke simulation. In Figs. 8 and 9, we show that BFECC can reduce dissipation and diffusion significantly. As shown in [13], BFECC is linearly stable in the l^2 sense, *i.e.*, $||a||_{l^2} = \sum |a_{ij}|^2$ is bounded, when the velocity field is constant, where a is the smoke density. However, density values a_{ij} can become negative or greater than 1.0 for some grid points. In our simulation, the excess

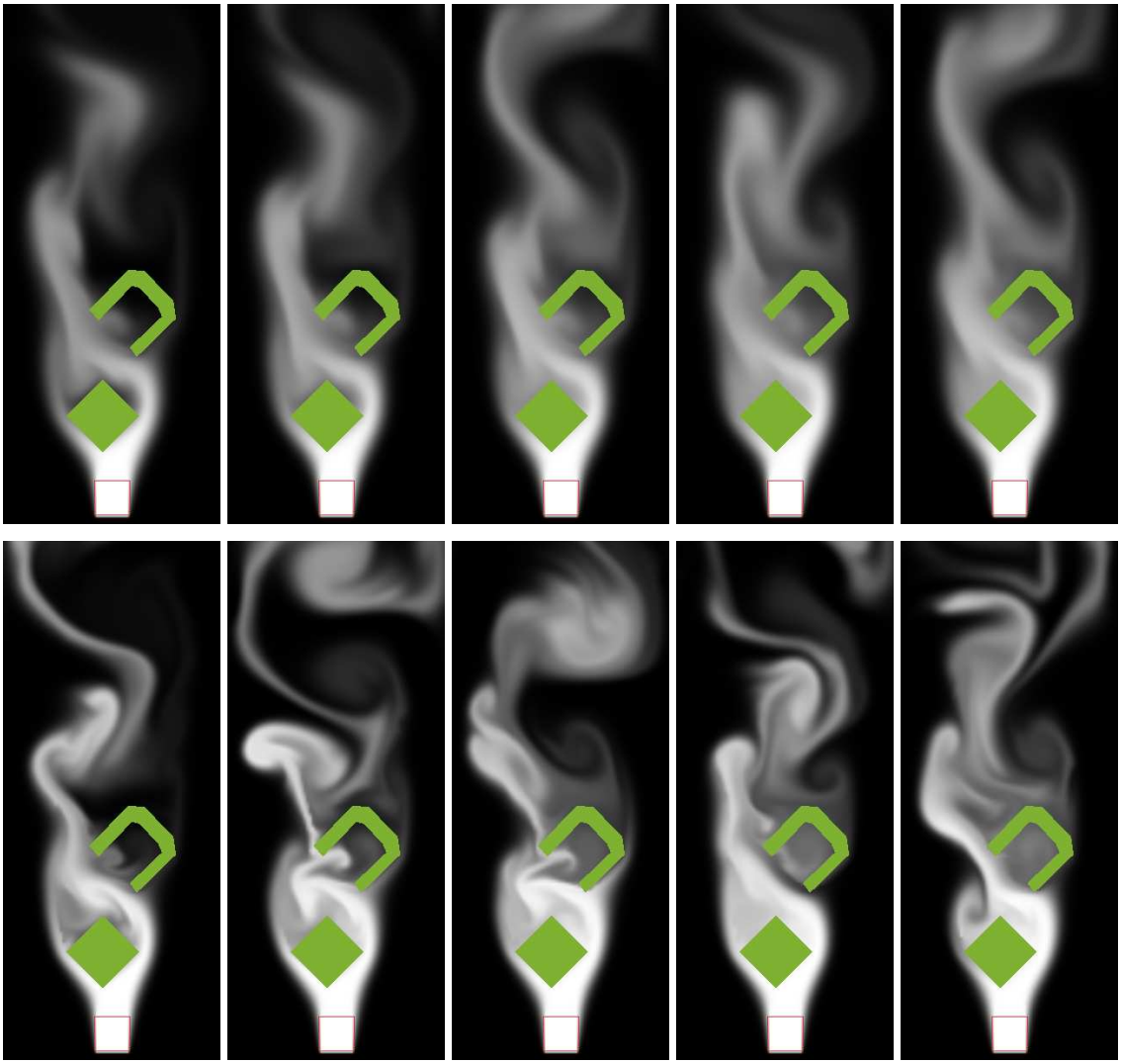

Figure 5: On the top, we used first-order velocity advection that shows damped fluid motion. On the bottom, we have added the simple BFECC method. Notice the small scale details as well as large scale fluctuations. The grid size is 80×200. We used BFECC for the smoke advection all simulations in the figure.

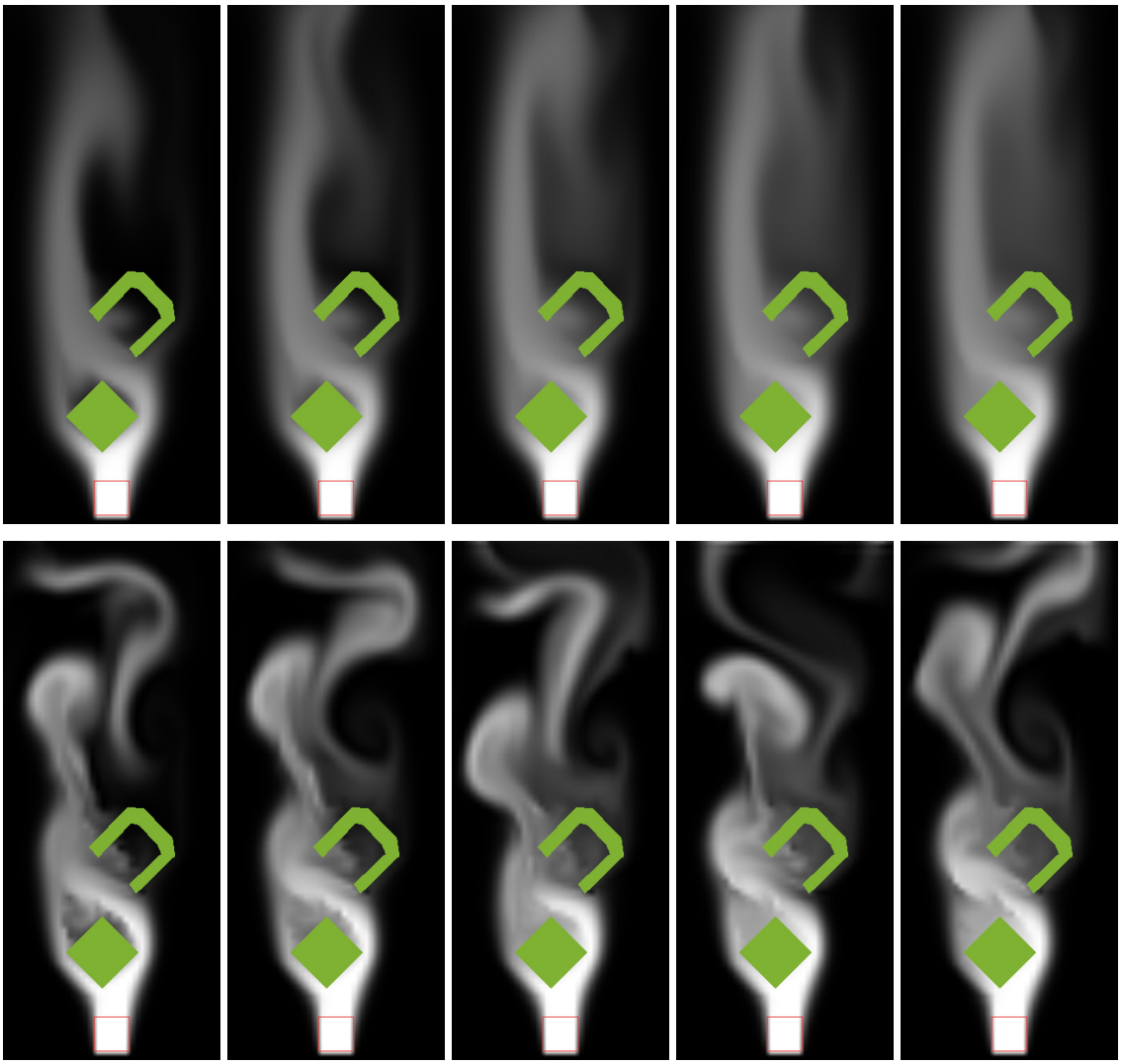

Figure 6: In a low resolution grid (40×100), the first-order velocity advection produces severely damped fluid motion (Top). On the bottom, we have added the simple BFECC method and the small scale details as well as large scale fluctuations appear.

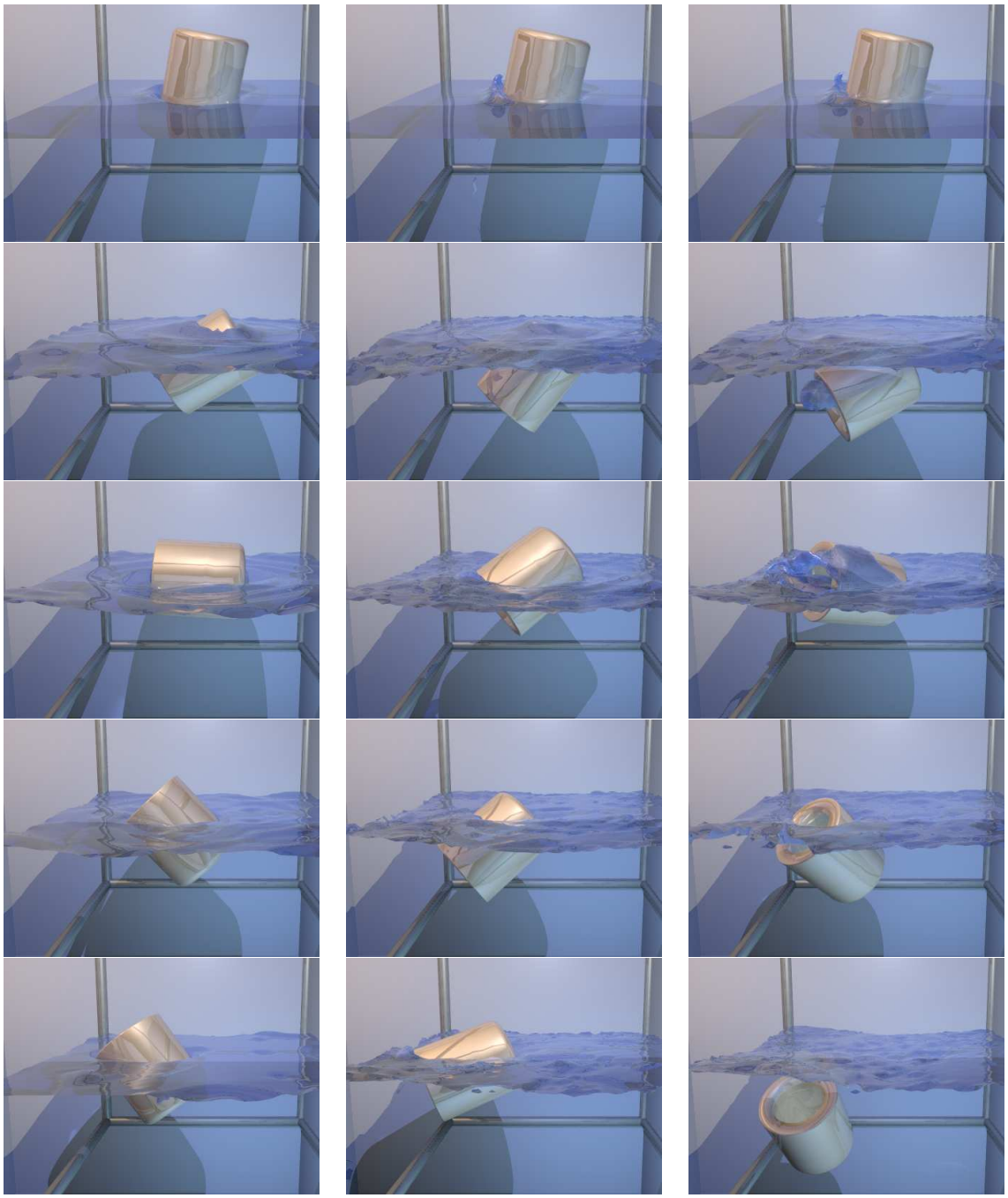

Figure 7: Simulation of a sinking cup. The left column is simulated without the BFECC in both level set and velocity advection steps, where the motion of the cup is damped (the cup does not dive deep into the water) and the detail of the surface is poor. In the center column, this poor surface detail is enriched by turning BFECC on for the level set step, but the cup motion is still damped (the cup goes deeper but it does not tumble). Finally, in the right column, the dampening in motion of the cup is remedied by using BFECC for the velocity advection step as well, making the cup sink deeper and tumble as well.

amount was negligible, so we clamped those values to stay in $[0, 1]$.

To measure the diffusion/dissipation amount, we design a test problem similar to the Zalesak's problem. Instead of the notched disk in the Zalesak's problem, we use a color image and rotate it 360 degrees (in 400 time steps and by using bilinear interpolations only) and then compare it with the original image as shown in Fig. 10. As shown in (d), the dissipation of the color is significantly reduced with BFECC. During advection, the image is also diffused to the neighboring region. To visualize the diffusion amount, we plot background pixels as blue to show the region where the image has been diffused. As shown in (d), the color of the object has almost no diffusion into neighboring region when BFECC is used. Also notice that the position of the image is different from the original location in (f) due to the error accumulated during time integration. This error is also fixed in (g), where BFECC is used, showing that due to the second-order time accuracy of BFECC, the image follows the vector field more precisely. The computation time was 0.156 seconds (without BFECC) and 0.36 seconds (with BFECC) per frame on a 3GHz Pentuim4.

2.4.3 Dye Advection for Vector Field Visualization

One way to visualize a vector field is to advect a dye on it. A natural approach would be to use the first-order semi-Lagrangian advection method. However, it introduces severe diffusion and dissipation. As shown in the left image of Fig. 11, the result contains a large amount of diffusion and dissipation. This problem has been addressed by several researchers. Weiskopf [75] applied the level set method to advect the dye without diffusion. This approach requires the implementation of the level set method together with a careful treatment to redistancing to prevent volume loss. It also allows only one dye color. In [33], Jobard et al combined semi-Lagrangian and Lagrangian particle advection of dye, which requires a significant amount of additional software. In contrast, BFECC requires only a trivial amount of code, and allows a convincing visualization of the vector field, as shown in the right image in Fig. 11.

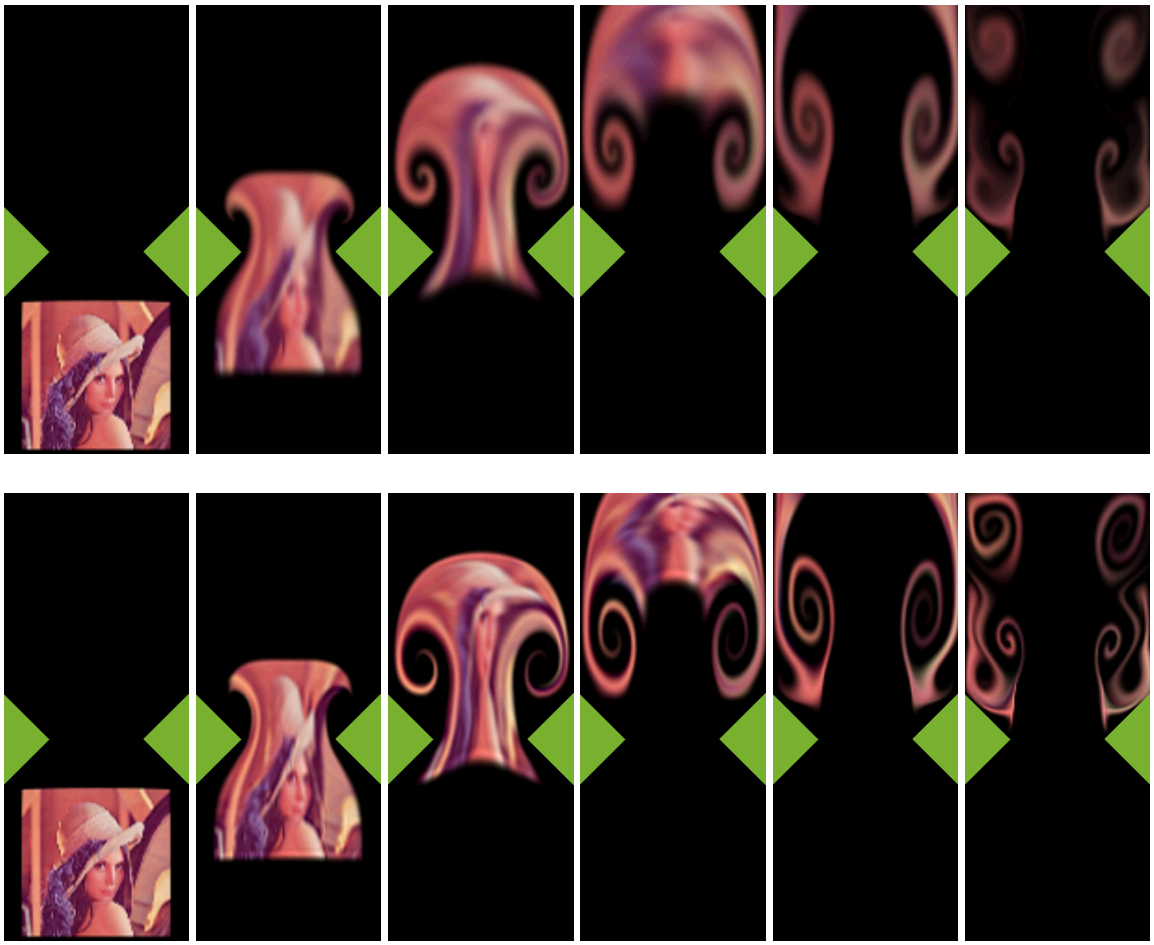

Figure 8: Advection of an image along with the up-going flow field on 100×250 grid. The left image shows the initial location of the image. The top images is computed without the BFECC, where the dissipation/diffusion are significant. The bottom images are computed with the BFECC, where the dissipation is greatly reduced and the features of the image can be identified.

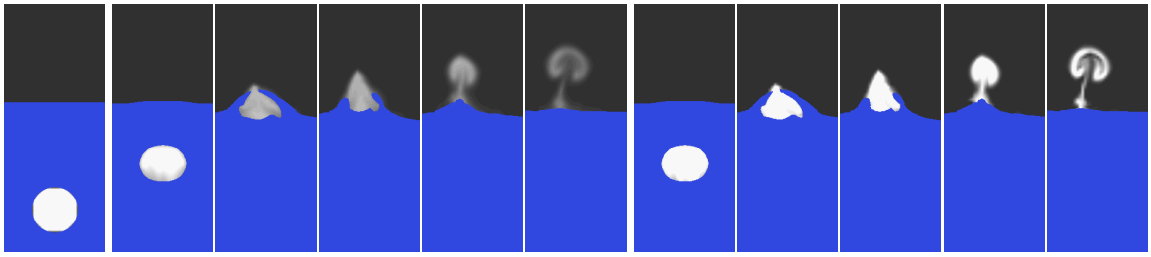

Figure 9: Simulation of smoke in a bubble rising and bursting on a 41×101 grid ($\Delta x = 0.0025m, \Delta t = 0.01sec$). The far left image shows the initial bubble. The next five images are without BFECC, where the dissipation/diffusion in the semi-Lagrangian step deteriorate the density of smoke. The last five images simulated with BFECC show significantly reduced dissipation/diffusion, and the smoke is in full density throughout the simulation. All simulation parameters between the two runs are identical, except for the usage of BFECC in smoke advection. Therefore, the only difference is the density of smoke. Also, notice that the simulation time differs by less than 1% since the bulk of the computation time is dominated by the pressure projection step.

2.4.4 BFECC for Level Set Advection

Even though BFECC does not completely prevent volume loss in fluid simulation, particularly for small droplets or thin filaments, the benefit of BFECC in the fluid simulation is valuable since it is easy to implement and fast. Notice that, for more demanding simulation, the slight volume loss in free surface flow may be reduced by combining BFECC with the particle level set method [17].

When we use the BFECC for level set advection, *i.e.*, $\varphi = \phi$, redistancing is needed to keep the level set function close to a signed distance function. We use the following redistancing equation [69]

$$\phi_\tau + \mathbf{w} \cdot \nabla\phi = \mathrm{sgn}(\phi) \quad \text{where } \mathbf{w} = \mathrm{sgn}(\phi)\frac{\nabla\phi}{|\nabla\phi|}, \tag{14}$$

where $\mathbf{w}$ is the velocity vector for redistancing. This equation can be solved by applying the first-order upwinding in discretizing the term $\mathbf{w} \cdot \nabla\phi$. An alternative is the semi-Lagrangian style integration, *i.e.*, $\phi^{n+1} = \phi^n(\mathbf{x} - \mathbf{w}\Delta\tau) + \mathrm{sgn}(\phi^n)\Delta\tau$, where $\mathbf{x}$ is the location of each grid point. Hence, $\phi^n(\mathbf{x} - \mathbf{w}\Delta\tau)$ is the ϕ value of the previous location.

When these integration formulae for (14) are combined with BFECC, redistancing tends

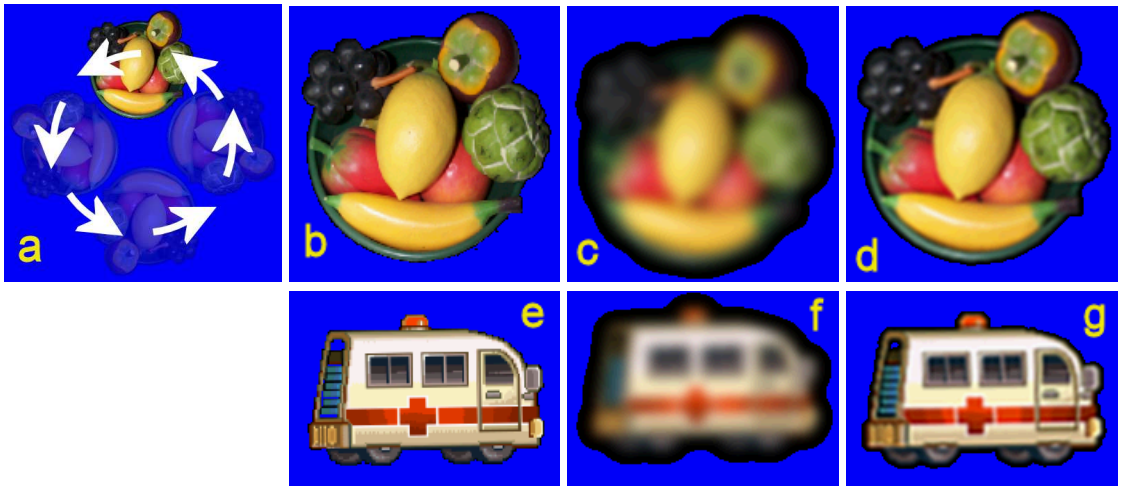

Figure 10: Test of dissipation and diffusion on an image advection problem along a circular vector field (800×800 grid, CFL = 6.29). (b) is the top center portion of the original image (a). (c) is obtained by rotating it 360 degrees using the first-order semi-Lagrangian scheme, where one can see a large amount of dissipation, diffusion, shrinkage of image, and position error. These errors are significantly reduced in (d), where BFECC is used. The blue background region is, in fact, in black, but it is rendered as blue to illustrate the region in which the color is not diffused. (e)-(g) are the same test with another image. Notice the position error in (f) due to the lack of accuracy in time. This is fixed in (g), where BFECC is applied.

to corrupt good ϕ values computed from the second-order accurate BFECC. Thus, if redistancing is turned off near the interface, good ϕ values are not corrupted. The conditions in which redistancing is turned off are provided in [13], where significant improvement was shown in the Zalesak's problem. This simple redistancing is crucial for preserving volume [13], but easy to implement since it simply requires redistancing at points where at least one of the following two conditions is met.

- When the grid point is not close to the interface, *i.e.*,

  when $\phi_{i,j}$ has the same sign as its eight neighbors.

- When the slope is sufficiently high, *i.e.*,

  when $|\phi_{i,j} - \phi_{i\pm1,j}| \geq 1.1\Delta x$ or $|\phi_{i,j} - \phi_{i,j\pm1}| \geq 1.1\Delta y$.

(15)

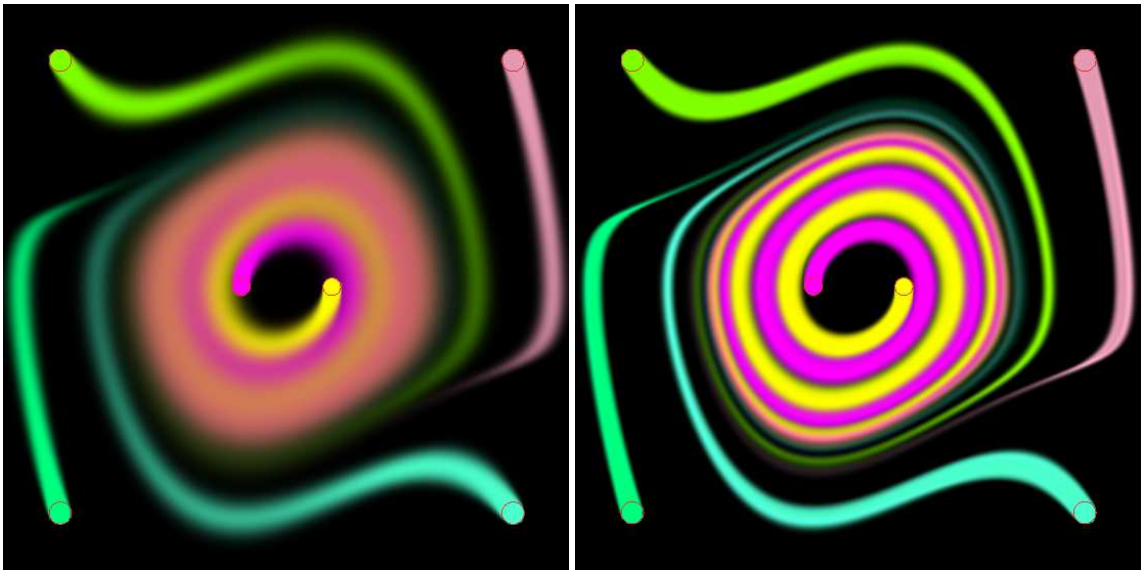

Figure 11: Visualization of the Van der Pol oscillator. The left image shows dye advection with first-order advection. The right image shows dye advection with BFECC, which produces reasonable visualization of the vector field.

2.4.5 Level Set Advection on Triangulated Surfaces

We have applied the BFECC to the advection of a scalar field on a triangulated surface. On this surface domain, we explore the advections of a level set. For the first-order semi-Lagrangian advection of this level set, one needs to trace the characteristic line of a vector field defined on a curved surface. The tracing on this curved surface is more complicated than that on a planar domain since the velocity vector should be steered to remain tangent to the surface. On a triangulated piecewise flat domain, this steering occurs when the trajectory moves to a different triangle by crossing an edge or a vertex. The solution to this steering problem is similar to [61], but we explain it in the following way. We perform this edge or vertex crossing steps in a refraction-free manner, i.e., we always proceed in a direction in which the angles of the two sides of the resulting trajectory are identical. Using this approach, we can follow the velocity field on the surface, and therefore, we can implement the first-order semi-Lagrangian advection step.

Once this first-order advection is implemented, BFECC can be trivially added by calling it three times, as explained in section 2.3.3. We have implemented the simple redistancing

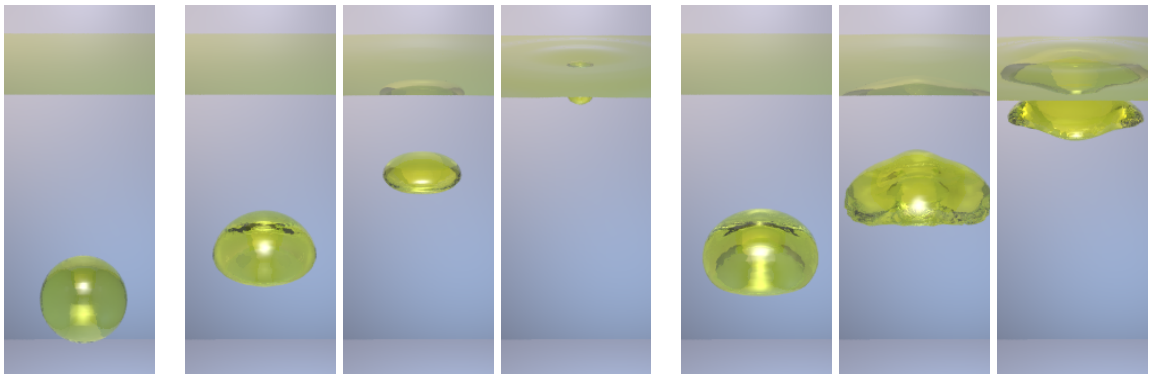

Figure 12: The far left image shows an air bubble placed in olive oil at time zero. The next three images are first-order semi-Lagrangian implementation of level set advection. The last three images are produced using BFECC and simple redistancing and show significantly reduced volume loss. The grid resolution is $60 \times 100 \times 60$ ($\Delta x = 0.0008333m, \Delta t = 0.001sec$).

