A Multiscale Approach to Mesh-based Surface Tension Flows

Nils Thürey, ETH Zurich
Chris Wojtan, Georgia Institute of Technology
Markus Gross, ETH Zurich
Greg Turk, Georgia Institute of Technology

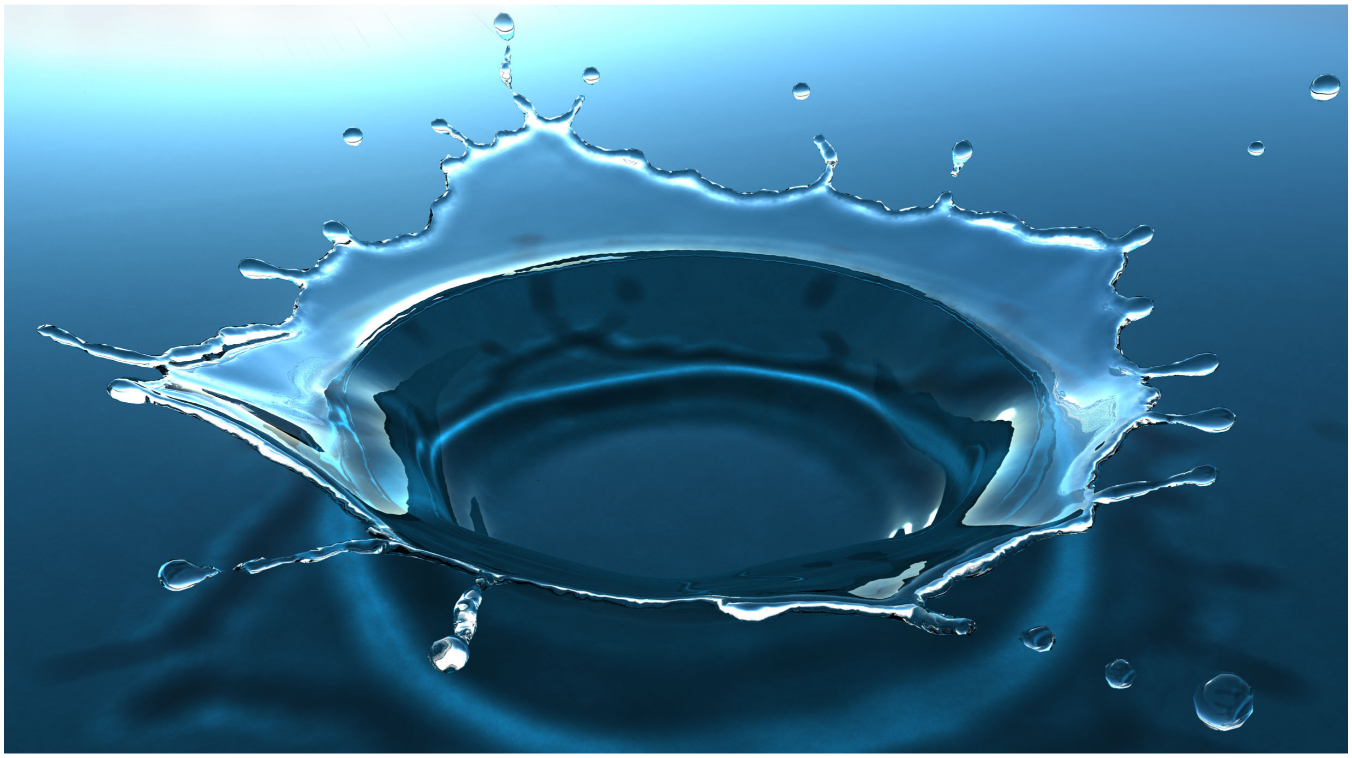

Figure 1: *Our method allows us to efficiently simulate complex surface tension phenomena such as this crown splash. The small scales are handled with our surface approach, while the larger scales are computed with the Eulerian simulation. For the shown simulation, our method requires only 22.3 seconds per frame on average.*

Abstract

We present an approach to simulate flows driven by surface tension based on triangle meshes. Our method consists of two simulation layers: the first layer is an Eulerian method for simulating surface tension forces that is free from typical strict time step constraints. The second simulation layer is a Lagrangian finite element method that simulates sub-grid scale wave details on the fluid surface. The surface wave simulation employs an unconditionally stable, symplectic time integration method that allows for a high propagation speed due to strong surface tension. Our approach can naturally separate the grid- and sub-grid scales based on a volume-preserving mean curvature flow. As our model for the sub-grid dynamics enforces a local conservation of mass, it leads to realistic pinch off and merging effects. In addition to this method for simulating dynamic surface tension effects, we also present an efficient non-oscillatory approximation for capturing damped surface tension behavior. These approaches allow us to efficiently simulate complex phenomena associated with strong surface tension, such as Rayleigh-Plateau instabilities and crown splashes, in a short amount of time.

Keywords: Physically Based Animation, Fluid Simulation, Surface Tension

1 Introduction

Surface tension forces are responsible for many phenomena that contribute essential details to liquids and interfaces in nature. The formation of a water droplet can be attributed to a growing surface tension instability, and small ripples on the surface of a liquid are primarily driven by surface tension. In the typical Eulerian solvers used for computer animation, surface tension effects are often neglected, and effects such as droplet pinch off occur purely as a result of insufficient computational resolution. Once tension effects are explicitly modeled with forces at the liquid interface, many cells are required to resolve the shape of a single droplet. In addition, surface tension forces impose a strict time step restriction on the solver. These requirements of a high resolution and small time steps make the simulation of large bodies of liquid infeasibly expensive.

We present an algorithm for efficiently and robustly simulating surface tension effects that is based on an a triangle mesh discretization of the fluid surface. As such, our method is decoupled from the resolution of the computational grid and allows for an efficient simulation of sub-grid scale surface tension effects, including droplet pinch off and surface tension waves. At the same time, it allows us to separate the small scales of detail that are best simulated on the surface from the larger scales that are suitable to be resolved within an Eulerian simulation. A key benefit of the surface mesh is that it allows us to use a robust finite element method to discretize our model for the surface tension dynamics. Furthermore, as surface tension phenomena are driven by the curvature of the surface, we can use the large body of work on discrete surface operators to accurately calculate curvature and curvature flows of our fluid surface. As our method enforces a local conservation of mass, it leads to realistically bulging surfaces and reproduces natural instabilities that result in pinch off behavior.

The key attributes of our approach to surface tension are as follows:

- A novel method for efficiently simulating sub-grid surface tension effects that computes wave dynamics on the discretized fluid surface.
- An algorithm that has a significantly relaxed time step restriction in comparison to previous approaches in graphics.
- The efficient and robust handling of pinch off, as well as merging effects with local volume preservation on sub-grid scales.
- A simple and efficient non-oscillatory approximation to sub-grid surface tension using mesh-based volume-preserving mean curvature flow

These contributions combine to produce highly detailed animations of surface tension phenomena with subtle secondary effects at a fraction of the cost of previous methods.

