DOI: 10.1111/j.1467-8659.2012.03062.x
EUROGRAPHICS 2012 / P. Cignoni, T. Ertl
(Guest Editors)

Volume 31 (2012), Number 2

Explicit Mesh Surfaces for Particle Based Fluids

Jihun Yu[1], Chris Wojtan[2], Greg Turk[3] and Chee Yap[4]

[1]Industrial Light and Magic, [2]IST Austria
[3]Georgia Institute of Technology, [4]New York University

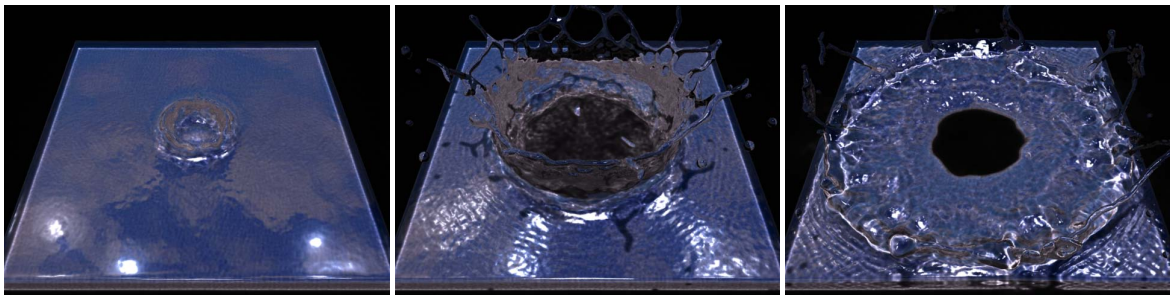

Figure 1: *A drop falling into a shallow pool creates a water crown.*

Abstract

We introduce the idea of using an explicit triangle mesh to track the air/fluid interface in a smoothed particle hydrodynamics (SPH) simulator. Once an initial surface mesh is created, this mesh is carried forward in time using nearby particle velocities to advect the mesh vertices. The mesh connectivity remains mostly unchanged across time-steps; it is only modified locally for topology change events or for the improvement of triangle quality. In order to ensure that the surface mesh does not diverge from the underlying particle simulation, we periodically project the mesh surface onto an implicit surface defined by the physics simulation. The mesh surface gives us several advantages over previous SPH surface tracking techniques. We demonstrate a new method for surface tension calculations that clearly outperforms the state of the art in SPH surface tension for computer graphics. We also demonstrate a method for tracking detailed surface information (like colors) that is less susceptible to numerical diffusion than competing techniques. Finally, our temporally-coherent surface mesh allows us to simulate high-resolution surface wave dynamics without being limited by the particle resolution of the SPH simulation.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Within the field of computer graphics, there is a diverse set of different techniques for animating liquids. Each technique has its own merits and drawbacks: Eulerian techniques can produce very realistic animations, but they are computationally expensive and unsuitable for simulating certain effects like spray and foam. Shallow water discretizations run at real-time rates, but they are restricted to height fields. Particle-based simulations, like smoothed particle hydrodynamics (SPH), easily retain volume and momentum, and they have been used in a wide range of applications like liquids, deformable solids, multi-phase fluids, viscoelastic materials, controlled liquids, and porous flow. However, these particle-based methods traditionally produce noisy and unrealistic surfaces.

The standard technique for producing a surface for a particle simulation involves the creation of an implicit surface that essentially wraps around all of the particles in the simulation. While recent research has greatly improved the visual quality of the surface that results from such an implicit formulation, this implicit surface strategy generally has difficulty producing and retaining high resolution surface details like waves, textures, and ripples.

We introduce a new *explicit* method for tracking the surface of a particle-based fluid simulation, which overcomes many of these difficulties. Our surface tracking approach be-

gins by constructing a surface mesh for the fluid at the first time step in the simulation. This initial mesh is an isosurface from a traditional implicit surface representation. In subsequent time steps the old mesh vertices are advected using the velocity of the fluid particles. We then project the new vertex locations back onto the implicit surface to ensure that numerical errors do not accumulate to cause the surface to diverge from the simulation particles. Where the mesh has been stretched or compressed, we perform edge splits or collapses in order to improve the triangle shapes, and when the fluid surface splits or merges we use a robust technique for changing the mesh topology.

This mesh surface gives us several advantages over previous particle surface tracking techniques. First, we demonstrate a new method for surface tension calculations that yields higher quality results than prior methods for SPH surface tension for computer graphics. We also introduce an intuitive method for tracking detailed surface information (like colors) that is less susceptible to numerical diffusion than competing techniques. Finally, our temporally-coherent surface mesh allows us to simulate high-resolution surface wave dynamics without being limited by the number of particles in the particle simulation. The contributions of our paper are as follows:

- We introduce the first *mesh-based* surface tracker for a particle-based simulation.
- Our method allows for accurate tracking of surface data like colors and textures that is virtually free of common artifacts like numerical diffusion.
- We present a new surface tension model that outperforms the state of the art in SPH for computer graphics.
- We introduce a new method for adding subtle ripple details to an animation that has *already been simulated* — this is the first technique that adds surface tension dynamics as a post-process to an already-computed simulation.

2. Related Work

2.1. Surface Tracking

A large body of techniques in surface tracking have been developed by researchers in recent years. The levelset method [OS88] has been successfully applied to track the free surface of a fluid in an Eulerian simulation scheme. To overcome volume loss and the blurring of features of the basic levelset method, Enright et al. [EFFM02] introduced a particle levelset method that advects explicit particles along with an implicit signed distance function. Lossaso et al. [LGF04] proposed an adaptive simulation method that captures fine surface details by using an octree data structure to increase the simulation resolution. Bargteil et al. [BGOS06] introduced the semi-Lagrangian contouring method. They create a new mesh by advecting a signed distance field to maintain a more accurate surface representation, and their method can track free surfaces of liquid along with surface properties such as colors and texture maps.

While these implicit surface techniques easily handle changes in surface topology, the size of the features that they can represent is limited by a grid resolution. To alleviate this limitation, researchers have turned to explicit tracking methods. In CFD literature, Hieber and Koumoutsakos [HK05a] introduced a Lagrangian particle levelset method that uses discrete particle samples for representing and advecting a signed distance field. In the computer animation literature, researchers have used approaches that maintain and advect a triangle mesh along with the fluid velocity field. The method presented by Müller [MÖ9] globally re-samples the surface using a marching cubes grid, but then retains previous mesh samples in order to preserve fine surface features. Misztal et al. [MBE*10] explicitly track fluid surfaces on a triangle mesh which is embedded as a subcomplex of a deformable tetrahedral mesh. Their tetrahedral grid is locally re-meshed in the region where the surface mesh components collide. Bridson [BB09] use a mesh surgery technique that is purely based on geometric intersections. They later combined this with an Eulerian simulation in which pressure samples are placed in order to capture fine surface geometry [BBB10]. To handle changes in topology when using a mesh surface representation, Du et al. [DFG*06] and Wojtan et al. [WTGT09] use local re-meshing techniques in regions where the mesh topology differs from that of an isosurface representation of the fluid interface. In [WTGT10], a local convex hull procedure is used for local re-meshing to preserve thin fluid features. The mesh-based surface tracking in our research makes use of techniques from [WTGT09] and [WTGT10].