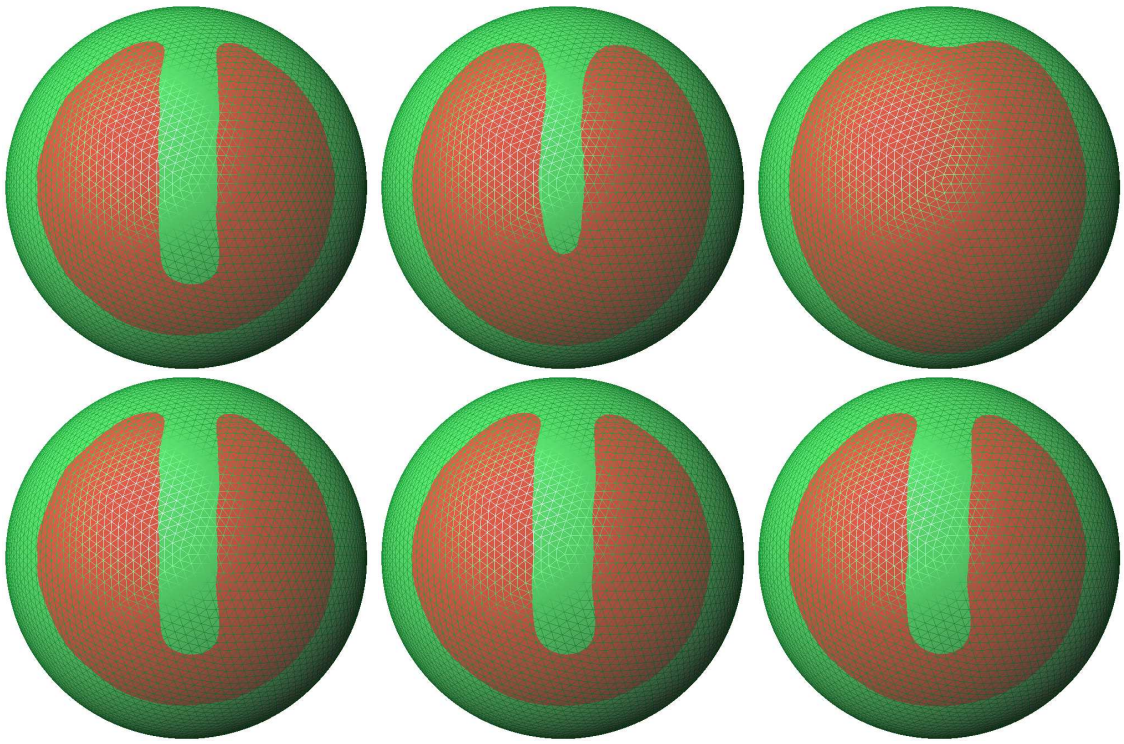

Figure 13: Advections of the Zalesak's disk on a sphere. The left column show initial disks. The next two columns show the disk after one and two rotations about the vertical axis with first-order advection (top) and BFECC advection (bottom).

strategy similar to (15). Let ϕ_i be the level set value at the i$^{\text{th}}$ vertex, and $\mathcal{N}_i$ be the set of indices of the vertices neighboring to the i$^{\text{th}}$ vertex and let $\mathbf{x}_i$ be the location of i$^{\text{th}}$ vertex. The simple redistancing conditions on a triangle mesh are

- ϕ_i has the same sign as its neighbors.
- $|\phi_i - \phi_j| \geq 1.1||\mathbf{x}_i - \mathbf{x}_j||$, for some $j \in \mathcal{N}_i$.

These two simple conditions reduce smoothing and volume and shape changes of the Zalesak's disk significantly, as shown in Fig. 13, where we rotated the Zalesak disk on the surface about the vertical axis passing the center of the sphere. Notice that we rotate Zalesak disk not by moving the sphere mesh but by updating the level set value sampled at each vertex. The sphere mesh does not move.

For fluid simulation on a triangulated surface, one needs to transport a velocity field as well. The velocity vector should always remain on the surface. Shi et al [61] also provide a solution to this problem by steering the coordinate frame where the vector is represented. Steering was performed in a way that minimized the twist. Notice that this approach is a discretized version of the parallel transport, which is known as a method for transporting a coordinate frame on manifolds [12]. Using this vector field transport idea, one may implement fluids with free surfaces on a triangulated surface. Since BFECC can significantly improve level set advection as illustrated in Fig. 13, it could be combined with [61] to create a fluid simulator on a triangulated surface.

2.4.5.1 Computation of Gradient

The redistancing velocity $\mathbf{w}$ defined in (14) contains a gradient term $\nabla\phi$. In this section, we show the computation of the gradient using the following precomputed gradient operator.

Let $\Delta\phi_{ij} = \phi_j - \phi_i$ and let $\Delta x_{ij} = x_j - x_i$, where x_i is the x-coordinate of $\mathbf{x}_i$. Similarly, define Δy_{ij} and Δz_{ij}. Then, we can express the variation of ϕ in discrete form.

$$\Delta\phi_{ij} \approx \phi_x \Delta x_{ij} + \phi_y \Delta y_{ij} + \phi_z \Delta y_{ij} \quad , \quad \forall j \in \mathcal{N}_i \tag{16}$$

This system of equation can be written in a matrix form.

$$\mathbf{d} = A\nabla\phi \tag{17}$$

$$\text{where} \quad \mathbf{d} = \begin{bmatrix} \Delta\phi_{ij_1} \\ \Delta\phi_{ij_2} \\ \cdots \\ \Delta\phi_{ij_n} \end{bmatrix} \quad , \quad \mathbf{A} = \begin{bmatrix} \Delta x_{ij_1} & \Delta y_{ij_1} & \Delta y_{ij_1} \\ \Delta x_{ij_2} & \Delta y_{ij_2} & \Delta y_{ij_2} \\ \cdots & \cdots & \cdots \\ \Delta x_{ij_{n_i}} & \Delta y_{ij_{n_i}} & \Delta y_{ij_{n_i}} \end{bmatrix} \tag{18}$$

and n_i is the number of neighbors of i$^{\text{th}}$ vertex. The least square solution of (18) is

$$\nabla\phi = \mathbf{A}^{\dagger}\mathbf{d} \tag{19}$$

where $\mathbf{A}^{\dagger}$ is the Moore-Penrose pseudo inverse of $\mathbf{A}$. Now, suppose that we represented $\nabla\phi$ in local coordinate of any two orthonormal tangent vectors $\mathbf{s}_1, \mathbf{s}_2$ and the normal vector $\mathbf{n}$. Let $\mathbf{S} = [\mathbf{s}_1\ \mathbf{s}_2] \in \mathbb{R}^{3\times 2}$. Then, each row-vector of $\mathbf{A}$ can be represented by the $\mathbf{s}_1, \mathbf{s}_2$ and $\mathbf{n}$, *i.e.,*

$$\mathbf{A} = \begin{bmatrix} \mathbf{AS} & \mathbf{An} \end{bmatrix} \begin{bmatrix} \mathbf{S}^T \\ \mathbf{n}^T \end{bmatrix} \quad , \quad \mathbf{AS} \in \mathbb{R}^{n_i\times 2}\ ,\ \mathbf{An} \in \mathbb{R}^{n_i\times 1} \tag{20}$$

Suppose the surface is locally smooth, then $\mathbf{An}$ is small and can be neglected.

$$\mathbf{A}^{\dagger} \approx \left(\mathbf{AS}\,\mathbf{S}^T\right)^{\dagger} = \mathbf{S}\,(\mathbf{AS})^{\dagger} \tag{21}$$

Finally, $\nabla\phi$ is computed as

$$\begin{aligned} \nabla\phi &\approx \mathbf{S}\ (\mathbf{AS})^{\dagger}\mathbf{d} \\ &= \mathbf{S}\ \left(\mathbf{A}_S^T\mathbf{A}_S\right)^{-1}\mathbf{A}_S^T\ \mathbf{d} \quad , \quad \mathbf{A}_S = \mathbf{AS} \end{aligned} \tag{22}$$

Notice that when the mesh is not deforming, $\mathbf{S}\ \left(\mathbf{A}_S^T\mathbf{A}_S\right)^{-1}\mathbf{A}_S^T \in \mathbb{R}^{3\times n_i}$ is constant and can be precomputed as a gradient operator.

2.4.6 BFECC for Adaptive Mesh

In an adaptive mesh such as an octree [44], the interpolation required for semi-Lagrangian advection is more complicated to implement. This complexity is often already high in the

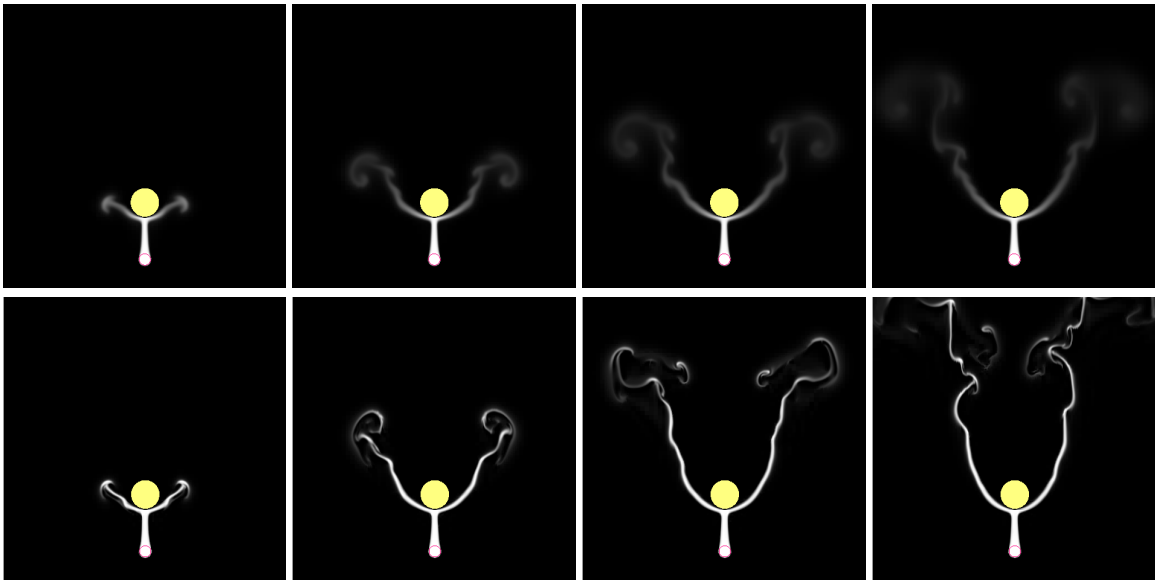

Figure 14: Simulation of smoke on an adaptive quadtree mesh of maximum resolution 512^2 ($\Delta t = 0.0001, \Delta x = 0.1m/512$ at maximum resolution). The top row is without BFECC, where the smoke diffusion and dissipation are large. When a lesser diffusive and dissipative smoke is needed, one can trivially implement BFECC and generate a smoke with significantly reduced dissipation and diffusion, as shown in the bottom row, which is simulated with BFECC. The amount of diffusion and dissipation is controlled by adjusting the diffusion and dissipation coefficients in the diffusion and dissipation steps, respectively. Thus, BFECC decouples advection from dissipation and diffusion steps.

simplest linear interpolation. Therefore, higher order nonlinear interpolation tends to be much more complex. This complexity of high-order interpolation can be easily avoided when one uses BFECC. To verify the applicability of BFECC on an adaptive mesh, we here implemented a smoke simulator on a quad tree mesh similar to [44] and show that it can reduce the diffusion of smoke. Due to small amount of diffusion obtained by BFECC, we can simulate a thin filament of smoke. Since the smoke remains in a thin region, the mesh is refined only in the thin region. Notice that when one uses first-order advection, smoke will diffuse into the neighborhood quickly, hence a mesh needs to be refined in a larger region. The benefits of BFECC on this quadtree mesh are illustrated in Fig. 14.

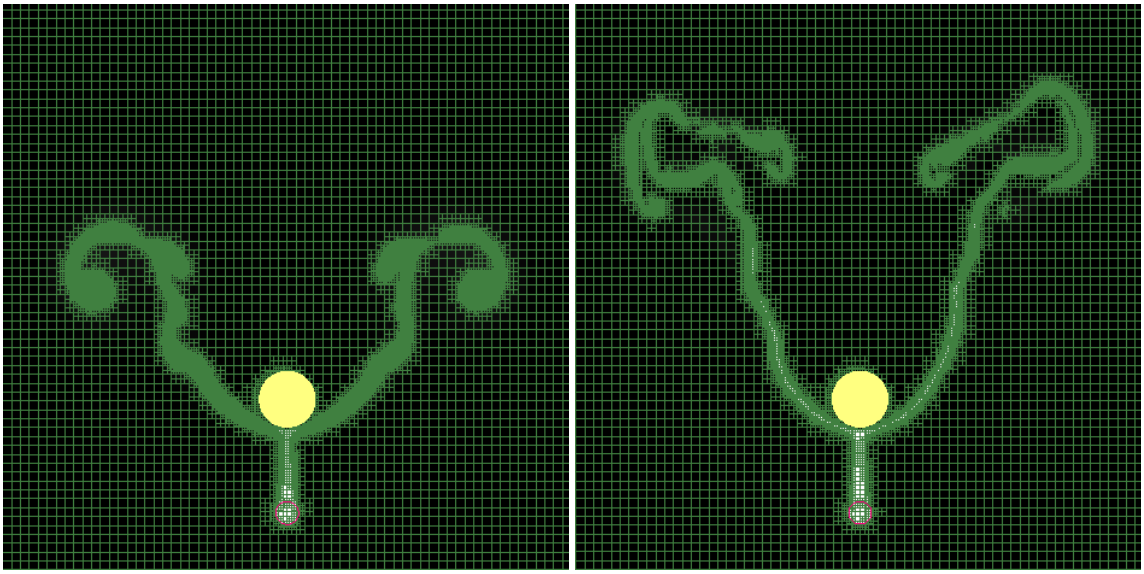

Figure 15: The adaptive quad tree grids without BFECC (left) and with BFECC (right).

2.5 Additional Discussions on Fluid Simulation

We here tested BFECC in different fluid simulations. We simulate air-water and olive oil-air interactions. Water is rendered in a bluish color and olive oil is rendered in a yellowish color. We use the PovRay (http://povray.org) to render the images.

In Fig. 7, we simulated interactions between a cup, air, and water. The cup is released upside down near the water surface. Due to its weight, the cup sinks into water, but it soon rises again because of the air in it. However, in the top row, where we turned BFECC off for velocity advection, the water is dissipative, and does not have enough velocity to tumble the cup. In the bottom row, we use BFECC for velocity advection where the velocity dissipation is small, and hence, the cup tumbles. This example indicates that reducing velocity dissipation could be important in simulating fluid and rigid body interactions.

We here implemented the *rigid fluid* method [7] to simulate rigid body and fluid interaction in Figs. 4 and 7, where buoyancy is automatically obtained by applying variable density projection similar to [27]. We use multiple pressure projections to address the seeping problem mentioned in [7]. We found that the angular momentum of the rigid body tends to

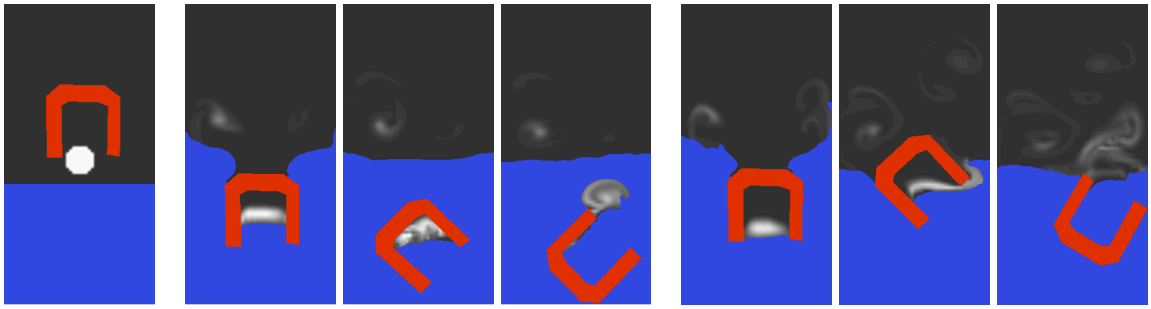

Figure 16: Dropping a cup into water on a 51×101 grid ($\Delta x = 0.01m, \Delta t = 0.005sec$). The far left image shows the initial configuration. The next three images in the middle are with two projections and show a significant amount of air lost. The three images on the right are with nine projections. They show that enough air is trapped inside the cup, causing the heavy cup ($\rho = 1300kg/m^3$) to rise again. Also, notice the patterns of the smoke that are diffused or dissipated little, thanks to the smoke advection, using BFECC in both cases.

be reduced per projection. Therefore, per each projection step, we added rotational velocity to the rigid body so that the angular momentum loss is corrected. This multi-projection treatment is slow but easy to implement. The effect is illustrated in Fig. 16.

The computation time varies with the complexity of the fluid motions. In a simple bubble rising situation without a rigid body, it took a few seconds per time step using a 50^3 mesh. The cup example in Fig. 7 has multiple pressure projections, taking about 30 to 130 seconds per time step on a 70^3 grid.

CHAPTER III

SIMULATION OF THIN LIQUID FILMS ON A CELL-CENTERED OCTREE GRID

3.1 Previous Works

In contrast to the simulation of gaseous phenomena such as smoke, the simulation of liquids produces complex changes in the liquid surface. Therefore, a method suitable to represent the liquid volume and surface and capable of handling changes of the liquid surface is needed. One such method that has been widely used recently is the level set method [51, 53, 60]. In [23], the level set method was used in fluid simulation to create a realistic liquid simulation.

Even though the level set method allows simulation of a liquid's surface, the complexity of the surface that can be simulated is limited by the grid resolution. A significant improvement can be obtained by using an adaptive grid such as an octree [44, 62]. In particular, Losasso et al [44] showed that detailed liquid surfaces can be simulated by using an octree grid.

Prior research in liquid simulation [18, 44, 7, 25, 59, 17] simulated liquids only, ignoring air. Therefore, liquid and air interactions, such as bubbles, requires an extension. In [31], the air bubbles were successfully simulated. More general liquid/air phenomena can be simulated by the variable density pressure projection method that has been broadly studied in mathematics and fluid mechanics [69, 49, 35, 29]. This variable density pressure projection has been used by [64, 37], in which splash and bubbles are simulated. We also notice the bubble simulation using Lattice-Boltzmann-Method (LBM) was studied in [73, 56].

Although the variable density projection method can simulate bubbles, the practical

simulation of foam, in particular, dry foam, is more challenging since the micrometer-thin liquid film cannot be represented by an Eulerian grid. Consequently, most foam simulations have been performed by Lagrangian methods. For example, Kuck et al [41] approximated foam bubbles as spheres and then studied interactions between them. The interaction between bubbles and liquid is not studied. In contrast, Takahashi et al [70] used particle to represent foams and simulated interactions with liquids. However, the formation, bursting, or merging of foam bubbles have not been addressed.

Thin liquid films are governed by molecular interactions rather than by the Navier-Stokes equations. An interesting outcome of such molecular interactions is the disjoining pressure, which acts along the direction normal to the film surface. This disjoining pressure causes a capillary suction, slowing down the drainage of liquid in the film. Thus, the thin film is maintained. This disjoining pressure, caused by molecular interactions such as electrostatic, van der Waals, steric, and adsorptional interactions, is the subject of several engineering and scientific studies [57, 19, 74, 67]. Besides those efforts to understand thin film, its simulation is rather rare. Weaire and Hutzler [74] constructed piecewise curves and patches to simulate a dry foam. Durian [16] simulated 2D dry foam using circular bubbles. Because circular bubbles avoid the difficulties related to the thin films, it greatly simplifies the simulation of foam. The drawback of these methods would be the lack of bursting, merging, and interactions with liquid. Bazhlekov et al [3] simulated the behavior of a foam drop that is made of a few bubbles. They used a Lagrangian mesh to represent the liquid/gas interface. Simulations of various phenomena such as wet foam, formation of dry foam, bubble merging, and bursting were not previously attempted.

Recently, Zheng et al [79] proposed two promissing methods for the bubble and thin liquid film simulation. First, they proposed the region based level set method in fluid simulation. Each bubble has different region code. Therefore, two adjacent bubbles does not have to be separated by a thin liquid region. Therefore the simulation of thin films between different regions is simplified.

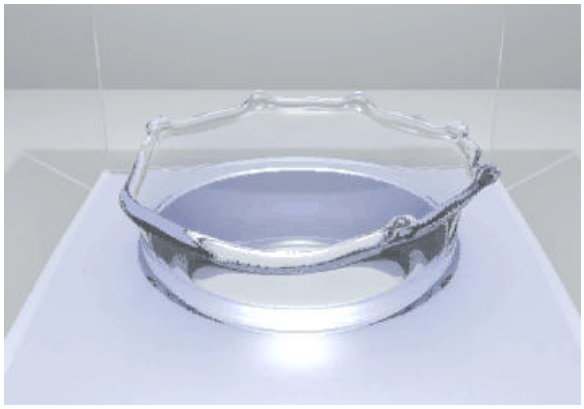

Figure 17: Thin liquid film occurs during a liquid splash simulation.

In contrast, our disjoining force idea be used for thin film that has boundary. An example of such film can be found during liquid splash simulations as shown in Fig. 17. The second contribution of [79] is the semi-implicit implementation of surface tension, which allows larger time step. Application of these approaches will be an interesting future direction.

3.2 Fluid Simulation on an Octree Grid

3.2.1 Nonstaggered Octree Grid

Simulation of bubbles with thin films requires a high resolution mesh to represent thin films. In addition, liquid with multiple bubbles has a complex interface. Since a high-resolution octree grid containing a complex interface has a high memory cost, a memory-efficient implementation of an adaptive grid is desirable. Towards this goal, we simplify the octree grid representation used in [44], by storing pressure and velocity at the center of the octree cell. Since all values are stored at the center of the octree cell, we do not need a parallel data structure for the cell corner, where velocities and level set values were stored in [44]. In addition, velocity transfer [27] to the cell face is not needed. Therefore, implementation is simpler and memory-efficient. In our experiment, the 512^3 grid with a flat water surface and 63 bubbles underneath (see Fig. 1) has 14 million octree leaves. In this case, 2.5G bytes of memory was used. The 1024^3 grid with five bubbles under a water surface requires 2.2G

bytes of memory. About 65% of this memory cost is used for the octree. The remaining 35% is used for the symmetric pressure projection matrix and for temporary vectors needed for conjugate gradient iteration. This 35% cost can be saved when the pressure projection is implemented directly on the octree. In our experiments, this implementation was about twice slower during the first few steps and then quickly become more than five times slower, because of the cache misses resulting from the address fragmentation produced during the dynamic allocation of memory for the octree nodes.

One drawback of our collocation is the complexity of the interpolation in semi-Lagrangian advection. However, we found that a continuous interpolation is not necessary, but a simple interpolation with neighboring cells suffices. We interpolate values with neighboring cells assuming that the cells are in the same tree depth, even when they are not. Since we do not allow more than a two-to-one depth ratio between adjacent cells, the largest tree depth difference ignored would be one. Suppose that we are performing an interpolation of ϕ at a point (x,y,z), which is contained in a leaf cell A. Assume that cell A has tree depth d. Interpolation involves the ϕ values of the neighboring nodes of A. First assume that a particular node B incident upon a face of A has depth d. If B is a leaf, we use its ϕ. If not, we use the average of its children. Now assume that the depth of B is less than d. It can only be $d-1$, since we ensure a 2-to-1 ratio between neighboring cells. We will use its ϕ.

Following [44], we perturb the sampling points to obtain a symmetric pressure projection matrix. The two phase flow formulation is similar to [64], except for the differentiation operator across different grid resolution. Notice that all our variables are defined at the cell center. Therefore, the differentiations of these variables are the same as the differentiation of pressure in [44].

3.2.2 Multigrid Solver

A high resolution mesh with a complex interface makes the pressure projection step very slow even in an adaptive grid. In addition, large surface tensions require small time steps.

Therefore, simulation may be very slow. To reduce this computational cost, we use a multigrid solver. We applied the simple nested iteration discussed in [5].

1. We first build a pressure projection operator and compute the divergence of the velocity at a coarse level.
2. We then solve the pressure projection equation using the conjugate gradient (CG) method with the Jacobi preconditioner. This will produce the pressure solution in a coarse grid.
3. We use the computed pressure as a starting estimate for the child-nodes at the next finer resolution.
4. We repeat steps 1, 2, and 3 until the maximum depth is reached.

We tested the effect of this approach in several experiments. As shown in Fig. 18, CG iteration tends to vary greatly when we do not use the nested iteration. This often occurs when the time step is small and the CG error bound is large. However, as shown in Fig. 18, when we used the nested iteration, the computation time become uniform. When the complexity of surface is high the benefit is significant. We conclude that the nested iteration is beneficial, especially when the interface is complex or is in high-resolution. When nested iteration is used, the pressure projection takes about $40 \sim 50\%$ of the total computation time.

3.2.3 Level Set Advection

The BFECC method applied to level set advection tends to induce high-frequency noise on the interface wherever the velocity field is not smooth. Therefore, Dupont and Liu [14] proposed a remedy using the following semi-Lagrangian advection

$$\phi^{n+1} = \frac{1}{2}\left[\ \phi^n(-\mathbf{u}\Delta t + \varepsilon\mathbf{e}) + \phi^n(-\mathbf{u}\Delta t - \varepsilon\mathbf{e})\ \right], \tag{23}$$

where $\mathbf{e} = (1,1,1)$. We found $\varepsilon = 0.3\Delta x$ works well. BFECC is implemented by performing this step three times in each time step. This oversampling and averaging adds a small

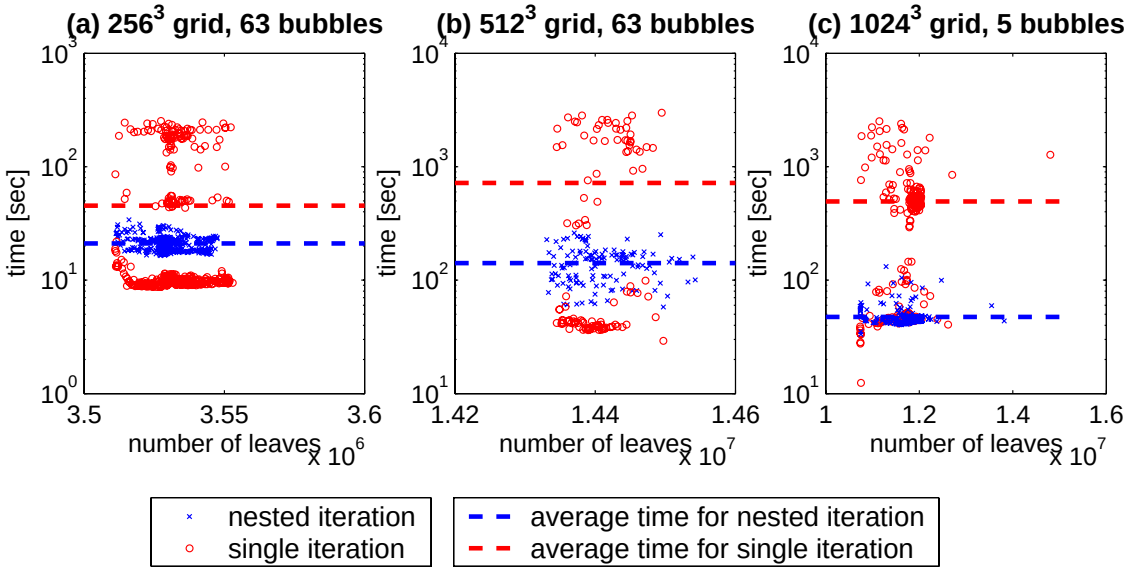

Figure 18: The computation time for the pressure projection step is reduced by nested iteration. The benefit increases as the grid resolution and number of leaves increases. Average pressure projection times are (a) nested:21.1sec, single:45.3 sec, (b) nested:141.2sec, single:718.6sec, (c) nested:47.5sec, single:494.4sec.

amount of diffusion, making the surface smooth. Since this oversampling point is always displaced along $\pm\mathbf{e}$, it may cause some artifacts along $\pm\mathbf{e}$. A possible solution would be oversampling in more directions, but it would be computationally expensive. Our solution is to randomly choose $\mathbf{e}$ from the four directions $\{(1,1,1),(1,-1,1),(-1,1,1),(-1,-1,1)\}$ in each time step.

3.3 Making Thin Film Last Longer

In our experiments, if ε in (23) is as large as $0.2 \sim 0.3\Delta x$, thin liquid films were well preserved. However, the liquid has the thickness of about $5\Delta x$. Films with this thickness can be observed unrealistic as shown in Fig. 26, where 256^3 equivalent grid was used. When we use smaller ε is chosen, this thickness can be reduced, but then, the film tends to rupture easily.

In this section, we identify that the curvature computed from level set gradient inside thin film causes the early rupture of thin film, and then we discuss a possible solution. In addition, inspired from the disjoining pressure, we propose to use the disjoining force to

strengthen the film further. Using these approaches, as shown in Fig. 22, we obtain liquid films that are about $3\Delta x$ thick.