We will now briefly outline the different steps of our simulation algorithm, which are also illustrated in Figure 2. For simulating the fluid, we use a standard solver in combination with a triangle-mesh embedded in the Eulerian grid to represent the liquid surface, as described in [Wojtan et al. 2009]. The input to our algorithm is a triangle mesh F representing the liquid interface. First, we advect F through the velocity field given by the previous step in the fluid simulation. Second, we compute a simplified and smoothed version S of this surface for each connected component of F using a volume-preserving curvature flow with a strength proportional to the spatial and temporal resolution. Next, we calculate the closest points on S for all vertices of F, and we use the distance from each vertex to its closest point to initialize the height values of the simulated surface waves. We solve the wave equation on F using an implicit Newmark scheme with a wave propagation speed given by the surface tension strength. We then reposition the vertices of F according to the resulting solution, giving us an updated position of the fluid surface. Meanwhile, the curvature of S represents the remainder of the surface tension forces that can be represented within the Eulerian fluid simulation. We apply a similar framework for animating surface tension on the scale of the fluid grid by computing a second step of volume-preserving curvature flow on S, yielding a surface T. Similar to previous work, the pressure boundary conditions for the surface tension on the Eulerian grid can be computed using the distance of a point on S towards its closest point on T. Finally, the simulation proceeds to solve for a divergence-free velocity field, taking into account the surface tension boundary conditions computed as outlined above.

2 Related Work

While some of the first fluid simulations in computer graphics were performed by Kass and Miller in [1990] as well as [Foster and Metaxas 1996], a large body of work is now based on the simulation approach introduced in [Stam 1999; Foster and Fedkiw 2001; Enright et al. 2002], and many extension of this approach have been proposed in the following years. Examples of such extensions are coupling with thin shells [Guendelman et al. 2005], improved boundary conditions for rigid objects [Batty et al. 2007], a more accurate advection step [Selle et al. 2008], and stable two-way coupling with deformable objects [Robinson-Mosher et al. 2008]. A detailed description can be found, e.g., in the book by Bridson [2008], and we also employ this type of solver in our work. However, while level-sets [Osher and Sethian 1988] are used often to represent the free surface of a liquid, we use the method of [Wojtan et al. 2009] to track the fluid surface and handle topology changes. This method is based on an accurate Lagrangian surface representation, and produces temporally coherent surfaces, which makes it a good basis for surface tension simulations. In addition, it allows us to overcome problems typically associated with computing curvature driven flows on meshes: the mesh is re-sampled to prevent a clustering of vertices, and topological changes are handled by robustly and locally re-constructing the mesh. Apart from this class of solvers, different approaches have been introduced, such as particle based discretizations [Müller et al. 2003], or model reduction [Treuille et al. 2006]. We will focus on the commonly used Eulerian fluid solvers in this work.

Surface tension was recognized as an important aspect of fluids, and the method of Kang et al. [2000] is a popular choice to compute the surface tension boundary conditions. The discontinuities across the liquid-air interface for boundary conditions have been addressed in [Hong and Kim 2005], while [Müller et al. 2003] discuss surface tension forces for SPH simulations. Because purely Eulerian surface tension simulations do not behave properly at low grid resolutions, [Losasso et al. 2004] used an octree data structure to simulate surface tension effects. Contact angles of liquids with surface tension are discussed by [Wang et al. 2005], while [Wang et al. 2007] use a shallow water solver on obstacle surfaces to compute the motion of drops and streams of liquid. We similarly use a surface based wave solver, but use a temporally changing discretization of the liquid surface itself to solve the wave dynamics. We make use of a solver similar to [Angst et al. 2008], where waves were simulated on animated characters with a fixed connectivity.

Specialized techniques have been proposed for simulating surface tension phenomena such as drops and bubbles. Bubbles and frothing liquids were simulated in [Cleary et al. 2007], [Kim et al. 2007] focused on volume control for foam structures, and [Kim and Carlson 2007] presented a fast model for strongly bubbling fluids. [Zheng et al. 2006] introduced a regional level set method with a semi-implicit surface tension scheme for simulating bubbles. A model for both drops and bubbles, motivated by the Weber number of the fluid, was proposed by [Mihalef et al. 2009]. While these models handle their respective effects very well, our work focuses on the accurate modelling of the underlying surface tension dynamics to recreate phenomena such as droplet pinch off.

One inherent difficulty in direct simulations of surface tension flows is the strict limitation of the timestep. This topic was addressed in [Cohen and Molemaker 2004], who proposed a method to perform multiple surface tension driven advection steps per solver iteration. This approach better resolves the surface tension dynamics in time while safely taking larger time steps in the rest of the fluid simulation, but is not closely related to the underlying physics because it decouples the surface tension and pressure forces. Recently, Sussman and Ohta [2009] proposed a method to relax the time step restrictions from $O(\Delta x^{3/2})$ to $O(\Delta x)$ by performing a volume preserving mean curvature flow for the computation of the surface tension boundary conditions. While our approach is similar in nature, we compute surface tension on a mesh instead of a regular grid, and we perform an additional simulation at sub-grid scales.

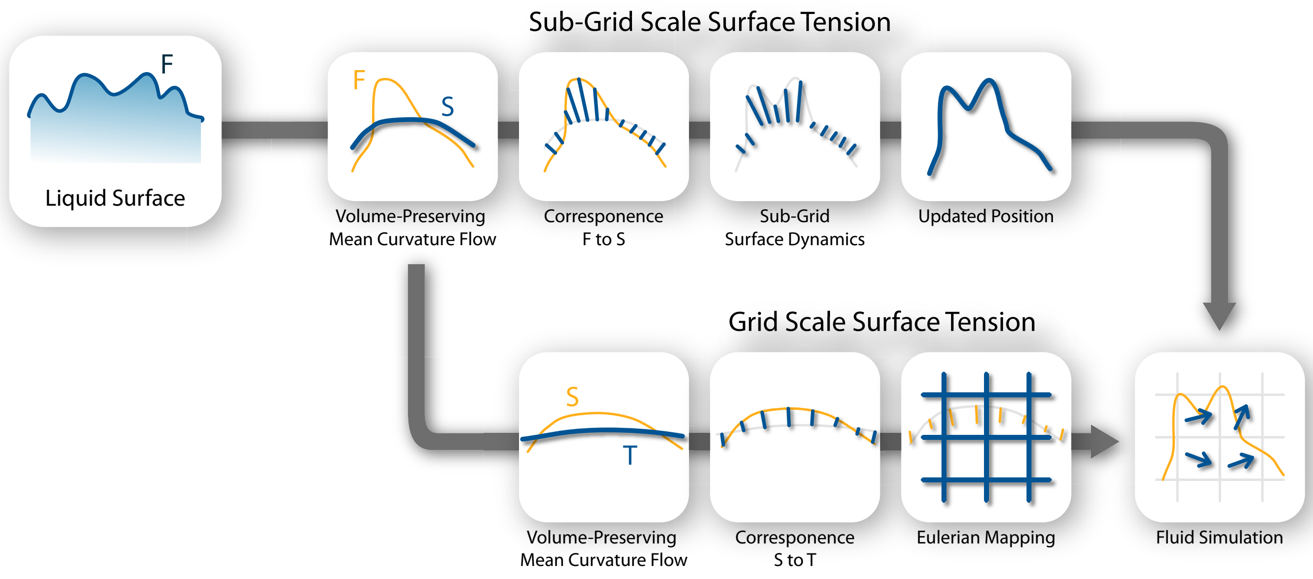
Sub-Grid Scale Surface Tension
Liquid Surface
F
F
S
Volume-Preserving Mean Curvature Flow
Corresponence F to S
Sub-Grid Surface Dynamics
Updated Position
Grid Scale Surface Tension
S
T
Volume-Preserving Mean Curvature Flow
Corresponence S to T
Eulerian Mapping
Fluid Simulation

Figure 2: *Overview of our surface tension approach. The two rows illustrate the steps performed for the sub-grid surface tension dynamics in the top row, and the Eulerian surface tension in the bottom row.*

As we make use of an explicit surface discretization, we can draw from the many works on shape operators for discrete surfaces. A good overview can be found in [Botsch et al. 2007]. Smoothing operations on meshes have been widely investigated, e.g., in [Desbrun et al. 1999], where an implicit scheme was presented. We will make use of volume preserving mean curvature flows on meshes, as was described in [Eckstein et al. 2007].