Surface tracking is difficult in particle-based fluid simulation schemes, because particles do not retain any connectivity information. Researchers have developed a number of methods for reconstructing surfaces from such particle simulations. The blobby sphere approach was introduced by Blinn [Bli82], and uses a sum of isotropic Gaussian basis functions to construct a surface. Zhu and Bridson [ZB05] defined a new implicit surface model by averaging particle locations and their radii to enhance the surface smoothness of the blobby method. Adams et al. [APKG07] improved Zhu's method by tracking the particle-to-surface distances across simulation time steps. The approach of Williams [Wil08] smooths an explicit mesh extracted from an implicit function by applying Laplacian and bi-Laplacian smoothing. Museth et al. [MCZ07] blurred implicit metaballs according to anisotopic diffusion. Later, Yu and Turk [YT10] used an anisotropic kernels and Laplacian smoothed particle positions to define a smoothed implicit surface. Recently, Bhattacharya et al. [BGB11] improved the method of Williams [Wil08] by minimizing thin-plate energy on an implicit level-set surface instead of smoothing an explicit mesh.

One limitation of these aforementioned surface reconstruction methods for particle based fluids is that carrying surface properties on the surface from one frame to a next frame is not straightforward, because a new surface mesh is constructed at each rendering step. In contrast, our new approach facilitates surface tracking by advecting an explicit mesh surface at each frame.

An important prior method for tracking the surface of particle-based implicit surfaces is the active surface approach of Desbrun and Cani [DCG98]. Like active contours from computer vision, they create a surface that is attracted to a moving zero set of an implicit function, but that can also be affected by other forces such as surface tension. Their surface representation is a discretely sampled implicit field on a regular grid. The advection field is determined by the difference between a target field and a current field. Because they use Marching Cubes to create a polygonal surface on a per-frame basis, it is not clear how their method could be used to carry additional quantities such as colors on the surface. They demonstrated surface smoothing due to surface tension forces, but did not give examples of capillary waves.

Recently, Stam and Schmidt [SS11] proposed a new surface tracking approach for time-evolving implicit surfaces. They uniquely defined a velocity field on implicit surfaces over time by enforcing a constraint on the evolution of the normal field. They apply this approach to track a moving surface and to motion blur moving implicit surfaces.

In addition to surface front tracking, special techniques have been developed for tracking surface characteristics such as colors, texture coordinates and physical properties. Mihalef et al. [MMS07] used surface marker particles for carrying color information and enhancing the level-set resolution. Texture mapped particles are advected through the fluid velocity field in [REN*04], while texture color and local orientation is advected and used as a constraint to synthesize temporally coherent textures in [KAK*07] and [BSM*06]. Bargteil et al. [BGOS06] showed that semi-Lagrangian surface tracking can be used to carry surface properties during a simulation.

2.2. Surface Tension

Surface tension forces have been applied in a number of Eulerian grid simulation methods. Kang et al. [KFL00] estimate surface curvature from a levelset function in order to produce surface tension forces. Lossaso et al. [LGF04] calculated the surface tension forces at free surfaces more accurately by employing an octree structure. Hong and Kim [HK05b] treat surface tension effects as discontinuous boundary conditions at the interface. Using an explicit surface representation, discrete curvature operators such as that of Desbrun et al. [DMSB99] can be used to compute accurate surface tension forces. Surface tension effects are formulated as an optimization approach to surface energy in [MBE*10, EMB11]. Brochu et al. [BBB10] added a discontinuous pressure jump based on the mean curvature of the surface. The method of Thürey et al. [TWGT10] computes the surface tension forces from a mesh using volume preserving mean curvature flow, and these forces are used as a boundary condition to the grid-based pressure solve.

There have also been several approaches to applying surface tension forces in particle-based simulations. Müller et al. [MCG03] approximate the surface tension force as the divergence of the surface normal field. Clavet et al. [CBP05] used a double density relaxation to achieve effects similar to surface tension. Hu and Adams [HA06] discussed a surface tension model multi-phase SPH simulation, and Becker and Teschner [BT07] proposed a molecular cohesive force approach for surface tension effects. Zhang [Zha10] detects boundary particles and measures the curvature from a local surface representation constructed from moving least squares (MLS). Andersson et al. [AJM*10] use the Radial Basis Function (RBF) framework to define a new implicit color filed that yields a smooth surface. Then they apply surface tension forces to particles by evaluating the curvature along the surface.

All previous methods in particle-based simulations use particle samples to generate tension effects. Consequently, the particle resolution explicitly limits the resolution of the tension force. Our approach is able to push beyond this limit by combining a standard SPH solver with a detailed explicit surface mesh.

2.3. Dynamic Surfaces

One method of creating finely detailed surfaces at a low computational cost is to couple a high resolution surface representation with a lower resolution solver for the bulk of the fluid. Goktekin et al. [GBO04] and Kim et al. [KSK09] couple a high resolution particle levelset with the low resolution fluid solver. Bargteil et al. [BSM*06] used an octree contouring method that is coupled with the uniform grid fluid solver. Sifakis et al. [SSIF07] embedded high resolution particle samples on the simulation mesh, and the low resolution finite element solver of [BWHT07] is coupled with a detailed surface mesh in [WT08].

Our surface dynamics approach has commonalities with the work of Tessendorf [Tes02], Wang et al. [WMT07] and Thürey et al. [TWGT10]. Tessendorf simulates ocean surface waves on a height field using Fast Fourier Transform (FFT) approach. Wang et al. solve a general shallow water equation on a surface mesh, while Thürey et al. use a dynamic surface mesh to solve the wave equation. In our approach, capillary waves of variable speed are generated on the mesh by minimizing thin plate energy. We use a per-vertex displacement along the normal of the surface mesh in order to create surface dynamics.