3.3.1 Surface Tension

Real world bubbles have round and smooth shapes because the surface tension flattens the liquid/gas interface. In thin liquid films, surface tension can be the dominating force. Therefore, implementing surface tension is important in the simulation of thin films. Since the surface tension is proportional to the curvature normal vector of the interface, computation of the curvature normal vector is necessary. In the level set framework, this curvature normal is computed as $\nabla \frac{\nabla \phi}{|\nabla \phi|}$. However, along the center of the thin film, the level set function has a singularity, as shown in Fig. 19, and hence the gradient computed would not be accurate. Therefore, the curvature computed from the gradient of ϕ is noisy. In particular, this noise is severe along the thin film's centroid, which is only about two grid cells away from the interface. Therefore, we want to use the ghost fluid method [35, 31] that use curvature values of the cell that is neighboring the interface. Since the curvature computation on these cells requires access to one neighbors, ghost fluid requires to use ϕ values about two grid cells away from the interface.

In contrast, surface tension methods such as the smeared delta method [4] applies a surface tension force up to $2 \sim 3$-neighboring cells away from the interface. To compute curvature on these cells, we use ϕ values about $3 \sim 4$ grid cells away from the interface. Therefore, the surface tension force computed from the gradient of ϕ is noisy in thin films. In our experiments, when one uses the smeared delta method, the surface tension computed from the derivatives of ϕ produced artifacts on surface and an early breaking of thin films, even with a large disjoining force (described later) was active. In contrast, the ghost fluid method showed an improvement. Since it uses ϕ values at most two-neighbors, the thin film is better preserved than in the smeared delta method. However, in 2D, the ghost fluid method fails to preserve the thin film, but in 3D, the ghost fluid method showed that the

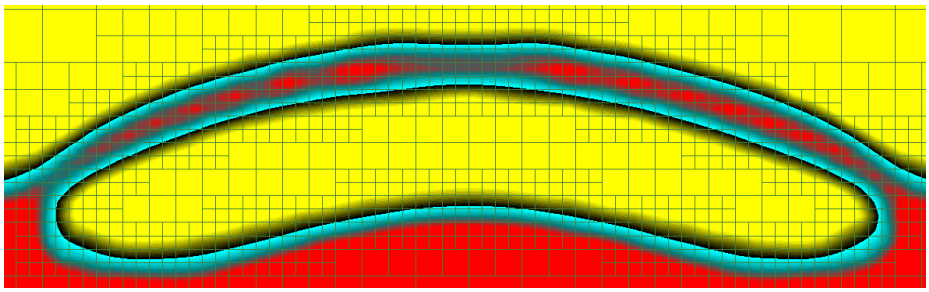

Figure 19: Contour of the level set function ϕ. Thin films contain series of local minima, and therefore, the gradient $\nabla\phi$ is not always perpendicular to the film surface. Therefore, the curvature vector computed from $\nabla\phi$ can be inaccurate.

thin film can be preserved well. However, in small bubbles bursted as shown in Fig. 26. In this section, we develop a method that can further improve this by computing the surface tension from the ϕ values of 1-neighbors.

First, notice that even though the gradient of ϕ is inaccurate inside a thin film, the liquid surface of the film is smooth as shown in Fig. 19. This observation led us to the idea of computing the surface tension from the liquid/gas interface. Assume that we have a sufficient octree refinement near the interface so that all the interfaces are contained in cells of maximum depth. Consider two octree cells whose centers are A and B, as shown in Fig. 20. Suppose that ϕ has different signs at A and B and that the line $\overline{AB}$ is parallel to x-axis. The interface point P can be computed by linear interpolation of ϕ. The neighboring interface points Q_0, Q_1, Q_2, Q_3 can be easily computed. The local surface neighborhood around P contains more neighbors, but computing those additional neighbors and their connectivity would require extraction of iso-surface . Since extracting the iso-surface at every time step is expensive, we developed a method to compute the curvature normal vector on P using the four neighborhood points $\{Q_0, Q_1, Q_2, Q_3\}$.

In estimating the curvature normal, the discrete Laplace-Beltrami operator [11] would be a good candidate. However, the vertex P has only four neighbors $\{Q_0, Q_1, Q_2, Q_3\}$, and moreover, those neighboring vertices are not always evenly distributed, as shown in the right image of Fig. 20. By the Taylor series expansion [76] of the surface around P, it can be shown that the accuracy of the discrete Laplace-Beltrami operator is low. Therefore, we

develop a method that first computes a valid normal direction and then a mean curvature value, using only $\{P, Q_0, Q_1, Q_2, Q_3\}$.

As shown in the left illustration of Fig. 20, consider the two difference vectors of the two tangent vectors $\mathbf{t}_{z_1} - \mathbf{t}_{z_0}$ and $\mathbf{t}_{y_1} - \mathbf{t}_{y_0}$. The normal vector is computed as a normal to these two vectors, i.e., $\mathbf{n} := (\mathbf{t}_{z_1} - \mathbf{t}_{z_0}) \times (\mathbf{t}_{y_1} - \mathbf{t}_{y_0})$, $\mathbf{n} := \mathbf{n}/|\mathbf{n}|$. In the right illustration of Fig. 20, the surface has small curvature but the angle between the two tangents $\mathbf{t}_{y_{0,1}}$ is small. Notice that a valid normal vector is produced in this case as well. Once the normal

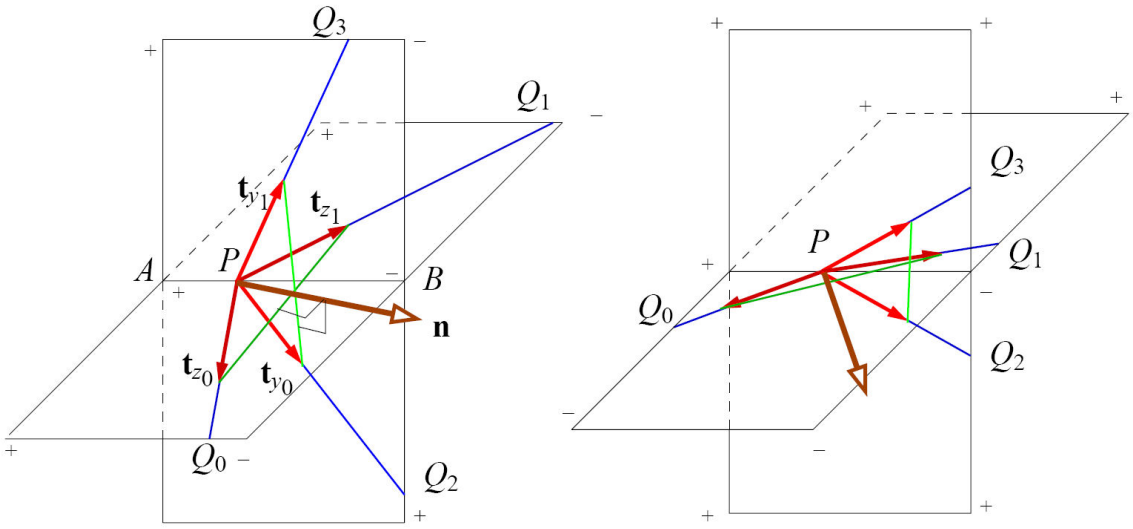

Figure 20: Computing a normal vector on a point where the liquid surface and $\overline{AB}$ are intersecting. In the right, we illustrate the case when the iso-surface has low curvature but the angle $\angle Q_2PQ_3$ is small. Notice that the normal vector $\mathbf{n}$ is computed properly.

vector is computed, we estimate the mean curvature from the angles between $\mathbf{n}$ and each of the tangent vector $\mathbf{t}_{z_0}, \mathbf{t}_{z_1}$, $\mathbf{t}_{y_0}$, and $\mathbf{t}_{y_1}$. We first compute the radius of a circle that encloses $\Delta x\mathbf{n}$ and $\Delta x\mathbf{t}_{z_0}$ by $r_0 = \Delta x\mathbf{t}_{z_0} \cdot \mathbf{n}$. Similarly, we compute $r_1 = \Delta x\mathbf{t}_{z_1} \cdot \mathbf{n}, r_2 = \Delta x\mathbf{t}_{y_0} \cdot \mathbf{n}$ and $r_3 = \Delta x\mathbf{t}_{y_1} \cdot \mathbf{n}$. We then average those radii, computing $r = (r_0 + r_1 + r_2 + r_3)/4$. The mean curvature is estimated as $\kappa = 2/r$, which allows us to compute the surface tension as $\gamma\kappa\mathbf{n}$, where γ is the surface tension coefficient.

After $\gamma\kappa\mathbf{n}$ is computed, we apply it to A and B as the surface tension force. We also tested it with the ghost fluid method by using $\gamma\kappa$ as the pressure jump required for the ghost fluid method proposed in [35, 31]. However, it caused square-like bubbles, in particular

when a large surface tension is used. Therefore, we do not use the ghost fluid method together with the $\kappa\mathbf{n}$ computed by the above method.

Examples of bubbles simulated by this surface tension approach can be found from Fig. 24 and from Fig. 23.

3.3.2 The Disjoining Force

In real fluid, thinning of a film slows down due to the disjoining pressure. This disjoining pressure is active typically when the thickness is smaller than a micrometer, which is too thin to be captured by an Eulerian grid. Therefore, in this section, we explore an idea to simulate a film that is about $3 \sim 4\Delta x$ thick.

To maintain this thickness this thickness, we apply a disjoining force in a film that is thinner than $4\Delta x$. This disjoining force is normal to the film and is applied in both directions to make the film thicker. As a result, a part of a film that became thinner than the threshold recovers its thickness. To this goal, we first detect cells inside the thin film, compute the normal vector and film thickness, compute the disjoining force, and finally, apply a symmetric set of forces that repulse the two film surfaces, making the film thicker.

To determine whether a cell is a thin film cell or not, we check the sign changes of ϕ in each of the x, y, and z directions. We only explore two grid cells in each direction, since our threshold thickness is $4\Delta x$. If sign changes in both $\pm x$ directions are detected, the cell is marked as a thin film cell. Similarly, we apply this to $\pm y$ and $\pm z$ directions.

Figure 21 shows a fluid cell A that is inside a thin film and is surrounded by four air cells A, B, C, and D. We calculate forces that push the two surrounding interface away from A. In each of such thin film cell, we compute the normal direction to the thin film. The normal $\mathbf{n}$ at A is computed as the orthogonal vector of the average of the two tangents at P and Q. Thus, we avoid using the level set gradient that is not differentiable in the thin film, similarly to the surface tension case. The thickness of the film is computed as a projected length. For example, in Fig. 21, the thickness is computed by $l = d\mathbf{n} \cdot \mathbf{i}$. When sign changes

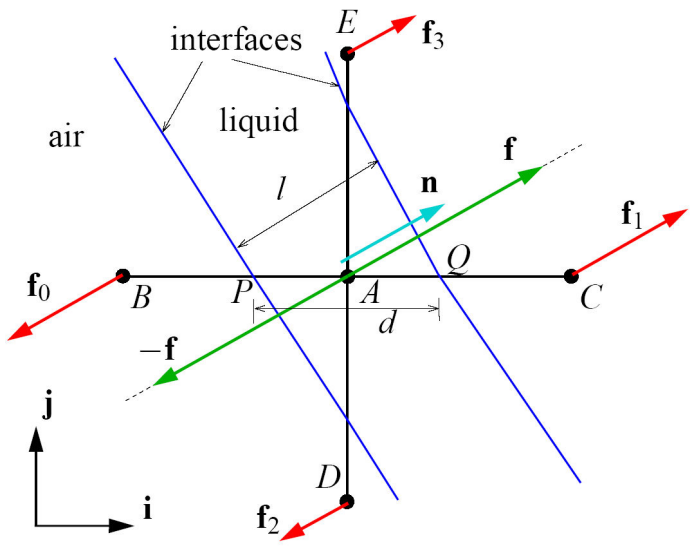

Figure 21: Disjoining force that prevents a thin film from bursting.

are detected in more than one direction, we perform this operation on each direction and average them. Thus, we computed the normal direction **n**, and the thickness of the thin film l at each film cells.

We then design the magnitude of force as a smoothly increasing function that is zero when the thickness is larger than the $4\Delta x$. As the thickness become smaller than $4\Delta x$, the function smoothly increase from zero and then saturate to η as the thickness approaches to zero. To meat this goal we compute the force **f** by

$$\mathbf{f} = \begin{cases} 0, & \text{if } l \geq 4\Delta x \\ \eta\left(2\left(\frac{l}{4\Delta x}\right)^3 - 3\left(\frac{l}{4\Delta x}\right)^2 + 1\right)^{\alpha}\mathbf{n}, & \text{if } l < 4\Delta x \end{cases}, \tag{24}$$

where η is the coefficient that determines the magnitude of force. In our experiments, when $\eta = 5,000$, thin films rupture rarely. The power α determines the thickness of the film. In most case, we use $\alpha = 1$, which produces film slightly thinner than $4\Delta x$.

The next step is to apply this force $\pm\mathbf{f}$ to the neighboring cells B, C, D, and E so that the interfaces are pushed apart. Since the forces applied to neighbors should not introduce motions other than thickening, we apply $\pm\mathbf{f}$ to B, C, D, and E so that the force and torque

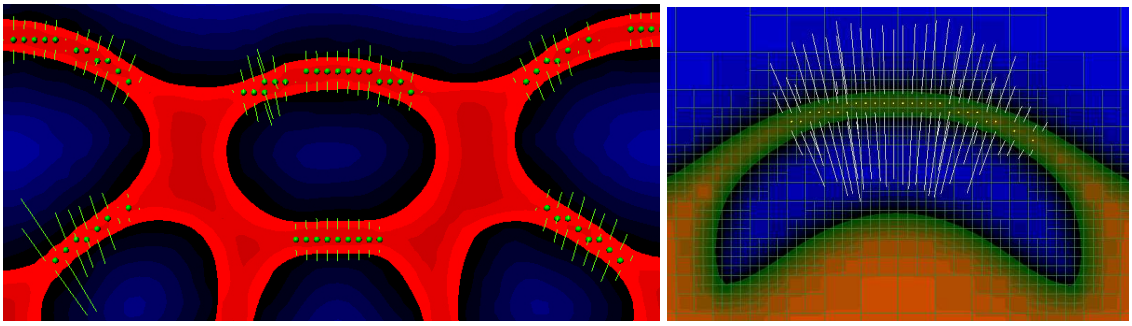

Figure 22: Disjoining forces acting on film cells (low-resolution grids).

resultants are zero, i.e.,

$$\begin{aligned} \mathbf{f}_0 + \mathbf{f}_1 + \mathbf{f}_3 + \mathbf{f}_3 &= 0 \\ (-\mathbf{i}) \times \mathbf{f}_0 \;+\; \mathbf{i} \times \mathbf{f}_1 \;+\; (-\mathbf{j}) \times \mathbf{f}_2 \;+\; \mathbf{j} \times \mathbf{f}_3 &= 0 \;. \end{aligned} \tag{25}$$

Let $\mathbf{n} = (n_x, n_y)$. The forces $\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2$, and $\mathbf{f}_3$ that satisfy (25) are computed as

$$\begin{aligned} \mathbf{f}_0 &= -\mathbf{f}_1 = -n_x \mathbf{f} \\ \mathbf{f}_2 &= -\mathbf{f}_3 = -n_y \mathbf{f} \;. \end{aligned} \tag{26}$$

Finally, $\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2$ and $\mathbf{f}_3$ are added to B, C, D and E, respectively.

The extension to 3D is straightforward. Let $\mathbf{n} = (n_x, n_y, n_z)$, and let $\mathbf{f}_4$ and $\mathbf{f}_5$ be the forces on the two neighboring nodes in back and front of A, respectively. Then, the forces $\mathbf{f}_4$ and $\mathbf{f}_5$ are computed by $\mathbf{f}_4 = -\mathbf{f}_5 = -n_z \mathbf{f}$, while $\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2$, and $\mathbf{f}_3$ are computed by (26).

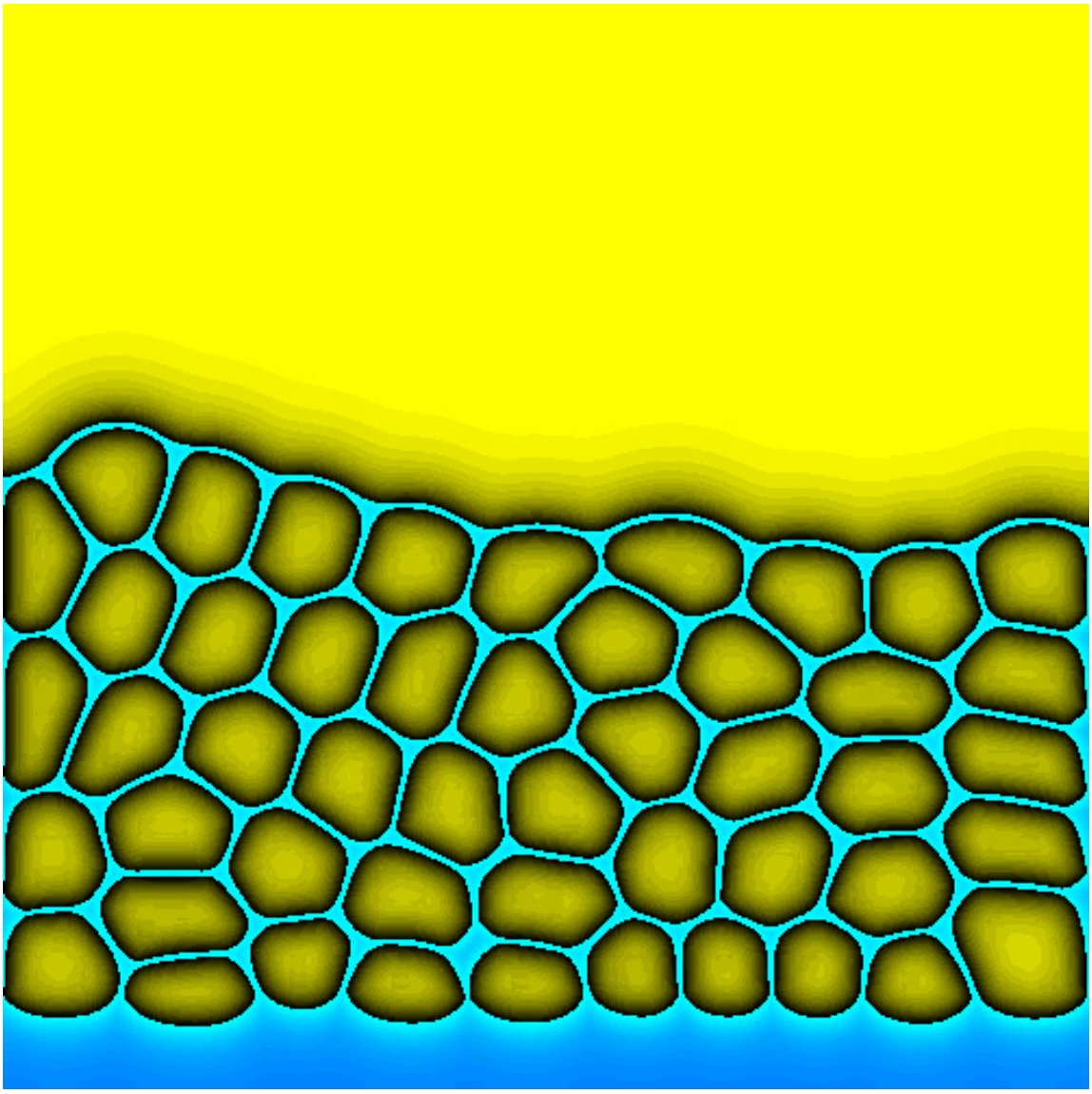

Figure 23: A dry-foam-like structure obtained by using the surface tension based on the ghost fluid method with the curvature computed from the local interface. We inflated the 2D bubbles using the volume control method discussed in chapter 4.

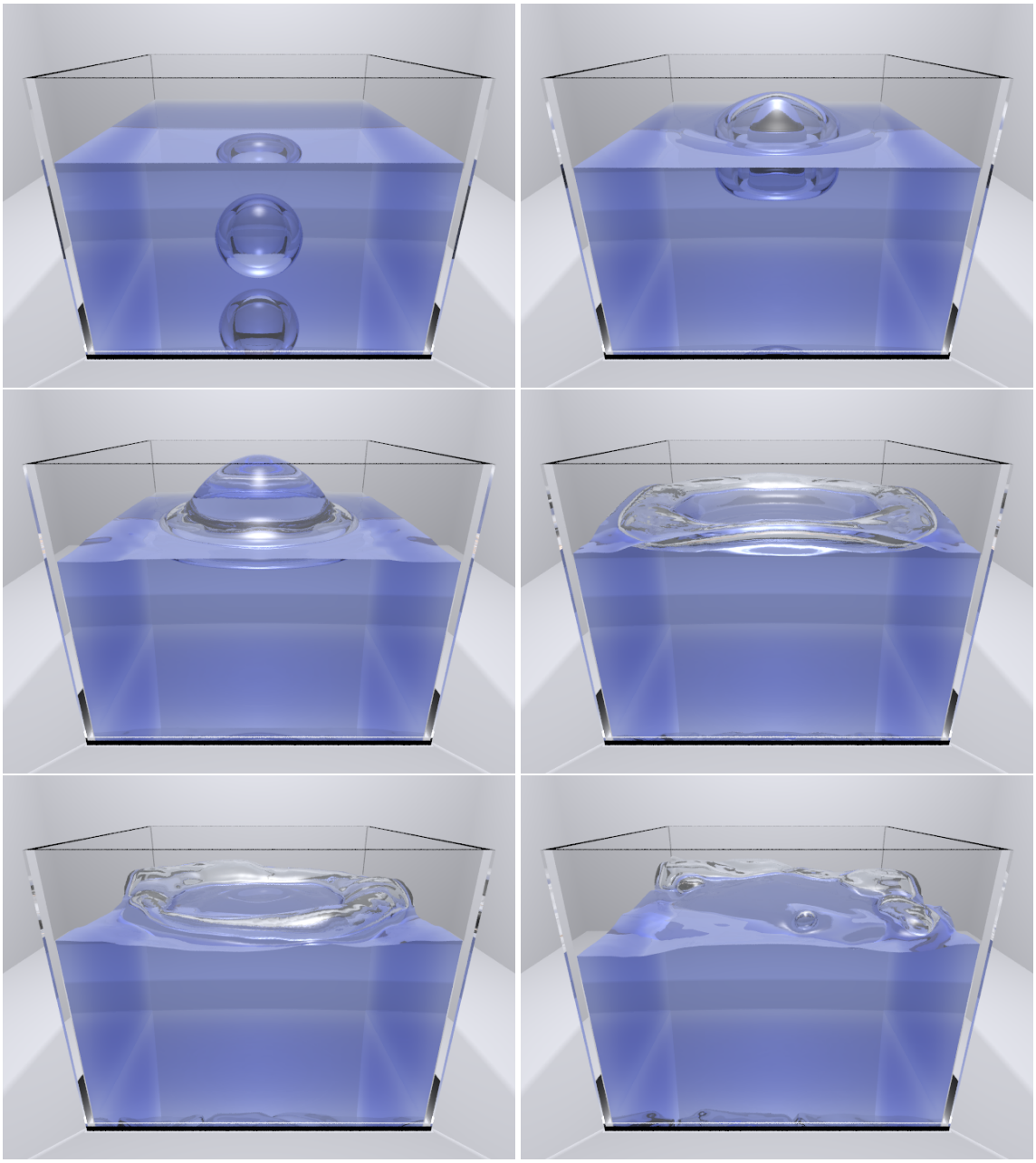

Figure 24: Simulation of one bubble forming a bubble ring and then splitting into many small bubbles. Surface tension proposed in section 3.3.1 is used.

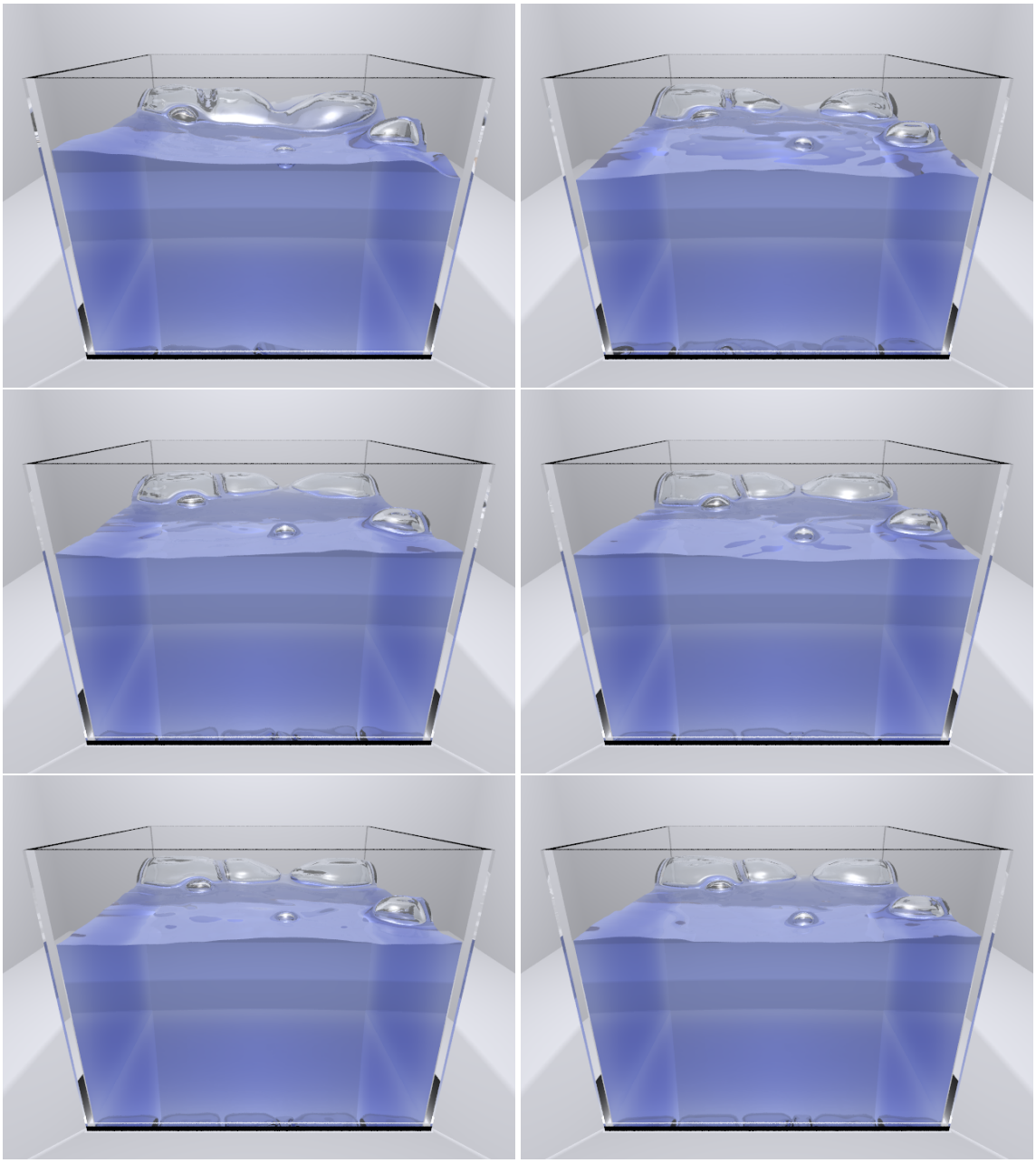

Figure 25: Simulation of one bubble forming a bubble ring and then splitting into many small bubbles (continued). Surface tension proposed in section 3.3.1 is used.