3 Fluids with Surface Tension

The dynamics of an incompressible viscous fluid with surface tension can be described by the *Navier-Stokes* (NS) equations:

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \,\triangle\mathbf{u} + \mathbf{g} + \sigma\kappa \qquad (1)$$

$$\nabla\cdot\mathbf{u} = 0\,. \qquad (2)$$

Here, $\mathbf{u}$ is the velocity of the fluid, $\mathbf{u}_t$ its temporal derivative, ρ the fluid density, $\mathbf{g}$ gravity, and σ denotes the surface tension coefficient, which, multiplied with curvature normal of the liquid interface κ , gives the surface tension force. Note that the surface tension terms are only evaluated at the liquid-gas interface. The NS equations are typically discretized on a Eulerian grid with cell size Δx and a time step of Δt, using a staggered MAC grid for storing the velocity information.

For level-set based surface representations, an established way to implement surface tension effects is to compute κ with the second derivatives of the signed distance function. The surface tension forces are included as a pressure jump across the interface, and the method of [Kang et al. 2000] can be used to include these boundary conditions in the fluid solver. This approach gives stable results for a moderate range of surface tension strengths. To improve stability, [Sussman and Ohta 2009] propose a method to compute the $\sigma\kappa$ term not directly from the level-set, but by first computing a volume-preserving mean curvature flow (VCF) of the liquid interface. The distance of the original towards the evolved surface is then used to compute the pressure boundary conditions. This strategy effectively integrates the forces for the upcoming time step taking into account the surface evolution, and it leads to a significant increase in stability.

However, the methods above still have to fully rely on the Eulerian grid to resolve features of the interface. As such, several cells are required to accurately resolve a single drop, and for features close to the size of a cell Δx the methods become inaccurate and unstable. When directly evaluating the curvature from the level-set, this can lead to noticeable ghost forces, and smaller drops can start to move randomly through the air. Furthermore, the surface tension method based on VCF computes a level set advection to resolve the curvature flow, and as such, small features can easily disappear, resulting in a significant mis-estimation of the surface tension forces.

These problems are closely related to the fact that the size of the largest stable time step for surface tension flows is strongly governed by the surface tension coefficient σ. Even though the methods outlined above can overcome this restriction, a small time step is required to accurately resolve the motion of capillary waves. Surface waves in a liquid are typically driven by gravity as well as surface tension. Under the assumption that the gas around the liquid has a negligible density, the dispersion relation, relating angular frequency ω to wave number k, simplifies to

$$\omega^2 = \left(|\mathbf{g}| + \frac{\sigma}{\rho}k^2\right)|k|\,. \qquad (3)$$

Note that for large k, which means spatially very small waves, surface tension dominates due to the k^2 factor, while large waves with small k are driven primarily by gravity. The phase velocity $\mathbf{v}_p = \omega/k$ of a water wave is thus given by $\sqrt{gk}$ for gravity driven waves, and by $|k|^{3/2}\rho/\sigma$ for surface tension driven waves. This directly implies that the smallest waves resolved on the grid would require a time step proportional to $\Delta x^{3/2}$. Note that even though semi-Lagrangian methods are unconditionally stable, surface waves do not transport mass, but are represented locally by circular motions, and thus would still require a very small time step to be resolved.

This stringent restrictions caused by surface tension have led to algorithms that perform multiple computational steps for the surface tension during a single step of the fluid simulation. [Hochstein and Williams 1996] predict the curvature of the surface for the next time step, and [Cohen and Molemaker 2004] perform additional advection operation to evolve the surface based on surface tension forces. [Sussman and Ohta 2009] can be seen in a similar line of thought, performing multiple iterations to evolve the volume-preserving curvature flow for calculating the surface tension forces. Our method takes this idea further, by using a wave equation solver

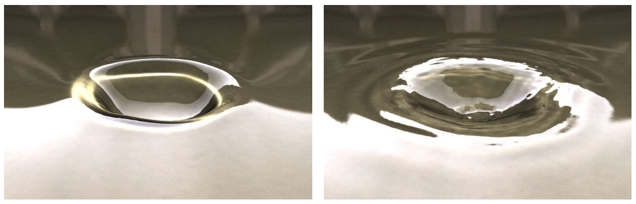
Figure 3: *This image shows a liquid surface after a drop impact. A normal simulation is shown on the left, while the right image demonstrates the small scales waves our method can resolve at the surface.*

to compute the dynamics of the capillary waves. We use an implicit, symplectic solver for this step, which means that we can handle waves of arbitrary propagation speed while conserving energy. In addition, we use a VCF to compute a base surface for the wave equation solver, which ensures that we simulate waves on the correct scale and re-introduces non-linear behavior into the linearized wave equation dynamics.

Given a surface tension coefficient σ, we split the overall strength into two components $\sigma = \sigma_s + \sigma_g$, where σ_s parametrizes the surface wave simulation, while the remaining strength σ_g is included in the grid-based simulation. Note that we could in theory split the forces arbitrarily: for $\sigma = \sigma_s$ all surface tension dynamics would be handled with our sub-grid model, while for $\sigma = \sigma_g$, the full surface tension strength would be calculated on the grid. In practice, we will use a geometric separation based on the grid resolution, so that the fluid simulation includes as much as it can resolve on the grid, while all smaller surface details are resolved with the wave equation. This is made possible by the fact that the algorithm to compute surface tension using a VCF resembles a surface fairing operation and effectively smoothes spatial frequencies according to the strength of the flow.