3. Surface Tracking

3.1. Mesh Advection and Topology Changes

Our new approach to surface tracking maintains an explicit mesh that is consistent with an isosurface defined by the position of the SPH particles. The SPH particles define an implicit function $\phi(x)$ that is positive for values inside of the surface and is negative for outside values, and our isosurface is extracted as a zero level-set of $\phi(x)$. We use the isosurface in two ways: 1) to create the initial surface mesh, and 2) as a surface to project mesh vertices onto during subsequent time steps. We use the anisotropic kernel method of Yu and Turk [YT10] to define our isosurface based on the particle positions. This method is similar to defining an isosurface based on the sum of per-particle radial basis functions, but

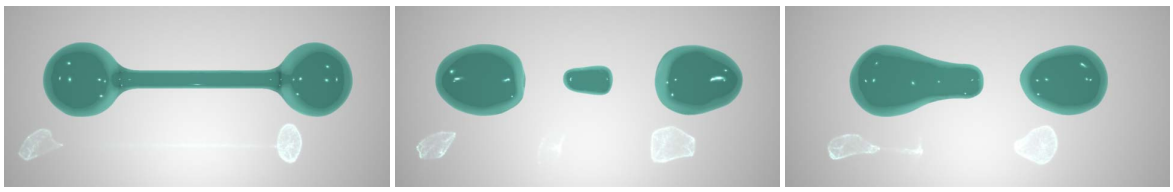

Figure 2: *A dumbbell in zero gravity exhibits pinch-off due to surface tension effects (Rayleigh-Plateau instability). Later in this animation the separate components re-join each other.*

differs in that each basis function may be stretched according to the particle distribution in a local neighborhood. The anisotropic kernel method yields an isosurface that is quite close to the center of SPH particles that are on the boundary, and this helps us to interpolate boundary particle velocities onto the mesh vertices.

Our method begins by constructing an initial surface mesh by applying Marching Cubes to the isosurface of $\phi(x)$. Once this initial mesh is created, our surface tracking method consists of repeating the following four tasks for each time step:

1. Advect the mesh vertices according to SPH particle velocities.
2. Improve triangle shapes using edge split and edge collapse operations.
3. Project the mesh vertices onto the implicit surface.
4. Perform topological changes when the fluid merges or splits apart.

To prepare for the advection of a surface mesh vertex at a given time step, we retrieve all of the simulation particles within the SPH smoothing radius of the vertex using a spatial hash grid. We calculate normalized weights for each of these selected particles by evaluating SPH's smoothing kernel on their distances to the vertex. The vertex velocity is given by summing the weighted velocity of these neighborhood particles. Therefore, the velocity of vertex i is given by:

$$\mathbf{v}_i = \sum_j W_{ij}\mathbf{v}_j / \sum_j W_{ij}, \tag{1}$$

where W_{ij} is the SPH smoothing kernel evaluation on a distance between vertex i and particle j, and $\mathbf{v}_j$ is the velocity of particle j. For all of our simulation examples, we use the B-cubic spline kernel from [BT07].

Then we advect both the surface mesh and the SPH particles. We chose symplectic Euler as our numerical advection scheme because it is stable, energy preserving, and simple to implement. We also perform edge collapses and edge splits at the end of each advection step to maintain a high mesh quality. We define an average edge length l_{avg} relative to the particle spacing. We split edges when they are longer than $2l_{avg}$ and collapse edges when they are shorter than $0.5l_{avg}$. We also perform edge collapses if the triangles have very small angles (less than $\pi/30$), or the dihedral angles between triangles are more than a $\pi/3$ deviation from being flat.

Although the advection step keeps the surface vertices near to the isosurface, accumulated numerical errors may cause inconsistency between the explicit mesh and the isosurface. In addition, topological changes to the fluid such as merging and splitting will produce surface vertices that are off the isosurface. Our projection step is designed to keep the surface mesh closely matched to the isosurface. In addition, it is during this projection step that we detect the regions where topological changes occur (to be discussed below). For an updated vertex v_i at position x_i after the advection step, we project the vertex onto the isosurface of $\phi(x)$ by performing a binary search along the ray segment from x_i to $x_i + \varepsilon n_i$ where n_i is the vertex normal obtained from the discrete mean curvature computation. The value of ε is $sign(\phi(x_i)) \cdot r_a$ where $\phi(x_i)$ is the evaluation of the isovalue at v_i and r_a is the average particle spacing. When $\phi(x_i) \cdot \phi(x_i + \varepsilon n_i) \leq 0$, we refine the interval until the new vertex is located where the isovalue is close enough to zero or until the maximum number of iterations has been performed. For all examples in the paper, we use four iterations and the value of 0.001 for the isovalue threshold.

This projection step will fail if $\phi(x_i) \cdot \phi(x_i + \varepsilon n_i) \geq 0$, because we cannot locate the zero isovalue position along the ray. This condition signals a topological change — the isosurface of $\phi(x)$ has either merged or split, so we need to merge or split the surface mesh as well in order for it to remain consistent with the simulation. Topological merges may produce vertices inside the fluid volume with a positive isovalue, while topological splits may produce vertices outside the fluid volume with a negative isovalue. For both cases, we first move v_i to $x_i + \varepsilon n_i$ in case $|\phi(x_i + \varepsilon n_i)| < |\phi(x_i)|$. We also set a topology change flag that indicates that we need to identify and fix such local changes. We then use the method of Wojtan et al. [WTGT09] to repair the topology of the mesh.

We refer the reader to [WTGT09] for details of resolving topology changes of the mesh, but we briefly outline the steps here. First, we calculate a grid-sampled signed distance field for the surface mesh. This signed distance field (and not the implicit function $\phi(x)$) will be used to recognize regions that require changing the mesh. We then mark each cell in which the eight corner samples of the distance field disagrees with the mesh in terms of topology. The surface mesh is clipped to these cells, and the polygons inside such cells are removed. New triangles are created using Marching Cubes inside these marked cells, and these triangles are sewn together with the mesh at the cell faces. This approach handles most of the topology change mesh events, including surface merging, surface splitting and other self-intersection events at the resolution of the grid.

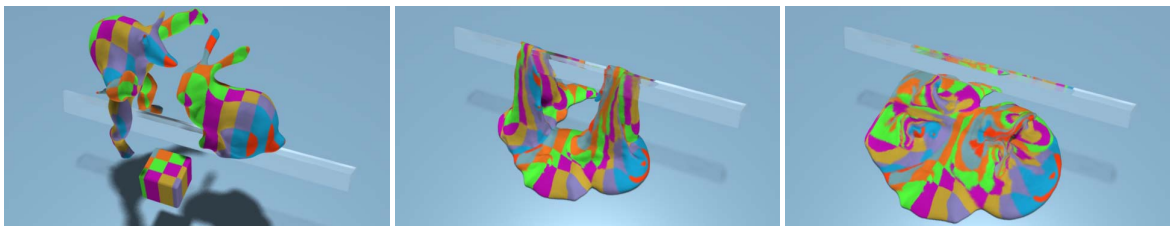

Figure 3: *Three viscous figures with surface colors are dropped on a bar.*

The topological event that the above procedure does not handle is a sub-grid scale splitting event, as occurs with a droplet pinch-off. We treat this case as in [WTGT10], during edge collapse operations. When a short edge is identified for collapse, we first check to see whether the edge is part of a thin tunnel surrounded by a ring of three edges that would be flattened if the edge were to be collapsed. Such a case indicates a topological split, and we do not perform the edge collapse. Instead, we replicate the vertices along the ring of edges, separate the mesh into two parts at this ring, and cap off these rings with one triangle each.