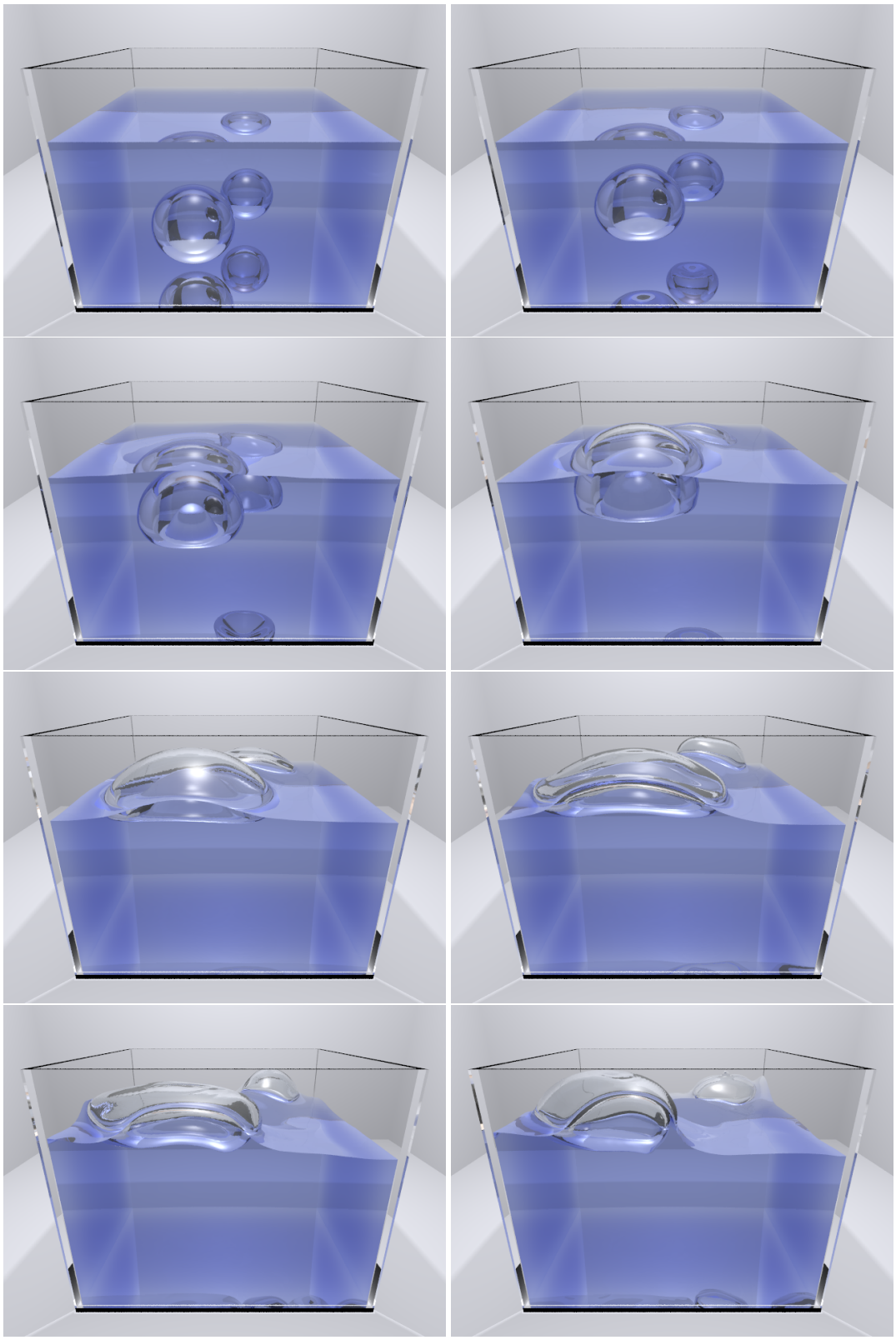

Figure 26: Simulation of the rise of two bubbles, one of which is bursting. Surface tension using the ghost fluid method [31] is used.

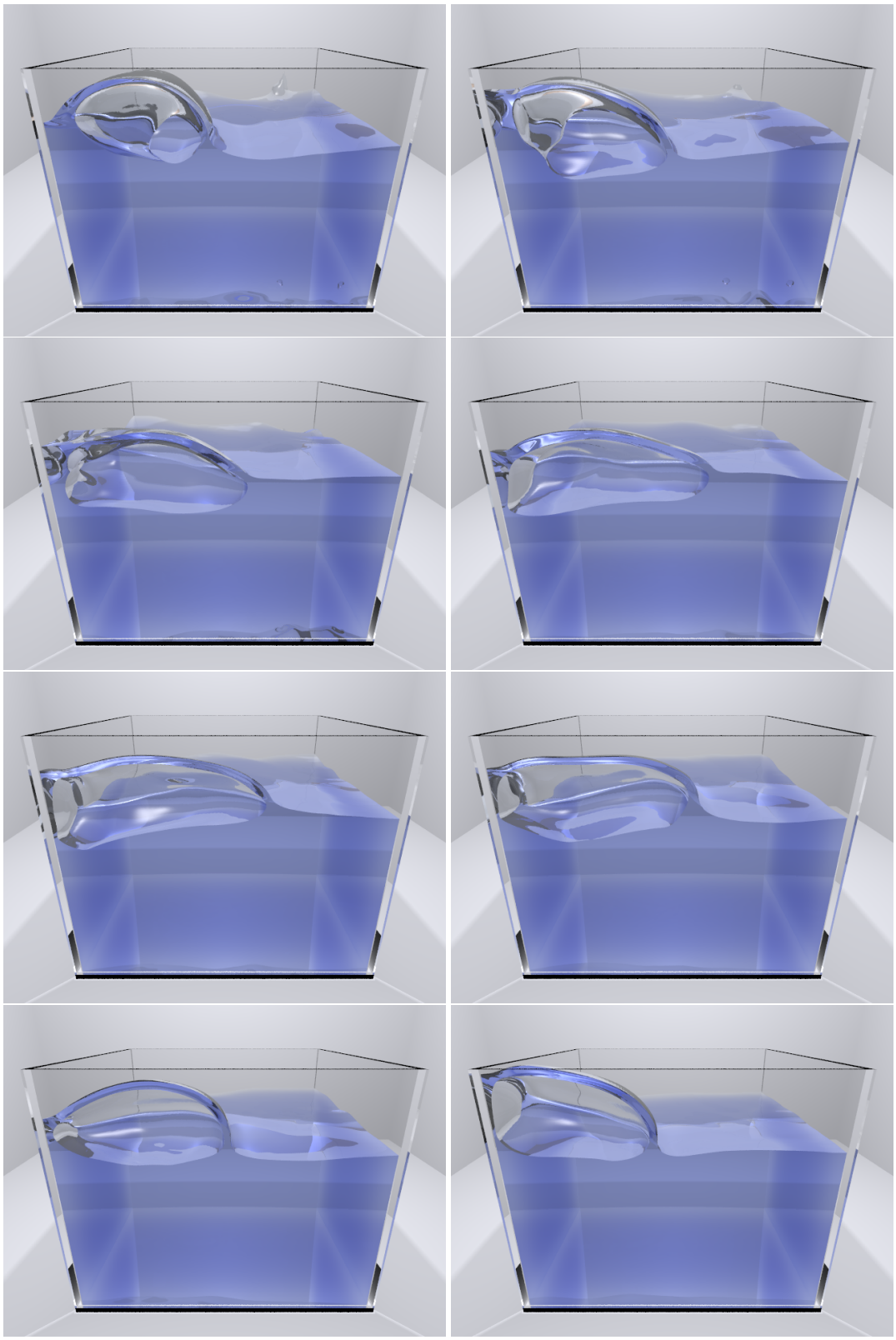

Figure 27: Simulation of the rise of two bubbles, one of which is bursting (continued). Surface tension using the ghost fluid method [31] is used.

CHAPTER IV

FLUID VOLUME CONSERVATION AND CONTROL USING NONLINEAR FEEDBACKS

4.1 Introduction and Previous Works

Fluid simulation using level set suffers from an undesired volume loss (See Fig. 38). In a two phase flow, one fluid can lose volume while the other fluid gain volume. Those volume gain or loss is the volume error. Our goal is to remove the volume error and to ensure that the volume of liquid preserved regardless of the simulation time (See Fig. 39). We also want to be able to control the volume of fluid directly to simulate bubbles that change volumes. An example of such bubbles will be bubbles inflated in boiling water.

We achieve these goals by a volume control method that is applying divergence values for fluid regions to compensate their volume error. For each of these regions, the divergence is computed carefully by first modeling the volume change equation in terms of the volume error and then constructing a nonlinear feedback controller similar to the proportional and integral feedback controller. We also validate the volume change equation and controller by several experiments.

We begin with the brief overview of volume control. First, we segment the fluid domain into several liquid or gas regions. We then compute the volume error as a difference between the desired and current volume per each region. Using this volume error as a feedback input, we compute the divergence using a nonlinear feedback control strategy. The computed divergence is the control input used to inflate/deflate each region to correct volume error. Finally, we project the velocity field to the vector field so that it has the computed divergence. This step is performed in the pressure projection step. Notice that we apply divergence to all cells in each region.

We use BFECC for the level set advection. Since BFECC produces relatively a small amount of volume loss, only a small divergence is required to correct this volume loss. Therefore, the overall fluid motion is affected minimally by the volume control. One also can use better volume preserving method such as particle level set method, with which one will need much smaller divergence.

The time for segmenting fluid into regions (connected components) is linear in the total number of octree cells. In additon, the computation of the divergence is performed in each region. Since the number of regions is much smaller than the number of cells, the cost of the computation of divergence can be ignored. Thus, volume control does not increase the runtime complexity.

Now, we review the related work. The level set method introduced by Foster et al [23] has volume loss, which is resolved by the particle level set methods [18, 17]. This particle level set method is now a widely used scheme [7, 25, 59, 17, 31] due to its volume conservation property and some other benefits such as increased surface detail and visual effects obtained by rendering escaped particles [27]. This particle level set method has a very small but continuous volume loss [46] that may eventually accumulate to a visible level [25] especially in a long simulation. In addition, particle level set methods require large amount of memory and high computational cost, in particular, when an octree grid is used [44]. In contrast, the BFECC method has a low memory and computation costs. However, even though the volume loss of BFECC is much slower than the first-order level set advection, BFECC still suffer from an observable amount of volume loss in most of the fluid animations. Therefore, the benefit of volume control is maximized in the BFECC method. To this reason, we only test our volume control approach on the fluid simulation, where the level set is advected by the BFECC method.

We apply divergence to inflate or deflate a region so that the region maintain the desired volume. A similar idea had been explored in fluid animation. The scheme called divergence sourcing was used in [45, 22]. In [22], particle explosion is achieved by treating particles as

divergence sources. In [45], divergence is applied to model the expansion from solid fuel to gas fuel. Our contribution is to apply this idea to the volume conservation, model the volume change equation, construct controllers, and provide gain synthesis methods from simulation parameters so that one can easily compute gains that provides fast and stable volume correction.

4.2 Segmentation and Tracking

A step needed for the volume control is segmenting gas and liquid regions, computing their volumes, and then tracking the movement, merge, and split of regions in the following time steps. The segmentation is rather trivial in the Eulerian grid using level set. We first consider the leaf cells as a graph, where two leaf cells are connected if they share a face. On this graph, we start a region from a leaf cell and grow that region by visiting a neighborhood in a breadth first manner. When no connected cell has the same sign of the level set variable ϕ, we start a new region. Since this requires only a normal queue rather than a priority queue, this is a linear time algorithm. The volume of the region is computed as the sum of the volumes of cubical leaf cells that belong to the region. We ignore the liquid-gas interface cutting the cells. Although this is an approximation, it only adds small amount of error.

The next problem is tracking regions in two consecutive time steps. Let $\mathscr{R}^n$ be the set of regions at the n^{th} time step, and let $\mathscr{R}^{n+1}$ be the set of regions at the $(n+1)^{\text{th}}$ time step. Let $R_i^n \in \mathscr{R}^n$ be the i^{th} region at the n^{th} time step. Similarly, let $R_j^{n+1} \in \mathscr{R}^{n+1}$ be the j^{th} region at the $(n+1)^{\text{th}}$ time step. We define $w_{i,j}$ as the volume fraction transferred from R_i^n to R_j^{n+1}.

$$w_{i,j} = \frac{|R_j^{n+1} \bigcap R_i^n|}{\tilde{V}_{i^n}}, \tag{27}$$

where $|R_j^{n+1} \bigcap R_i^n|$ is the volume of region that was the i^{th} region at the n^{th} time step and become j^{th} region at the $(n+1)^{\text{th}}$ time step. Notice that $|R_j^{n+1} \bigcap R_i^n|$ can be easily computed by counting cells that belonged to R_i^n at the n^{th} time step and belongs to R_i^{n+1} at the $(n+1)^{\text{th}}$ time step.

Suppose that there are two adjacent regions $R_{i_1}^n$ and $R_{i_2}^n$ at n^{th} time step, e.g., air and water. Suppose that $R_{i_1}^n$ is moved and segmented as $R_{j_1}^{n+1}$ in the next time step. However, we only have segmentations of n^{th} and $(n+1)^{\text{th}}$ time steps, and we do not know that $R_{i_1}^n$ is moved to $R_{j_1}^{n+1}$ yet.

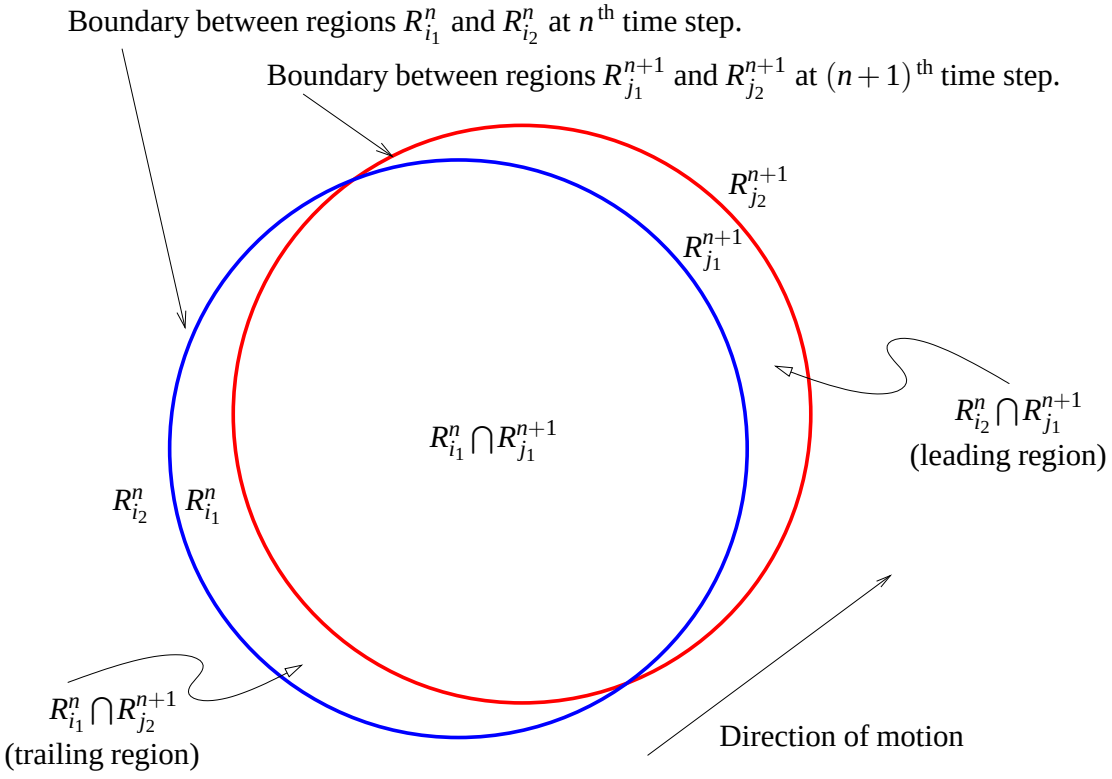

Figure 28: Tracking Region.

As illustrated in Fig 28, suppose that $R_{i_1}^n$ is a moving region. In next time step, cells that belonged to $R_{i_1}^n$ belong to the two regions $R_{j_1}^{n+1}$ and $R_{j_2}^{n+1}$. We may call $R_{i_2}^n \bigcap R_{j_1}^{n+1}$ as the leading region, and $R_{i_1}^n \bigcap R_{j_2}^{n+1}$ as the trailing region.

Since surface tension requires small time step, the level set CFL (Courant-Friedrichs-Lewy) number is less than one in most cases, and therefore, R_i^n typically moves less than a grid cell per time step. This implies that $R_{i_2}^n \bigcap R_{j_1}^{n+1}$ and $R_{i_1}^n \bigcap R_{j_2}^{n+1}$ has small volume fraction, while the region $R_{i_1}^n \bigcap R_{j_1}^{n+1}$ has volume fraction close to one. Therefore, we can consider regions of volume fractions smaller than the threshold (we chose 0.1) as leading and trailing regions due to the movement of regions. We can conclude that $R_{j_1}^{n+1}$ should

have the volume of $R^n_{i_1}$.

After zeroing volume fractions that are below the threshold, we normalize the remaining fractions so that their sum is one. Therefore, we set $w_{i_1,j_1} = 1$ and $w_{i_1,j_2} = w_{i_2,j_1} = 0$. Notice that when multiple regions, say $R^n_{i_1}, R^n_{i_2}, ...R^n_{i_m}$, are merged and split to multiple regions $R^{n+1}_{j_1}, R^{n+1}_{j_2}, ..., R^{n+1}_{j'_m}$, the above strategy is sill valid for most of cases. In this way, we do not have to track the motion of each cells. However, when the volume splits to many small volumes, the above tracking method may not work. This approach worked correctly in our experiments. One can improve this by advecting the region as in [79], but this solution requires an extended advection algorithm.

4.3 Projection to a Controlled Divergence Field

Let $\mathbf{u}^*$ be the velocity computed before the pressure projection is applied, and let c_n be the desired divergence in a region. The modified pressure projection projects $\mathbf{u}^*$ to a velocity $\mathbf{u}^{n+1}$ that has divergence c^n, i.e.,

$$\nabla \cdot \mathbf{u}^{n+1} = c^n \qquad \Rightarrow \qquad \nabla \cdot \frac{\nabla P}{\rho} = \frac{1}{\Delta t}(\nabla \cdot \mathbf{u}^* - c^n), \tag{28}$$

Since c^n is simply subtracted from the divergence, the complexity of the pressure projection step is not increased. The divergence value c^n is constant at all cells that belong to a given region. In general, different regions have different values of c^n.

4.4 The Volume Change Equation

A fluid region, such as a gas bubble, gains or loses little volume per time step. It typically takes almost 1,000 time steps until a bubble loses half of its volume even in a relatively coarse 128^3 grid. In other words, the volume change is not a high frequency behavior when BFECC is used for level set advection. This observation indicates that volume control should be relatively easy. In addition, this slow dynamic nature implies that a derivative feedback is not necessary and proportional feedback would suffice. However, when we applied this proportional feedback to a rising bubble simulation, we observed that a rising

bubble tends to saturate to a slightly smaller volume. Although this shrinkage is often negligible, the amount of shrinkage is not known. Therefore, we perform further analysis and design an additional control strategy to remove this shrinkage.

In designing a control system, the first step is to define the state variable x. We use the normalized volume error

$$x_i^n = \frac{V_i^n - \tilde{V}_i}{\tilde{V}_i}, \tag{29}$$

where V_i is the current volume of the i^{th} region, and $\tilde{V}_i$ is its desired volume. The next step is obtaining the system equation that describes the changes of this state variable x_i^n. However, modeling how the x_i^n changes at each discrete time step is difficult as the volume changes by complex simulation procedure. Therefore, we use a continuous time model, i.e., we use $x_i(t)$ and $c_i(t)$ instead of x_i^n and c_i^n. The volume rate is computed using the divergence theorem as

$$\dot{V}_i = \int_{S_i} \mathbf{u} \cdot \mathbf{n} dS \;=\; \int_{V_i} \nabla \cdot \mathbf{u} dV \;=\; c_i V_i, \tag{30}$$

where S_i is the boundary of the fluid volume V_i. Therefore, ideally, when $c_i = 0$, the volume should not change, but simulation experiments show a volume loss due to the inaccuracies in the level set advection and the pressure projection step. After several simulation experiments, we conclude that this volume loss can be modeled by the following equation using an unknown factor $\tilde{b}$.

$$\dot{V}_i = c_i V_i + \tilde{b} \tag{31}$$

When $c_i = 0$, simulation experiments show volume plot as a function of time has large linear segments connected by curved transitions. In each linear segment, the slope is constant, indicating piecewise constant $\tilde{b}$. Rewriting (31) in terms of x_i, we obtain

$$\dot{x}_i \;=\; \frac{\dot{V}_i}{\tilde{V}_i} \;=\; \frac{c_i V_i + \tilde{b}}{\tilde{V}_i} \;=\; c_i(x_i + 1) + b, \tag{32}$$

where $b = \tilde{b}/\tilde{V}$.

4.4.1 Experiments on Volume Loss

The unknown factor b primarily depends on the level set advection method, but it also depends on simulation parameters such as time step Δt, Δx, redistancing parameters, the size of bubble or liquid drop, the surface tension, and others. In addition, b changes during the simulation. For example, b is large when a liquid film is formed. However, we can observe that b changes slowly. In this section, we perform various experiments to understand the volume loss rate b. As shown in Fig. 29, 30, 31, and 32

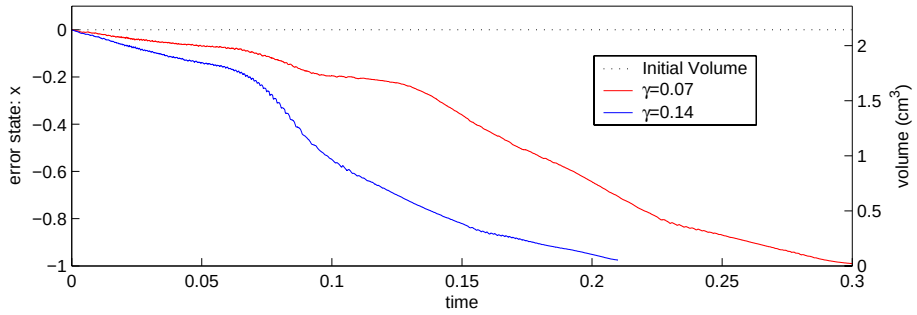

Figure 29: The volume loss rate b of a rising bubble strongly depends on the surface tension ($c_i = 0$).

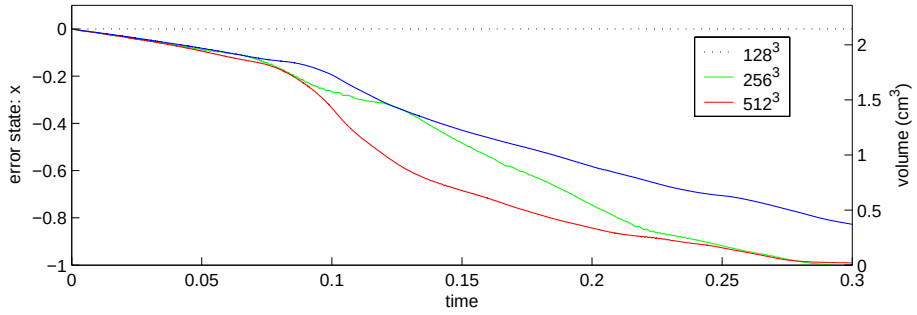

Figure 30: The volume loss rate b of a rising bubble depends on the mesh resolution ($c_i = 0$).

4.5 A Proportional Feedback Controller

The volume loss observed in the previous section can be compensated by adding a divergence c_i. This divergence c_i is applied uniformly inside each region, but different regions

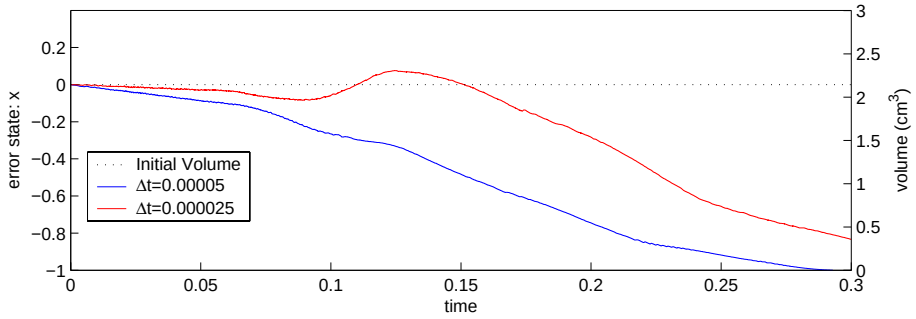

Figure 31: The volume loss rate b of a rising bubble depends on the time step ($c_i = 0$).

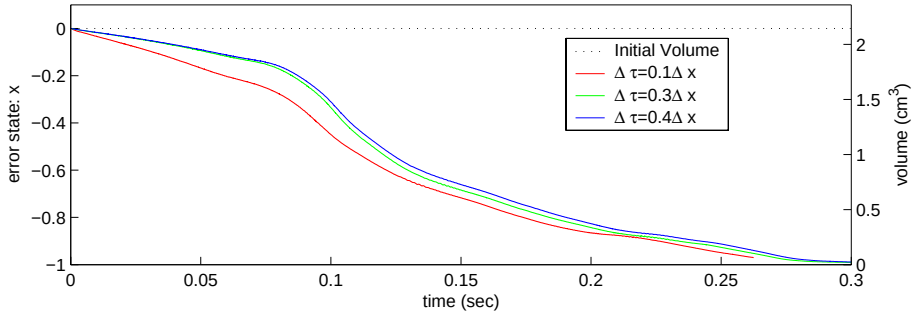

Figure 32: The volume loss rate b of a rising bubble depends on the redistancing. ($c_i = 0$).

have different values of c_i. We first consider the feedback

$$c_i = -k_P \frac{x_i}{x_i + 1}. \tag{33}$$

This is a nonlinear feedback, but since $|x_i| \ll 1$, $c_i \approx -k_P x_i$. Due to the similarity with the proportional feedback $-k_P x_i$, (33) will be called the pseudo proportional feedback or proportional feedback for simplicity.

Plugging (33) into (32), we obtain a linear equation

$$\dot{x}_i + k_p x_i \;=\; b, \tag{34}$$

which has the following explicit solution

$$x_i(t) = x_i(t_i) e^{-k_p(t-t_i)} + \frac{b}{k_P}\left(1 - e^{-kp(t-t_i)}\right), \tag{35}$$

where t_i is the time when the i^{th} region is created. If $k_P > 0$, this explicit solution shows that the error $x_i(t)$ does not converge to zero, but

$$x_i(t) \to \frac{b}{k_P} \quad \text{as} \quad t \to \infty. \tag{36}$$

This steady state error $\frac{b}{k_P}$ is the drift error we observed with the proportional feedback in the rising bubble experiment in Fig. 34 and 35. This is due to the fact that a rising bubble loses volume until the loss $V_i - \tilde{V}_i$ is large enough for the proportional controller to produce large enough divergence c_i. As a result, the volume of a rising bubble tends to saturate to a slightly smaller volume. This volume loss may not be visible (See Fig. 34), and therefore, the volume drift error may be often negligible, if a sufficiently large k_P is used. However, the amount of error is unknown since it is a function of many different simulation methods and parameters. Moreover, the volume error could be visible when the bubble is large. In particular, when multiple bubbles are stacked, the small volume losses of individual bubbles are added together produces a lowering of the surface of the foam that is clearly visible.

4.5.1 Computing Proportional Gain k_P

The equation (36) shows that the volume error can be reduced down to b/k_P, which may be very small if k_P is sufficiently large. With this observation, a natural approach to compute k_P is first defining a error tolerance ε_v, and then computing the gain as $k_P = |b|/\varepsilon_v$ so that the steady state error is smaller than the tolerance. Since b is unknown, one may estimate b by a number of experiments. However, as shown in the previous section, b is a function of many factors and it is hard to estimate. More importantly, if the error tolerance ε_v is arbitrarily small, the gain $k_P = |b|/\varepsilon_v$ will be arbitrarily large . This will produce an arbitrarily large divergence input $c_i = -k_P x_i$, which can halt the simulation if the time step is not sufficiently small. Therefore, we do not compute k_P from the steady state error tolerance.

Instead, we use the *rising time* criterion to design the gain k_P. We first assume that a fluid region has volume error x. The user specifies the number of time steps n_P, during which the volume error x is reduced down to $x/10$. The time $n_P \Delta t$ is called the *rising time*. The use of rising time criterion provides an important advantage. Since we specify the number of time steps n_P, the resulting divergence input will try to fix the volume error in n_P

time steps. For example, if we set $n_P = 25$, the volume error will be reduced down to 10% in $25\Delta t$ seconds. If we decrease the time step Δt into half, the gain will be automatically adjusted and the volume error will be reduced twice faster as $25\Delta t$ is decreased down to half, but still in $n_p = 25$ time steps. Conversely, if we increase the time step, the volume error will be fixed slowly, but still in $n_p = 25$ time steps.

Notice that instability would occur if n_P is too small. For example, if $n_P < 1$, the divergence input will be large and the volume of the region may grow or shrink more than necessary in a single time step, making the simulation unstable.

Now, we derive k_P. First, using the proportional feedback $c_i = -k_P x_i$, the system equation (41) is written as

$$\dot{x}_i = -k_P x_i + b. \tag{37}$$

As will be shown in the following experiments, the steady state volume error is small. Thus, b can be ignored for the controller construction. Ignoring b, the volume error evolves in the following discrete form

$$x_i^n = \left(e^{-k_P \Delta t}\right)^n x_i^0, \tag{38}$$

where x_i^n is the volume error of i^{th} region in n^{th} time step. Since n_P is the number of time steps required to reduce the initial error x_i^0 down to 10%, the proportional gain k_P is computed from the condition $\left(e^{-k_P \Delta t}\right)^{n_P} = 0.1$ as

$$k_P = \frac{-\ln 0.1}{n_P \Delta t} = \frac{2.3}{n_P \Delta t}. \tag{39}$$

To validate the volume change equation, and the proposed controller, we set the volume error $x_i = -0.1$ and then apply the volume control. As shown in Fig. 33, the error x_i decreases towards zero in approximately n_P time steps in various grid resolutions.