4 Mean Curvature Flow

First we will explain how to compute the smoothed version of the original fluid surface that is represented as a triangle mesh. It was shown by Sussman et al. in [2009] that the surface tension forces integrated over the length of a time step can be related to the distance of the liquid surface to another version of the surface advected by a volume preserving mean curvature flow (VCF). In the limit, the solution of the VCF will converge to one or more spheres with the same overall volume as the fluid component, which is exactly the result an isolated surface tension fluid simulation with viscosity would give. Note that a single component might be split into pieces by the surface tension, and thus form multiple spheres. We compute the smoothed surfaces S and T using a VCF, but solve the VCF directly on a surface mesh instead of a grid. In addition, we use this method for both layers of our simulation: for the surface tension forces on the grid, and the sub-grid scales on the fluid surface. Thus, in our case, S provides a separation of small spatial scales which will be handled by our surface dynamics model and larger scales for the Eulerian fluid simulation. As a first step we have to solve for a VCF of the initial and possibly very detailed surface of the liquid. Here, we can make use of the large body of research on curvature flows for meshes. A general mean curvature flow of the set of vertex positions $\mathbf{X}$ of a triangle mesh can be formulated as

$$\mathbf{X}_t = \gamma \nabla^2 \mathbf{X} \, , \tag{4}$$

where γ is the strength of the curvature flow, and the t subscript denotes a temporal derivative with respect to the smoothing time. Note that, by construction, Eq. (4) is equivalent to $\mathbf{X}_t = -2\gamma\kappa\mathbf{n}$, with κ denoting the mean curvature , and $\mathbf{n}$ the surface normal. This basic form evolves the vertices locally according to their Laplacian, which we discretize with the standard Laplace-Beltrami operator according to [Desbrun et al. 1999]. For a function f that takes the value f_i at a vertex v_i, the discrete Laplace-Beltrami operator can be formulated as

$$\nabla^2 f = \frac{1}{2A_i} \sum_{v_k \in \mathcal{N}(v_i)} (\cot\alpha_k + \cot\beta_k)(f_k - f_i) \, , \tag{5}$$

where the area of the barycentric dual cell associated with a vertex v_i is denoted by A_i, and the neighborhood of a vertex v is denoted as $\mathcal{N}(v)$, consisting of the adjacent vertices v_k.

This general form can be turned into a volume preserving surface flow by averaging the local deformations around a neighborhood of each vertex, as described by Eckstein et al. in [2007]. Similarly, we compute an averaged curvature κ_{avg} for each connected component of our surface mesh with

$$\kappa_{avg} = \frac{\sum_{v_i} A_i \kappa_i}{\sum_{v_i} A_i} \, , \tag{6}$$

and evolve the surface with

$$\mathbf{X}_t = -2\gamma(\kappa - \kappa_{avg})\mathbf{n} \, . \tag{7}$$

This system of equations is solved iteratively for a duration t^e. For the boundary conditions of the Eulerian grid, t^e is given by $t^e_{grid} = \Delta t \sigma_g$, while for the surface dynamics it is proportional to the grid resolution Δx, and is computed as $t^e_{grid} = \Delta t(\alpha + \Delta x)$. We introduce a parameter α here to manually increase the strength of the VCF for the sub-grid scales. We have found that the Eulerian solver typically has difficulties resolving scales on the size of one to two grid cells, and we make sure these scales are resolved by the surface wave dynamics by choosing α on the order of the grid cell size. This ties the geometric differences between the surfaces F and S to the grid size of the simulation. It is a result of our desire to cleanly separate grid-scale volumetric physics from sub-grid-scale surface physics.

We can speed up this process by simplifying the mesh to match the grid resolution in a first step. We do this by computing a signed distance field from the mesh, and then reconstruct the mesh from a level-set of this distance field using marching cubes. This immediately gives us a mesh that closely resembles the details that are representable with the given grid resolution. Note that an alternative would be to continue collapsing all edges that are smaller than Δx, but we have found the results from the level-set reconstruction are temporally more stable and are cheaper to compute. To ensure that even fine details of the mesh such as thin sheets and small drops are resolved on the grid, we triangulate the isosurface not at zero but for a thickened level set of $\sqrt{3}/2\ \Delta x$. We handle each disconnected component of the mesh separately, to make sure two separate components do not erroneously merge during the reconstruction from the signed distance field. This means, that we can use a strength of $t^e = \Delta t \alpha$ for the VCF of the wave solver.

Note that, despite the thickened level-set, we can guarantee that the surface after the VCF has the same volume as the input surface using the volume rescaling technique of Section 8. Our experiments have shown that the thickened grid-based re-sampling scheme mentioned above is faster and more consistent than the purely Lagrangian approach based on edge collapses. The grid-based scheme prohibits topological artifacts due to thin regions, although it can cause close regions of a single component to merge together. We did not notice any artifacts in practice, so we used this scheme to generate the examples of Section 10.

Another advantage of simplifying the initial mesh in a first step is that this typically reduces the number of vertices significantly, and the complexity for solving Eq. (7) directly depends on the number of vertices. In addition, the mesh S typically has a feature size on the comparably large scale of the underlying grid, which makes

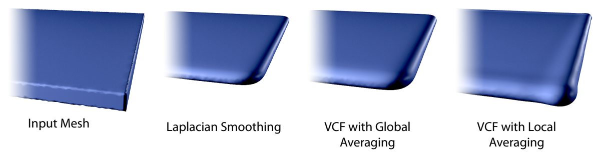

Figure 4: *Here a comparison of the different techniques for curvature flow can be seen. The input mesh is shown on the left, Laplacian smoothing second from left. Note that the VCF with global averaging behaves similarly to Laplacian smoothing, but conserves volume by pushing the whole surface outward. In contrast to this, local averaging conserves volume by locally bulging the surface.*

the curvature flow cheap to compute. This is important, as the step size of our explicit curvature flow scheme is proportional to the feature size squared. For highly detailed meshes, the implicit scheme described by [Eckstein et al. 2007] can yield a significant speed up. A faster, but less accurate alternative to computing this type of volume-preserving curvature flow is to use a global κ_{avg} in equation Eq. (7) by computing a single volume-weighted average curvature for each component. We will use both methods interchangeably in the following. While Figure 7 and Figure 1 make use of the local computation of Eq. (6), the other examples below make use of the faster per component averaging.

In a next step, we use the mesh S computed with the volume-preserving mean curvature flow to solve for the surface dynamics on the input mesh F. We will describe how to compute correspondences between the two meshes in the following. This step will be used to initialize the wave equation in Section 6 and set the boundary conditions on the grid in Section 9.

5 Mesh Correspondence

We now have the initial, unmodified liquid surface F, and a simplified version S that was evolved in a VCF. As S represents the target state towards which F should evolve due to surface tension, we next map the vertices of F to their closest points on S.

To do this efficiently, any spatial data structure, such as grids, trees or hash tables is applicable. For our implementation, we have chosen to insert the locations of the vertices $\mathbf{x}^S$ of S into a kd-tree and query the closest point $\tilde{\mathbf{x}}_i^S$ to each vertex $\mathbf{x}_i$ of F using this tree. We then check whether there is an even closer point $\mathbf{x}_i^S$ to $\mathbf{x}_i$ on the triangle fan around $\tilde{\mathbf{x}}_i^S$. Note that due to the VCF the surface might have shifted, and we have to make sure not to compute the closest point on the wrong side of thin fluid surfaces. To prevent this, we only return points from the kd-tree query whose normals align with the normal of $\mathbf{x}_i$. After this step, each vertex $\mathbf{x}_i$ on F has a corresponding point $\mathbf{x}_i^S$ on S. The set of points $\mathbf{x}_i^S$ on S will later on represent the reference surface with respect to which we solve the wave equation. The signed distance h_i between the two corresponding points can be computed with

$$h_i = (\mathbf{x}_i^S - \mathbf{x}_i) \cdot \mathbf{n}_i^S \ , \tag{8}$$

where $\mathbf{n}_i^S$ is the normal of the surface S at $\mathbf{x}_i^S$. These distances can now be used to solve for the surface dynamics. Given a new water height h_i' computed from the wave dynamics, the updated positions of the vertices is computed with

$$\mathbf{x}_i = \mathbf{x}_i^S + h_i'\mathbf{n}_i^S \ . \tag{9}$$

6 Wave Simulation on the Mesh

To efficiently compute the motion of capillary waves on the mesh surface, we linearize the surface tension dynamics with the classical wave equation. The wave equation is an often-used second order differential equation in which the normal acceleration of vertical elevations h is related to their second derivatives:

$$h_{tt} = c^2 \nabla_{v_i}^2 h \ . \tag{10}$$

Here h_{tt} denotes the second derivative with respect to time. The values of h for each vertex of F are given by Eq. (8), and $c^2 = \sigma_s$.

