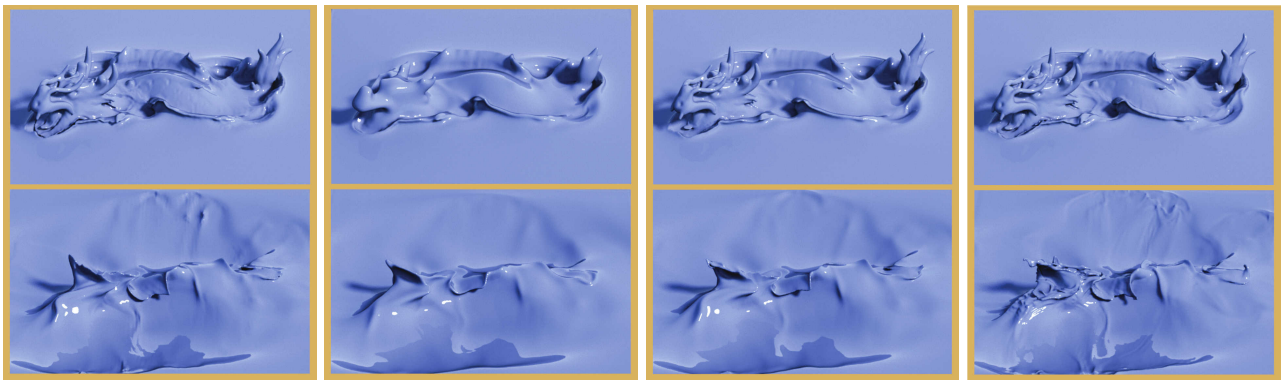
Liquid Surface Tracking with Error Compensation

Morten Bojsen-Hansen* IST Austria

Chris Wojtan* IST Austria



(a) Original simulation (b) Smoothed (c) Our smoothing (d) Our dynamics

Figure 1: *Our method permits high-resolution tracking of a low-resolution fluid simulation, without any visual or topological artifacts. The original simulation (a) exhibits sharp details and low-resolution banding artifacts. Smoothing the surface tracker (b) hides the artifacts but corrodes important surface features. We propose a smoothing technique (c) that preserves sharp details while selectively removing surface tracking artifacts, and a force generation method (d) that removes visual artifacts with strategically placed surface waves. Our algorithms are general and apply to both level sets as well as mesh-based surface tracking techniques.*

Abstract

Our work concerns the combination of an Eulerian liquid simulation with a high-resolution surface tracker (e.g. the level set method or a Lagrangian triangle mesh). The naive application of a high-resolution surface tracker to a low-resolution velocity field can produce many visually disturbing physical and topological artifacts that limit their use in practice. We address these problems by defining an error function which compares the current state of the surface tracker to the set of physically valid surface states. By reducing this error with a gradient descent technique, we introduce a novel physics-based surface fairing method. Similarly, by treating this error function as a potential energy, we derive a new surface correction force that mimics the vortex sheet equations. We demonstrate our results with both level set and mesh-based surface trackers.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

Keywords: liquid simulation, surface tracking, surface fairing, vortex sheets, level set method, triangle mesh

Links: DL PDF

*(mortenbh|wojtan)@ist.ac.at

1 Detailed surface tracking

This paper addresses the problem of tracking a liquid surface in an Eulerian fluid simulation. Within the field of computer graphics, Eulerian fluid simulation has become commonplace, with standard methods relying on a rectilinear grid or tetrahedral mesh for solving the Navier-Stokes equations [Bridson 2008]. The problem becomes significantly more complicated when we wish to simulate a free surface, such as when animating liquid. Correct treatment of this free surface requires special boundary conditions as well as some additional computational machinery called a *surface tracker*, such as the level set method [Osher and Fedkiw 2003] or a moving triangle mesh [Wojtan et al. 2011].

When animating a free surface, almost all of the visual detail is directly dependent on this surface tracker, because the surface is often the only visible part of the resulting fluid simulation. In order to make a simulation as detailed and visually rich as possible, we must add detail to the surface tracker. The computational cost of solving the Navier-Stokes equations scales with the volume of the simulation. Therefore, adding details to the surface by simply increasing the number of computational elements quickly becomes intractable. The problem can be somewhat alleviated by speeding up computational bottlenecks like the pressure projection step [Lentine et al. 2010; McAdams et al. 2010], but ultimately the volumetric complexity remains an obstacle. On the other hand, the costs of surface tracking only scales with the surface area, so the immediate temptation here is to increase the resolution of the surface tracker while keeping the fluid simulation resolution fixed. This strategy of only increasing the surface resolution has produced some beautiful results in the past [Goktekin et al. 2004; Bargteil et al. 2006; Heo and Ko 2010; Kim et al. 2009; Wojtan et al. 2009], but it introduces visual and topological errors that limit its usefulness with extremely detailed surfaces (Figure 2).

To see where these errors come from, we consider the relationship between the surface tracker and the fluid simulation. While the surface tracker certainly acts as the source of visual detail, it is also responsible for communicating the location of the free surface to

the fluid simulation. The fluid simulation then converts the shape of this free surface into Dirichlet boundary conditions for a Poisson equation. After solving this Poisson equation, the fluid simulation then adds pressure forces to ensure that any subtle variations near the free surface are accounted for in a manner consistent with the Navier-Stokes equations. However, a problem occurs if we lose information when conveying the free surface shape from the surface tracker to the fluid simulation; if the surface tracker is significantly more detailed than the fluid simulation, then there is no way to adequately encode all of the subtleties of the free surface into the boundary conditions. As a result of these mismatched levels of detail, the fluid simulation cannot recognize highly detailed surface features, and it cannot supply the necessary high-resolution pressure forces. Consequently, high resolution surface structures will clearly violate natural fluid motion, because they ignore the pressure term of the Navier-Stokes equations — the fluid simulation simply does not have enough degrees of freedom to prevent unphysical states in the surface tracker.

Previous methods have either ignored these errors, applied surface smoothing, or added additional detail to the fluid simulation in order to address these problems. While surface smoothing eventually removes unphysical high-resolution details, it also removes important physically valid motions, and it has no physical basis (or is based on unphysically strong and over-damped surface tension). Refining the fluid simulation detail near the surface is certainly a valid strategy, but perfectly matching the resolution of a detailed surface tracker often comes with significant extra implementation effort and computational overhead.

Our paper presents a fundamentally different approach for reconciling the difference between a high resolution surface tracker and a low resolution velocity field. We first propose a novel error metric that identifies and quantifies any unphysical surface behaviors by contrasting the current state of the surface tracker with the set of physically valid surface states. Once we have this information, we can use the gradient of this error function in a couple of different ways. We first introduce a novel physics-based surface fairing method that quickly removes surface artifacts while preserving physically-valid surface details. Next, we derive a novel surface correction force that removes artifacts with strategically placed gravity and surface tension waves. We show that this approach conveniently mimics the vortex sheet form of the Navier-Stokes equations while seamlessly integrating into an Eulerian simulation.