4.6 A Proportional-Integral Feedback Controller

BFECC applied to level set advection makes volume loss or gain occur slowly. Therefore, the proportional control may keep the volume loss unnoticeable. Indeed, in most cases,

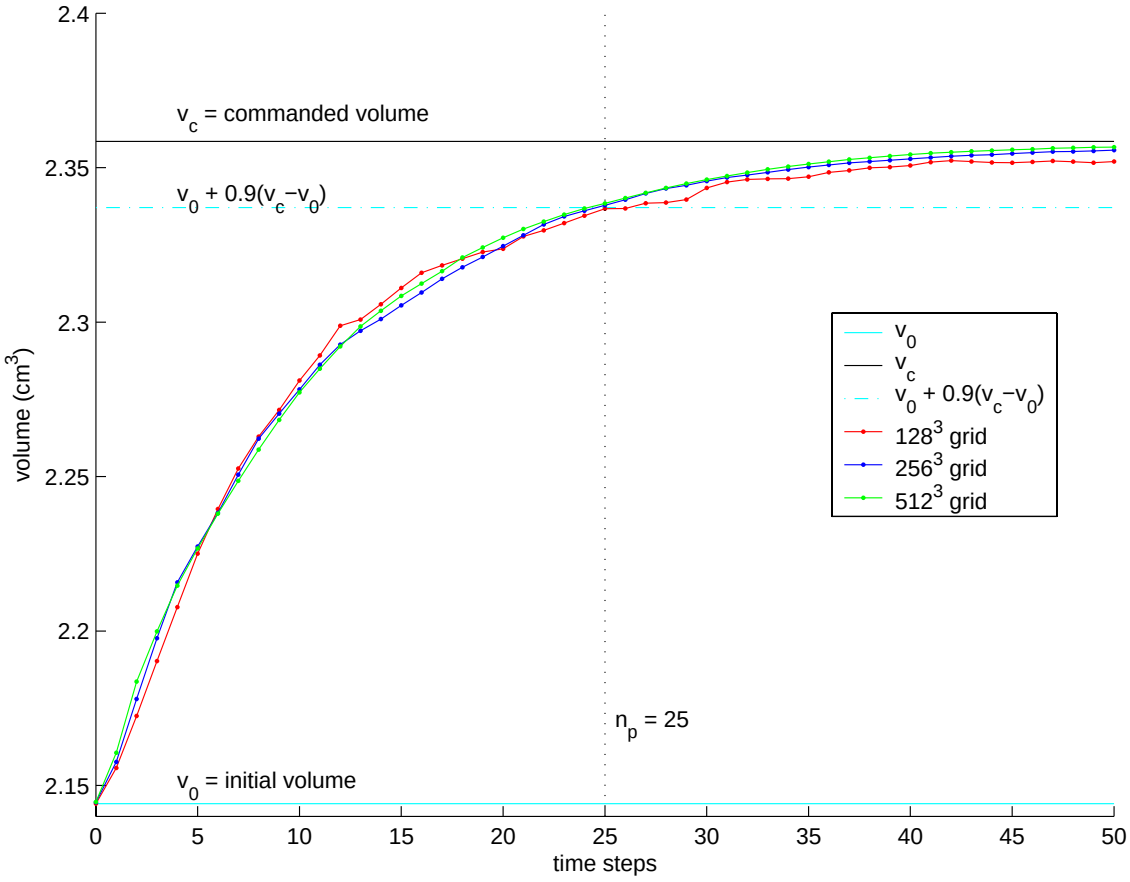

Figure 33: Step responses of volume show that the rise time is indeed n_p.

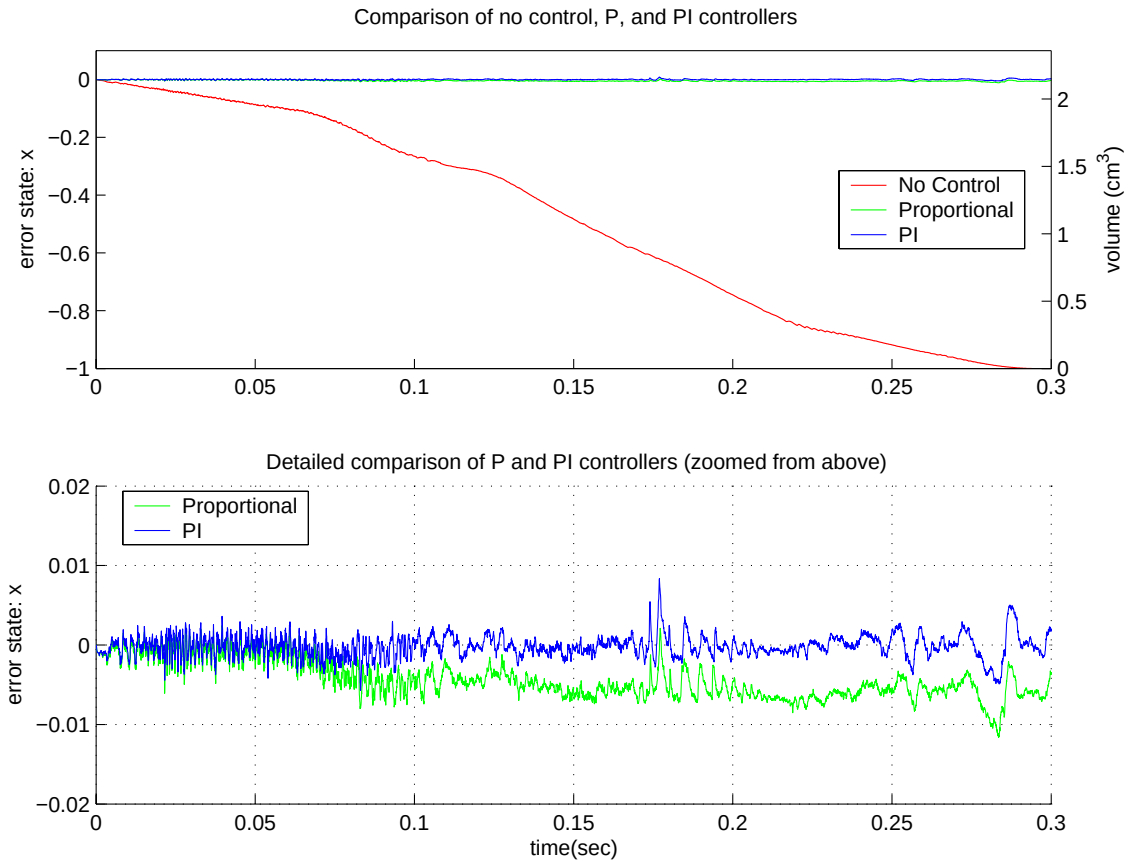

Figure 34: Comparison of controllers in a 128^3 grid. The proportional controller produces a very small error, and the PI controller does not improve much. The fluctuation may come from the error in the volume computed by counting grid cells. Notice that the controller is robust against this error.

the proportional control seems sufficient. For example, Fig. 34 shows that the volume error in 128^3 grid is less than 1%, when the proportional feedback with $n_P = 25$ is applied. However, in some cases, when a low resolution grid is used with large surface tension, the volume error tends to increase in proportional controller as shown in Fig. 35.

In the classical control strategy [63], a natural choice for removing drift error is using integral feedback, by which the small drift error will be integrated over time and then used as an additional control input. This is an efficient strategy since the small drift error can be accumulated to produce large control input. Consider the following nonlinear feedback

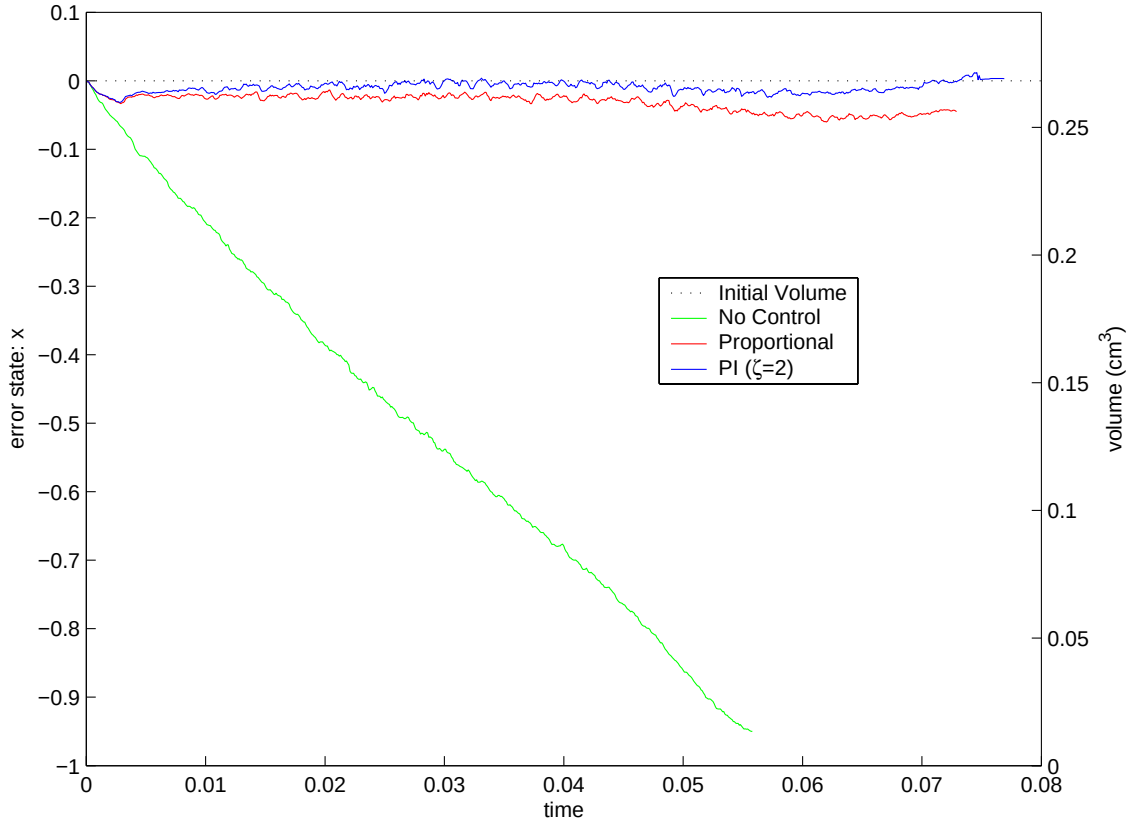

Figure 35: Comparison of controllers in a low-resolution 64^3 grid with high surface tension $\gamma = 1.0$. The proportional controller produces somewhat large error and the PI controller improves it.

with integral term

$$\begin{aligned} c_i &= \frac{-k_P x_i - k_I \int_{-\infty}^{t} x_i dt}{x_i + 1} \\ &= \frac{-k_P x_i - k_I \int_{t_i}^{t} x_i dt - k_I \int_{-\infty}^{t_i} x_i dt}{x_i + 1}, \end{aligned} \tag{40}$$

where t_i is the time when the i^{th} region, for example a bubble, is created. This is a nonlinear feedback, but since $|x_i| \ll 1$, the feedback is similar to the proportional integral feedback $c_i = -k_P x_i - k_I \int_{-\infty}^{t} x_i dt$. Therefore, we call (40) as the proportional-integral (PI) feedback. The term $\int_{-\infty}^{t_i} x_i dt$ is computed by combining the error integrals of previous components that participated to the the i^{th} component. Using this feedback, we obtain a linear equation

$$\dot{x}_i + k_p x_i + k_I \int_{-\infty}^{t} x_i dt \;=\; b, \tag{41}$$

which has the following solution

$$x_i(t) = \frac{1}{\lambda_1 - \lambda_2} \left[x_i(t_i) \left(\lambda_1 e^{\lambda_1 (t-t_i)} - \lambda_2 e^{\lambda_2 (t-t_i)} \right) + \left(b - k_I \int_{-\infty}^{t_i} x_i dt \right) \left(e^{\lambda_1 (t-t_i)} - e^{\lambda_2 (t-t_i)} \right) \right], \tag{42}$$

where λ_1, and λ_2 are the solutions of the characteristic equation ($\lambda^2 + k_P \lambda + k_I = 0$), computed as $\lambda_1 = -\frac{k_P}{2} + \frac{\sqrt{k_P^2 - 4k_I}}{2}$ and $\lambda_2 = -\frac{k_P}{2} - \frac{\sqrt{k_P^2 - 4k_I}}{2}$. If we choose positive gains, i.e., $k_P > 0$ and $k_I > 0$, the exponents $\lambda_{1,2}$ are always negative or have negative real parts, implying $e^{\lambda_{1,2} t} \to 0$ as $t \to \infty$. Therefore, the volume error tends to disappear, i.e.,

$$x_i(t) \to 0 \quad \text{as} \quad t \to \infty. \tag{43}$$

In addition, by taking time derivative of (42), we can show that

$$\dot{x}_i(t) \to 0 \quad \text{as} \quad t \to \infty. \tag{44}$$

Therefore, from (41), we obtain

$$k_I \int_{-\infty}^{t} x_i dt \;=\; b - \dot{x}_i - k_p x_i \quad \to \quad b \qquad \text{as} \quad t \to \infty, \tag{45}$$

which shows that $k_I \int_{-\infty}^{t} x_i dt$ is an estimate of the unknown factor b, and that the PI-controller uses this estimate to cancel the b factor.

The next question is the computation of $\int_{-\infty}^{t_i} x_i dt$ When the bubbles merge or split to create new bubbles, the terms $\int_{-\infty}^{t_i} x_i dt$ in (42) of the new bubbles can be computed in the following way. Suppose that bubbles identified as $j_1{}^{\text{th}}$, $j_2{}^{\text{th}}$, ..., $j_n{}^{\text{th}}$ regions at the n^{th} time step are merged to from a bigger bubble that is identified as the i^{th} region at the $(n+1)^{\text{th}}$ time step. To further generalize this, suppose that the volume fractions of $w_1, w_2, ...w_n$ of each bubble went to the i^{th} bubble. Therefore the desired volume of i^{th} bubble is $\tilde{V}_i = \sum_{k=1}^{n} w_{j_k} \tilde{V}_{j_k}$, and the current volume is $V_i = \sum_{k=1}^{n} w_{j_k} V_{j_k}$. The error integral of the new bubble $\int_{-\infty}^{t_i} x_i dt$ is computed from the error integrals of old bubbles $\int_{-\infty}^{t_{j_k}} x_{j_k} dt, k = 1, 2..n$ as

$$\begin{aligned}
\int_{-\infty}^{t_i} x_i dt &= \int_{-\infty}^{t_i} \frac{V_i - \tilde{V}_i}{\tilde{V}_i} dt \\
&= \frac{1}{\tilde{V}_i} \int_{-\infty}^{t_i} \left(\sum_{k=1}^{n} w_{j_k} V_{j_k} - \tilde{V}_i \right) dt \\
&= \frac{1}{\tilde{V}_i} \int_{-\infty}^{t_i} \left(\sum_{k=1}^{n} w_{j_k} \left(\tilde{V}_{j_k} x_{j_k} + \tilde{V}_{j_k} \right) - \tilde{V}_i \right) dt \\
&= \frac{1}{\tilde{V}_i} \int_{-\infty}^{t_i} \left(\sum_{k=1}^{n} w_{j_k} \tilde{V}_{j_k} x_{j_k} + \sum_{k=1}^{n} w_{j_k} \tilde{V}_{j_k} - \tilde{V}_i \right) dt \\
&= \frac{1}{\tilde{V}_i} \int_{-\infty}^{t_i} \left(\sum_{k=1}^{n} w_{j_k} \tilde{V}_{j_k} x_{j_k} + \tilde{V}_i - \tilde{V}_i \right) dt \\
&= \frac{1}{\tilde{V}_i} \sum_{k=1}^{n} \left(w_{j_k} \tilde{V}_{j_k} \int_{-\infty}^{t_i} x_{j_k} dt \right).
\end{aligned} \tag{46}$$

This way, one can apply the integral feedback to a new bubble. In contrast, if one start a new error integral, treating $\int_{-\infty}^{t_i} x_i dt = 0$ for newly created bubble, the bubble may shrink by a small amount for a short period of time. This shrinkage is often small and recovered quickly, and hence not visible in most cases, but we occasionally observed the volume of bubble shrank for a short period of time. Therefore, computing the error integral of a new bubble by (46) is often necessary.

4.6.1 Computing Integral Gain k_I

The drift error of proportional control can be removed by adding integral feedback. However, improperly chosen integral gain k_I can cause undesired oscillations in volume, which

indeed occurred in our simulation of stacked bubbles. This oscillation was removed when we increased the damping. Therefore, we propose a method to compute a gain k_I that provides a good damping.

Let $y = \int_0^t x_i dt$. Then, from (41), we obtain a second order system

$$\ddot{y} \; + \; k_P \dot{y} \; + \; k_I y = 0, \tag{47}$$

whose natural frequency is $\omega_n = \sqrt{k_I}$ and the damping coefficient is $\zeta = \frac{k_P}{2\sqrt{k_I}} = \frac{k_P}{2\sqrt{k_I}}$. Our goal is now choosing good values of ζ that provides enough damping. By the classical control theory, a system that has a good balance between fast error correction and damping would have $\zeta = 0.7$, which contains a small amount of oscillation that settles down quickly. When $\zeta \geq 1$, the system is critically damped or over-damped, and therefore, oscillation in x_i does not exist. Therefore, it would be safe to choose ζ larger than 0.7. In our experiments, $\zeta = 2$ worked well. After ζ is chosen, the integral gain k_I is computed as

$$k_I = \left(\frac{k_P}{2\zeta} \right)^2 \quad = \quad \frac{k_P^2}{16}, \tag{48}$$

where the proportional gain k_P is computed from (39).

4.7 Controller Implementation

The proportional controller is implemented as

$$c_i^n \quad = \quad -k_P x_i^n. \tag{49}$$

The proportional-integral (PI) controller is defined by

$$c_i^n \quad = \quad -k_P x_i^n \quad - \quad k_I \sum_{m=0}^{n} x_i^m \, \Delta t. \tag{50}$$

Therefore, it is implemented as

$$\begin{aligned} y_i^n \quad &= \quad y_i^{n-1} + x_i^n \, \Delta t \\ c_i^n \quad &= \quad -k_P x_i^n \quad - \quad k_I y_i^n. \end{aligned} \tag{51}$$

The volume loss factor b can be used to measure the volume loss property of the level set advection scheme. When PI controller is used, this factor b can be estimated as

$$b_i^n = k_I y_i^n \tag{52}$$

As mentioned earlier, the computed divergence b_i is applied to all cells that belong to i^{th} region.

4.8 Computation Time

The pressure projection is solving a matrix equation in the form $Qx = y$, and the volume control adds divergence to right hand side y. Although the matrix remains the same, added divergence may cost more iterations in the pressure projection. In experiments, the slow down in pressure projection is negligible as shown in Fig. 36. In this figure, segmentation is more expensive. However, the time for segmentation and pressure projection increases total computation time only by about $3 \sim 4\%$ as shown in Fig. 37.

4.9 Discussions on the Order of Accuracy

An interesting question left is whether the volume control affects the overall order of accuracy or not. To answer this question, first observe that the volume control adds the divergence term c to the pressure projection step, and that if c is zero, the simulation is not changed by the volume control. Therefore, if c has the order of accuracy higher than or equal to the accuracies of other simulation steps, the overall order of accuracy may not changed. On the other hand, if c has the order of accuracy lower than that of the other simulation steps, the accuracy of the entire simulation step can be lowered.

Notice that the overall simulation is first order accurate due to the use of the operator splitting. In addition, the velocity diffusion, and the pressure projection steps are also first order accurate in space and time. In particular, the accuracy of the pressure projection step implies that the divergence equation has the residual error $\nabla \cdot \mathbf{u} = O(\Delta x^2, \Delta t^2)$ per each time step. Therefore, although the level set advection is second order accurate thanks to BFECC

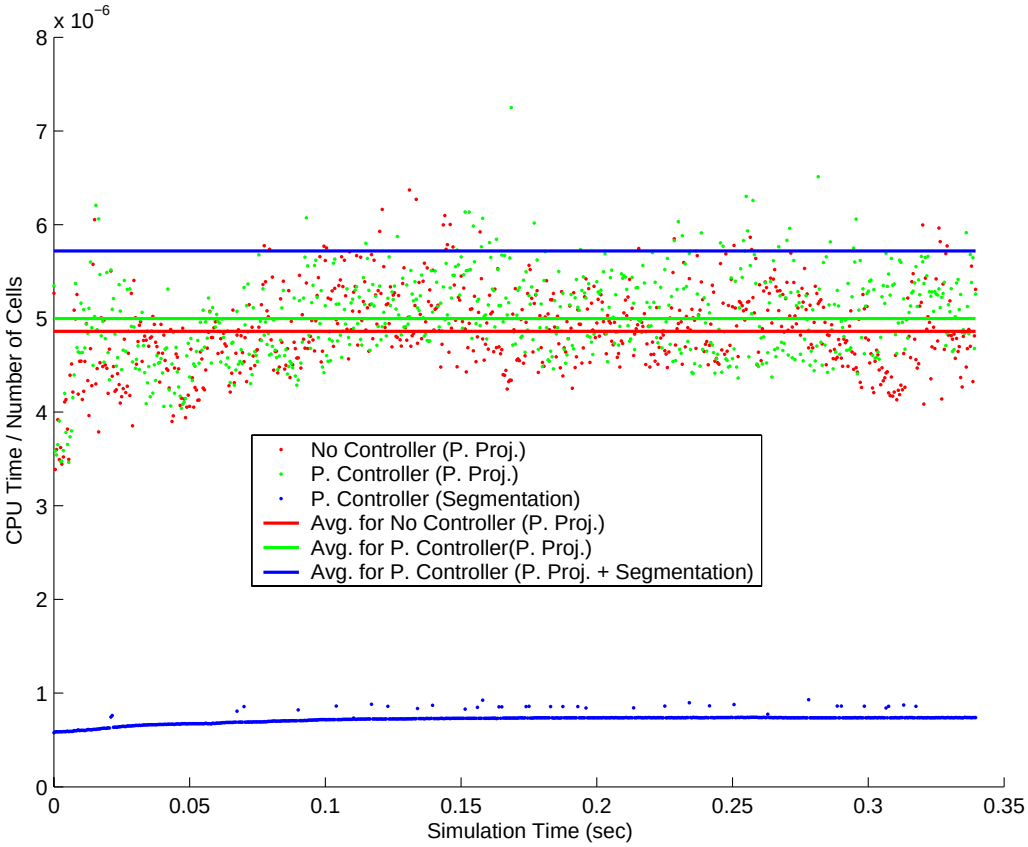

Figure 36: Computation times for pressure projection and segmentation with and without volume control. Computation time is normalized by the number of grid cells.

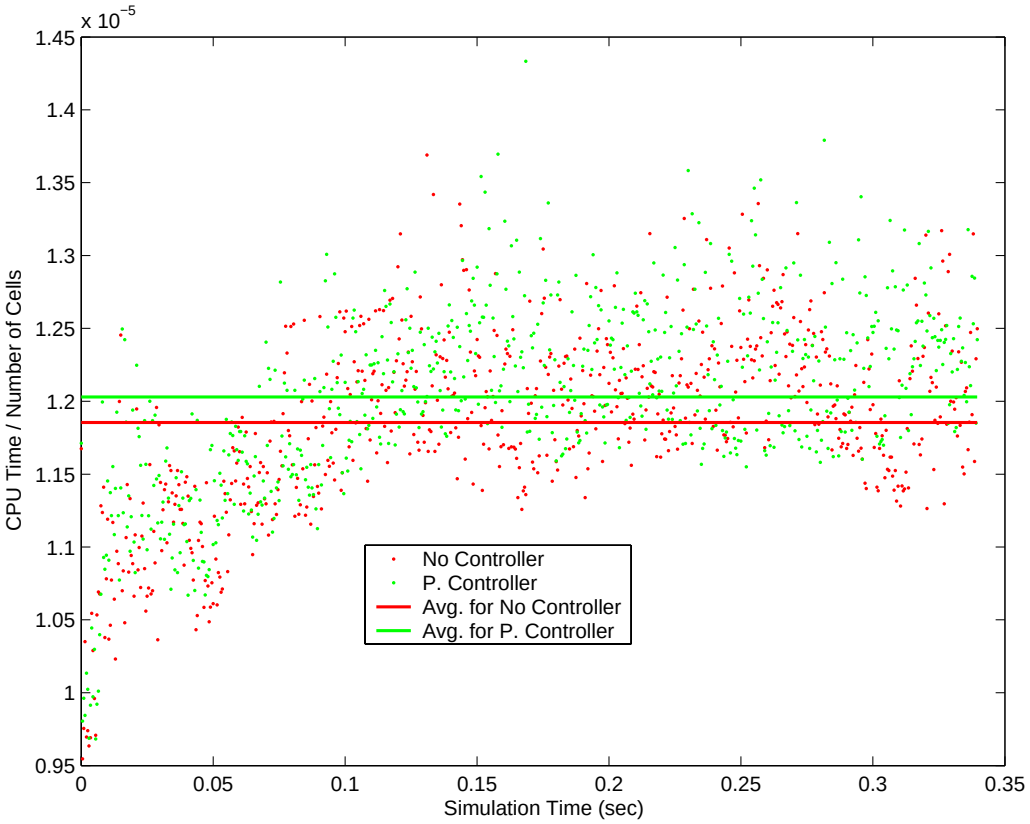

Figure 37: Total computation times with and without the volume control. Computation time is normalized by the number of grid cells.

(the residual error is $O(\Delta x^3, \Delta t^3)$ per each time step), the volume error accumulated over time remains first order. Roughly speaking, since c is proportional to the volume error, the order of accuracy may remain the same. A further analysis is required to verify this statement.

In addition, we introduced an additional error in the volume computation. Since we counted number of cells, the volume computation has an error whose magnitude is $O(\Delta x S)$, where S is the area of a region's boundary. Since the divergence c is proportional to the normalized volume error x whose magnitude is computed as $x = \frac{V-\tilde{V}}{\tilde{V}} = O(\Delta x S/V) = O(\Delta x)$, the divergence c is also $O(\Delta x)$. Although this is larger than the residual error, the volume computation error is not accumulated over time. Therefore, the error in the volume computation may not decrease the overall accuracy. We plan to perform further analysis in future.

4.10 Results

Using the volume control method, we can simulate fluid without volume loss. The severe volume loss observed in Fig. 38 can be corrected by using the volume control method as shown in Fig. 39. For the simulations in Fig. 38 and 39, we used 256^3 equivalent grid, with the surface tension $\gamma = 0.07$ N/m. The size of the computational domain is 5cm^3.

As discussed earlier, the volume control method is not only used to correct the volume error, but it can also be used to change the volume of fluid. In Fig. 40, we demonstrate this by inflating bubbles. We let the desired volume linearly increase over time. As shown in Fig. 40, the volume control produces divergence that inflate bubbles to follow the desired volume. We use a 512^3 equivalent grid, and the surface tension coefficient $\gamma = 0.07$ N/m. The size of the computational domain is 10cm^3.

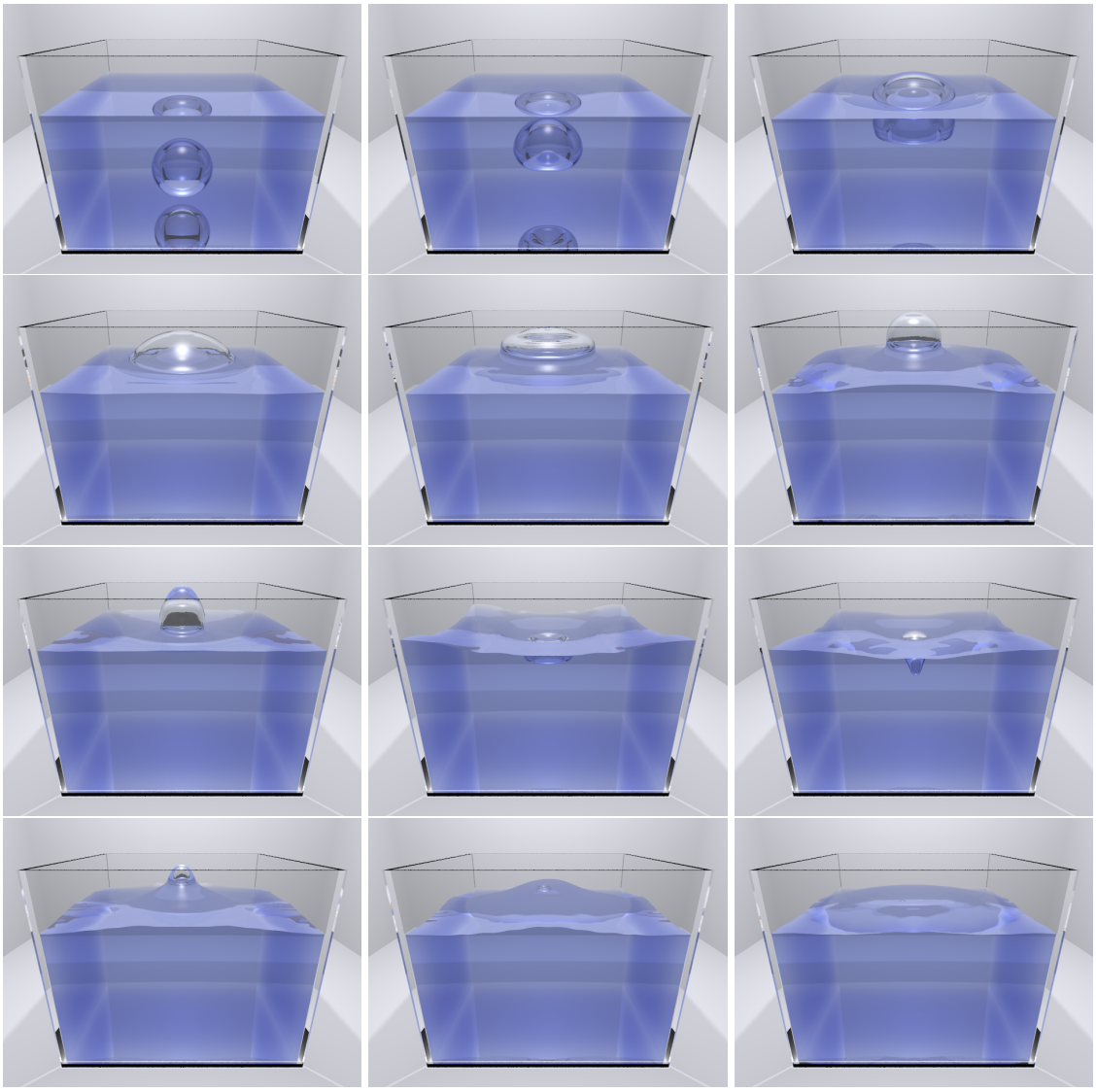

Figure 38: Simulation of a bubble without controllers on a 256^3-equivalent octree grid. The volume is almost lost after a while.