We perform the wave simulation on the triangle mesh F that represents the liquid surface. Similar to [Angst et al. 2008], we use a finite element discretization of the height values. The surface height is a scalar function that is represented with linear basis functions, and whose values h_i are located at the vertices of the triangle mesh. This leads to the commonly used operator with cotangent weights for the second derivatives according to Eq. (5).

To overcome the problem of energy dissipation while ensuring robustness, we make use of the Implicit Newmark time integration scheme [Newmark 1959] that was proposed in [Angst et al. 2008]. In the following, we will denote the vector of the height values for all vertices with $\mathbf{h}$ and write the evaluation of the Laplacian of $\mathbf{h}$ as the multiplication with the matrix L. Now, given a time step Δt, the Newmark integration step in its unconditionally stable form is given by

$$\Big(I - \frac{\Delta t^2}{4} c^2 L\Big)\mathbf{h}^{n+1} = \mathbf{h}^n + \Delta t \mathbf{h}_t^n + \frac{\Delta t^2}{4}\mathbf{h}_{tt}^n. \tag{11}$$

These equations represent a sparse system of linear equations; the unknowns are the positions of the next time step $\mathbf{h}^{n+1}$ that can be solved with a standard iterative solver such as a conjugate gradient method.

As the Newmark integrator is a symplectic scheme, this leads to a conservation of energy when solving the wave equation. We can manually introduce viscosity into the wave equation solve by manually scaling the surface heights by a factor of slightly less than one relative to the average height of the surface. After solving the wave equation, the surface F is updated using Eq. (9).

7 Non-Oscillatory Approximation

While the method for simulating dynamic surface tension proposed in Section 6 produces visually-appealing ripple effects on the surface, we can also efficiently simulate damped surface tension effects by neglecting surface wave phenomena entirely. As mentioned previously, we can use volume-preserving mean curvature flow to produce a reference surface S. The capillary waves oscillate about this surface, and they will eventually converge to it after the wave motion damps out. Therefore, S represents a type of steady-state solution to the surface tension dynamics. To efficiently simulate surface tension in the absence of small-scale capillary waves, all we have to do is run volume-preserving mean curvature flow on the original surface F by an amount proportional to the surface tension at each time step of the simulation. Because this approximation can be efficiently computed even on high resolution surface meshes, it can be used to simulate detailed droplet effects (See Figure 7).

8 Mass Conservation

A beneficial property of the triangle mesh surface representation is that we can accurately and efficiently compute its volume, e.g., as described in [Müller 2009]. This is necessary, as the overall conservation of mass can not be guaranteed despite the accurate advection calculation of the mesh vertices, e.g., with a fourth-order *Runge-Kutta* method. In addition, steps such as surface subdivision, topological changes, or the re-initialization from the wave equation solve can cause violations of the conservation of mass. This is especially crucial for smaller liquid volumes, which could completely disappear due to these errors. We use a rescaling along the surface normals to correct for these errors iteratively. Given a desired volume V_t and a current volume V_c with surface area A_c, we choose a step size of $h = \frac{V_t - V_c}{A_c}$, and iterate the following steps. First the positions of all surface vertices are updated with $\mathbf{x}_i = h\mathbf{n}_i$, where $\mathbf{n}_i$ is the normal of vertex i. We then recompute V_c, and when the relative error $\varepsilon = (V_t - V_c)/V_t$ changes sign from one iteration to the next, reinitialize the step size with $h' = -h/2$. This is repeated until converging to the desired accuracy. We have used $|\varepsilon| < 0.005$ for the examples below, and usually one or two iterations suffice to reach this accuracy.

Note that similar to the other steps of our algorithm, it is important to perform the volumetric rescaling separately for each connected component of the surface. Components with a negative volume, like bubbles, can be treated in the same way. Using the algorithm described in [Wojtan et al. 2009] to handle topology changes of the surface, we only need to identify disconnected components and update the target volume V_t for each component when a topology change was registered. This happens infrequently compared to the overall number of simulations steps.

9 Eulerian Surface Tension Forces

To compute surface tension forces for the grid-based fluid simulation we can re-use several steps of the framework described above for computing the sub-grid scale wave dynamics. As in [Sussman and Ohta 2009], we compute the term $\sigma\kappa$ from the distance between two surfaces, one of which is evolved in a VCF corresponding to the strength of the surface tension. As the difference between the original surface F and the ground surface for the wave equation S is handled as described in Section 6, we perform another step of VCF with Eq. (7) on S. For this step, the strength is given by σ_g, yielding an even smoother surface T.

Now we perform another correspondence computation from Section 5 to match the position of the vertices $\mathbf{x}_i$ of S with their closest points on T, denoted by $\mathbf{x}_i^T$. Note that $\mathbf{x}_i$ is the actual position of a vertex, while $\mathbf{x}_i^T$ can lie anywhere on the surface of T. Next we compute the surface tension strength for all cells of the grid at the interface with

$$\sigma_g \kappa = (\mathbf{x}_i^T - \mathbf{x}_i) \cdot \mathbf{n}_i. \qquad (12)$$

As the points in S will typically not align with the centers of the grid cells, we again use a kd-tree to retrieve the closest point on S with respect to the cell center. We interpolate the data if a closer point is found on an adjacent triangle. For this step, we can estimate the normal at the cell from the signed distance field of S and only query points from the kd-tree with aligning normals. While various approaches could be used to compute the distance from S to T, we decided to use a kd-tree primarily because of its temporally consistency. After using this surface tension value as a boundary condition for the Eulerian fluid solver, we then proceed to advect the surface mesh F according to the fluid velocities and handle topological changes with a local re-sampling method. The details of this process can be found in [Wojtan et al. 2009]. Next we will demonstrate the capabilities of our method with several test cases.