While we chose to use the explicit mesh tracking method of [WTGT09] in this paper, the method of [BB09] is a valid alternative. The main functional difference between these methods is the strategy for handling topological changes: [WTGT09] efficiently changes topology based on underlying implicit surface, while [BB09] uses collision detection and proximity cues. Because the method of [WTGT09] allows the use of a high resolution mesh with the same topology of a lower resolution isosurface, it is ideal for our application (tracking detailed surface geometry while preserving the connectivity of a lower resolution particle simulation).

It is worth mentioning that by projecting the surface mesh onto the isosurface, we sacrifice one benefit of using an explicit mesh — that it can easily represent thin features. We could easily allow these thin sheets if we advect the surface mesh without projection. Unfortunately, ignoring the projection would lead to drift in the simulation (either the surface drifting too far away from the particles, or the particles drifting far outside of the surface mesh). An alternative way to correct this drift is to re-sample the particles inside the surface mesh as is proposed in [AT11]. However, continual re-sampling negates the Lagrangian benefits of particle-based fluids. In graphics literature, fairing the Lagrangian surface with the evolving isosurface is often addressed by the projecting approach: Witkin and Heckbert [WH94] projected the velocity of surface particles, and Stam and Schmidt [SS11] projected the surface mesh in order to prevent the mesh drifting away from the isosurface. In our approach, we also chose to project the surface mesh onto the isosurface in order to remain faithful to the Lagrangian simulation.

3.2. Surface Property Advection

By maintaining the explicit surface mesh representation, we are easily able to track surface quantities such as colors, textures or physical properties. We choose to carry surface quantities on the mesh vertices because the interpolation and extrapolation schemes at vertices are straightforward. As the surface mesh is advected, a local re-meshing happens when we perform edge operations or topological changes.

When a new vertex is created by the edge split operation, the properties in one-ring neighbor vertices are linearly averaged according to the edge length weight. When two vertices are merged by the edge collapse operation, two properties are averaged on a new vertex. In the event of topological changes, we remove local invalid mesh patches and replace them with newly created correct topology triangle patches. To assign surface properties to the newly created vertices, all new vertices connected to an original mesh vertex (one that was not replaced by the topological operation) are assigned extrapolated properties from the original vertices and pushed into a queue. Then we propagate the properties of the original vertices to the interior of the patch by iteratively popping the vertex at the front of the queue, extrapolating its properties onto the unvisited one-ring neighbor vertices, and then pushing the neighbors into the queue.

As an alternative to this flood-fill approach, we could solve for a Harmonic or Biharmonic equation that interpolates properties from the original mesh vertices onto the new ones. Although a Biharmonic interpolant will produce very smooth data, we found that our flood-fill approach is simple and accurate enough for the demonstrations in this paper, especially because the size of newly created patches is usually fairly small.

The main benefit of tracking surface quantities using the explicit mesh is that we introduce little dissipation of the quantities over time. Most mesh vertices are carried across many time steps undisturbed, and thus the properties at these vertices remain untouched. This is in contrast to mesh-based surface tracking methods that perform global re-meshing at each time step.

Another benefit of keeping an explicit representation is that we can delay the quantity tracking and perform it as a post-process. Instead of transferring surface quantities during the simulation, we can make a separate sequence by storing each quantity transferring process per-timestep. That is, for each newly created vertex we make a list of the influencing vertices and their weights. Once we build this transferring sequence, we can track different quantities on the identical simulation. One application of this post-processing approach to tracking surface properties is the generation of surface waves on a mesh sequence. We refer readers Section 6 for the details of this method.

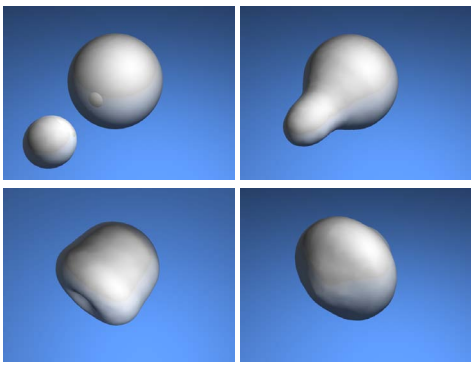

Figure 4: *Two spheres that merge and exhibit surface waves.*

4. Surface Tension Model

We apply surface tension to the SPH particles as a body force. The curvature normal at each mesh vertex is computed by applying the discrete mean curvature operator on the mesh. We then create a per-vertex surface tension force and transfer these forces to the SPH particles. To do this, we locate the nearby mesh vertices within the SPH smoothing radius of each particle p_i. Then the surface tension force $\mathbf{f}_i$ is given by:

$$\mathbf{f}_i = \gamma \sum_j W_{ij} A_j \kappa_j / \sum_j W_{ij} A_j, \tag{2}$$

where γ is the surface tension coefficient, W_{ij} is the SPH smoothing kernel evaluation on a distance between particle i and vertex j, and κ_j is the mean curvature normal of vertex j. The value A_j is the area of the mesh closest to vertex j, that is, A_j is one-third of the sum of the area of the triangles adjacent to j. We use the shape operator of Meyer et al. [MDSB02] to calculate the curvature κ_j from the one-ring of triangles around vertex j. We use area-based weights to generate the surface tension forces of Equation 2 since small area triangles are often obtuse and Meyer's approach is unstable for such poorly-shaped triangles. By weighting these curvature estimates using triangle areas, we are able to attenuate the noise caused by these cases.

There have been prior methods for calculating surface tension forces for SPH simulations, including the double density relaxation method [CBP05] and the molecular cohesive force approach [BT07]. In these formulations, the surface tension is rather an emergent feature and the tension force is not induced from the mean curvature of the fluid surface. In contrast, our surface tension model captures the detailed curvature from the explicit surface representation and the tension force is explicitly induced from the fluid surface. The color field approaches to SPH surface tension [MCG03], [HA06] suffer from numerical errors caused by irregular particle samples. In contrast, our approach is unaffected by the particle distribution because the curvature information on the surface is computed from the high resolution mesh representation. We refer readers to Section 6 for the comparison of our approach with the molecular cohesive force approach.