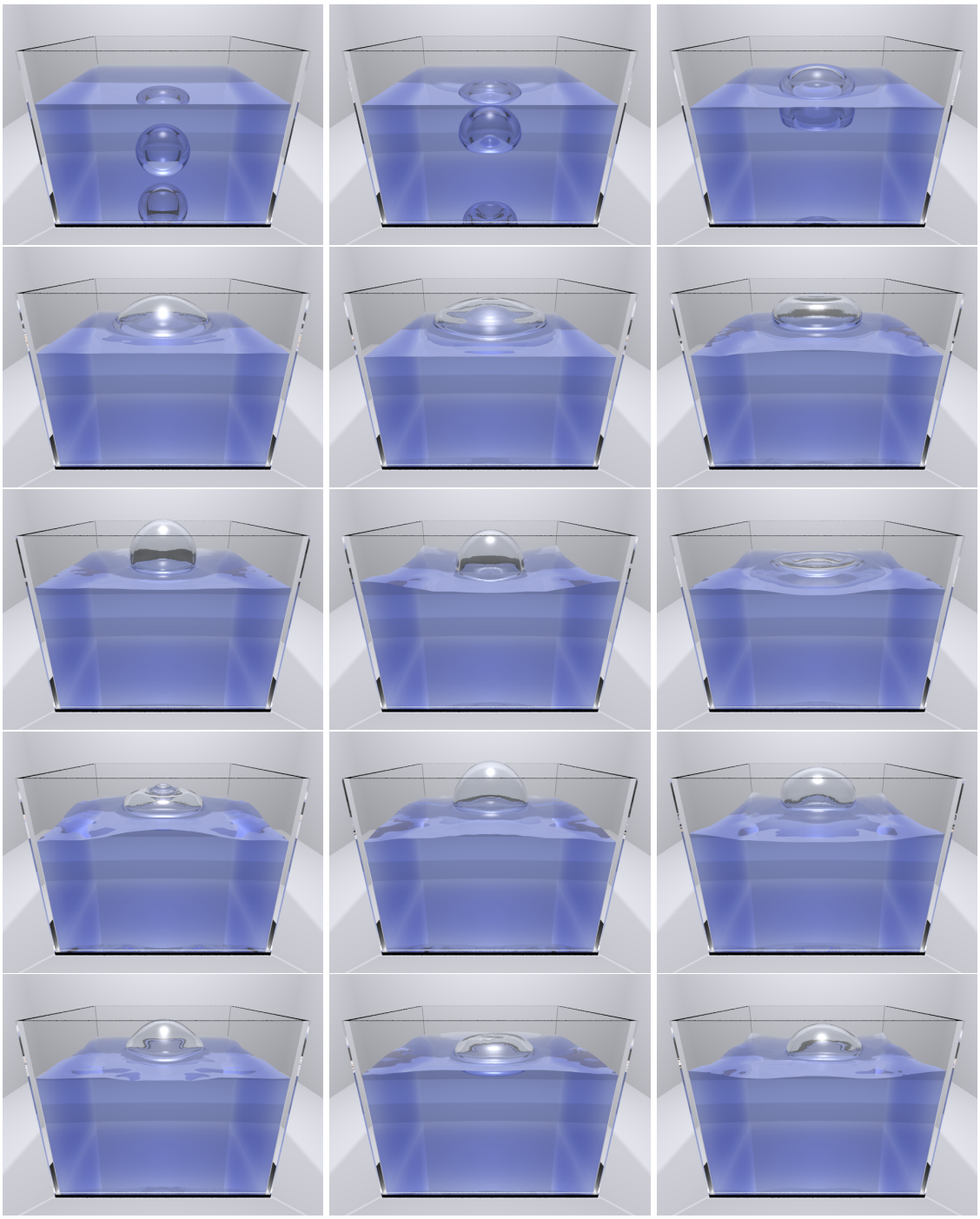

Figure 39: Simulation of a bubble with proportional controller on a 256^3-equivalent octree grid. The volume of bubble is preserved.

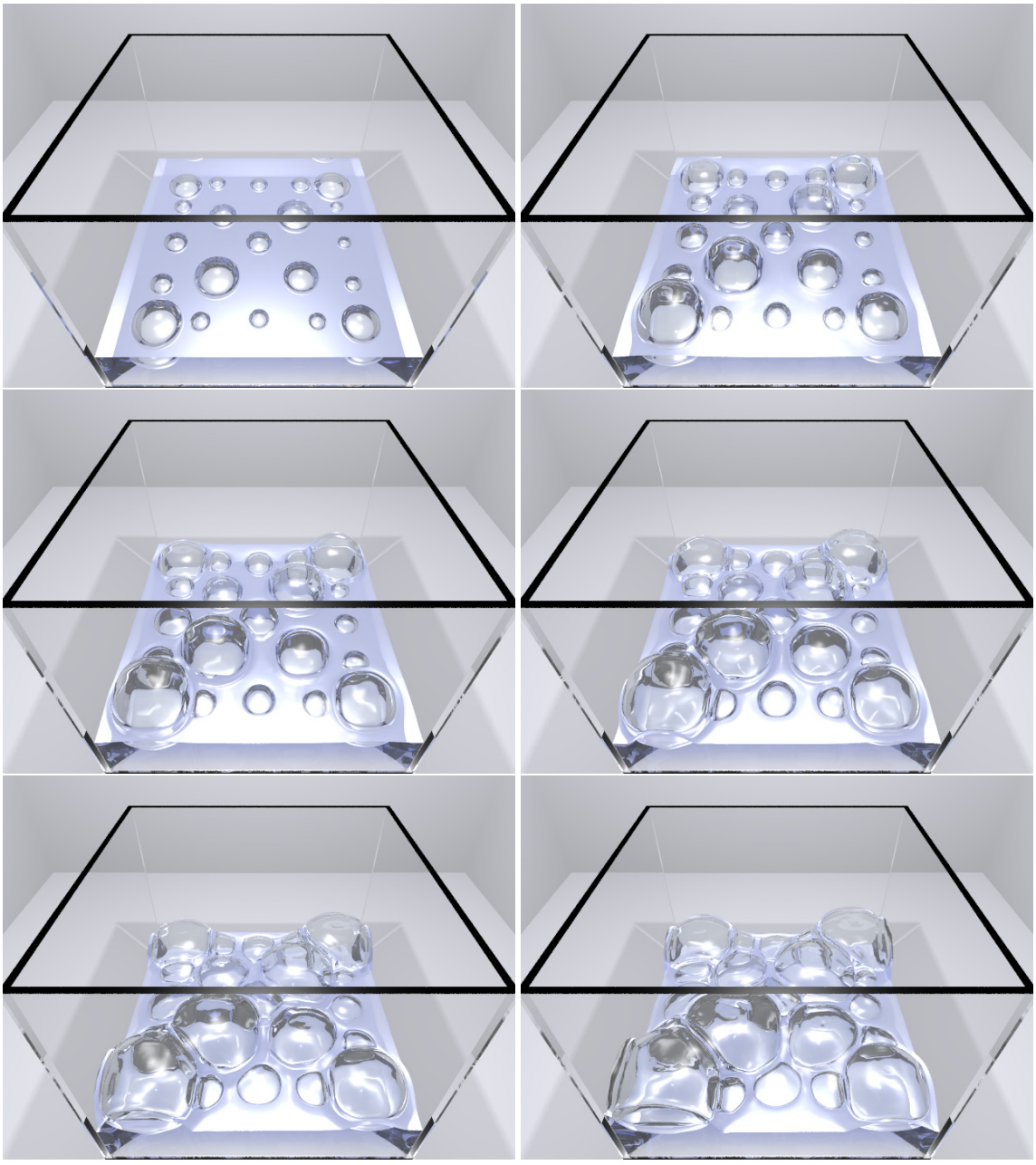

Figure 40: Simulation of inflated bubbles.

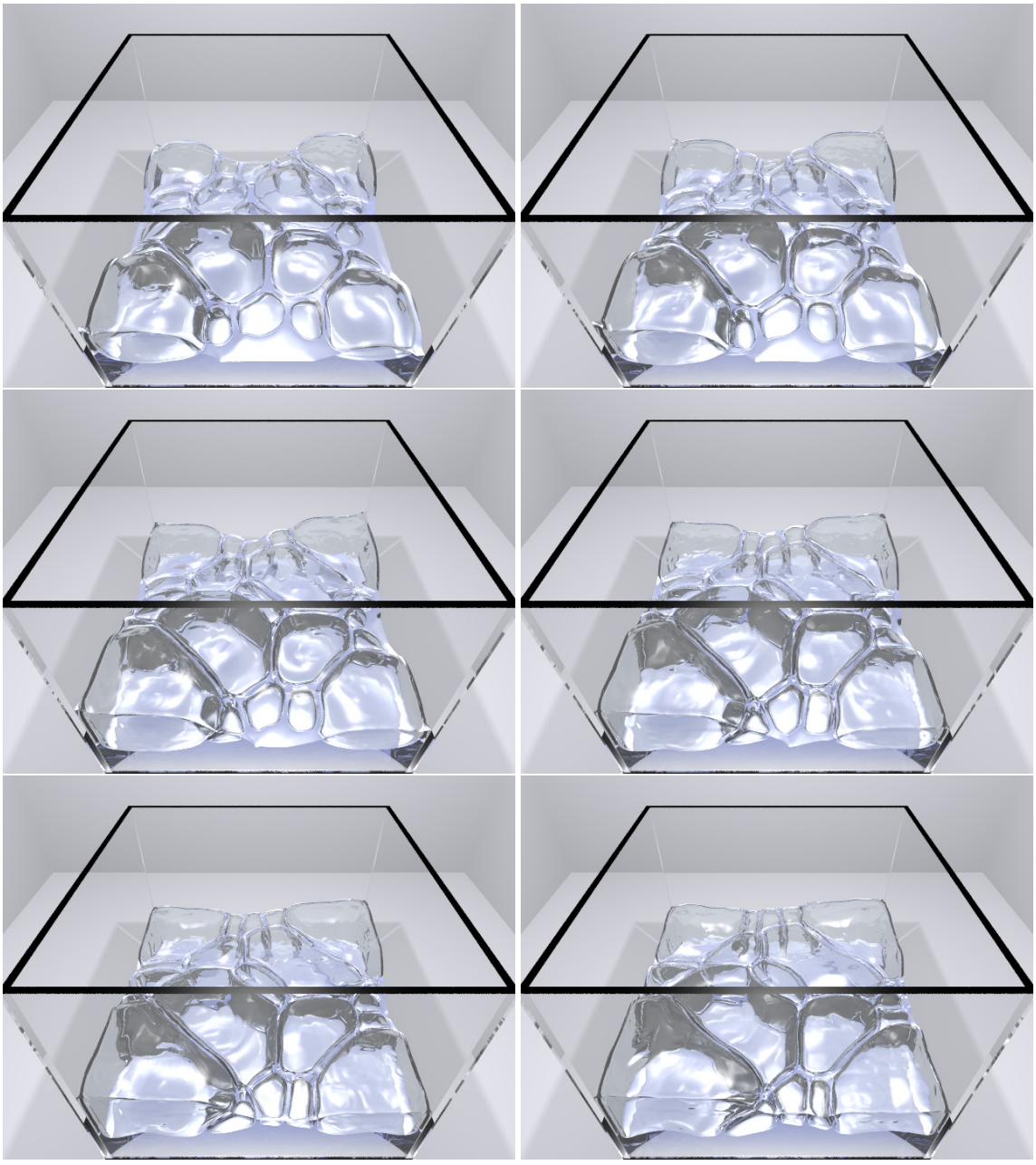

Figure 41: Simulation of inflated bubbles (continued).

CHAPTER V

A MESH FILTER IN RATIONAL FORM

5.1 Surface Smoothing

5.1.1 Choice of Discrete Laplacian Operator

The algorithms for smoothing triangle meshes have been studied extensively. In Taubin's work [72], the low frequency modes in a mesh are preserved, while high frequency modes are attenuated. The underlying theory is that the eigenvector of the negative Laplacian operator of a surface represents different frequency modes of the surface, *i.e.*, the larger the associated eigenvalue is, the higher the frequency mode it represents. However, the quantitative relation between the eigenvalues of the discrete Laplacian matrix and the frequency of the noise on the surface is not known. Furthermore, the discrete Laplacian operator does not approximate the continuous Laplacian and hence one cannot expect that the eigenvalue/eigenvector pair approximates the eigenvalue/eigenfunction pair of the smooth surface. Indeed, all the eigenvalues of the discrete Laplacian operator fall into the interval $[0,2]$, whereas the eigenvalue of the smooth surface can be infinitely large. In the discrete mesh, the finer the mesh, the larger the frequency the mesh can represent and therefore eigenvalue associated with such a high frequency mode can be very large. In [78], the discrete operator used for filtering is the affinity matrix, which does not have a known corresponding continuous operator for a smooth surface.

Hence, we have decided to use a discrete Laplacian operator that is convergent to the continuous one, as the mesh is refined. A popular formulation of the discrete Laplacian operator is based on the cotangent weights [55] that is used for mesh filtering in [11]. It can be shown by Taylor series expansion [76] that when the cotangent weights are divided by one third of the area of neighboring triangle ($\tilde{A}$ in (53)) this formulation approximates

the continuous Laplace-Beltrami operator if the mesh is regular, i.e., the edge length and triangle angles are uniform. We show that, when this operator is used, the cutoff frequency can be selected using a more intuitive measure such as the geometric dimension of a feature.

Let $\mathbf{L} \in \mathbb{R}^{n_v \times n_v}$ be the discrete Laplacian operator defined as

$$(-\mathbf{L})_{ij} = \begin{cases} \dfrac{1}{\tilde{A}_i/3} \displaystyle\sum_{k \in \mathscr{N}_i} (\cot \alpha_{ik} + \cot \beta_{ik}) & , \quad i = j \\ -\dfrac{1}{\tilde{A}_i/3} \left(\cot \alpha_{ij} + \cot \beta_{ij}\right) & , \quad i \neq j \end{cases} \tag{53}$$

where $(\mathbf{L})_{ij}$ is the i,j-component of $\mathbf{L}$, n_v is the number of vertices, $\tilde{A}_i$ is the sum of area of triangles around i^{th} vertex, $\mathscr{N}_i$ is the set of indices of vertices around the i^{th} vertex and α_{ik}, β_{ik} are angles of the corners facing the edge connecting i^{th} and k^{th} vertices.

5.1.2 Decomposition of the Operator

The matrix $\mathbf{L}$ is not symmetric, but it can be decomposed into a multiplication of a diagonal matrix $\mathbf{M}$ with a symmetric matrix $\mathbf{K}$ defined as

$$\begin{aligned} -\mathbf{L} &= \mathbf{M}^{-1}\mathbf{K} \ , \text{ where} \\ (\mathbf{M})_{ii} &= \tilde{A}_i/3 \\ (\mathbf{K})_{ij} &= \begin{cases} \sum_{k \in \mathscr{N}_i} (\cot \alpha_{ik} + \cot \beta_{ik}) & , \quad i = j \\ -\left(\cot \alpha_{ij} + \cot \beta_{ij}\right) & , \quad i \neq j \end{cases} \end{aligned} \tag{54}$$

Note that the matrices $\mathbf{M}$ and $\mathbf{K}$ appear when one constructs a linear finite element formulation of the PDE $\dot{\phi} - \Delta\phi = 0$ of a scalar filed ϕ over the triangle mesh. The finite element formulation yields $\mathbf{M}\{\dot{\phi}\} + \mathbf{K}\{\phi\} = 0$, where $\{\phi\}$ is the long vector of ϕ sampled at the vertices, where $\mathbf{K}$ is the stiffness matrix corresponding to the negative Laplacian term, and $\mathbf{M}$ is the lumped mass matrix.

Notice that if all triangles are properly oriented and there is no triangle with negative or zero area, the element stiffness matrix $\mathbf{K}_e \in \mathbb{R}^{3\times 3}$ is positive semi-definite with a one dimensional null space. Since $\mathbf{x}^T\mathbf{K}\mathbf{x} = \sum_{\forall e} \mathbf{x}_e^T \mathbf{K}_e \mathbf{x}_e$, where $\mathbf{x}_e \in \mathbb{R}^3$ is the collection of entries in $\mathbf{x}$ that belong to the e^{th} triangle, $\mathbf{x}^T\mathbf{K}\mathbf{x} \geq 0$ ($\mathbf{x}^T\mathbf{K}\mathbf{x} = 0$ if and only if all entries

of **x** are the same when the mesh has one connected component). Consequently, **K** is (symmetric) positive semi-definite with one zero eigenvalue and **M** is a diagonal matrix with positive elements. Consider the eigen decomposition of the positive semi-definite matrix $\mathbf{M}^{-1/2}\mathbf{K}\mathbf{M}^{-1/2} = \tilde{\mathbf{V}}\Lambda\tilde{\mathbf{V}}^T$, where $\tilde{\mathbf{V}}$ is orthogonal and Λ is diagonal. By multiplying it by $\mathbf{M}^{-1/2}$ to the left and $\mathbf{M}^{1/2}$ to the right, we obtain an eigen decomposition of $-\mathbf{L}$

$$\begin{aligned} -\mathbf{L} &= \mathbf{M}^{-1/2}\left(\mathbf{M}^{-1/2}\mathbf{K}\mathbf{M}^{-1/2}\right)\mathbf{M}^{1/2} \\ &= \mathbf{M}^{-1/2}\tilde{\mathbf{V}}\ \Lambda\ \left(\mathbf{M}^{-1/2}\tilde{\mathbf{V}}\right)^{-1} = \mathbf{M}^{-1/2}\tilde{\mathbf{V}}\ \Lambda\ \tilde{\mathbf{V}}^T\mathbf{M}^{1/2} \end{aligned} \tag{55}$$

If we define $\mathbf{V} \equiv \mathbf{M}^{-1/2}\tilde{\mathbf{V}}$, then $\mathbf{V}^{-1} = \tilde{\mathbf{V}}^T\mathbf{M}^{1/2} = \mathbf{V}^T\mathbf{M}$ and we have

$$-\mathbf{L} = \mathbf{M}^{-1}\mathbf{K} = \mathbf{V}\Lambda\mathbf{V}^{-1} = \mathbf{V}\Lambda\mathbf{V}^T\mathbf{M} \tag{56}$$

V is an eigenvector matrix of $-\mathbf{L}$.

5.1.3 Construction of Filter

Let $\mathbf{p} \in \mathbb{R}^{n_v \times 3}$ be the matrix whose columns are the x, y and z coordinates of vertices. Then, $-\mathbf{L}\mathbf{p}$ is the normal vector whose magnitude is twice the mean curvature at each vertex, and hence $-\mathbf{L}$ is the discrete Laplace-Beltrami operator [6].

If we can compute **V** exactly, we can drop high frequency $\mathbf{v}_i$, yielding an ideal filtering. However, in a large mesh, it is not practical to compute more than the first few eigenvectors. Moreover, even if possible, ideal filtering often creates ripples and hence not always ideal in practice. Thus, the approach in [72] is very efficient, since a carefully designed filter can keep the low frequency while reducing high frequency without having to compute the eigenvector **V**.

We propose the following filtering formula inspired by the linear discrete system used in DSP and control fields.

$$\left(a_0\mathbf{I} - a_1\mathbf{L} + a_2\mathbf{L}^2 - ...\right)\mathbf{p}' = \left(b_0\mathbf{I} - b_1\mathbf{L} + b_2\mathbf{L}^2 - ...\right)\mathbf{p}$$

Applying the eigen decomposition of $-\mathbf{L}$ in (55) yields

$$\left(a_0\mathbf{I} + a_1\Lambda + ...\right)\mathbf{V}^{-1}\mathbf{p}' = \left(b_0\mathbf{I} + b_1\Lambda + ...\right)\mathbf{V}^{-1}\mathbf{p} \tag{57}$$

Let s_i be the i^{th} diagonal element of Λ. Then, the i^{th} row of (57) is

$$\mathbf{V}^{-1}\mathbf{p}' \;=\; \text{diag}\,[G(s_i)]\;\mathbf{V}^{-1}\mathbf{p} \tag{58}$$

where $\text{diag}\,[G(s_i)]$ is a diagonal matrix with $G(s_i)$ in its diagonal. The function $G(s)$ is called the transfer function and is the amplification factor for the eigenmode associated with the eigenvalue s. One can see that $G(s)$ is in the following form

$$G(s) = \frac{b_0 + b_1 s + b_2 s^2 + \ldots}{a_0 + a_1 s + a_2 s^2 + a_3 s^3 + \ldots} \tag{59}$$

Finally, we have

$$\mathbf{p}' \;=\; \mathbf{V}\,\text{diag}\,[G(s_i)]\;\mathbf{V}^{-1}\mathbf{p} \tag{60}$$

which shows that $\mathbf{p}' = \mathbf{p}$ if $G(s) = 1$ for all s. If $G(s)$ is small (large) for some s, then $\mathbf{p}'$ will have small (large) contribution from the mode associated with it. Thus, if $-\mathbf{L}$ is the discrete Laplacian operator that approximates the continuous one, one can assume that the eigenvectors in $\mathbf{V}$ and the associated eigenvalues approximate the physical frequencies in the surface. Hence, one can design $G(s)$ to exaggerate/attenuate certain frequency pattern in the surface.

By choosing different coefficients, we can design a variety of filters, such as lowpass, high pass, bandpass, or notch filters. Other filters or transfer functions design methods such as the classical pole-zero placements, butterworth, Chebyshev and other filters in analog/digital controls and DSP literatures could also be used with slight modifications as needed.

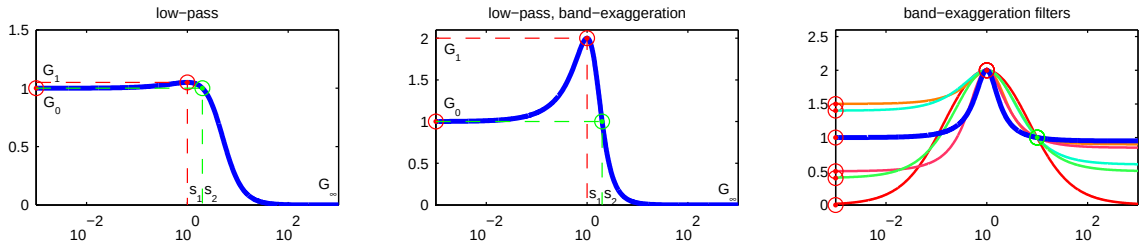

Figure 42: Lowpass filter and its variations constructed from (67) with $0 < s_1 < s_2, G_\infty < 1 < G_1, G_0 < G_1$

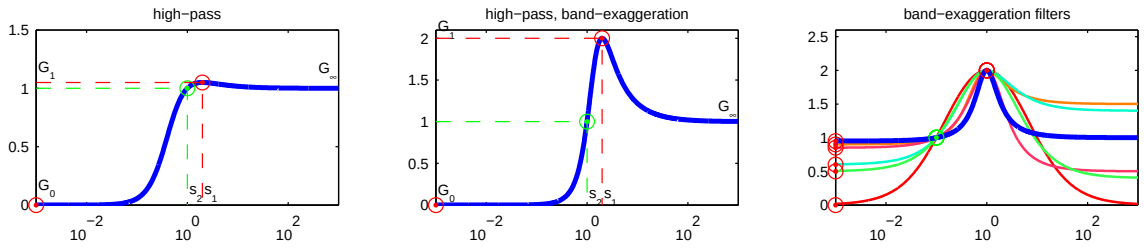

Figure 43: Highpass filter and its variations constructed from (67) with $0 < s_2 < s_1, G_0 < 1 < G_1, G_\infty < G_1$

5.1.4 Converting to Symmetric Matrix Equation

Since $\mathbf{L} = -\mathbf{M}^{-1}\mathbf{K}$ is not symmetric, a bi-conjugate gradient method has been used [11]. Instead, we propose to left-multiply the equation by the diagonal $\mathbf{M}$, yielding

$$\left(a_0\mathbf{M} + a_1\mathbf{K} + a_2\mathbf{K}\mathbf{M}^{-1}\mathbf{K} + \ldots\right)\mathbf{p}' = \left(b_0\mathbf{M} + b_1\mathbf{K} + \ldots\right)\mathbf{p} \tag{61}$$

Now, we have a symmetric matrix and the equation can be solved by the simpler conjugate gradient methods. We use a simple Jacobi preconditioner. Notice that the sparse matrix $\mathbf{M} + a_1\mathbf{K} + a_2\mathbf{K}\mathbf{M}^{-1}\mathbf{K} + \ldots$ does not have to be computed. The CG iteration only requires matrix vector multiplications. Thus, $\left(\mathbf{M} + a_1\mathbf{K} + a_2\mathbf{K}\mathbf{M}^{-1}\mathbf{K} + \ldots\right)\mathbf{p}$ can be conveniently computed as the sum of a cascaded series of simpler operations, such as $\mathbf{Mp}$, $\mathbf{Kp}$. The efficient computation of the Jacobi preconditioner is not trivial but it can be performed in linear time by taking advantage of the sparsity information that is available from the connectivity of the mesh.

5.2 Previous Works

The λ/μ filter [72] is in the following form.

$$\mathbf{p}' = \left(\mathbf{I} + b_1(-\mathbf{L}) + b_2(-\mathbf{L})^2\right)^n \mathbf{p} \tag{62}$$

Thus, the transfer function is

$$G(s) = \left(1 + b_1 s + b_2 s^2\right)^n \tag{63}$$

Since the operator $-\mathbf{L}$ chosen in [72] has eigenvalue less than two, a proper choice of b_1, b_2 will keep $G(s)$ small on the interval [0,2]. However, when one uses the Laplacian operator in (53), its eigenvalues are large, yielding large $G(s)$ for large eigenvalue s. Thus, the explicit filter formulation will not suffice for any operator that approximates the continuous Laplace-Beltrami operator. Hence, Desbrun et al used an implicit formula [11] using an operator nearly similar to (53).

$$(\mathbf{I} + a_1(-\mathbf{L}))^n \mathbf{p}' = \mathbf{p} \tag{64}$$

The transfer function is

$$G(s) = \frac{1}{(1 + a_1 s)^n} \tag{65}$$

They used a uniform scaling factor to preserve the volume. In implicit form, $G(s)$ is safe for very large s since it will result in a very small value of G, attenuating the high frequency mode associated with it. The drawback is the lack of a flat lowpass band, which may yield an annoying shrinkage of features that one may want to preserve. Even though the operator chosen was close to (53), the quantitative meaning of the filter frequencies was not studied.

Another implicit filter is the butterworth filter found in [78]. In this work, the transfer function is in the form

$$G(s) = \frac{1}{1 + a_2 s^2} \tag{66}$$

Again, the gain is a monotonically decreasing function of s and hence the attenuation of the low frequency is inevitable. A typical example is the shrinkage of the bunny ear. A higher order butterworth filter will produce more flatness at low frequencies [10], since the filter is designed to have maximal flatness at zero frequency. However, to maintain a flat lowpass band, the order of the filter needs to be high and hence the cutoff rate will be very steep, approaching the ideal filtering, which can cause ripples, an effect known as ringing. This phenomenon can happen even in the second order filter, whose maximum cutoff rate is -40db. However, in our filter construction, it can be easily reduced by choosing a larger s_2, which is defined in section 5.3.1.

It should be mentioned that the Laplacian operators found in [72, 40, 36] regularize the mesh while performing the filtering operation. The reason is complex. An intuitive understanding may be gained by deriving the Laplacian operator in [40] from the finite element framework, where edges correspond to string elements with nominal length zero, and their stiffness are proportional to lengths. This yields long edges pull harder and shrink, while stretching short edges, which yields mesh regularization. Unfortunately, (53) does not have a similar effect. A remedy was proposed by [50], where they constructed a hybrid operator that uses (53) for normal displacement of the vertex and the umbrella operator in [40] for tangential motion with some adaptation. In contrast, our approach applies the mesh filter only once and hence the mesh regularity deteriorates little. Therefore, we do not need to embed mesh regularization.