The contributions of our paper are as follows:

- Theoretical insight into the problem of coupling a high resolution surface tracker to a low resolution fluid simulation.
- A novel error metric for quantifying the physical validity of a fluid surface tracker.

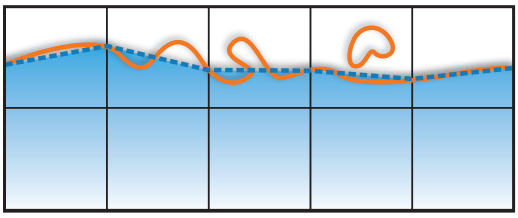


Figure 2: *If the surface tracker (orange) is much more detailed than the simulation grid (black squares), then the simulation can only work with a rough approximation of the surface (blue). The mismatch between the orange and blue surfaces can create visual artifacts like permanent surface kinks and floating droplets.*

- A surface fairing algorithm for fluid-like surfaces that clearly out-performs standard smoothing techniques.
- A surface correction force that removes high-resolution artifacts while preserving physically-valid details.
- Applications to both level set and mesh-based surface trackers.

2 Previous Work

Our algorithm concerns the combination of an Eulerian discretization of the Navier-Stokes equations with free-surface boundary conditions. The book by Bridson [2008] provides an excellent overview of the common fluid simulation methods in the field of computer animation. In order to simulate a free surface (for example, at the boundary of a liquid), Dirichlet boundary conditions must be enforced by specifying an exact value for the fluid pressure. First order boundary conditions simply set the pressure at the center of a boundary cell, but this strategy leads to aliasing artifacts due to the assumption that the fluid boundary precisely lines up with the simulation grid. Significantly higher accuracy can be achieved by using second order "ghost fluid" boundary conditions [Enright et al. 2003], which use linear extrapolation to set the surface pressure.

Several methods exist for representing and tracking a moving liquid surface. The most common Eulerian surface tracking method for fluid simulation is the level set method [Osher and Fedkiw 2003], which maintains a signed distance function and implicitly represents the surface wherever the function is equal to zero. The volume of fluid method [Hirt and Nichols 1981] also uses an Eulerian strategy to track the surface; by explicitly tracking the amount of fluid in each cell, the boundary can be observed by locating fractionally-filled fluid cells. Lagrangian surface tracking methods use particles or meshes [Wojtan et al. 2011] to explicitly represent the surface. Whereas topology changes are implicit for Eulerian surface tracking routines, they must be carefully computed for Lagrangian mesh-based methods [Brochu and Bridson 2009; Wojtan et al. 2009]. Hybrid surface tracking techniques, like the particle level set method [Enright et al. 2002] attempt to combine the merits of both Lagrangian and Eulerian surface tracking techniques.

High resolution surface, low resolution grid. As mentioned above, many researchers seek to extract additional richness from a simulation by significantly increasing the amount of detail in the surface tracker while fixing the resolution of the underlying fluid simulation. Goktekin et al. [2004] used a high-resolution particle level set to track a low-resolution viscoelastic fluid simulator. Similarly, Wojtan et al. [2008; 2009] used a Lagrangian mesh to retain detail at a much higher resolution than a viscoelastic simulation. Bargteil et al. [2006] used an octree and Heo et al. [2010] used a pseudo-spectral method to maximize the resolution of an Eulerian surface tracker for a fixed fluid simulation.

The main drawback with intentionally mismatching the surface and simulator resolutions is that the surface tracker tends to retain details that are invisible to the simulation [Brochu et al. 2010]. This is less of a problem for a viscoelastic motion, because unphysically-retained surface features may resemble rigid and elastic behavior. Eulerian surface representations, particularly with semi-Lagrangian advection, will naturally lose detail over time; so an overly-detailed tracker can help make up for this loss in detail, although they will preserve visual artifacts if they are too successful. See Figure 5 for an example of these visual artifacts.

Removing artifacts. One strategy for eliminating the artifacts caused by a high-resolution surface tracker is to adaptively in-

crease the fluid simulation resolution near the boundary. Losasso et al. [2004], Hong et al. [2005] and Kim et al. [2007] used an octree to adapt a fluid simulation to a high-resolution level set. Brochu et al. [2010] introduced a simulator based on an adaptive Voronoi diagram in order to match the resolution of a high resolution mesh surface. The general strategy of adding detail to the fluid simulation will naturally remove artifacts due to mismatched resolutions, but it can be computationally expensive and handling spatial adaptivity may introduce further simulation artifacts.

Instead of adapting the fluid simulation to the surface tracker, several methods try to make the high-resolution surface conform to the low-resolution physics. Wojtan and Turk [2008], and Kim et al. [2009] attempt to remove high-frequency visual artifacts using smoothing algorithms, while Yu and Turk [2010] use anisotropic smoothing kernels to bias the loss of surface detail. Williams [2008] and Thürey et al. [2010] attempt to make up for volume-loss artifacts with bi-Laplacian smoothing. While such smoothing approaches may be effective in small doses, they will destroy many interesting surface details when applied with too much enthusiasm, and they do not produce physically correct surface motions. Smoothing in this manner is related to over-damped surface tension, which may be appropriate for small-scale viscous flows but is inaccurate for inviscid liquid phenomena.

In an attempt to make up for the lack of detailed surface motions when combining a low-resolution simulation and a detailed surface, Thürey et al. [2010], Bojsen-Hansen [2011] and Yu et al. [2012] propose using high-resolution dynamic surface waves. These methods mask high resolution surface artifacts with rippling motions, but they are based on inappropriate restrictions such as shallow water, height-field, and constant wave speed assumptions. These restrictions also require that the surface tracker must be homeomorphic to a low-resolution simulation boundary in order to function properly, while our method naturally removes topological inconsistencies with Rayleigh-Taylor-like instabilities. The method of Wojtan et al. [2010] removes topological inconsistencies in the surface tracker by re-sampling the surface, but it does not address the problem of removing surface noise.

Vortex methods. Our method results in equations that resemble the vortex sheet equations. Several researchers have used vortex particle methods [Selle et al. 2005; Park and Kim 2005; Pfaff et al. 2009; Kim et al. 2012] and vortex sheet methods [Pfaff et al. 2012; Brochu et al. 2012] to add details and generate turbulence in fluid simulations, though these methods do not directly address the difficulties of simulating high-resolution motion with a free-surface. The method of Kim et al. [2009] uses a high-resolution surface tracker coupled with the vortex sheet equations to drive a low-resolution vorticity confinement force. This strategy serves to add interesting low-resolution turbulence to the simulation, but it also enhances high-resolution surface noise. Our method only adds turbulence in areas that exhibit unphysical behavior and uses it to remove surface noise artifacts.