10 Results

The following simulations and performance measurements were performed on a standard PC with 3GHz without multi-threading. First, we compare our method to a surface tension computation performed by calculating the curvature of a level set surface representation. Several frames of a simulation of two merging drops with strong surface tension can be seen in Figure 5, where the top row was performed with the level-set, and the bottom row with our method. In each simulation, the drops have an initial diameter of 9 cells. While both simulations result in a similar behavior, our method is able to capture the waves on the surface of the drop that are caused by the merge. In addition, our method does preserve the volume of the fluid, and does not start to drift. However,

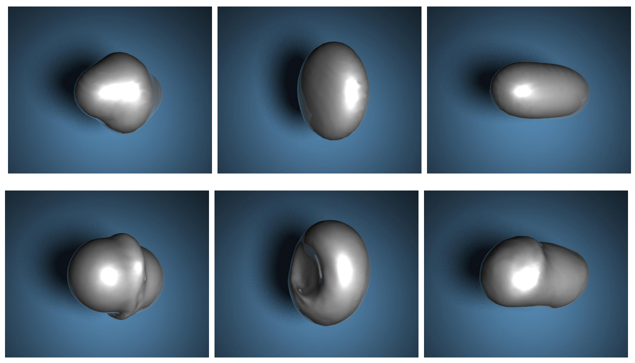
Figure 5: *Comparison of a level set based surface tension forces (top row) with our method (bottom row). While both simulations show an overall similar behavior of the two merging drops, our method is able to capture a capillary wave travelling around the merged drop with a high speed. In addition, our method is much better at conserving volume.*

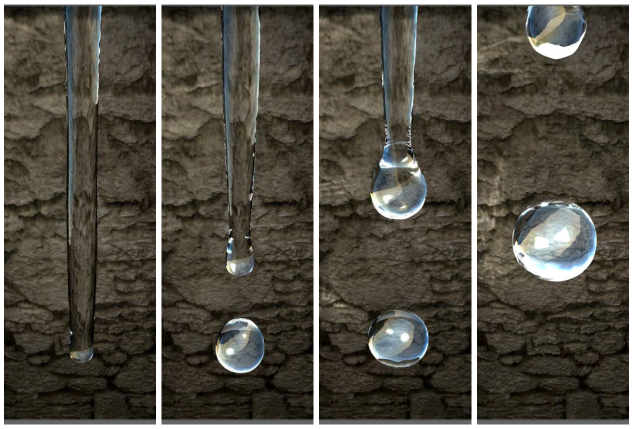
Figure 6: *Example of the self-reinforcing instability of a fluid jet causing droplet pinch off. The images from left to right show the change in behavior when increasing the surface tension of the liquid (σ_s = 3e-5 to 0.01). In Table 1, the settings for the second simulation from the left can be seen.*

the level-set style simulation took less time to compute. We also compared the stability of both methods with this setup: while level-set based simulation quickly becomes unstable with an increase in surface tension, our approach gives violently moving, but reasonable, results even for a 50 times higher surface tension coefficient. Because the time step restriction increases super-linearly for such explicit schemes, this difference in stability increases strongly at higher simulation resolutions.

The simulations shown in Figure 6 highlights that our method is able to very efficiently resolve droplet pinch off for liquid jets. This form of instability is caused by slight perturbations of the initial cylinder that grow over time. If the diameter of the jet is small enough, this ultimately leads to a pinch off. This phenomena is known as the *Rayleigh-Plateau* instability, and has been widely studied in experiments and simulations, see, e.g., [Bush 2004] for details. For our simulation, the jet is resolved with less than 5 grid cells in diameter, and the simulation ran at five frames per second. The images from left to right show how the behavior of the jets differs when increasing surface tension. The left-most image has a surface tension coefficient close to zero, resulting in barely any droplet pinch off, while the right-most one shows a simulation with strong surface tension, causing drops to pinch off right below the inlet.

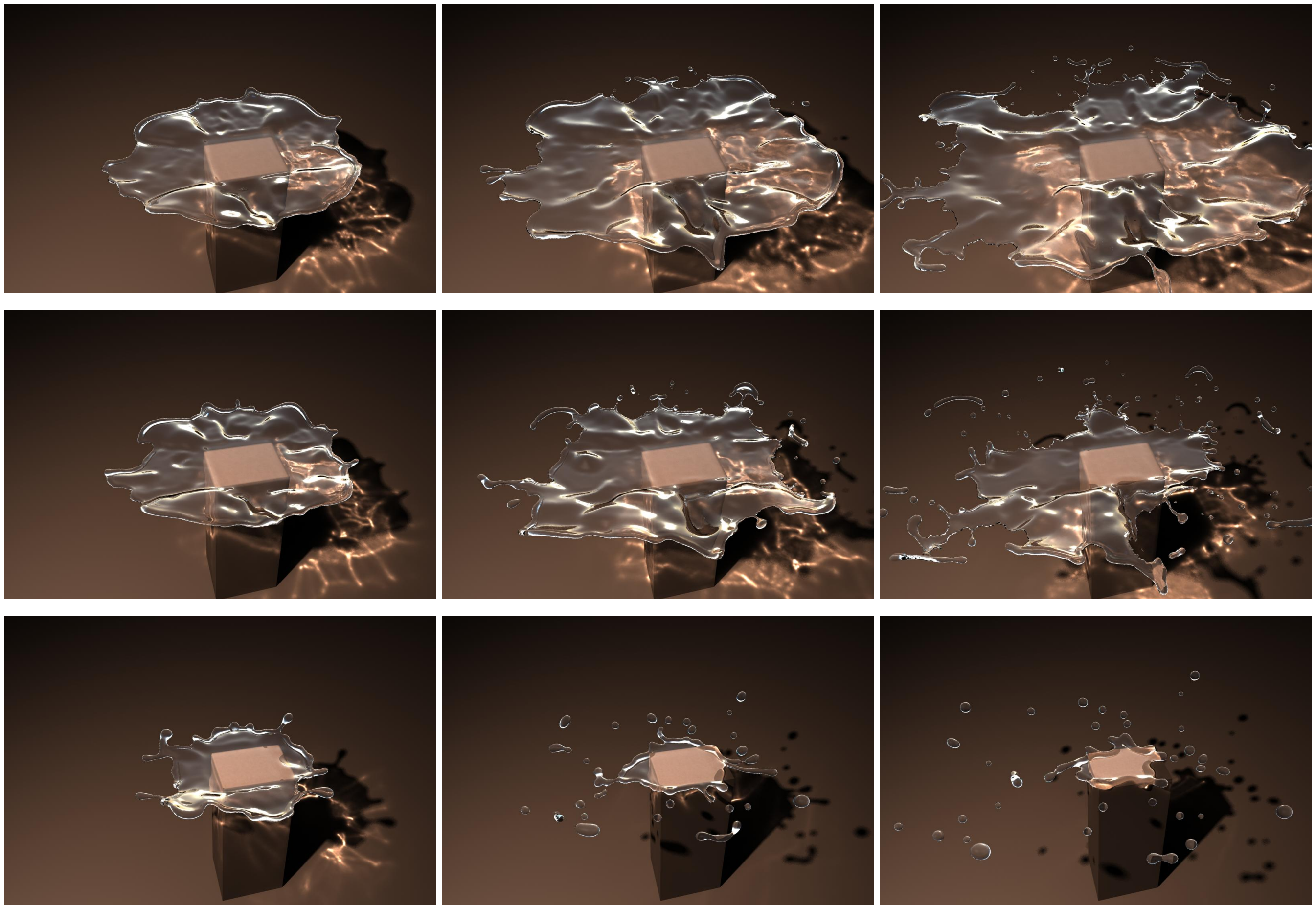

Figure 7: *A fast drop is colliding with a planar obstacle, resulting in a horizontal sheet of liquid. The images show simulations with increasing surface tension (from top to bottom) at the same instants in time. Stronger surface tension causes the liquid sheet to break up earlier.*

An example of a larger fluid simulation with a resolution of 128^3 can be seen in Figure 9. Here, a larger drop is falling into a body of water, and the resulting back splash causes a single drop to pinch off at its end. In this case, the high fluid simulation resolution is necessary to counter the numerical viscosity of the fluid simulation's semi-Lagrangian advection scheme.