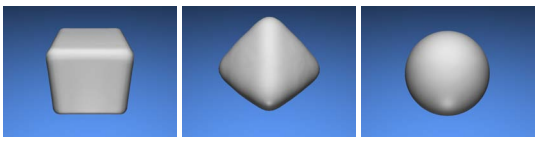

Figure 5: *A cubic water drop that oscillates and settles into a sphere.*

5. Surface Physics

Our mesh-based surface tension induces forces on the SPH particles and causes smoothing of the fluid that is characteristic of small-scale flows. There is, however, another phenomena due to surface tension that would be computationally prohibitive to simulate using forces on particles, namely capillary waves. We take a different approach to simulating this phenomena, and in particular we simulate this form of wave propagation directly on the surface mesh. In order to do this, we treat these waves as displacements from a base mesh in the normal direction. These displacements are properties that we store directly on the mesh vertices, much like color values.

We perform our surface wave physics on a dynamic surface mesh $\mathbf{D}$ that is constructed from a base surface mesh $\mathbf{B}$ that represents the fluid surface. Initially, $\mathbf{D}$ is just a duplicate of the base mesh $\mathbf{B}$. At each simulation step, we compute new positions of $\mathbf{D}$ by performing surface wave dynamics. These updated displacements of $\mathbf{D}$ are stored back on $\mathbf{B}$ and carried to the next step by advecting $\mathbf{B}$. Similar to the wave equation model used in [WMT07] and [TWGT10], we constrain the vertices of $\mathbf{D}$ to move along the normal rays of the corresponding vertices of $\mathbf{B}$. We store the height scalar h and a scalar representing the normal component of the velocity v on the corresponding vertices in $\mathbf{B}$. By maintaining the configuration of $\mathbf{D}$ as scalar quantities on the vertices of $\mathbf{B}$, tracking $\mathbf{D}$ is simple under topological events or edge operations. That is, we treat h and v as surface quantities (similar to color) and assign influence values on the newly created vertices using the method described in Section 3.2

We use a thin-plate bending energy model that is similar to [BWH*06] for our surface wave simulation. We first describe our model on a smooth surface $\mathbf{S}$ in a continuous setting, and later describe its discretization on the mesh $\mathbf{D}$.

Continuous formulation: The bending energy E_b of a smooth surface $\mathbf{S}$ is formulated as

$$E_b(S) = \frac{1}{2} \int_S \|\Delta \mathbf{x}\|^2 dA, \tag{3}$$

where $\|\cdot\|$ is the l^2 norm and Δ is the Laplace-Beltrami operator. In addition to E_b, we introduce a shape constraint energy E_s in order to prevent excessive deformation and volume loss of $\mathbf{D}$ with respect to $\mathbf{B}$ under bending energy minimization. The energy E_s between a surface $\mathbf{S}$ and its deformation (corresponding to $\mathbf{B}$ in a discrete setting) under a homeomorphism h is defined as

$$E_s(S) = \frac{1}{2} \int_S \|\mathbf{x} - h(\mathbf{x})\|^2 dA. \tag{4}$$

The wave dynamics of $\mathbf{S}$ is determined by both conservative

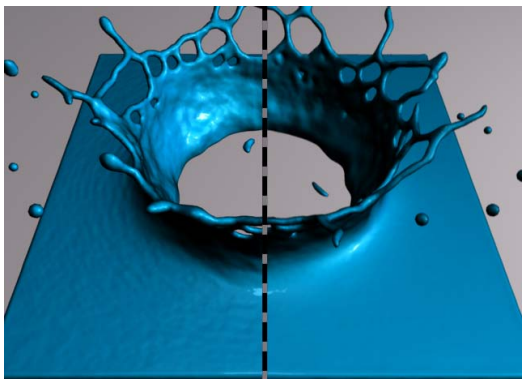

Figure 6: *A water crown augmented with capillary waves as a post-process (left), and the original water crown (right).*

and dissipative forces — the conservative behavior is governed by the gradient of total energy and the dissipation is dependent upon the velocity of the surface. The force f at $\mathbf{x}$ on $\mathbf{S}$ is given by

$$f_s(\mathbf{x}) = -\frac{k_b}{2}\nabla_{\mathbf{x}} E_b(S) - \frac{k_s}{2}\nabla_{\mathbf{x}} E_s(S) - k_d m(\mathbf{x})\dot{\mathbf{x}}, \quad (5)$$

where k_b, k_s and k_d are coefficients for bending, shape and damping respectively, and $m(\mathbf{x})$ is a point mass of S at $\mathbf{x}$ and we use the value dA for our discretization.

Numerical discretization: We discretize bending energy as a weighted surface bi-Laplacian as described in [Wil08]. Given a dynamic mesh $\mathbf{D}$ with n vertices, let W be a $3n \times 3n$ matrix representing angle cotangent weights, let A be a $3n \times 3n$ diagonal matrix representing scaled vertex areas, and let X be a $3n \times 1$ vector representing vertex locations, so that the discretized curvature normal is expressed as $A^{-1}WX$. We approximate bending energy as

$$E_b(D) = k_b X^T (W^T A^{-1} W) X. \quad (6)$$

By assigning lumped mass on the surface, the discrete shape energy is formulated as

$$E_s(D) = k_s (X - X_b)^T A (X - X_b), \quad (7)$$

where X_b is a vector representing vertex locations of the base mesh $\mathbf{B}$. During the surface dynamics integration, X_b remains unchanged to provide a reference mesh location. We formulate the momentum equation by

$$\ddot{X} = -k_b (A^{-1}W)^2 X - k_s (X - X_b) - k_d \dot{X}, \quad (8)$$

that is obtained by discretizing (5) using (6) and (7). Note that we linearize W and A as constants in our system by recomputing them at every iteration as $\mathbf{D}$ is updated, although they are inherently nonlinear in $\mathbf{X}$. We use a symplectic Euler (also known as Euler-Cromer) method for our time integration rule to ensure that our capillary waves propagate over long distances and do not dissipate too early. For each SPH simulation step, we perform $5 \sim 10$ iterations of wave dynamics. The wave frequency is proportional to k_s and the wave height is proportional to k_b.