3 Method

As mentioned in Section 1, our main source of unphysical behavior is the conversion from the detailed surface tracker into pressure boundary conditions. We first aim to quantify these errors.

3.1 In the absence of Surface Tension

We observe that, in an analytical solution to the Navier-Stokes equations in the absence of surface tension, the pressure at the free surface is equal to that of the surrounding air. Following standard

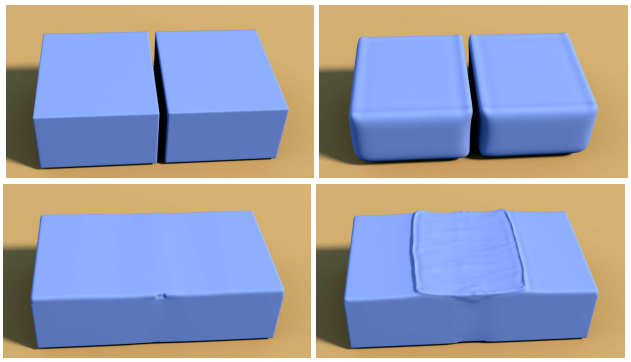


Figure 3: *This instructive example inserts a high-resolution cut in the surface of a low-resolution simulation. The cut is smaller than a grid cell, so the original fluid simulation (top left) ignores it. Smoothing the surface tracker (top right) leads to extreme loss of volume and surface details. Our smoothing algorithm (bottom left) quickly fills in the gap, and our dynamics algorithm (bottom right) converts the artifact into fluid energy.*

practices for simulating liquids, we assume that the air pressure is a constant zero along the interface [Bridson 2008]. Next, we note that the zero level set of pressure perfectly coincides with the location of the free surface. Because the gradient of a function is always orthogonal to its level sets, we infer that the pressure gradient should be perpendicular to the free surface. Accounting for the fact that the pressure is positive inside the liquid, we can further state that the pressure gradient must be anti-parallel to the surface normal in order for a flow to be consistent with the Navier-Stokes equations (Figure 4(a)). We use this information to define an energy function:

$$E_0 = \int_{\partial\Omega} \mathbf{n} \cdot \nabla p \, dA \tag{1}$$

where $\mathbf{n}$ is the surface normal at a point on the surface, ∇p is the pressure gradient, dA is the area of a surface tracking element, and $\partial\Omega$ is the entire free surface according to the high resolution surface tracker. Intuitively, the energy is minimized when the free surface is physically valid (when the normal is anti-parallel with the pressure gradient). We evaluate the surface normal from the surface tracker and the pressure gradient from the fluid simulation, so any deviation from the minimum energy state encodes an unphysical disagreement between the surface tracker and the fluid simulation.

We propose to reduce this error by following the direction of steepest descent of the energy function. The energy gradient is the derivative of Equation 1 with respect to its free variables. In our case, the surface tracker is overly-detailed and under-constrained, so we will only adjust the control variables of the surface tracker. We approximate divergence-free motion by constraining our surface tracker adjustments to be local rotations; thus, our degrees of freedom are the orientations of the surface tracker normals $\mathbf{n}$.

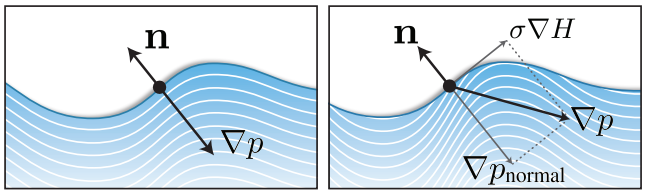
Taking the partial derivative of Equation 1 with respect to $\mathbf{n}$, the direction of steepest descent for surface normal is:

$$-\frac{\partial E_0}{\partial \mathbf{n}} = -\nabla p \, dA. \tag{2}$$

However, the normals must remain unit length, so we reformulate the energy gradient as a local rotation and arrive at the equation:

$$\mathbf{n} \times (-\frac{\partial E_0}{\partial \mathbf{n}}) = \nabla p \times \mathbf{n} dA, \tag{3}$$

which encodes the area-weighted rotational velocity and axis of rotation as a vector magnitude and direction respectively. This equation tells us that we should rotate the normal away from the pressure



(a) without surface tension (b) with surface tension

Figure 4: *A schematic of the simulated liquid (blue) and its level sets of constant pressure (white lines). The surface normal* $\mathbf{n}$ *is perpendicular to the fluid surface, while the pressure gradient* ∇p *is perpendicular to the surfaces of constant pressure. In the presence of surface tension, the tangential component of* ∇p *is proportional to the gradient of mean curvature.*

gradient, with a strength proportional to the magnitude of the pressure force, if we wish to decrease the surface error.

3.2 Including Surface Tension

For large scale flows, surface tension forces are practically negligible, and the above analysis is sufficient. For smaller scale flows, however, we must incorporate effects due to surface tension.

In the presence of surface tension, the pressure at the free surface is $p = \sigma H$, where σ is the surface tension coefficient and H is the mean curvature of the free surface at that point. This pressure can vary along the surface, so the pressure gradient will have a tangential component (Figure 4(b)). As a result, we can no longer assume that the pressure gradient is normal to the free surface. However, the tangential variation of the pressure is fully defined by our free surface boundary conditions, so the tangential component of the pressure gradient is equal to the gradient of the surface pressure: $\nabla p_{\text{tangent}} = \sigma \nabla H$. We decompose the pressure gradient into normal and tangential parts and solve for the normal component:

$$\nabla p_{\text{normal}} = \nabla p - \sigma \nabla H \tag{4}$$

Using the same reasoning as in the previous section, the normal component of this pressure gradient should be anti-parallel to the surface normal in a fully resolved fluid simulation. Our surface tracker is more detailed than the fluid simulation and the surface normals will vary, so we introduce the following energy function:

$$E_{ST} = \int_{\partial\Omega} \mathbf{n} \cdot (\nabla p - \sigma \nabla H)\, dA \tag{5}$$

which is again minimized when the surface tracker represents a physically valid configuration. We compute the partial derivative of Equation 5 with respect to the surface normals:

$$-\frac{\partial E_{ST}}{\partial \mathbf{n}} = (-\nabla p + 2\sigma \nabla H)\, dA \tag{6}$$

This derivative is slightly more complicated, because H depends on $\mathbf{n}$, as explained in the Appendix. We then encode the result as a rotation

$$\mathbf{n} \times (-\frac{\partial E_{ST}}{\partial \mathbf{n}}) = \nabla p \times \mathbf{n}\, dA + 2\sigma \mathbf{n} \times \nabla H dA \tag{7}$$

The first term of this update rule is identical to the case without surface tension. The second term indicates that we should rotate the surface normal towards the direction of increasing curvature if we want surface tension to smooth out the surface.