The non-oscillatory approximation of Section 7 is demonstrated in Figure 7, where no grid based surface tension was active. The setup is a well studied experimental one: a liquid drop (or jet) with high velocity is hitting a planar or curved obstacle, causing a horizontal liquid sheet to develop [Clanet and Villermaux 2002]. The sheet breaks up at a specific radius inversely proportional to the surface tension strength. Our method allows us to re-create the effect that high surface tension causes an early break up, while low surface tension results in a large spreading sheet. For this simulation, we make use of the technique presented in [Wojtan et al. 2010] to track the very thin liquid sheets.

Finally, a simulation of a crown splash can be seen in Figure 1. This phenomenon develops when a drop hits a shallow volume of water, and the evolving circular liquid sheet breaks up at its outer boundary, resulting in regularly spaced droplets. For this simulation, we have used a combination of the grid-based surface tension forces and the non-oscillatory approximation. By varying the strength of the grid-based component, we are able to control the evolution of the liquid sheet, as can be seen in the accompanying video. To our knowledge, we are the first in the graphics and computational science field to simulate a full crown splash with droplet pinch off, and our method allows us to efficiently perform a very detailed simulation with a relatively low grid and surface resolution.

11 Discussion

The complexity of our algorithm predominantly depends on the resolution of the surface discretization. For a liquid surface mesh F with n nodes, and a smoothed version S with $m < n$ nodes, we observe that the following steps have a complexity of $O(n)$: the signed distance field computation, the calculation of correspondences from F to S, and solving the wave equation on F. On the other hand, the following steps have a complexity of $O(m)$: computing the volume preserving mean curvature flow for S and T, the correspondences from S to T, and the calculations of the boundary conditions for the grid-based simulation. The magnitude of both n and m predominantly depends on the surface area of the liquid interface, instead of the grid resolution.

The simulation settings and computation times for the shown simulations can be seen in Table 1. The fastest simulation was the Rayleigh-Plateau instability from Figure 6 with 0.2 seconds per timestep on average, while larger simulations, such as the crown splash from Figure 1 require up to 22.3 seconds per timestep. For the simulations shown, are always able to perform a single timestep per frame of animation. The resolution of the fluid simulations is typically determined by how many cells the solver requires to achieve a viscosity that is low enough to produce the desired motion. Our surface meshes are typically parametrized relative to the grid resolution, and the small simulations such as Figure 6 have

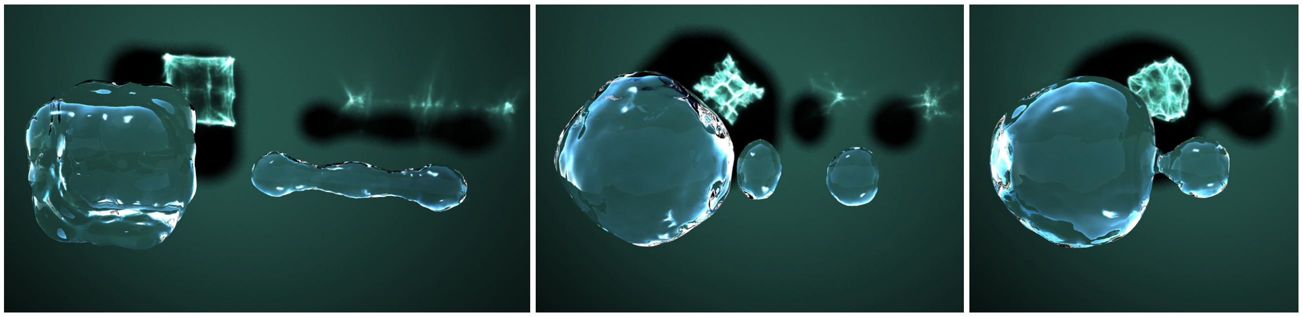

Figure 8: *These images show a simulation purely with our sub-grid wave equation solver, and without Eulerian surface tension forces. While the right drop is moving towards the large drop, it splits up, and finally merges with the large drop. Throughout the simulation, detailed capillary waves can be seen on the surfaces of the drops.*

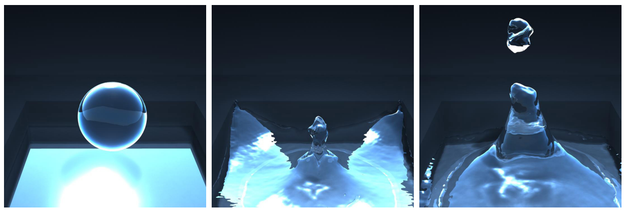

Figure 9: *A drop is falling into a body of water. The strong surface tension causes a reinforcement of the instabilities during the back splash, and finally results in a drop pinching off.*

around 2100 vertices for the mesh, while larger ones such as Figure 1 resolve the surface with up to 270 thousand vertices. Similarly, the amount of time that is required for our algorithm is determined by the complexity of the surface. For larger fluid simulations, such as Figure 9, on average one third of the overall time is spent for the surface tension calculations, while others, such as the large surfaces of Figure 7, require two thirds of the overall time to compute the surface tension dynamics.

Note that we model the non-linear surface tension behavior of a fluid simulation with a linear wave equation. We originally implemented a fully non-linear wave solution for the sub-grid scale dynamics, but our early attempts were plagued with self-intersections and numerical instabilities. The linearized version avoids these problems. However, the linearization also means that we can only simulate a single wave propagation speed given by the choice of the parameter c in Eq. (10). However, we achieve a non-linear behavior even without an underlying fluid simulation due to the fact that the positions updated by the wave equation solve are the input for the next mesh simplification and smoothing step, as described in Section 4. In the limit, our wave equation solver will iteratively converge towards a uniform height on the smoothed mesh S, which in turn will converge towards one or more spheres. This means that our method shows the desired behavior for a flow driven by surface tension and converges towards spheres as the final shapes for each disconnected component. However, some effects that are missing with only surface waves are the lower frequency oscillations of the surface, which occur on the scale of the smoothed surface S. In contrast to the small scale capillary waves, these larger oscillations can by construction be resolved on the Eulerian grid, and we can reuse many components of the algorithm above to compute surface tension boundary conditions for the grid based fluid simulation.

An important property of the wave equation is its ability to locally preserve the volume of the surface height function. This means that it is able to capture, e.g., the effect of a bulging front of a fluid sheet even when the sheet is much smaller than a grid cell. It can thus capture effects such as the breakup of thin sheets and droplet pinch off without having to rely on a Eulerian fluid simulation.

Our approach to surface tension does have some limitations. The surface wave equation does not completely conserve the mass of the overall fluid, but more specifically, that of the represented heights. This fact, in combination with the inaccuracies of the iterative solve make it necessary to enforce mass conservation with the method explained in Section 8. Luckily, we can accurately measure the initial volume of each component and keep its mass constant. In addition, despite the energy conserving nature of our implicit solver, the equation can lose energy due to the re-sampling of the underlying mesh. We currently only linearly interpolate the wave equation variables for triangle subdivisions and edge collapses, so higher-order interpolations could help to conserve energy.

In addition, as with any grid based method, our Eulerian approach for surface tension forces can be inaccurate for fluid components the size of a grid cell, because the boundary conditions can not be accurately represented on this scale anymore. We can however, reduce the strength of the Eulerian surface tension for connected components with a volume on the order of a grid cell, because we can rely on our sub-grid model to handle its dynamics.