Our surface wave dynamics model is governed by bi-Laplacian of the surface. In the continuum limit, the bi-Laplacian flow smooths the surface while preserving the original volume as shown in [XPB06]. The bi-Laplacian has also been applied for extracting smooth surfaces of particle based fluids in [Wil08] and [BGB11]. Although our discrete model does not guarantee exact volume preserving, our approach simulates physically plausible waves by characterizing the smoothing and the volume preserving nature of the real waves. Compared to the method of Thürey et al. [TWGT10], their surface tension waves are modeled by solving a wave equation on the surface. As mentioned in their paper, this linearized approximation is limited by an user-specified constant wave speed, while natural capillary waves experience a wave speed dependent on the wave number. Our method generates more complex behavior by considering the full 3D geometry of the surface mesh, instead of basing its calculations on a simplified height field. By directly modeling bending energy instead of the linearized wave equation, we can generate small waves at varying speeds. In our experience, these factors produce waves that are visually less disturbing; however a full experimental and theoretical analysis of our model's dispersion behavior is beyond the scope of this paper.

Our surface dynamics model can have numerical issues if the surface parametrization introduces bad triangulations. However, as described in Section 3.1, we clean up all the flipped and degenerate triangles on the mesh by applying the topology repairing step along with edge collapses. Therefore, solution of our surface dynamics remains faithful on the high quality mesh.

We found that the bending force becomes noisy when $\mathbf{D}$ is displaced more than the average edge length from $\mathbf{B}$ due to poorly-shaped triangles. To maintain a higher mesh quality, we project the location of vertices in $\mathbf{D}$ onto the vertex normal ray of the corresponding vertices in $\mathbf{B}$. As long as these displacements are small enough, this is justifiable since the volume preserving curvature flow runs along the normal direction. Taming the wave height can be done by clamping the height above a fixed limit. We also turn off surface waves for small features and high curvature regions based on the curvature information of the base mesh $\mathbf{B}$.

6. Results

We have used our mesh-based surface tracker to produce several SPH animations, and we describe them in this section. Our simulations were run single-threaded on an Intel Xeon E5620 workstation with two 2.4 GHz processors and 6 GB of main memory. The simulation code was written in C++, and the images were rendered using Maya, except in the case of Figures 3, 7 and 9 which were rendered with Gelato. Please watch our video to see these animations.

Figure 2 shows an example of a Raleigh-Plateau instability. The initial condition for this example is a dumbbell in zero gravity. Due to the surface tension forces (induced by the surface mesh), portions of the bar become thin and eventually pinch off. This illustrates that our mesh surfaces can

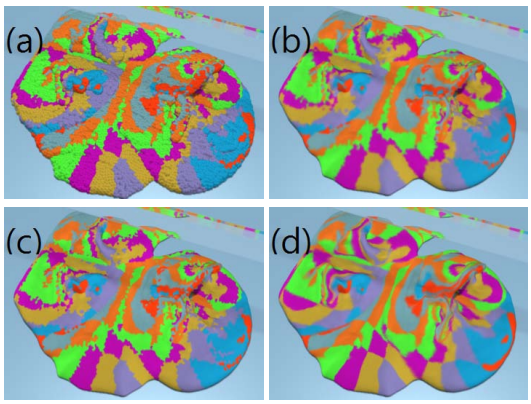

Figure 7: *Comparison between different surface color tracking approaches on the viscous figures animation. (a) per-particle colors, (b) distance-weighted average from particle colors, (c) nearest neighbor colors from particles, (d) our mesh-based method.*

undergo topological splits. As can be seen in our video, the separation of these components also induce ripples on the surface of the drops. In this example, these ripples are purely based on the positions of the SPH particles without use of mesh-based dynamics. This simulation used 7,279 particles.

Figure 3 shows our method for carrying properties such as color on the surface of an object that is represented using particles. In this sequence, three highly viscous figures (a bunny, a cube, and an armadillo) are dropped. The bunny and the armadillo spill over both sides of a bar, and they all pool onto the floor. We use the vertices of our surface mesh to carry color information for each of these three objects. We use an average edge length that is one quarter of the average inter-particle distance, and this allows us to carry a more fine resolution of color information than per-particle colors. Our method blends color values when a mesh triangle becomes overly stretched, at which time we subdivide the triangle and use interpolation to create a color value at the new vertices. We also blend colors during topological changes to the mesh. The surface of this material in this example is heavily stretched, yet the colors do not blend together at their common borders. This simulation used 29,172 particles.

Figure 7 shows how our method of carrying colors compares to the logical alternatives. Part (a) of this figure shows a color-per-particle view, which is the information used for parts (b) and (c). Part (b) uses a distance-weighted average of per-particle colors. Part (c) demonstrates coloring the surface mesh based on the color of the nearest neighbor particle. Part (d) shows our mesh-based method of carrying colors. When blending per-particle colors, as in part (b), we must choose a maximum blending distance d. In part (b) of this figure, $d = s$, where s is the average inter-particle distance. In the accompanying video, we show results both for $d = s$ and $d = 2s$.

Figure 8 demonstrates the importance of projecting the mesh vertices onto the moving implicit surface. The left image shows that when this projection is not performed, the

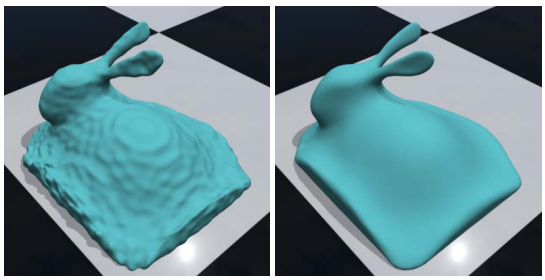

Figure 8: *Comparison between simulation with no projection (left) and projection of the surface mesh onto the moving implicit function (right).*

mesh quickly develops wrinkles in the surface. When projection is used, as shown on the right image, no such wrinkles form. As second example, included only in our video, shows that if projection is not used, an unnatural bridge is maintained between two separating particles.

The simulation of Figure 4 demonstrates two spheres of water merging in zero gravity. This animation was produced using 12,532 particles. When the drops merge, this causes ripples on the water's surface. Along with the simulation, we added surface dynamics to these drops based on the surface bi-Laplacian. To see the difference between using no surface dynamics and incorporating surface dynamics, please see the accompanying video.

Figure 5 shows the effect of surface tension on a cube of water in zero gravity, inspired by the example of [BBB10]. Our mesh-based surface tension rounds off the corners of the cube and causes the bulk of the fluid to be drawn to a more spherical shape. Due to the momentum of the fluid, the sphere shape is overshot and the fluid moves to a nearly octahedral shape before being drawn back towards the sphere. As can be seen in the accompanying video, this oscillation between the cube and octahedron occurs several times before the fluid settles into a sphere. In the video, we compare our results to the method of Becker and Teschner [BT07]. Their particle-based surface tension approach also draws the cube towards a sphere, but this motion is considerably more damped than the motion from our mesh-based surface tension forces. This simulation used 4,096 particles.