As before, we evaluate the pressure gradient from the fluid simulation and we evaluate the normals from the surface tracker. We are left with a choice of whether to evaluate the mean curvature H using the fluid simulation or the surface tracker. We first note that small values of H will be computed correctly regardless of where we evaluate it, because low curvatures are easily represented on low-resolution fluid grids. However, large values of H cannot be accurately computed on the low-resolution simulation grid and will be clamped to some maximum value related to its Nyquist limit. As a result, high-resolution surface tension motions will be ignored if we compute H using the low resolution fluid simulation. On the other hand, computing H from the surface tracker will allow high-resolution motions, but it will not necessarily give us the same value of ∇p_{normal}. However, the second term of Equation 7 will always act to reduce the surface curvature, so we can be confident that it will at least move in the right direction until H is reduced and its computation becomes consistent with the fluid simulation. For this reason, we choose to evaluate H on the surface tracker.

4 Applications

Now that we have defined error functions and their gradients, we propose a few different strategies for reducing unphysical behaviors in the surface tracker.

4.1 Physics-based surface fairing

Our first application uses the above analysis to derive a gradient-descent technique for reducing unphysical behavior. The gradients derived in the previous section are weighted by area, so we first divide the equation by the area of the surface element to get a local rotational motion. Then, by assigning local rotations to each point on the surface, we present a physically-based surface fairing rule for fluid simulations:

$$\omega = \alpha(\nabla p \times \mathbf{n} + 2\sigma \mathbf{n} \times \nabla H) \tag{8}$$

where ω represents the angular velocity at a point on the surface tracker and α is a user-tunable smoothing parameter. In our implementation, we convert this rotational velocity ω into a local high-resolution velocity field using a finite-kernel approximation to the Biot-Savart law. We provide implementation details in Section 5.

When applied steadily throughout the progress of the fluid simulation, this procedure has the remarkable effect that it filters the sur-

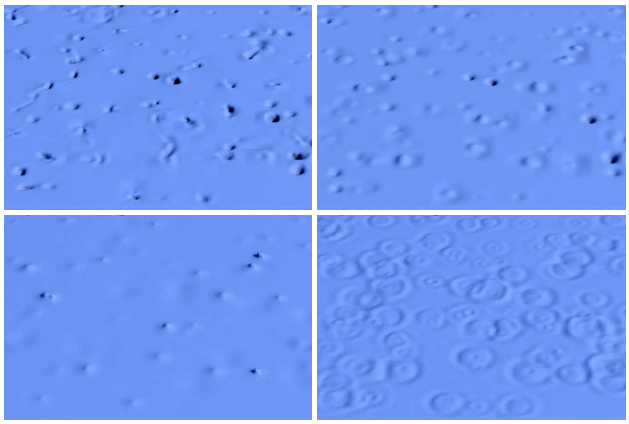


Figure 5: *The original simulation (top left) cannot remove high resolution noise. After many iterations, Laplacian smoothing (top right) slowly diffuses errors across the surface. Our smoothing method (bottom left) immediately targets and flattens artifacts, and our dynamics algorithm (bottom right) converts artifacts into waves.*

face at a rate proportional to the magnitude of the pressure gradient — nearly static liquid surfaces are quickly smoothed out while ballistic motions are left untouched (Figure 8). Note that this update has no effect if the surface tracker is the same resolution as the fluid simulation; in this case, the normal component of the pressure gradient will precisely line up with the surface normal and the effect of our update rule will disappear. This procedure only acts to correct errors due to a mis-match between the surface tracker and the boundary conditions of the fluid simulation, and the effect smoothly fades away as the fluid simulation accuracy increases.

4.2 Fluid simulation on the surface tracker

Another option is to utilize Equation 5 as a physical potential energy. We derive surface forces from the gradient of this energy and factor out the per-element area to get an equation for angular acceleration. The result is essentially the time derivative of our surface fairing algorithm:

$$\dot{\omega} = \beta(\nabla p \times \mathbf{n} + 2\sigma \mathbf{n} \times \nabla H) \tag{9}$$

where $\dot{\omega}$ is the angular acceleration at some point on the surface tracker, and β is a user-tunable parameter analogous to a spring constant or squared wave speed. Instead of simply smoothing out errors in the surface tracker, this approach transforms surface artifacts into water waves (Figure 5).

We note that our error correction forces are remarkably similar to the buoyancy and surface tension terms of the vortex sheet equations [Pozrikidis 2000], hinting that a good choice for our tuning parameter β is a value of twice the Atwood ratio:

$$\beta \approx 2(\rho_{\text{liquid}} - \rho_{\text{air}})/(\rho_{\text{liquid}} + \rho_{\text{air}}) = 2$$

The main differences between our method and previous vortex sheet discretizations [Kim et al. 2009; Pfaff et al. 2012; Brochu et al. 2012] are that we omit the Boussinesq approximation, and we use a low-resolution pressure gradient from the fluid simulation instead of the total acceleration of the surface. By tying the low-resolution simulation into our dynamics, our surface tracker is guaranteed to oscillate about a low-resolution surface representation. In contrast, a high-resolution vortex sheet discretization may easily drift from the simulation and become arbitrarily complicated. The derivation of our method also indicates that the vortex stretching and dilation terms of the vortex sheet equations are unrelated to the reduction of surface tracking errors; omitting these terms will only affect wave propagation speeds.

Because Equation 9 is based on the inadequacy of a low-resolution pressure gradient, the resulting dynamics are allowed to gracefully interact with an Eulerian fluid simulation without double-counting forces. As the resolution of the fluid simulation increases relative to that of the surface tracker, the effect of these additional fluid dynamics diminishes, until they disappear when the resolutions are equal. We found this approach to be an effective strategy for adding high-resolution dynamics to a low-resolution fluid simulation without requiring much computational overhead. Again our implementation uses an approximation to the Biot-Savart law to convert our angular acceleration into a velocity field, with details given in Section 5.

5 Implementation Details

In our fluid simulation implementation, we use a regular MAC grid fluid simulation with second order ghost fluid boundary conditions at the free surface [Bridson 2008]. We use the liquid-biased filter of Kim et al. [2009] to represent thin liquid features on the coarse fluid grid. We convert the rotational velocities in Section 4 into a velocity field using using the Biot-Savart Law with a finite kernel size, as in [Pfaff et al. 2012]. Because any size kernel will create a local rotation and will thus reduce the errors in the surface tracker, the kernel size is irrelevant when considering error minimization. Larger kernel sizes simply allow for lower-frequency surface rotations, which may have visual importance depending on the application. Both the first-order smoothing (Section 4.1) and second-order dynamics (Section 4.2) are integrated with a symplectic Euler method.