An alternative filtering approach is to build a multi-resolution hierarchy that contains different level of detail and then selectively reduce or amplify various detail levels. In [28], the progressive mesh is used to build the multi-resolution and then different refinement steps are zeroed, kept or amplified. Again, the mesh refinement step n and size of feature are not explicitly related and hence the user need to choose n by trial and error. Exaggeration of mesh feature can also be found in [77], where Zelinka et al picked a feature using a geodesic fan and then searching the mesh for similar features to exaggerate.

5.3 Filter Design

5.3.1 Exaggeration Filter Design

In this section, we explain how we designed a filter that exaggerates certain frequencies. We may also remove high frequencies and keep low frequencies, or vice versa. We use a second order polynomial for both the numerator and denominator, which gives us choices for six coefficients: $a_0, a_1, a_2, b_0, b_1, b_2$. A higher order polynomial would afford more flexibility and sharper cutoffs, but would considerably slow down computation. Therefore, in order to support interactive design, we have opted for second order polynomials.

As is shown in the far left image of Fig. 42, we allow the user to specify the DC gain $G(0) = G_0$ and the high frequency gain $G(\infty) = G_\infty$, which should be zero if one wants to attenuate the high frequencies. The user can also select the location of the maximum gain such that $G(s_1) = G_1$ to design the frequency and amplification factor. We also allow the user to specify another frequency s_2, such that $G(s_2) = 1$, to specify when the gain falls off to one. Then, given the five parameter $s_1, s_2, G_0, G_1, G_\infty$, the filter coefficients are computed as

$$\begin{aligned} a_0 &= 1 \ , \quad b_0 = G_0 \ , \quad a_2 = -\frac{G_0 - G_1}{G_1 - G_\infty}\frac{1}{s_1^2} \ , \quad b_2 = a_2 G_\infty \\ a_1 &= -\frac{G_0 - G_1}{1 - G_1}\frac{2}{s_1} - \frac{1 - G_0}{1 - G_1}\frac{1}{s_2} - a_2\frac{1 - G_\infty}{1 - G_1}s_2 \\ b_1 &= \frac{G_0 - G_1}{1 - G_1}\frac{s_2 - 2s_1}{s_1^2} - \frac{1 - G_0}{1 - G_1}\frac{G_1}{s_2} - a_2 G_\infty \, s_2 \end{aligned} \tag{67}$$

where $G_1 \neq 1$.

If $s_2 > s_1 > 0, G_1 > G_0$ and $G_1 > G_\infty$, one obtains a set of filters shown in Fig. 42 that includes lowpass, band exaggeration filters with the option of high frequency reduction. In lowpass filter design, when stronger attenuation in high frequency is needed, one may choose $G_\infty = 0$. In this case, $b_2 = 0$ and the high frequency cutoff of 20db is achieved. This can be increased to 40db if $G_0 = 1$ and $s_2 = 2s_1$ since $b_1 = b_2 = 0$. The example of this steep cutoff can be found in the second image in Fig. 52, where the high frequency noise in the chin, shoulder and ear have disappeared. When we choose $G_0 = G_\infty = 0$, we obtain a bandpass filter as is shown in red in the last image of Fig. 42. If $s_1 > s_2 > 0, G_1 > G_\infty$ and $G_1 > G_0$, one obtains the highpass filter with options of various exaggerations and bandpass filter as illustrated in Fig. 43. Notice that those highpass filters will collapse the mesh since they remove low frequencies and hence are less useful than lowpass filters. However, they may still be used in some application that needs to compute the strength of high frequency signal or transfers high frequency details of a mesh to other mesh after constructing a mapping between two surfaces. One can also obtain a notch filter by $G_0 > 1, G_\infty > 1$ and assigning small value to G_1, as is shown in Fig. 44. An example of this notch filter can be found in the third image of Fig. 49. Also, when one increases $G_\infty > 1$ and G_1 close to one,

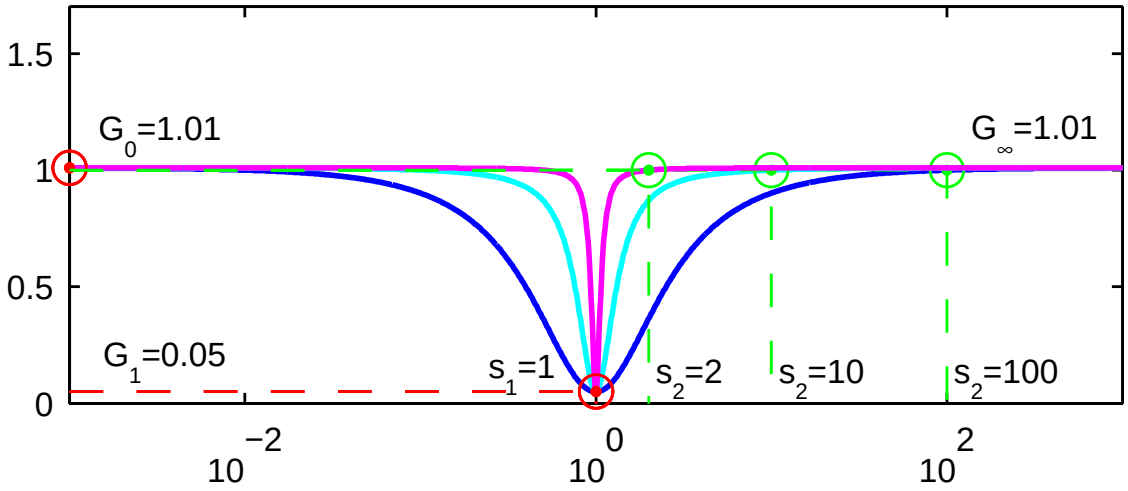

Figure 44: Notch filters with different stop-band widths.

a high frequency exaggeration filter is obtained, as is shown in Fig. 45.

In conclusion, our filter formulation in (61) and (67) can be used for a variety of mesh processing applications.

5.3.1.1 Note on Computation Time and Higher Order Filters

Our filter (67) is second order. The higher order filter can provide more freedom in designing $G(s)$. For example, Chebyshev filter can reduce ripples in the pass band. However, as the order of the filter grows, the condition number of the matrix in the left hand side of (61) will grow exponentially and hence applying it will be very slow. Alternatively Zhang et al [78] suggest an idea that can possibly remedy this by factoring high order polynomials into a product of quadratic or linear polynomials. Zhang et al. even factored the denominator of (66) into $\left(1+\sqrt{a_2}s\right)\left(1+j\sqrt{a_2}s\right), j=\sqrt{-1}$ and then solved the two first order complex matrix equations.

Our filter process takes a few seconds for a model with 3,291 vertices. For the dinosaur model with 28,098 vertices, it requires 14$\sim$17 seconds for high frequency filters in Fig. 49 and a few minutes for low-frequency filtering as is shown in Fig. 50. Note that low frequency filtering is much slower than high frequency one. We provide timing results for

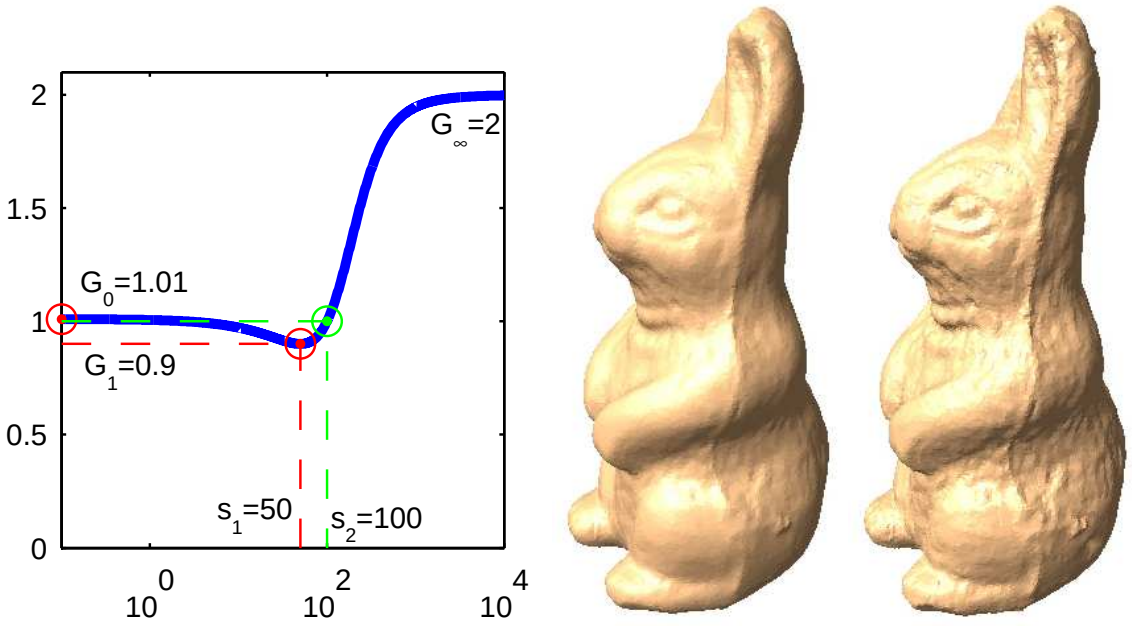

Figure 45: High frequency amplification filter applied to a rabbit model(33,519 vertices), which took 26 seconds in 2.4GHz Pentium4.

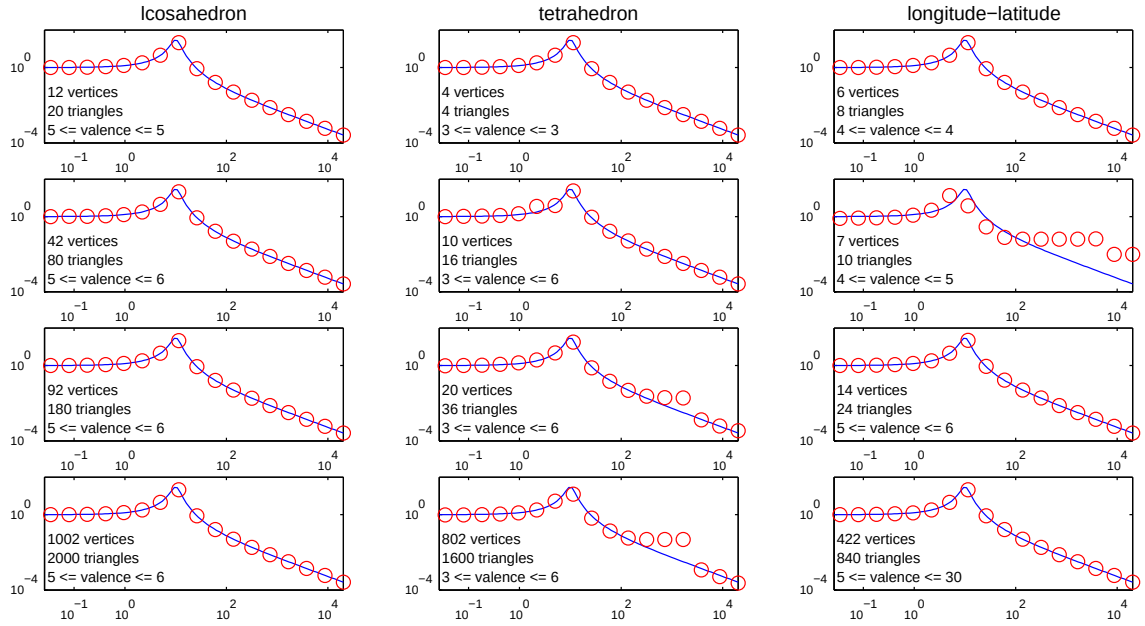

Figure 46: Exaggeration filter($s_1 = 10$, $s_2 = 25$, $G_0 = 1.0$, $G_1 = 30$, $G_\infty = 0$) applied to sphere meshes of various radii and connectivities. The solid blue line is $G(s)$ computed from (59), while red dots are samples of experimental gains r_O/r_I, where r_I, r_O are the average radii of input and output spheres, respectively. Notice that y-axes are in log scale.

all models measured in 2.4GHz Pentium4 PC with 512MB of main memory. This computation time could be significantly improved by using complex valued conjugate gradient solver [78] or the multigrid solver [48].

5.3.2 Filters Decomposable to First Order Ones

We explore the idea of constructing a filter that can be factored into real polynomials of degree one, since such an approach allows the left hand side of (61) to be factored into products of $\mathbf{M}^{-1}(\mathbf{K}+p_i\mathbf{M})$, which can be solved much faster.

$$G(s) = \left(G_0 \frac{p_1 p_2 p_3 \ldots}{z_1 z_2 z_3 \ldots}\right) \frac{(s+z_1)(s+z_2)(s+z_3)\ldots}{(s+p_1)(s+p_2)(s+p_3)\ldots} \tag{68}$$

where p_i and z_i are real numbers. A classical control theory [63], pages 213–226, provides an easy guideline in choosing z_i and p_i using the asymptotic lines that turns 20db at z_i and -20db at p_i as is illustrated in Fig. 48. As is shown in Fig. 47, it can be set to approximate the exaggeration filter too. However, it is difficult to make the exaggeration band narrow. In low pass filtering, it is hard to obtain a sharp cutoff rate while maintaining flat pass band. Thus, filters decomposable to first order ones can be intuitively designed by asymptotic lines and faster than the second order filters but their filtering capability is not sufficiently powerful.

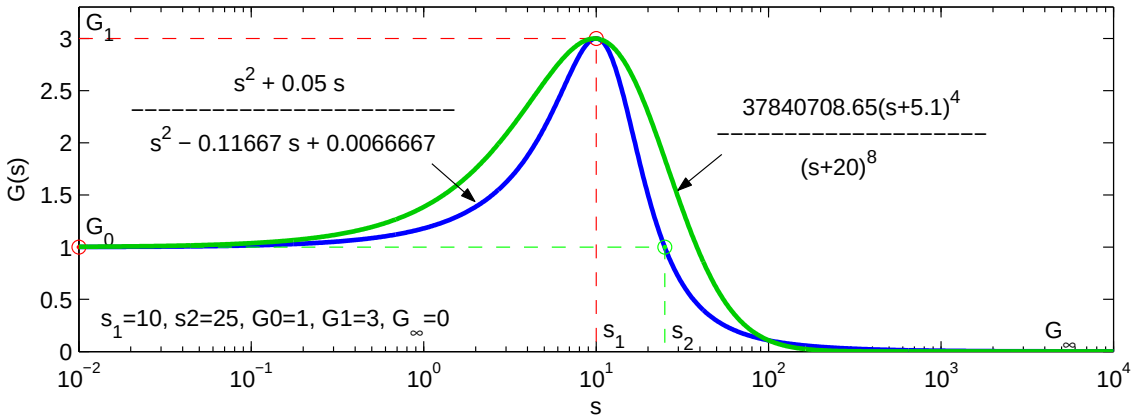

Figure 47: Comparison of exaggeration filter(blue) to a filter designed by asymptotic lines(green).

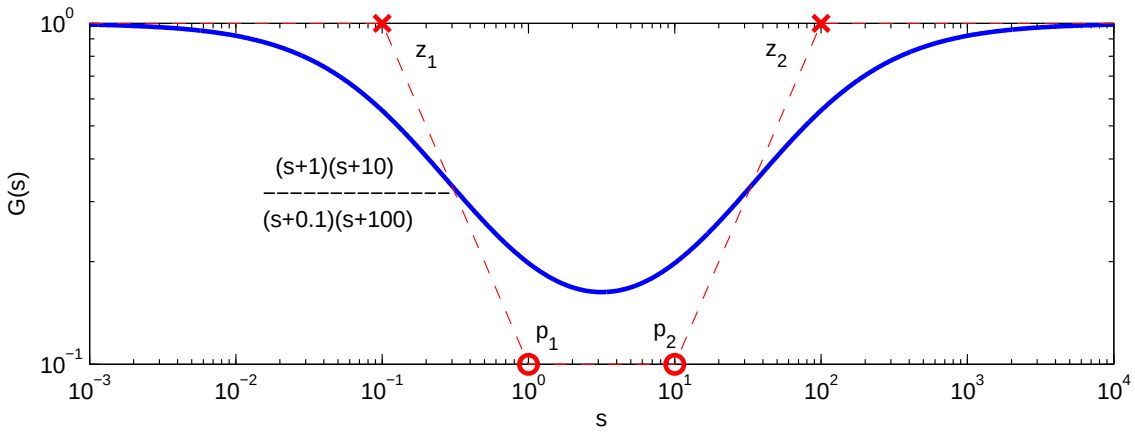

Figure 48: Designing a band stop filter by choosing four frequencies. $G(s)$ tends to turn 20db(-20db) at each $z_i(p_i)$.

5.4 Tests of Filtering Framework

We now validate the assumption made in section 1.5 under the proposed filtering framework. Since the curvature of a sphere is unique, we can predict the shrinkage ratio theoretically. We apply the filter to sphere models of various radii and then measure their shrinkage/expansion ratios and compare them to the theoretically predicted ones.

Consider a sphere equation $\mathbf{p} \cdot \mathbf{p} = r^2$. Since $-\mathbf{L}$ computes the normal vector whose length is twice the mean curvature, $-\mathbf{L}$ computes

$$-\mathbf{L}\mathbf{p} = \frac{2}{r}\frac{\mathbf{p}}{|\mathbf{p}|} = \frac{2}{r^2}\mathbf{p} \tag{69}$$

Thus, the eigenfunction is the sphere itself and the eigenvalue is $2/r^2$, which implies that if we apply the filter to spheres of radius r, it should shrink or expand by the ratio of $G\left(2/r^2\right)$.

We apply the filters for a sphere mesh obtained from three different tessellation methods: longitude-latitude model and subdividing tetrahedron and icosahedron. As is shown in Fig. 46, the filter output and the theoretical gain matches well in the mesh obtained from icosahedron since the triangulation is near regular. In other models, the mesh includes vertices whose valence is significantly different from six. This leads to some imprecise results for certain frequencies. However, in most frequency, the expected frequency and filter throughput match well.

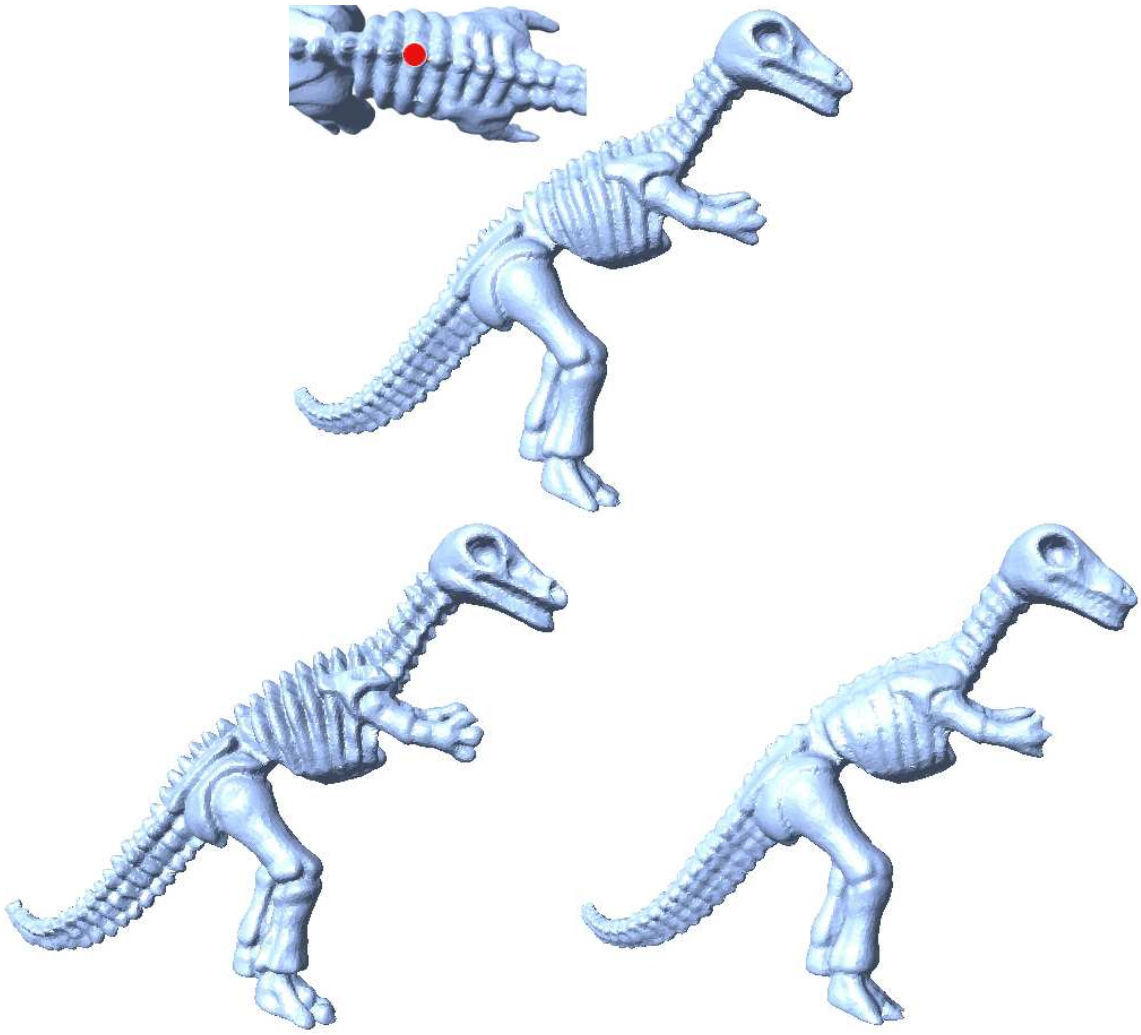

Figure 49: Various filtering results for a dinosaur model. The top image is the initial model with 28,098 vertices and 56,192 triangles. The bottom two are produced by a band exaggeration and a band stop filter for $s_1 = 630$ (frequency of back-bone marked in red circle), $s_2 = 1260, G_0 = 1$ and $G_1 = 3, G_\infty = 0.9$ (bottom left), and $G_1 = 0.01, G_\infty = 1.1(bottomright)$. The computation times are 17 and 14 seconds, respectively.

5.5 Selecting Feature and Computing Filter Frequency

In the proposed GeoFilter framework, the filter frequencies are chosen from the physical size of a user selected mesh feature. For example, a sphere like feature can be considered to have frequency of approximately $2/r^2$ and a cylinder like feature has $1/r^2$. Thus, if the user knows the size of the feature, the filter frequency can be easily computed.

To facilitate this process, we first allow the user to select a portion of mesh by picking a triangle graphically and then by automatically expanding the selection to neighboring triangles as long as the user keep pressing the mouse button. The operation may be repeated to extend the selection areas in a less regular fashion. The feature size is computed automatically by fitting a bounding box around optimally aligned principal axes computed as the eigenvectors of the covariance matrix [26]. The bounding box is shown as a transparent ellipsoid in Fig. 1. The dimensions of the bounding box are used to derive the desired filter frequencies.

In Fig. 49, the bump of the back bone has the radius of 0.0563, which corresponds to the frequency of $2/0.0563^2 \approx 630$. This frequency is exaggerated in the center image and reduced in the right image. We apply similar a strategy in various examples. In Fig 51, we pick the horse shoe that gives a frequency of approximately 60. In Fig. 52, we pick the nose that has frequency of approximately 25.

In conclusion, we have proposed a new approach to mesh filtering. The examples demonstrate that the filter frequency can be computed from the sizes of graphical features rather than via a time consuming trial-and-error process.

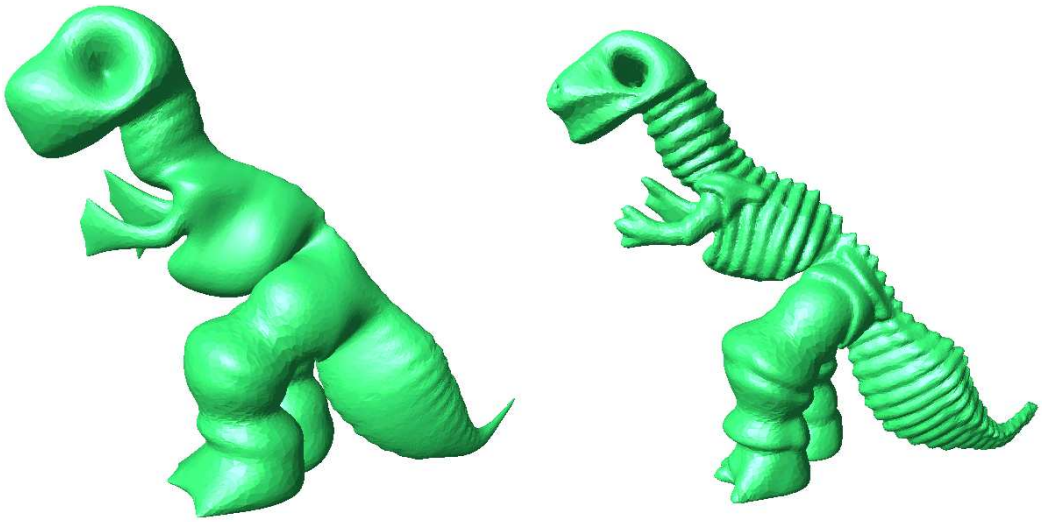

Figure 50: A band exaggeration filter applied to the dinosaur model(far left in Fig. 49) with and without high frequency attenuation. $s_1 = 25$ is chosen as the frequency of the leg: $s_2 = 60, G_0 = 1, G_1 = 3$ $G_\infty = 0.0$ (left), and $G_\infty = 0.9(right)$. The computation times are 142 and 337 seconds, respectively

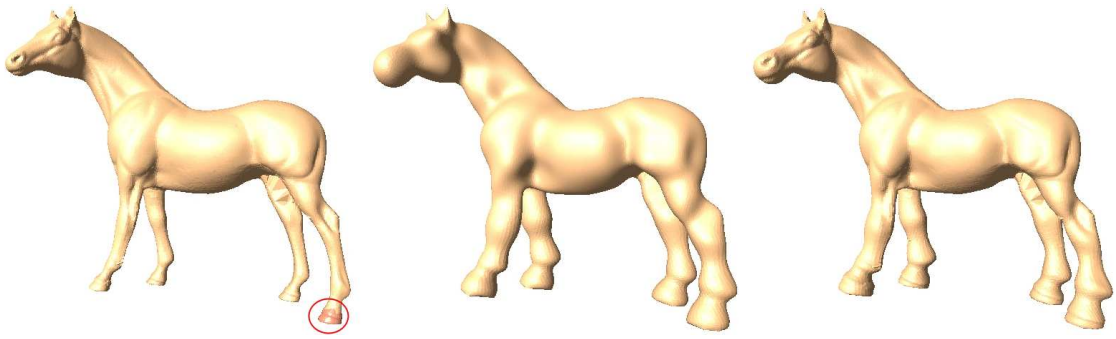

Figure 51: Exaggeration of the legs of a horse model with 48,485 vertices. Far left image is original model. The next two are exaggeration results ($G_0 = 1, G_1 = 2, s_1 = 60, s_2 = 120$). The seconds and third are respectively with($G_\infty = 0$, 89 sec.) and without($G_\infty = 0.9, 182sec.$) eliminating high frequency.