Our final example is a water crown that was created by a falling drop of water hitting the surface of a still pool (Figure 1). Note that this example demonstrates numerous topological changes to the surface, and these are handled by the local topology repair method. In Figure 6, we show opaque renderings from two versions of this animation, one without surface dynamics and one that adds surface ripples using our surface dynamics post-process. Note that in the version without the capillary ripples the surface around the water crown looks unnaturally flat. The images in Figure 1 include the surface dynamics for added realism. This was our largest simulation, and it required 200,000 particles to simulate.

Table 1 shows the per-frame computational costs for the various components of our simulations. Note that the cost of the surface tracking is proportional to the surface area of

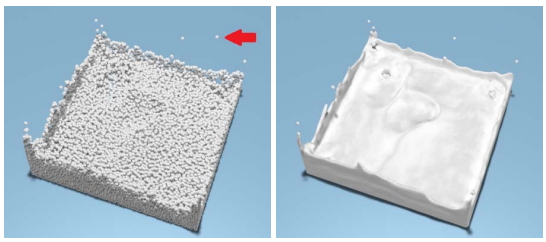

Figure 9: *Comparison between simulation particles (left) and the explicit surface mesh (right). The explicit mesh loses tracking of the particle that is identified by the red arrow.*

the fluid, so surface tracking will be faster for more compact shapes. In most of our examples, the surface tracking took less time than the SPH simulation of the particle motion. The exception to this is in the simulations for Figures 3 and 7, where we used a finer mesh resolution for color tracking. In most of our examples, the mesh resolution was one-half of the average particle spacing, but for the viscous figures and the water crown, this mesh to particle spacing ratio was 0.25 and 0.37, respectively. Compared to the timings of the method of Yu and Turk [YT10] (the third column), our surface tracking (the second column) adds a similar amount of overhead to the simulation.

Although our method incurs a per-timestep cost of the mesh vertex projection, it turns out that this projection does not adversely affect our running time. In a typical timestep, most of our projections require just one implicit surface evaluation. In contrast, a high-quality Marching Cubes surface extraction requires a root-finding step to locate the zero-value point along each edge, and this typically requires around ten implicit surface evaluations (see [Blo94] for details). Although the Marching Cubes surface extraction is needed only per-frame and not per-timestep, the higher cost of Marching Cubes is roughly equal to our per-timestep projection cost for the examples in this paper. In the zero-gravity cube example of Figure 5, our surface tracking requires 0.78 seconds per frame, while the Marching cubes requires 0.67 seconds per frame.

6.1. Limitations

There are a few limitations of our mesh-based surface tracking method. When we use our surface wave dynamics as a post-process, we have to prohibit the creation of new waves by highly curved regions. If we do not take this step, then many tiny waves are initiated and this quickly fills the entire surface with a noisy wave field. Another limitation of our method is that our surface dynamics do not exactly preserve the volume of the surface. This is not noticeable for large fluid volumes, but very small drops can be seen to slightly oscillate in volume. We think that this can be corrected by calculating the volume of each connected component and then re-scaling. As shown in Figure 9, the surface mesh occasionally fails tracking of some particles. This occurs in regions where the mesh resolution becomes coarse, because the isolated particles are undetected by the projection step.

Example	Surface Tracking	Surface Recons.	Simul.	Surface Waves	Total
Cube	0.54	0.24	1.15	-	1.93
Dumbbell	0.54	0.20	2.67	-	3.42
Two Sphere	1.767	5.08	3.69	11.41	21.70
Viscous Figs.	41.08	24.37	22.49	-	87.94
Water Crown	46.87	38.54	63.32	24.57	173.31

Table 1: *Mesh resolution relative to the average particle spacing and average per frame timings (in seconds) for our simulation examples.*

We think that this can be corrected by detecting escaped particles using a signed distance field of the surface mesh and enclosing each of them with a sphere mesh.

7. Conclusion and Future Work

We have introduced a new way of tracking SPH fluids that carries an explicit surface mesh from one time step to the next. This approach allows us to create surface tension forces, carry properties on the surface such as colors, and add capillary waves to these dynamic surfaces.

There are several avenues for future work. One challenge is to use our surface mesh to carry foam on the fluid surface in order to increase the visual realism of SPH fluids. Another possibility is to perform dynamic texture synthesis on SPH fluid surfaces, similar to the work of [BSM*06] and [KAK*07]. Finally, there are other particle-based methods used for animation besides SPH, and it is likely that our mesh tracking approach can be useful for these simulators.

Acknowledgements

This work was funded by NSF grant IIS-1017014 and CCF-0917093.

References

[AJM*10] ANDERSSON B., JAKOBSSON S., MARK A., EDELVIK F., DAVIDSON L.: Modeling surface tension in sph by interface reconstruction using radial basis functions. In *Proc. of the 5th International SPHERIC Workshop* (2010). 3

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. Graph. 26*, 3 (2007), 48. 2

[AT11] ANDO R., TSURUNO R.: A particle-based method for preserving fluid sheets. In *Proc. of Symp. on Comput. Anim* (2011), pp. 7–16. 5

[BB09] BROCHU T., BRIDSON R.: Robust topological operations for dynamic explicit surfaces. *SIAM J. Sci. Comput. 31*, 4 (2009), 2472–2493. 2, 5

[BBB10] BROCHU T., BATTY C., BRIDSON R.: Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. 29*, 4 (2010), 1–9. 2, 3, 8

[BGB11] BHATTACHARYA H., GAO Y., BARGTEIL A. W.: A level-set method for skinning animated particle data. In *Proc. of Symp. on Comput. Anim* (Aug 2011). 2, 7

[BGOS06] BARGTEIL A., GOKTEKIN T., O'BRIEN J., STRAIN J.: A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graph. (TOG) 25*, 1 (2006), 19–38. 2, 3

[Bli82] BLINN J.: A generalization of algebraic surface drawing. *ACM Trans. Graph. (TOG) 1*, 3 (1982), 235–256. 2

[Blo94] BLOOMENTHAL J.: An implicit surface polygonizer. *Graphics gems IV 1* (1994), 324–349. 9

[BSM*06] BARGTEIL A. W., SIN F., MICHAELS J. E., GOKTEKIN T. G., O'BRIEN J. F.: A texture synthesis method for liquid animations. In *Proc. of Symp. on Comput. Anim.* (2006). 3, 9

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *Proc. of Symp. on Comput. Anim.* (2007), pp. 209–217. 3, 4, 6, 8

[BWH*06] BERGOU M., WARDETZKY M., HARMON D., ZORIN D., GRINSPUN E.: A quadratic bending model for inextensible surfaces. In *Proc. of the fourth Eurographics symposium on Geometry processing* (2006), pp. 227–230. 6

[BWHT07] BARGTEIL A., WOJTAN C., HODGINS J., TURK G.: A finite element method for animating large viscoplastic flow. *ACM Trans. Graph. 26*, 3 (2007), 16–1. 3