Once a pressure field has been computed in the fluid simulation, we compute its gradient and extrapolate both quantities past the free surface using the usual constant extrapolation in the normal direction. Subsequently, we use tri-linear interpolation when evaluating either quantity at points on the free surface. In some of our smoothing experiments, we found it useful to re-scale the pressure gradient by the difference between the extrapolated pressure evaluated at the interface and the outside air pressure instead of directly using the low-resolution pressure gradient. This adjustment has no effect in most cases near the fluid surface, but it causes bubbles and air pockets deep below the liquid surface to smooth out more quickly.

5.1 Solid boundaries

Our high resolution surface tracker can cause kinks and bubbles along solid boundaries as well as along the free surface. When the surface pushes up against a solid obstacle, its surface normal should be anti-parallel that of the boundary (Figure 7). Using similar reasoning to Section 3, we arrive at the following update rule:

$$\mathbf{n}_{\text{surface}} \times \left(-\frac{\partial E_{\text{boundary}}}{\partial \mathbf{n}}\right) = \mathbf{n}_{\text{boundary}} \times \mathbf{n}_{\text{surface}} dA \tag{10}$$

We can also multiply the strength of this motion by the fluid pressure if we wish to create more drastic behaviors in high pressure regions. However, we declined this option in order to reduce sources of potential instability.

In practice, we switch from the free surface update rules in Section 4 to the solid boundary update rule for each point on the surface tracker that is within a given distance of the boundary. In this paper, we use a distance of three times the size of the minimum resolvable detail in the surface tracker.

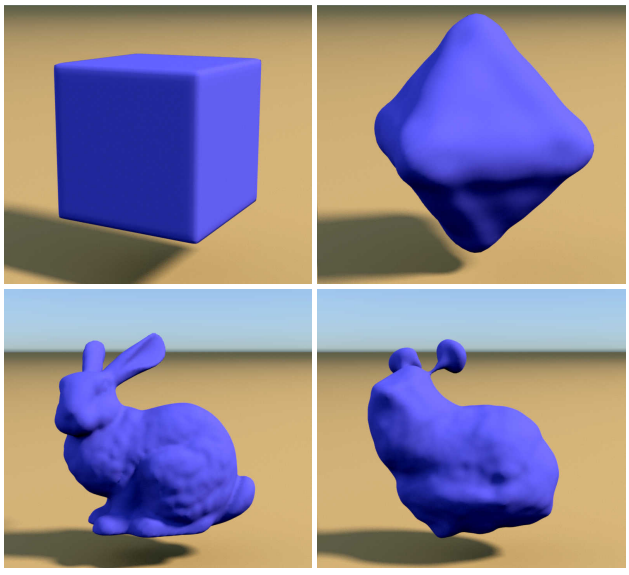


Figure 6: *The technique in Section 4.2 can create detailed surface tension dynamics of a cube (top) and a bunny (bottom), even with a low simulation resolution.*

We create a special case along corners where the low-resolution free surface borders a solid boundary, because we do not want a small change in position to result in a drastic change in motion. To remove the potential for such discontinuous gradients in the corners, we only allow rotations about an axis parallel to the boundary normal. We further add a small amount of Laplacian smoothing at the corners to compensate for errors that we ignore with this restriction.

We have tested our methods on both high resolution level set surface trackers as well as Lagrangian triangle mesh surface trackers. While the overall principles for integrating our error correction mechanism are independent of the tracker, we discuss some specific implementation details below.

5.2 Level set surface tracker

We use the sparse level set implementation OpenVDB [Museth 2013] with an adaptive WENO5 scheme for level set advection, because it minimizes artificial diffusion artifacts. We used a narrow band the size of one fluid cell, and we evaluate the surface normal by taking the gradient of the signed distance function using central differencing. We compute free surface boundary conditions by down-sampling the level set function onto the fluid grid and then performing the ghost fluid method on the low-resolution distance function. We typically set the level set resolution four to eight times higher than that of the fluid simulation.

We compute our energy gradients (Equations 8 and 9) on a high-resolution sparse grid, and we use a smeared delta function with a radius of 1.5 level set cells to confine them to the surface, similar to Kim et al. [2009]. We then use a Biot-Savart kernel with a diameter of five level set grid cells to convert local rotations into detailed velocities, and we store the resulting velocities on a grid co-located with the level set samples. We up-sample the low-resolution fluid velocity field and add it to this high resolution velocity field in order to advect the level set. For computing dynamics, we also store, extrapolate, and advect the angular velocities on a grid co-located with the level set.

5.3 Mesh surface tracker

We use a Lagrangian triangle mesh surface tracker with a voxelization-based method for computing topology changes, as in Wojtan et al. [2010]. We create a low-resolution signed distance function from the triangle mesh as part of the topology-changing procedure, which we use for the ghost fluid method. We keep the topology change grid at the same resolution as the fluid simulation, and we maintain an average mesh resolution 4 to 5 times higher than that of the simulation grid.

We evaluate our energy gradients at the centroid of each triangle face using the geometric normal of each triangle. We follow Pfaff et al. [Pfaff et al. 2012] to compute a regularized Biot-Savart kernel with a diameter of four fluid simulation grid cells, and we store

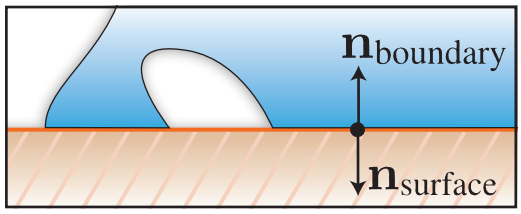


Figure 7: *When the surface of a liquid (blue) contacts a solid obstacle (orange), the two normals should be anti-parallel.*

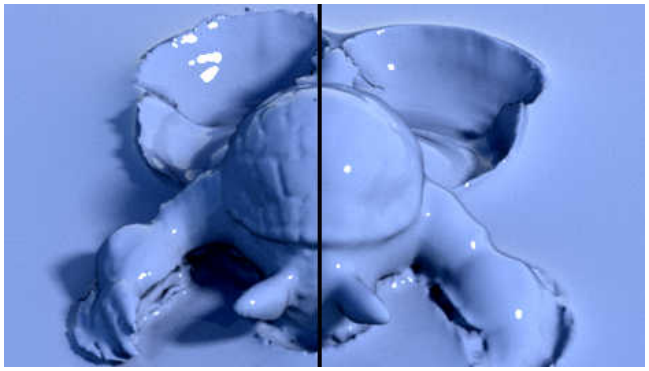


Figure 8: *Our mesh-based smoothing method (left) preserves details in low-pressure regions, unlike mean curvature flow (right).*

the resulting velocities on the mesh vertices. We also store and transport angular velocities by converting them to circulations on triangle faces, in the way of Stock et al. [2008]. We advect the mesh through the low-resolution velocity field of the fluid simulation with a fourth order Runge-Kutta method, and we advect the mesh through the high-resolution mesh velocity field using an Euler method.