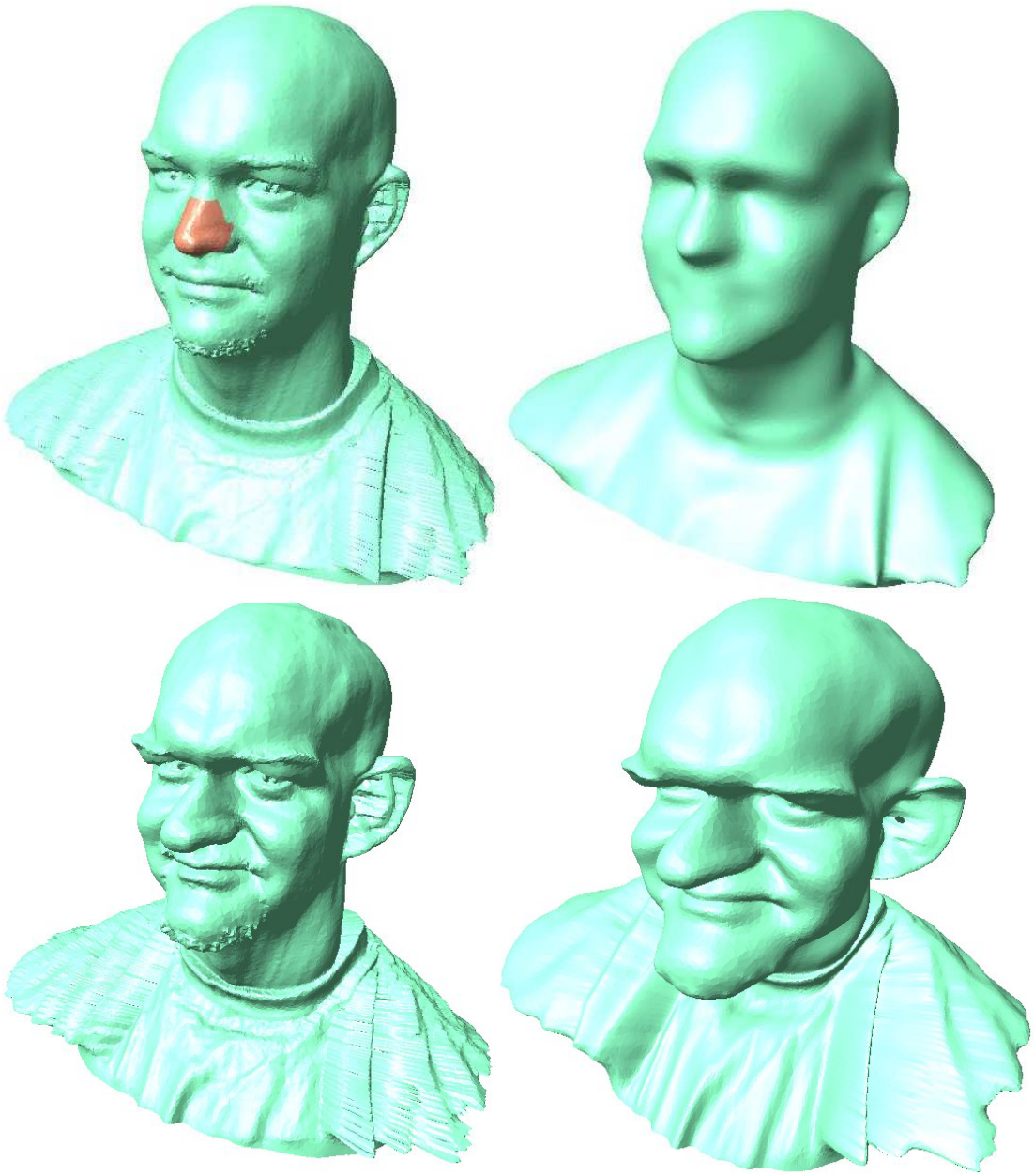

Figure 52: Various filtering results for a human model (75,948 vertices, 151,474 triangles). Top left is the original model. The top right image shows a lowpass filtered model ($s_1 = 25, s_2 = 50, G_0 = 1, G_1 = 1.1, G_\infty = 0$, 366 sec.). The bottom two are exaggerated models that look older ($s_1 = 200, s_2 = 1000, G_0 = 0.5, G_1 = 1.1, G_\infty = 0$,70 sec.) or like a cartoon character ($s_1 = 25, s_2 = 400, G_0 = 0.5, G_1 = 1.8, G_\infty = 0$, 481 sec.). Since the bottom two models have $G_0 = 0.5$, the filtered models are about half the size of the initial model. In this figure, we zoomed them in.

CHAPTER VI

CONCLUSION

Most applications of computer graphics deal with static or animated surfaces that approximate the boundaries of real or simulated objects and their behavior with time. A surface may be represented explicitly, for example by an approximating triangle mesh, or implicitly, for example as the iso-surface of a scalar field sampled along a regular lattice. We have used two representations of such surfaces: (1) a triangle mesh, (2) an iso-surface implicitly defined by the level set.

6.1 Fluid Simulation

Recently, with the growth of the entertainment industries such as movie and games, computer graphics applications have been expanding. Those industries demand accurate simulation methods for the synthesis of realistic behaviors in virtual scenes. These simulations typically involve objects moving according to Newtonian physics. Therefore, accurate simulation of Newton physics is an important tool to create a world that contains objects moving realistically. Rigid body motion, and even articulated structure kinematics and dynamics, have been addressed extensively in the field.

A challenging problem is fluid animation, especially for a fluid with a high degree of freedom. Because it would be tedious, or even impossible, to design fluid animation by hand from ground up, there is a strong demand for accurate simulation tools for synthesizing animations of fluids interacting with solids.

Unfortunately, fluid simulation is a complex process involving many steps and subproblems. Naive simulations suffer from shortcomings that make the results clearly unrealistic and hence unacceptable for commercial entertainment or scientific use. This thesis addresses three problems in fluid simulation: (1) Diffusion and dissipation in advection step,

(2) Breaking of thin films, and (3) Volume loss.

6.1.1 Reduced Diffusion/Dissipation

A fundamental step in fluid simulation is the advection step. A stable implementation of that step was proposed in [65] using the CIR (Courant-Isaacson-Rees) method. Unfortunately, the CIR advection methods produces significant amount of diffusion and dissipation. This diffusion makes the transported quantity spread to neighbors (resulting in blurred images or in the loss of shape details) and the dissipation makes the transported quantity disappear. This dissipation/diffusion in the advection step is a problem when simulating non-diffusive/non-dissipative fluids. We have addressed this dissipation problem by harnessing the BFECC method and have shown that the BFECC method can reduce dissipation and diffusion. Although other approaches exist for reducing diffusion and dissipation, the BFECC method has a strong advantage: it only requires the implementation of a simple linear CIR. Therefore, it is very easy to implement and has low computation cost. We applied BFECC to a number of problems. We have demonstrated its effectiveness on several simulation problems. First, we applied it to the velocity advection step, and have shown that the resulting velocity has richer flow details.

With this solution, at the boundary of the fluid in a two phase flow simulation (for example, the interaction of water and air), we observed that the velocity at the interface keeps increasing. We avoided this problem by disabling BFECC in a thin band near the interface. This way, flow details are preserved except at the interface. A future direction would be applying BFECC or its modification on this thin band, which can keep the flow detail at the interface, and therefore, can increase the surface details.

We also applied BFECC to the image advection problem, which will provide a useful tool in the movie industry when a detailed moving texture on a fluid surface is necessary [2, 42]. When BFECC is applied, the texture detail is preserved more accurately than when linear CIR was used.

The next application of BFECC that we have explored was the smoke density advection problem. When BFECC was applied, the smoke advected with only a small amount of diffusion and dissipation, allowing the simulation of non-diffusive smoke such as cigarette smoke.

The next application was the level set advection. The linear CIR level set advection yields fast volume loss. When the BFECC method was used, the volume loss was significantly reduced.

However, this solution still suffered from a blatant volume loss. For example, bubbles of air in water would shrink and eventually disappear. We solved this problem by using the volume control method discussed in Chapter 4.

We have also tested BFECC on irregular grids. In particular, we tested in on triangle meshes. To do so, we defined the level set value on each vertex of the mesh. We have developed a gradient operator. Then, we have used it to implement level set advection on triangle meshes.

To test this advection on a triangle mesh, we have designed a challenge similar to the Zalesak's problem, and have compared BFECC with the CIR advection on meshes. We have shown that BFECC advects the Zalesak disk with small deteriorations.

Finally, we here applied BFECC to a smoke advection problem on a quadtree mesh and showed that BFECC can be applied to an adaptive grid. Using BFECC, we observed that smoke was advected without diffused to neighbors, creating an animation of thin smoke.

6.1.2 Simulation of Thin Liquid Films

Fluid animations often contain splashing water or bubbles. In such animations, a thin liquid film is formed during splash or between bubbles. The thin film is difficult to simulate using fixed Eulerian grid level set methods, since inadequate sampling will result in a rupture of the film.

The solution proposed in [79] is robust and and can accurately simulate thin film that

separates two different fluid region. Independently, we have developed a method that can be used for any thin liquid film. Unfortunately, our approach cannot simulate arbitrarily thin films. Hence, we have decided to follow the approach proposed in [79] and improved upon it.

Our efforts to prevent the rupture of thin films were inspired by the disjoining pressure, which is a resultant of various molecular interactions. This disjoining pressure prevents two very close liquid and gas interfaces from approaching further. We have adopted this idea and implemented a step that simulates the disjoining pressure. We identified the thin film cells, and then, we applied the disjoining force that is designed to make the thin film interfaces (membranes separating the water film from two adjacent air bubbles) repulse each other. The force is a function of the film thickness. We compute this force for each thin film cell and then apply it force to neighboring cells so that the force and moment resultants are zero. We plan to verify this idea with more experiments in future.

However, we observed that an additional problem occurred in the simulation of the thin film. Since the film is thin, the level set is singular near the interface. Therefore, the level set gradient is noisy and unreliable. Because the surface tension uses this level set gradient, the surface tension force cannot be computed reliably from the level set values. This unreliability was causing a premature rupture of thin films.

To address the problem, we proposed to compute curvature from the local iso-surface shape rather than directly from the level set gradient. When the curvature is computed using this method, we obtain resilient bubbles even when multiple bubbles are stacked in 2D. We will continue the similar idea on 3D cases.

We anticipate that much of future work in this area will focus on improving the methods to simulate the thin film in bubbles, bubble junctions, and splash.

To decrease computation time and memory usage, we used an octree grid where all the simulation variables are collocated. This way, the implementation complexity and memory cost were lowered. We have applied a multi-grid method and have shown that it make the

pressure projection step 2∼10 times faster.

6.1.3 Volume Control Using Divergence

Our third problem was the volume loss problem in the level set method. As shown in Chapter 2, the volume loss in the CIR level set advection can be reduced significantly by using the BFECC method. However, a small but still visible amount of the volume loss occurred. Therefore, we have developed a volume control method that compensates volume loss using the divergence. First, the volume change equation was derived, and then, a nonlinear feedback controller was designed. Finally, the gain was computed so that the volume error is corrected in a predefined time. The developed volume change equation and controller was verified by several experiments.

The drift error in the proportional controller encouraged us to use an additional integral feedback. Since this additional feedback makes the resulting equation second order, the integral gain was chosen so that the oscillation does not occur. Using this integral feedback, the drift error was corrected and it was shown that the volume loss factor can be estimated by the integral of the volume error.

The volume control technique presented in Chapter 4 leaves many opportunities for future research. First, we believe that our volume control technique may be applied to different level set advection schemes in different simulation settings. We believe that the staggered grid approach will require a smaller divergence and that a level set method that incorporates particles [17] will require even smaller divergence to correct the volume error by our control method. We can also test the volume control for the liquid drops. Because the bubble's volume loss was maximized when the thin film was formed and because liquid drops do not have thin films, we believe that the liquid drop will also need smaller divergence to correct its volume error. Therefore, we believe that the volume control method will work as well with a staggered grid, particle level set, and for liquid drops.

If the region has only a few grid cells, our volume computation will be very noisy since

we only count the number of cells. To this reason, we only applied the volume control to the regions that have more than 50 grid cells. Therefore, computing volume more precisely and then tracking it more precisely will be necessary in order to apply volume control to small regions with small number of grid cells.

The proposed volume control can be applied to most fluid simulations that are based on an Eulerian grid. Since the volume error is no longer accumulated, fluid simulations can now run for a long time without the bubbles losing volume or having the water level decreasing.

In summary, we have developed a suite of tools for evolving surfaces in animations. We believe that these tools will help advance the field of animation towards simpler and more accurate solutions.

6.2 Mesh Filter

Triangle mesh representations are popular due to their simplicity and to the availability of hardware graphic accelerators for rendering them. They may be produced with Computer-Aided Design or Animation systems or through simulation. They may also be extracted from volumetric models through iso-surface extraction. These meshes may include a significant amount of geometric details, which may have to be eliminated to simplify analysis and shape matching or to accelerate transmission or further processing. They may also have to be exaggerated to attract the viewers attention or for artistic effects.

We have designed a new mesh filter that may be used to simplify or exaggerate features of a user-selected magnitude. Our filter improves upon prior art, which used either an explicit or an implicit formulation, by unifying them into a rational form that offers increased selectivity and flexibility. Our rational mesh filters may be viewed as a generalization of rational linear filters (such as butterworth, elliptic, or Chebychev filters) and their adaptation to triangle meshes.

Compared to the implicit or explicit form, this generalized filter form supports broader

set of filters. For example, explicit filters tends to have diverging amplification factor in high frequencies. In contrast, implicit filters have amplification factor decaying at high frequencies, but they suffer from the lack of flat pass band. Our rational form takes advantage of both, not only allowing decaying or diverging transfer function, but also allowing wider variety of filter shapes (such as a band-pass filter).

To perform the filtering on the whole mesh, we need to choose a Laplace-Beltrami operator so that we can reshape its spectrum by applying our filter. Among the various discrete formulations that approximate the Laplacian operators, the most popular one is the operator that uses cotangent weights. This operator approximates the Laplace-Beltrami operator of the continuous surface when the mesh is close to regular. The cotangent operator does not converge to the continuous operator as the mesh is refined. We verified that the cotangent operator is suitable to our needs by experimenting on a sphere, for which the eigen values of the continuous Laplace-Beltrami operator are known and for which we can compute the exact shrinkage or expansion amount.

A filter is only useful if the parameters (the filter coefficients) are set properly. It is impractical to expect the user of a CAD or animation system to tinker directly with such parameters, especially given the fact that for most filters, several such parameters must be adjusted. To address this problem, we have proposed to assist the user with a graphic interface for selecting a feature and an automatic measurement system, which computes feature size measures and helps the designer to translate them into filter parameter values.

The selected feature will not have one frequency but a set of frequencies, called the spectrum. We have decided not to analyze and filter this spectrum for two reason: (1) The computation of the spectrum would be time consuming and (2) the spectrum typically contains frequencies that should not be eliminated. In addition, mesh filter transfer function proposed here cannot have sharp cut-offs if one wants to avoid ringing problem. Therefore, we have focused on automating the choice of one dominant frequency for the selected feature. In finding one dominant frequency, we proposed to measure the size of the feature

and then use the frequencies of ellipsoid, sphere, or cylinder fit to the selected feature (depending the feature shape).

The graphic interface lets the user select a particular feature by painting directly on the mesh. Our approach is focused on the the filter: i.e., the users intention. Typically, a user may want to eliminate or enhance (exaggerate) features of a specific size, while not affecting larger or smaller features. For example, the user may want to remove the noise smaller than some size, or to exaggerate certain feature of mesh such as the ears of a bunny model without affecting the bumpy nature of the bunny's furry surface. Hence, we decided to compute the mesh filter parameters from the physical size of a user-selected feature (such as the ear).

We have validated the proposed approach on a small collection of shapes and features and concluded that it considerably simplifies the task of selecting and enhancing features of a given size in models of animals or human faces.

The limitations of the proposed approach stems from the use of a linear approximation of the local shape properties. The approximation is valid when one wants to filter signals defined on a mesh that is not moving. However, since the mesh is deforming in mesh filtering applications, the discrete operator is changing. Thus, mesh filtering is inherently a nonlinear problem. By ignoring this nonlinearity, our mesh filter is limited to relatively small deformations.

One may consider extending our work to higher order approximations. Unfortunately, there are little theoretical guidelines in nonlinear filter and the study of nonlinear filters is arduous. Nevertheless, we see important opportunities for future works.

First, there is a need for a more precise discrete operator since for irregular meshed, the cotangent operator will produce incorrect curvature. One may develop such an operator by taking the Taylor series approximation and then computing weights that zeros low-order terms.

REFERENCES

[1] ALMGREN, A. S., BELL, J. B., and SZYMCZAK, W. G., "A numerical method for the incompressible navier-stokes equations based on an approximate projection," *SIAM Journal of Scientific Computing*, vol. 17, March 1996.

[2] BARGTEIL, A. W., SIN, F., MICHAELS, J. E., GOKTEKIN, T. G., and O'BRIEN, J. F., "A texture synthesis method for liquid animations," in *Proceedings of ACM SIGGRAPH 2006 Symposium on Computer Animation*, pp. 345–351, 2006.

[3] BAZHLEKOV, I. B., VAN DE VOSSE, F. N., and MEIJER, H. E., "Boundary integral method for 3d simulation of foam dynamics," in *Lecture Notes in Computer Science*, pp. 401–408, 2001.

[4] BRACKBILL, J. U., KOTHE, D. B., and ZEMACH, C., "A continuum method for modeling surface tension," *Journal of Computational Physics*, vol. 100, no. 2, pp. 335–354, 1992.

[5] BRIGGS, W. L., HENSON, V. E., and MCCORMICK, S. F., *A Multigrid Tutorial*. Siam, 2000.

[6] BUSHER, P., *Geometry and Spectra of Compact Riemann Surfaces*. Birkhauser Boston, 1992.

[7] CARLSON, M., MUCHA, P. J., and TURK, G., "Rigid fluid: Animating the interplay between rigid bodies and fluid," in *SIGGRAPH*, ACM, 2004.

[8] CHORIN, A. J., "Numerical solution of the navier-stokes equations," *Mathematics of Computation*, vol. 22, no. 104, pp. 745–762, 1968.

[9] CHORIN, A. J., "On the convergence of discrete approximations to the navier-stokes equations," *Mathematics of Computation*, vol. 23, no. 106, pp. 341–353, 1969.

[10] CUNNINGHAM, E. P., *Digital Filtering: An Introduction*. Houghton Mifflin, 1992.

[11] DESBRUN, M., MEYER, M., SCHRÖDER, P., and BARR, A. H., "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proceedings of ACM SIGGRAPH*, pp. 317–324, 1999.

[12] DO CARMO, M. P., *Riemannian Geometry*. Birkhauser, 1994.

[13] DUPONT, T. F. and LIU, Y., "Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function," *Journal of Computational Physics*, vol. 190, no. 1, pp. 311–324, 2003.

[14] DUPONT, T. F. and LIU, Y., "Back and forth error compensation and correction methods for semi-lagrangian schemes with application to interface computation using level set method," *CDSNS2004-399, School of Mathematics, Georgia Institute of Technology*, 2004.

[15] DUPONT, T. F. and LIU, Y., "Back and forth error compensation and correction methods for semi-lagrangian schemes with application to level set interface computations," *Math. Comp., To appear*, 2006.

[16] DURIAN, D., "Bubble-scale model of foam mechanics: Melting, nonlinear behavior, and avalanches," *Physical Review*, vol. 55, no. 2, pp. 1739–1751, 1997.

[17] ENRIGHT, D., LOSASSO, F., and FEDKIW, R., "A fast and accurate semi-lagrangian particle level set method," *Computers and Structures*, vol. 83, pp. 479–490, 2005.

[18] ENRIGHT, D., MARSCHNER, S., and FEDKIW, R., "Animation and rendering of complex water surfaces," in *SIGGRAPH*, ACM, 2002.

[19] EXEROWA, D. and KRUGLYAKOV, P. M., *Foams and Foam Films*. Elsevier, 1998.

[20] FAN, Z., ZHAO, Y., KAUFMAN, A., and HE, Y., "Adapted unstructured lbm for flow simulation on curved surfaces," in *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005.

[21] FEDKIW, R., STAM, J., and JENSEN, H., "Visual simulation of smoke," in *SIGGRAPH*, pp. 23–30, ACM, 2001.

[22] FELDMAN, B. E., O'BRIEN, J. F., and ARIKAN, O., "Animating suspended particle explosions," in *ACM SIGGRAPH*, pp. 708–?715, 2003.

[23] FOSTER, N. and FEDKIW, R., "Practical animation of liquids," in *SIGGRAPH*, pp. 15–22, ACM, 2001.

[24] FOSTER, N. and METAXAS, D., "Realistic animation of liquids," *Graphical Models and Image Processing*, vol. 58, no. 5, pp. 471–483, 1996.

[25] GOKTEKIN, T. G., BARGTEIL, A. W., and O'BRIEN, J. F., "A method for animating viscoelastic fluids," in *SIGGRAPH*, pp. 463–468, ACM, 2004.

[26] GOTTSCHALK, S., LIN, M. C., and MANOCHA, D., "Obb-tree: A hierarchical structure for rapid interference detection," in *Proceedings of ACM Siggraph*, 1996.

[27] GUENDELMAN, E., SELLE, A., LOSASSO, F., and FEDKIW, R., "Coupling water and smoke to thin deformable and rigid shells," in *SIGGRAPH*, ACM, 2005.

[28] GUSKOV, I., SWELDENS, W., and SCHRÖDER, P., "Multiresolution signal processing for meshes," in *SIGGRAPH*, 1999.

[29] HAARIO, H., KOROTKAYA, Z., LUUKKA, P., and SMOLIANSKI, A., "Computational modelling of complex phenomena in bubble dynamics: Vortex shedding and bubble swarms," in *Proceedings of ECCOMAS 2004*, 2004.

[30] HONG, J.-M. and KIM, C.-H., "Animation of bubbles in liquid," in *EUROGRAPHICS*, vol. 22, 2003.

[31] HONG, J.-M. and KIM, C.-H., "Discontinuous fluids," in *SIGGRAPH*, ACM, 2005.

[32] JIANG, G. and SHU, C.-W., "Efficient implementation of weighted ENO schemes," *Journal of Computational Physics*, vol. 126, pp. 202–228, 1996.

[33] JOBARD, B., ERLEBACHER, G., and HUSSAINI, M. Y., "Lagrangian-eulerian advection of noise and dye textures for unsteady flow visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, 2002.

[34] JR., J. D. A., *Modern Compressible Flow With Historical Perspective*. Mc Graw Hill, 1990.

[35] KANG, M., FEDKIW, R. P., and LIU, X.-D., "A boundary condition capturing method for multiphase incompressible flow," *Journal of Scientific Computing*, vol. 15, Sep 2000.

[36] KARNI, Z. and GOTSMAN, C., "Spectral compression of mesh geometry," in *Proceedings of ACM SIGGRAPH*, pp. 279–286, 2000.

[37] KIM, B., LIU, Y., LLAMAS, I., and ROSSIGNAC, J., "Flowfixer: Using bfecc for fluid simulation," in *Eurographics Workshop on Natural Phenomena*, 2005.

[38] KIM, B., LIU, Y., LLAMAS, I., and ROSSIGNAC, J., "Advections with significantly reduced dissipation and diffusion," *IEEE Transactions on Visualization and Computer Graphics*, 2007.

[39] KIM, B. and ROSSIGNAC, J., "Geofilter: Geometric selection of mesh filter parameters," in *EUROGRAPHICS*, 2005.

[40] KOBBELT, L., CAMPAGNA, S., VORSATZ, J., and SEIDEL, H.-P., "Interactive multi-resolution modeling on arbitrary meshes," in *Proceedings of ACM SIGGRAPH*, pp. 105–114, 1998.

[41] KÜCK, H. VOGELGSANG, C., and GREINER, G., "Simulation and rendering of liquid foams," in *Proceedings of Graphics Interface*, pp. 81–88, 2002.

[42] KWATRA, V., ADALSTEINSSON, D., KWATRA, N., CARLSON, M., and LIN, M., "Texturing fluids," in *Technical Sketches Program, ACM SIGGRAPH*, 2006.

[43] LIU, X. D., OSHER, S., and CHAN, T., "Weighted essentially non-oscillatory schemes," *Journal of Computational Physics*, vol. 115, no. 1, pp. 200–212, 1994.

[44] LOSASSO, F., GIBOU, F., and FEDKIW, R., "Simulating water and smoke with an octree data structure," in *SIGGRAPH*, pp. 457–462, ACM, 2004.

[45] LOSASSO, F., IRVING, G., GUENDELMAN, E., and FEDKIW, R., "Melting and burning solids into liquids and gases," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, pp. 343–353, 2006.

[46] LOSASSO, F., SHINAR, T., SELLE, A., and FEDKIW, R., "Multiple interacting liquids," in *ACM SIGGRAPH*, pp. 812–819, 2006.

[47] MUSETH, K., BREEN, D., WHITAKER, R., and BARR, A., "Level set surface editing operators," in *ACM SIGGRAPH*, pp. 330–338, 2002.

[48] NI, X., GARLAND, M., and HART, J. C., "Fair morse functions for extracting the topological structure of a surface mesh," in *Proceedings of ACM SIGGRAPH*, pp. 613–622, 2004.

[49] OEVERMANN, M., KLEIN, R., BERGER, M., and GOODMAN, J., "A projection method for two-phase incompressible flow with surface tension and sharp interface resolution," Tech. Rep. ZIB-Report 00-17, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000.

[50] OHTAKE, Y., BELYAEV, A. G., and BOGAEVSKI, I. A., "Polyhedral surface smoothing with simultaneous mesh regularization," in *Proceedings of the Geometric Modeling and Processing*, pp. 229–237, 2000.

[51] OSHER, S. and SETHIAN, J. A., "Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations," *Journal of Computational Physics*, vol. 79, pp. 12–49, 1988.

[52] OSHER, S. and FEDKIW, R., *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2003.

[53] OSHER, S. J. and FEDKIW, R. P., *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[54] PARK, S. I. and KIM, M. J., "Vortex fluid for gaseous phenomena," in *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005.

[55] PINKALL, U. and POLTHIER, K., "Computing discrete minimal surfaces and their conjugates," *Experimental Mathematics*, vol. 2, no. 1, pp. 15–36, 1993.

[56] POHL, T., DESERNO, F., THUREY, N., RUDE, U., LAMMERS, P., WELLEIN, G., and ZEISER, T., "Performance evaluation of parallel large-scale lattice boltzmann applications on three supercomputing architectures," in *Supercomputing, 2004. Proceedings of the ACM/IEEE SC2004 Conference*, 2004.

[57] PRUD'HOME, R. K. and KHAN, S. A., *Foams: Theory, Measurements, and Applications*. Marcel Dekker, Inc, 1996.

[58] SCHNEIDER, R. and KOBBELT, L., "Geometric fairing of irregular meshes for free-form surface design," *Computer Aided Geometric Design*, vol. 18, no. 4, pp. 359–379, 2001.

[59] SELLE, A., RASMUSSEN, N., and FEDKIW, R., "A vortex particle method for smoke, water and explosions," in *SIGGRAPH*, ACM, 2005.

[60] SETHIAN, J. A., *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[61] SHI, L. and YU, Y., "Inviscid and incompressible fluid simulation on triangle meshes," *Computer Animation and Virtual Worlds*, vol. 15, no. 3-4, pp. 173–181, 2004.

[62] SHI, L. and YU, Y., "Visual smoke simulation with adaptive octree refinement," in *Computer Graphics and Imaging*, 2004.

[63] SHINNERS, S. M., *Modern Control System Theory and Application*. Addison Wesley, 1978.

[64] SONG, O., SHIN, H., and KO, H., "Stable but nondissipative water," *ACM Transactions on Graphics*, vol. 24, no. 1, pp. 81–97, 2005.

[65] STAM, J., "Stable fluids," in *SIGGRAPH*, pp. 121–128, ACM, 1999.

[66] STAM, J., "Flows on surfaces of arbitrary topology," in *SIGGRAPH*, pp. 724–731, ACM, 2003.

[67] STUBENRAUCH, C. and VON KLITZING, R., "Disjoining pressure in thin liquid foam and emulsion films - new concepts and perspective," *Journal of Physics Condensed Matter*, vol. 15, no. 3-4, pp. 173–181, 2004.

[68] SUSSMAN, M., ALMGREN, A., BELL, J., COLELLA, P., HOWELL, L., and WELCOME, M., "An adaptive level set approach for incompressible two-phase flow," *Journal of Computational Physics*, vol. 148, pp. 81–124, 1999.

[69] SUSSMAN, M., SMEREKA, P., and OSHER, S., "A levelset approach for computing solutions to incompressible two-phase flow," *Journal of Computational Physics*, vol. 114, no. 1, pp. 146–159, 1994.

[70] TAKAHASHI, T., FUJII, H., KUNIMATSU, A., HIWADA, K., SAITO, T., TANAKA, K., and UEKI, H., "Realistic animation of fluid with splash and foam," in *EUROGRAPHICS*, vol. 22, 2003.

[71] TAUBIN, G., ZHANG, T., and GOLUB, G., "Optimal surface smoothing as filter design," in *Fourth European Conference on Computer Vision (ECCV'96) and IBM Research Technical Report RC-20404*, March 1996.

[72] TAUBIN, G., "Signal processing approach to fair surface design," in *Proceedings of ACM SIGGRAPH*, pp. 351–358, 1995.

[73] THUREY, N. and RUDE, U., "Free surface lattice-boltzmann fluid simulations with and without level sets," in *Workshop on Vision, Modeling, and Visualization*, 2004.

[74] WEAIRE, D. and HUTZLER, S., *The physics of foams*. Oxford, 1999.

[75] WEISKOPF, D., "Dye advection without the blur: A level-set approach for texture-based visualization of unsteady flow," in *EUROGRAPHICS*, vol. 23, 2004.

[76] XU, G., "The convergent discrete laplace-beltrami operator over triangular surfaces," in *Proceedings of Geometric Modelling and Processing (GMP2004)*, pp. 195–204, 2004.

[77] ZELINKA, S. and GARLAND, M., "Similarity-based surface modelling using geodesic fans," in *Proceedings of the 2nd Eurographics Symposium on Geometry Processing*, 2004.

[78] ZHANG, H. and FIUME, E., "Butterworth filtering and implicit fairing of irregular meshes," in *Proceedings of Pacific Graphics*, pp. 502–506, 2003.

[79] ZHENG, W., YONG, Y.-H., and PAUL, J.-C., "Simulation of bubbles," in *ACM Siggraph/Eurographics Symposium in Computer Animation*, 2006.

[80] ZHU, Y. and BRIDSON, R., "Animating sand as a fluid," in *SIGGRAPH*, ACM, 2005.

VITA

ByungMoon Kim received a bachelors degree in aerospace engineering in Inha University, Inchon, Korea in 1994. After graduation, he worked as a software programmer until 1998. At the same year, he joined the Georgia Institute of Technology, where he earned masters degrees in aerospace engineering in 1999, in computer science in 2004, and in mathematics in 2005, and is currently a doctoral candidate in computer science. His research interests are in computer graphics, focusing on fluid and solid simulation, geometry processing such as mesh filtering and editing, and haptic devices. He is an author of papers on various topics: mobile robot control, a spacecraft simulator, collision prediction, mesh editing, nonphotorealistic video processing, a mesh filter, and fluid simulation.