[CBP05] CLAVET S., BEAUDOIN P., POULIN P.: Particle-based viscoelastic fluid simulation. In *Proc. of Symp. on Comput. Anim.* (2005), pp. 219–228. 3, 6

[DCG98] DESBRUN M., CANI-GASCUEL M.: Active implicit surface for animation. In *Graph. Inter.* (1998), pp. 143–150. 3

[DFG*06] DU J., FIX B., GLIMM J., JIA X., LI X., LI Y., WU L.: A simple package for front tracking. *J. Comput. Phys. 213*, 2 (2006), 613–628. 2

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc. of SIGGRAPH* (1999), pp. 317–324. 3

[EFFM02] ENRIGHT D., FEDKIW R., FERZIGER J., MITCHELL I.: A hybrid particle level set method for improved interface capturing. *J. Comput. Phys. 183*, 1 (2002), 83–116. 2

[EMB11] ERLEBEN K., MISZTAL M., BÆRENTZEN J.: Mathematical foundation of the optimization-based fluid animation method. In *Proc. of Symp. on Comput. Anim* (2011), ACM, pp. 101–110. 3

[GBO04] GOKTEKIN T. G., BARGTEIL A. W., O'BRIEN J. F.: A method for animating viscoelastic fluids. *ACM Trans. Graph. 23*, 3 (2004), 463–468. 3

[HA06] HU X., ADAMS N.: A multi-phase SPH method for macroscopic and mesoscopic flows. *J. Comput. Phys. 213*, 2 (2006), 844–861. 3, 6

[HK05a] HIEBER S., KOUMOUTSAKOS P.: A lagrangian particle level set method. *J. Comput. Phys. 210*, 1 (2005), 342–367. 2

[HK05b] HONG J.-M., KIM C.-H.: Discontinuous fluids. *ACM Trans. Graph. 24* (July 2005), 915–920. 3

[KAK*07] KWATRA V., ADALSTEINSSON D., KIM T., KWATRA N., CARLSON M., LIN M.: Texturing fluids. *IEEE TVCG* (2007), 939–952. 3, 9

[KFL00] KANG M., FEDKIW R., LIU X.: A boundary condition capturing method for multiphase incompressible flow. *Journal of Scientific Computing 15*, 3 (2000), 323–360. 3

[KSK09] KIM D., SONG O.-Y., KO H.-S.: Stretching and wiggling liquids. *ACM Trans. Graph. 28*, 5 (2009), 120. 3

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *ACM Trans. Graph. 23*, 3 (2004), 457–462. 2, 3

[MÖ9] MÜLLER M.: Fast and robust tracking of fluid surfaces. In *Proc. of Symp. on Comput. Anim* (2009), ACM, pp. 237–245. 2

[MBE*10] MISZTAL M., BRIDSON R., ERLEBEN K., BÆRENTZEN J., ANTON F.: Optimization-based fluid simulation on unstructured meshes. In *Proc. of Virtual Reality Interactions and Physical Simulations* (2010), pp. 11–20. 2, 3

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proc. of Symp. on Comput. Anim.* (2003), pp. 154–159. 3, 6

[MCZ07] MUSETH K., CLIVE M., ZAFAR N. B.: Blobtacular: surfacing particle system in "pirates of the caribbean 3". In *SIGGRAPH sketches* (2007). 2

[MDSB02] MEYER M., DESBRUN M., SCHRÖDER P., BARR A.: Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and mathematics 3*, 7 (2002), 34–57. 6

[MMS07] MIHALEF V., METAXAS D., SUSSMAN M.: Textured liquids based on the marker level set. In *Computer Graphics Forum* (2007), vol. 26, Wiley Online Library, pp. 457–466. 3

[OS88] OSHER S., SETHIAN J.: Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys. 79*, 1 (1988), 12–49. 2

[REN*04] RASMUSSEN N., ENRIGHT D., NGUYEN D., MARINO S., SUMNER N., GEIGER W., HOON S., FEDKIW R.: Directable photorealistic liquids. In *Proc. of Symp. on Comput. Anim.* (2004), pp. 193–202. 3

[SS11] STAM J., SCHMIDT R.: On the velocity of an implicit surface. *ACM Trans. Graph. 30* (May 2011), 21:1–21:7. 3, 5

[SSIF07] SIFAKIS E., SHINAR T., IRVING G., FEDKIW R.: Hybrid simulation of deformable solids. In *Proc. of Symp. on Comput. Anim.* (2007), pp. 81–90. 3

[Tes02] TESSENDORF J.: Simulating ocean water. In *Simulating Nature: Realistic and Interactive Techniques* (2002), SIGGRAPH Course Notes. 3

[TWGT10] THÜREY N., WOJTAN C., GROSS M., TURK G.: A multiscale approach to mesh-based surface tension flows. *ACM Trans. Graph. (TOG) 29*, 4 (2010), 48. 3, 6, 7

[WH94] WITKIN A., HECKBERT P.: Using particles to sample and control implicit surfaces. In *Proc. of the 21st annual conference on Computer graphics and interactive techniques* (1994), ACM, pp. 269–277. 5

[Wil08] WILLIAMS B. W.: *Fluid Surface Reconstruction from Particles*. Master's thesis, The University of British Columbia, Canada, February 2008. 2, 7

[WMT07] WANG H., MILLER G., TURK G.: Solving general shallow wave equations on surfaces. In *Proc. of Symp. on Comput. Anim* (2007), pp. 229–238. 3, 6

[WT08] WOJTAN C., TURK G.: Fast viscoelastic behavior with thin features. *ACM Trans. Graph. 27*, 3 (2008), 1–8. 3

[WTGT09] WOJTAN C., THÜREY N., GROSS M., TURK G.: Deforming meshes that split and merge. *ACM Trans. Graph. 28* (July 2009), 76:1–76:10. 2, 4, 5

[WTGT10] WOJTAN C., THÜREY N., GROSS M., TURK G.: Physics-inspired topology changes for thin fluid features. *ACM Trans. Graph. 29*, 4 (2010), 1–8. 2, 5

[XPB06] XU G., PAN Q., BAJAJ C.: Discrete surface modelling using partial differential equations. *Computer Aided Geometric Design 23*, 2 (2006), 125–145. 7

[YT10] YU J., TURK G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proc. of Symp. on Comput. Anim* (2010), pp. 217–225. 2, 3, 9

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. In *SIGGRAPH* (2005), ACM, p. 972. 2

[Zha10] ZHANG M.: Simulation of surface tension in 2D and 3D with smoothed particle hydrodynamics method. *J. Comput. Phys. 229* (2010), 7238–7259. 3