5.4 Stability

We use symplectic Euler time integration for our surface fairing and dynamics algorithms, so we inherit the expected stability criteria. For instance, this means that our gravity waves should obey a standard CFL condition and that the surface tension time step should shrink with the spatial resolution to the power of 1.5. Although we are currently unable to provide a formula for the stability of our smoothing algorithm, we believe it has similar stability behavior to other volume-preserving fairing algorithms like bi-Laplacian smoothing.

Nevertheless, we noticed the level set implementation was remarkably stable in practice. In particular, we were able to take much larger time steps with our surface tension dynamics than with standard ghost fluid-based surface tension. Additionally, while excessively large correction forces caused high frequency oscillations, the method is able to recover quickly from short periods of instability.

Our mesh-based implementation is a bit more delicate to instabilities. While a grid-based level set has a guaranteed minimum distance between any two samples, centroids on a triangle mesh can be arbitrarily close together when folded into a thin sheet. The regularizer in the Biot-Savart kernel mentioned above helps to limit large velocities for close particles, and we also found it convenient to artificially limit the size of the error gradients by multiplying by a constant slightly less than one. For large α and β parameters, we used sub-cycling to integrate the surface tracker through several small time steps in between large fluid simulation time steps.

6 Results

We generated several didactic animations to illustrate the behavior of our method, which can be seen in the accompanying video. For illustrative purposes, these examples were computed with an exaggerated low resolution of 32^3 for the fluid simulation and 256^3 for the level set surface tracker. One obvious benefit of our method is that it can selectively remove high-resolution surface noise from a low-resolution simulation (Figure 5). The method also addresses

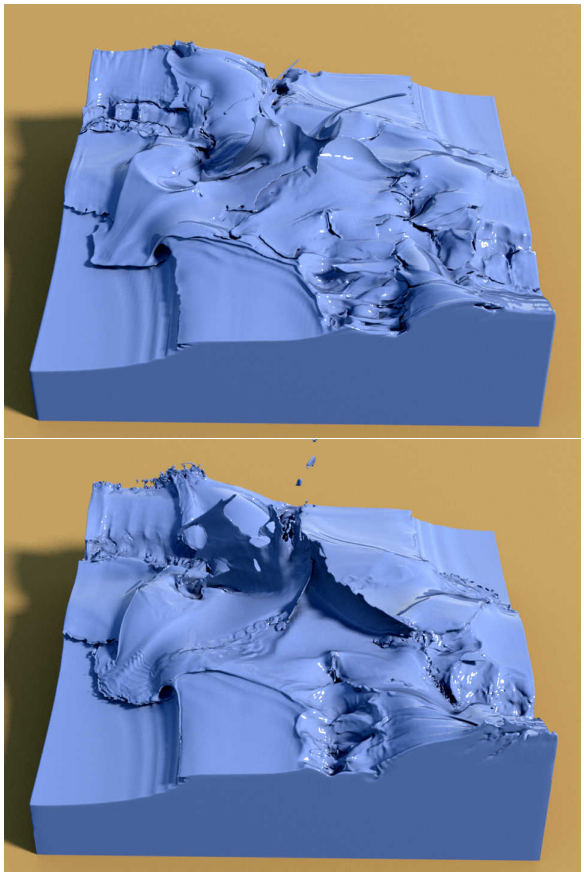


Figure 9: *The original simulation (top) is enhanced by our wave dynamics (bottom). Notice how the detailed motion creates numerous topological artifacts holes and crevices in the right side of the original simulation while exhibiting unnatural jagged edges on the left. Our method cleanly removes these topological artifacts while significantly enhancing the liquid motions.*

the important problem of removing *topological* noise, such as high-resolution cavities and droplets. Figure 3 shows how a thin slot can persist in a fluid simulation as long as the gap is smaller than the width of a fluid cell. Simply smoothing the surface could eventually fix the problem, but not before first removing details from the entire simulation. In contrast, our method specifically targets the unphysical topological structure and rectifies it using either smoothing or splashes.

We found it difficult to directly compare our method with a Laplacian smoothing approach, because the methods behave quite differently and their parameters are not analogous. Qualitatively, we noticed that Laplacian smoothing erodes high curvature regions first and progressively smoothes away larger bumps over time, removing both surface artifacts and surface details without prejudice. Our smoothing method instead selectively erodes artifacts and ignores physically plausible surface features. In addition, our method has a convenient upper limit to the amount of surface smoothing: upon convergence it simply forces the surface tracker to exactly conform to the low-resolution fluid boundary conditions. The limit behavior of Laplacian smoothing, on the other hand, has nothing to do with natural fluid motion.

Figure 6 shows how our method can be used to supplement surface tension dynamics with a level set surface tracker. By computing surface tension waves on the surface tracker instead of the fluid grid, we were able to create surprisingly stable dynamics with good volume conservation properties.

Figure 8 shows how our surface fairing algorithm removes artifacts from a mesh-based surface tracker. The fluid simulation and all simple mesh operations (including our smoothing algorithm) were implemented in parallel, while our topology change code is a serial implementation. Consequently, the topology changes were the most computationally expensive part of this simulation, and the fluid pressure solve was the second most expensive. Our algorithm has negligible overhead for our mesh-based implementation. While our smoothing algorithm successfully removes artifacts from mesh-based simulations, the topology change algorithm of [Wojtan et al. 2010] conflicts with the subtle ripples generated by our wave dynamics algorithm. This is because it identifies high-resolution surface concavities as topological mistakes that must be re-sampled. In the future, we wish to modify this topology change algorithm or switch to a different one (e.g. [Brochu and Bridson 2009]).

Figure 1 shows a highly detailed object interacting with a fluid simulation. Using a high resolution level set without any smoothing successfully preserves the dragon's detailed surface features, but it also preserves many unphysical gashes and interpolation artifacts in the fluid surface. We found it impossible to remove these visual artifacts with Laplacian smoothing unless we also smoothed out the surface details of the dragon. In contrast, our method perfectly preserves the dragon's surface features while it is falling and only smooths them out when they splash around in areas of high pressure. By adding our error-reducing surface waves, we introduce highly detailed surface ripples and Kelvin-Helmholz instabilities while still avoiding unphysical artifacts. Similarly, the original simulation in Figure 9 exhibits several thin sheets and subsequent surface artifacts. As seen in our accompanying video, our method preserves details better than traditional smoothing while simultaneously removing unsightly errors and greatly enhancing the motion.