12 Conclusion and Future Work

We have presented a method to simulate surface tension flows using a mesh-based surface representation. Our method handles grid based surface tension forces with high stability and allows for efficient simulations of fast capillary waves on the liquid surface. This enables detailed simulations featuring strong surface tension forces with droplet pinch off as well as capillary waves. In addition, we have shown how to use a fast approximation of the wave equation solution with a steady state surface flow to yield highly detailed surfaces. The mesh surface makes it possible to accurately compute volume preserving mean curvature flow, which is the basis for our grid based forces and sub-grid dynamics. Our approach allows us to simulate a wide range of surface tension phenomena with a low computational cost.

There are a number of extensions to our basic technique that we are considering as future work. We would like to include a non-linear wave solver, to more accurately handle dispersive capillary waves on the liquid surface. In addition, our method could be used to create interesting effects for bubbles and underwater phenomena, instead of only drops. For the grid based surface tension component, we would like to use more accurate immersed boundary methods from [Peskin 2002]. Finally, it would be intereting to include interactions with obstacle surfaces, such as enforcing certain contact angles as in [Wang et al. 2005].

Acknowledgements

This work was funded by NFS grants CCF-0811485 and CCF-0625264. We also thank the anonymous reviewers for their helpful comments.

	grid	mesh	time	σ_g	α	σ_s
Figure 5	75^3	0.5	1.0s	0.4	0.002	0.015
Figure 6	15^2x75	0.5	0.2s	5e-5	0.001	4e-3
Figure 7	160^2x80	0.25	12.6s	0	0	1.5e-4*
Figure 9	128^3	0.5	14.1s	2e-4	0.003	2e-4
Figure 8	50^3	0.45	1.0s	0	0.01	2.25e-3
Figure 1	180^2x50	0.55	22.3s	3e-6	0	3.5e-4*

Table 1: *This table shows the settings used for our simulations. The mesh resolution is given relative to the grid size, while the timing is the average simulation time per frame of animation. σs with an asterisk * denotes simulations with the non-oscillatory approximation.*

References

ANGST, R., THÜREY, N., BOTSCH, M., AND GROSS, M. 2008. Robust and Efficient Wave Simulations on Deforming Meshes. *Computer Graphics Forum 27 (7)* (October), 6, 1895 – 1900.

BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. 26*, 3, 100:1–100:7.

BOTSCH, M., PAULY, M., KOBBELT, L., ALLIEZ, P., LÉVY, B., BISCHOFF, S., AND RÖSSL, C. 2007. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, ACM, 1.

BRIDSON, R. 2008. *Fluid Simulation for Computer Graphics*. A K Peters.

BUSH, J. 2004. *MIT Lecture Notes on Surface Tension*. Massachusetts Institute of Technology.

CLANET, C., AND VILLERMAUX, E. 2002. Life of a smooth liquid sheet. *Journal of Fluid Mechanics 462*, 307–340.

CLEARY, P. W., PYO, S. H., PRAKASH, M., AND KOO, B. K. 2007. Bubbling and frothing liquids. *ACM Trans. Graph. 26*, 3, 97.

COHEN, J. M., AND MOLEMAKER, M. J. 2004. Practical simulation of surface tension flows. In *SIGGRAPH '04: Sketches*, ACM, New York, NY, USA, 70.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proc. SIGGRAPH*, 317–324.

ECKSTEIN, I., PONS, J.-P., TONG, Y., KUO, C.-C. J., AND DESBRUN, M. 2007. Generalized surface flows for mesh processing. In *SGP '07: Proceedings of the Eurographics symposium on Geometry processing*, 183–192.

ENRIGHT, D. P., MARSCHNER, S. R., AND FEDKIW, R. P. 2002. Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21*, 3, 736–744.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH 2001*, 23–30.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. In *Graphics Interface 1996*, 204–212.

GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. 2005. Coupling water and smoke to thin deformable and rigid shells. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, 973–981.

HOCHSTEIN, J. I., AND WILLIAMS, T. L. 1996. An implicit surface tension model. In *AIAA Meeting Papers*, vol. 96-0599.

HONG, J.-M., AND KIM, C.-H. 2005. Discontinuous fluids. *Proc. of ACM SIGGRAPH '05 24*, 3, 915–920.

KANG, M., FEDKIW, R. P., AND LIU, X.-D. 2000. A boundary condition capturing method for multiphase incompressible flow. *J. Sci. Comput. 15*, 3, 323–360.

KASS, M., AND MILLER, G. 1990. Rapid, stable fluid dynamics for computer graphics. In *the Proceedings of ACM SIGGRAPH 90*, 49–57.

KIM, T., AND CARLSON, M. 2007. A simple boiling module. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 27–34.

KIM, B., LIU, Y., LLAMAS, I., JIAO, X., AND ROSSIGNAC, J. 2007. Simulation of bubbles in foam with the volume control method. *ACM Trans. Graph. 26*, 3, 98.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. In *Proceedings of ACM SIGGRAPH 2004*, ACM Press, 457–462.

MIHALEF, V., METAXAS, D., AND SUSSMAN, M. 2009. Simulation of two-phase flow with sub-scale droplet and bubble effects. In *Proceedings of Eurographics 2009, CGF*, vol. 28:2.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. *Proc. of the ACM Siggraph/Eurographics Symposium on Computer Animation*, 154–159.

MÜLLER, M. 2009. Fast and robust tracking of fluid surfaces. *Proc. of Symposium on Computer Animation*.

NEWMARK, N. M. 1959. A method of computation for structural dynamics. *ASCE J. Eng. Mech. Div. 85*, 67–94.

OSHER, S., AND SETHIAN, J. 1988. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics 79*, 12–49.

PESKIN, C. S. 2002. The immersed boundary method. *Acta Numerica 11*, 1–39.

ROBINSON-MOSHER, A., SHINAR, T., GRETARSSON, J., SU, J., AND FEDKIW, R. 2008. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph. 27*, 3, 1–9.

SELLE, A., FEDKIW, R., KIM, B., LIU, Y., AND ROSSIGNAC, J. 2008. An unconditionally stable maccormack method. *J. Sci. Comput. 35*, 2-3, 350–371.

STAM, J. 1999. Stable fluids. In *the Proceedings of ACM SIGGRAPH 99*, 121–128.

SUSSMAN, M., AND OHTA, M. 2009. A stable and efficient method for treating surface tension in incompressible two-phase flow. *J. Sci. Comput. 31*, 4, 2447–2471.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics 25*, 3 (July), 826–834.

WANG, H., MUCHA, P. J., AND TURK, G. 2005. Water drops on surfaces. In *ACM SIGGRAPH 2005 Papers*, ACM, New York, NY, USA, 921–929.

WANG, H., MILLER, G., AND TURK, G. 2007. Solving general shallow wave equations on surfaces. In *Proc. of the ACM Siggraph/Eurographics Symposium on Computer Animation*, 229–238.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. In *ACM SIGGRAPH 2009 papers*, ACM, New York, NY, USA, 1–10.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2010. Physics-inspired topology changes for thin fluid features. 1–8.

ZHENG, W., YONG, J.-H., AND PAUL, J.-C. 2006. Simulation of bubbles. In *Proc. of the ACM Siggraph/Eurographics Symposium on Computer Animation*, 325–333.