The most computationally expensive part of each of our level set simulations was the surface advection. The detail in our surface dynamics tends to create a more complicated velocity field, which slows down the conditionally-stable adaptive WENO advection. Biot-Savart calculation was not the bottleneck in our simulations, but it was a significant expense. This is because the velocity has to be evaluated at all cells in the high-resolution narrow band, instead of exactly on the surface. Our employment of relatively small kernel sizes disables lower frequency surface rotations, but it allows for reasonably efficient surface updates. These simulations were dominated by the surface tracker, and our method's complexity increases with surface tracker resolution. Consequently, the level set simulations augmented with our method took about 2.5 longer to simulate than an analogous high-resolution level set without any high-resolution dynamics (though sub-stepping makes our simulation proportionally slower). We believe this computational overhead is acceptable for our goals of removing surface artifacts while adding convincing dynamic details.

Detailed information about our simulations is listed in Table 1. Most of our simulations were run on a standard desktop computer with 8-cores and 64GB of RAM, while the Dragon (LS) and 4-way (LS) simulations were run on a 64-core server with 256GB of RAM.

7 Discussion

Our method solves the problem of removing high-frequency artifacts from a liquid surface tracker by identifying inconsistencies and explicitly removing them. Minimizing all unphysical surface behaviors in one step will not only remove all visual artifacts, but it will also reduce the effective resolution of the surface tracker to exactly match that of the fluid simulation. By spreading out this min-

	Sim	Tracker	Base	Smth	Wave
Dragon (LS)	128^3	512^3	71	200	201
4-way (LS)	128^3	512^3	103	217	220
4-way (LS)	128^3	1024^3	594	—	*3260
4-way (Mesh)	96^3	480^3	106	106	—
Armadillo (Mesh)	96^3	360^3	74	85	—
Cube (LS)	64^3	64^3	50	—	6
Bunny (LS)	128^3	128^3	—	—	98

Table 1: *Summary of timings (in minutes) for our simulations. "Sim" and "Tracker" indicate the simulation and effective surface tracker resolution, respectively. "Base" indicates the high resolution surface tracker with no additional dynamics, "Smth" indicates our smoothing algorithm, and "Wave" corresponds to our wave dynamics. Dashes indicate simulations that were not run. A star indicates that four time steps were used per frame to ensure stability. Effective mesh resolution is based on the ratio between the average mesh edge length and the width of the computational domain. Timings exclude file I/O operations.*

imization process over time, we introduce a physics-based smoothing process. Similarly, by using the errors as a potential energy, we gain surface wave behaviors that specifically remove artifacts. Our method aggressively removes high-frequency surface noise wherever the pressure gradient is large, and it only gradually removes them in regions where the internal fluid forces are smaller. Intuitively, this means that our method will preserve surface details and thin sheets while splashing around in the air, but it will quickly remove noise from standing water. Our wave dynamics will physically over-compensate for sudden changes in the pressure gradient during collisions, which tends to add even more detail to splashes and thin sheets.

The tuning parameter α in our surface fairing intuitively controls how much we will allow the surface to diverge from the low-resolution simulation. Large values ensure consistency between surface and simulation but prohibit details that fall below the simulation grid, such as thin sheets, from developing. Smoothing with too small of an α value may not remove artifacts quickly enough for a high-quality simulation. We set $\alpha = 20$ in all of our examples.

As mentioned in Section 4.2, the vortex sheet equations give us a useful guide for tuning the β parameter. We found that using this setting of $\beta = 2$ usually creates extremely subtle waves that only fill in the gaps of the low-resolution fluid simulation. On the other hand, we are free to artistically tune high-frequency details because our method is not restricted to faithfully reproduce any physical equation. We found that increasing β effectively boosts the sub-grid wave speed, leading to the creation of convincing (though technically unphysical) rolling motions as waves crash down. We used a value of $\beta = 2$ or $\beta = 6$ in all dynamics examples.

Our method is directly guided by the pressure gradient of a fluid simulation. On one hand, this constraint limits the guiding force field to a low resolution. On the other hand, simplified pressure gradients enhance the stability of rotational motion (see the Boussinesq approximation, for example), and our second-order dynamics allow for interesting turbulent motions even with a constant pressure gradient. In addition, allowing our method to be guided by a full fluid simulation causes many of the difficulties in handling solid boundaries to disappear.

We implemented our method in both level set and mesh surface tracking frameworks, and each of these discretizations has its own drawbacks. The level set method is quite robust, but it smooths out desirable details and currently requires a narrow band of at least one fluid cell in thickness to perform the liquid-biased filter. The mesh tracker uses less memory but is more delicate — samples can come very close together during fold-overs and thin sheet formation, resampling the surface is more involved (we use edge subdivisions and collapses as described in [Wojtan et al. 2011]), and robust and efficient handling of topological changes is still an open problem.

One inconsistency between our method and the underlying fluid simulation is that our local rotations enforce incompressibility for both air and water phases, while the fluid simulation ignores conservation of the air's volume. The large scale fluid simulation will freely allow large air pockets to collapse beneath the weight of the liquid, but our surface tracker dynamics preserve the volume of air bubbles smaller than a fluid cell. These air bubbles still experience buoyancy forces though, which cause them to rise up and break through the fluid surface. We found this behavior quite beautiful for the sub-grid fluid dynamics (Section 4.2), but the motion may be undesirable when performing surface smoothing (Section 4.1). We should be able to quickly detect and remove such small bubbles if the behavior is objectionable.

The memory complexity of standard grid-based fluid simulation techniques is proportional to the volume of the simulation, while our algorithm scales only with the surface area. As a result, we were able to achieve very high resolution simulations on a single computer without experiencing memory problems. With our dynamic wave method, we were also able to increase the apparent resolution of a simulation by factors of 4^3 or 8^3 while only requiring a constant factor more computation from the surface tracker. Nevertheless, we would like to further speed up these operations. The stability of the method may be improved by replacing the explicit Euler integration scheme with an implicit one. We may also be able to speed up or sidestep Biot-Savart summations by reformulating our energy gradient in terms of different control variables.

Additionally, we would like to investigate boundary handling in more detail. We are open to new strategies for handling fluid cells that exhibit both free-surface and solid boundary conditions, and we plan to experiment with higher resolution solid boundaries in the future.

Acknowledgments

The authors would like to thank Keenan Crane for helpful discussions about energy gradients, as well as the anonymous reviewers for their helpful feedback on our work.

References

BARGTEIL, A. W., GOKTEKIN, T. G., O'BRIEN, J. F., AND STRAIN, J. A. 2006. A semi-lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics (TOG) 25*, 1, 19–38.

BOJSEN-HANSEN, M. 2011. *A Hybrid Mesh-Grid Approach for Efficient Large Body Water Simulation*. Master's thesis, Aarhus University.

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

BROCHU, T., AND BRIDSON, R. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing 31*, 4, 2472–2493.

BROCHU, T., BATTY, C., AND BRIDSON, R. 2010. Matching fluid simulation elements to surface geometry and topology. *ACM Transactions on Graphics (SIGGRAPH) 29*, 4, 47:1–47:9.

BROCHU, T., KEELER, T., AND BRIDSON, R. 2012. Linear-time smoke animation with vortex sheet meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 87–95.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics (SIGGRAPH) 21*, 3, 736–744.

ENRIGHT, D., NGUYEN, D., GIBOU, F., AND FEDKIW, R. 2003. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proceedings of FEDSM*, vol. 3, 4th.

GOKTEKIN, T., BARGTEIL, A., AND O'BRIEN, J. 2004. A method for animating viscoelastic fluids. *ACM Transactions on Graphics (SIGGRAPH) 23*, 3, 463–468.

HEO, N., AND KO, H.-S. 2010. Detail-preserving fully-eulerian interface tracking framework. *ACM Transactions on Graphics (SIGGRAPH Asia) 29*, 6, 176:1–176:8.

HIRT, C., AND NICHOLS, B. 1981. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of computational physics 39*, 1, 201–225.

HONG, J.-M., AND KIM, C.-H. 2005. Discontinuous fluids. *ACM Transactions on Graphics (SIGGRAPH) 24*, 3, 915–920.

KIM, B., LIU, Y., LLAMAS, I., JIAO, X., AND ROSSIGNAC, J. 2007. Simulation of bubbles in foam with the volume control method. *ACM Transactions on Graphics (SIGGRAPH) 26*, 3, 98:1–98:10.

KIM, D., SONG, O.-Y., AND KO, H.-S. 2009. Stretching and wiggling liquids. *ACM Transactions on Graphics (SIGGRAPH Asia) 28*, 5, 120:1–120:7.

KIM, D., LEE, S. W., YOUNG SONG, O., AND KO, H.-S. 2012. Baroclinic turbulence with varying density and temperature. *IEEE Transactions on Visualization and Computer Graphics 18*, 1488–1495.

LENTINE, M., ZHENG, W., AND FEDKIW, R. 2010. A novel algorithm for incompressible flow using only a coarse grid projection. *ACM Transactions on Graphics (SIGGRAPH) 29*, 4, 114:1–114:9.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics (SIGGRAPH) 23*, 3, 457–462.

MCADAMS, A., SIFAKIS, E., AND TERAN, J. 2010. A parallel multigrid poisson solver for fluids simulation on large grids. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 65–74.

MUSETH, K. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics (to appear) 32*, 3.

OSHER, S., AND FEDKIW, R. 2003. *Level set methods and dynamic implicit surfaces*, vol. 153. Springer.

PARK, S. I., AND KIM, M. J. 2005. Vortex fluid for gaseous phenomena. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 261–270.

PFAFF, T., THUEREY, N., SELLE, A., AND GROSS, M. 2009. Synthetic turbulence using artificial boundary layers. *ACM Transactions on Graphics (SIGGRAPH Asia) 28*, 5, 121:1–121:10.

PFAFF, T., THUEREY, N., AND GROSS, M. 2012. Lagrangian vortex sheets for animating fluids. *ACM Transactions on Graphics (SIGGRAPH) 31*, 4, 112:1–112:8.

POZRIKIDIS, C. 2000. Theoretical and computational aspects of the self-induced motion of three-dimensional vortex sheets. *Journal of Fluid Mechanics 425*, 335–366.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics (SIGGRAPH) 24*, 3, 910–914.

STOCK, M., DAHM, W., AND TRYGGVASON, G. 2008. Impact of a vortex ring on a density interface using a regularized inviscid vortex sheet method. *Journal of Computational Physics 227*, 21, 9021–9043.

THÜREY, N., WOJTAN, C., GROSS, M., AND TURK, G. 2010. A multiscale approach to mesh-based surface tension flows. *ACM Transactions on Graphics (SIGGRAPH) 29*, 4, 48:1–48:10.

WILLIAMS, B. 2008. *Fluid surface reconstruction from particles*. Master's thesis, The University Of British Columbia.

WOJTAN, C., AND TURK, G. 2008. Fast viscoelastic behavior with thin features. *ACM Transactions on Graphics (SIGGRAPH) 27*, 3, 47:1–47:8.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. *ACM Transactions on Graphics (SIGGRAPH) 28*, 3, 76:1–76:10.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2010. Physics-inspired topology changes for thin fluid features. *ACM Transactions on Graphics (SIGGRAPH) 29*, 4, 50:1–50:8.

WOJTAN, C., MÜLLER-FISCHER, M., AND BROCHU, T. 2011. Liquid simulation with mesh-based surface tracking. In *ACM SIGGRAPH 2011 Courses*.

YU, J., AND TURK, G. 2010. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 217–225.

YU, J., WOJTAN, C., TURK, G., AND YAP, C. 2012. Explicit mesh surfaces for particle based fluids. *Computer Graphics Forum (Eurographics) 31*, 2, 815–824.

Appendix: Energy gradient

We wish to compute the partial derivative of the following energy with respect to $\mathbf{n}$:

$$E_{ST} = \int_{\partial\Omega} \mathbf{n} \cdot \nabla p \, dA - \sigma \mathbf{n} \cdot \nabla H \, dA$$

The first term is identical to Equation 1, and the gradient is computed analogously. The second term is a bit more complicated because H is a function of $\mathbf{n}$ — specifically, the mean curvature is equal to the divergence of the normal field. In the following derivation, $\langle \cdot, \cdot \rangle$ notation denotes the inner product, and we use the fact that $\langle f, \nabla \cdot g \rangle = -\langle \nabla f, g \rangle$ from integration by parts on a closed

manifold domain:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{n}}\langle \mathbf{n}, \nabla H\rangle \sigma dA &= \left(\langle \frac{\partial}{\partial \mathbf{n}}\mathbf{n}, \nabla H\rangle + \langle \mathbf{n}, \frac{\partial}{\partial \mathbf{n}}\nabla H\rangle\right)\sigma dA \\
&= \left(\nabla H + \langle \mathbf{n}, \frac{\partial}{\partial \mathbf{n}}\nabla(\nabla\cdot\mathbf{n})\rangle\right)\sigma dA \\
&= \left(\nabla H - \langle \nabla\cdot\mathbf{n}, \frac{\partial}{\partial \mathbf{n}}(\nabla\cdot\mathbf{n})\rangle\right)\sigma dA \\
&= \left(\nabla H + \langle \nabla(\nabla\cdot\mathbf{n}), \frac{\partial}{\partial \mathbf{n}}\mathbf{n}\rangle\right)\sigma dA \\
&= \left(\nabla H + \langle \nabla H, \frac{\partial}{\partial \mathbf{n}}\mathbf{n}\rangle\right)\sigma dA \\
&= \left(\nabla H + \nabla H\right)\sigma dA \\
&= 2\sigma\nabla H dA
\end{aligned}$$

Therefore:

$$\frac{\partial}{\partial \mathbf{n}}E_{ST} = \left(\nabla p - 2\sigma\nabla H\right)dA$$