A Deformable Surface Model for Real-Time Water Drop Animation

Yizhong Zhang, *Student Member, IEEE,* Huamin Wang, *Member, IEEE*
Shuai Wang, *Member, IEEE*, Yiying Tong, *Member, IEEE,* and Kun Zhou, *Member, IEEE*

Abstract—A water drop behaves differently from a large water body because of its strong viscosity and surface tension under the small scale. Surface tension causes the motion of a water drop to be largely determined by its boundary surface. Meanwhile, viscosity makes the interior of a water drop less relevant to its motion, as the smooth velocity field can be well approximated by an interpolation of the velocity on the boundary. Consequently, we propose a fast *deformable surface model* to realistically animate water drops and their flowing behaviors on solid surfaces. Our system efficiently simulates water drop motions in a Lagrangian fashion, by reducing 3D fluid dynamics over the whole liquid volume to a deformable surface model. In each time step, the model uses an implicit mean curvature flow operator to produce surface tension effects, a contact angle operator to change droplet shapes on solid surfaces, and a set of mesh connectivity updates to handle topological changes and improve mesh quality over time. Our numerical experiments demonstrate a variety of physically plausible water drop phenomena at a real-time rate, including capillary waves when water drops collide, pinch-off of water jets, and droplets flowing over solid materials. The whole system performs orders-of-magnitude faster than existing simulation approaches that generate comparable water drop effects.

Index Terms—deformable surface model, surface tension, mean curvature flow, water drop simulation

1 Introduction

Water drops appear everywhere in our daily life, from wall panels when we take showers in the bathroom, to car windshields when we drive on a rainy day. Compared with simulation of large water bodies, how to efficiently simulate water drops is an even more challenging problem as described by Wang *et al.* [1]. This is mainly due to the large viscosity and surface tension effect under a small scale. When water drops are defined by 3D volumetric representations, the large viscosity and surface tension effects require a small time step in order to ensure accuracy and stability. A 3D representation also requires a large memory cost to maintain all of the surface details, making it difficult to handle scenes with numerous water drops that we usually see in the real world. In general, most generic fluid simulation techniques are not directly applicable to this domain. To the best of our knowledge, there exists no simple yet efficient algorithm to animate small-scale water phenomena.

While the viscosity and surface tension effects cause difficulties in simulating water drops as 3D volumes, we found that they can be beneficial when water drops are animated in a surface representation. Large viscosity is likely to smooth out the velocity field of a water drop, making it safe to ignore the water drop interior and focus on its boundary surface. At the same time, the surface tension force can be calculated directly using the mean curvature of the surface. This effect brings smoothness to the surface and makes topological changes easier to handle.

Based on these observations, we present an efficient Lagrangian deformable surface model for physically plausible animation of water drops and their interactions with solid surfaces. Under this model, we reduce the degrees of freedom of water drop motions to the boundary nodes of a dynamic system, thus dramatically decrease the running time. The model is made of two crucial components:

- **Deformation Operators:** We present a series of deformation operators to evolve water drops over time, so that they behave in a physically plausible way. For example, our implicit mean curvature flow operator efficiently produces surface tension effects, and our contact angle operator generates various hydrophilic behaviors when water drops flow on solid surfaces.
- **Mesh Operators:** A set of mesh operators optimize the mesh quality and handle topological changes, such as mesh merge and split. Without mesh operators, meshes cannot be robustly simulated and they cannot interact with each other in a proper way.

The whole system is an organic combination of individual operators and it allows us to produce water drop effects that each component alone is not able to generate. For example, although mesh operations have been studied for decades, their robustness remains questionable when they are applied in generic fluid simulation as Wojtan *et al.* pointed out

- *Yizhong Zhang, Shuai Wang and Kun Zhou are with the State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China, 310058. E-mail: kunzhou@acm.org, yizhongzhang.zju@gmail.com, clarkwang_phd@163.com*
- *Huamin Wang is with the Department of Computer Science and Engineering at the Ohio State University, Columbus, OH 43210. E-mail: wanghmin@hotmail.com*
- *Yiying Tong is with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824. E-mail: ytong@msu.edu*

(in [2], [3], [4]). We found that certain mesh operators can still be effective under the small-scale assumption. On the other hand, although surface tension flow and volume conservation techniques have been studied by Müller [5] and Brochu [6], they are not directly applicable to complicated scenes when water drops frequently collide and split. By combining these operators into a system, we recognized and verified their effectiveness under the small-scale water animation context through a variety of realistic examples, including capillary waves when water drops collide, pinch-off of water jets, and droplets flowing over various solid materials. A typical scene that contains 20,000 surface triangles can be simulated at 30 frames per second. We also validate our deformable surface model by comparing water drop simulation results with real-world experiment videos.

2 Previous Work

Since physically based fluid simulation was first introduced into the graphics community for animation purposes, various numerical techniques have been proposed using different water representations. Water bodies can be represented using implicit functions and animated using Eulerian approaches, such as [7], [8], [9], [10], [11], [12], [13], [14]. On the other hand, Lagrangian approaches explicitly track water surfaces by representing water as a set of smoothed particles [15], [16], [17], [18], or tetrahedral meshes [19], [20]. Hybrid methods that use more than one representation have also been proposed to generate detailed water surface behaviors for specific problems, including [3], [21], [22].

Among existing physically based water simulation techniques, our work is most related to numerical methods that handle surface tension forces, including surface tension forces in Smoothed Particle Hydrodynamics (SPH) [17], practical simulation of surface tension [23], liquid-air boundary conditions [24], a virtual surface method for water drops on solid surfaces [1], and surface tension in incompressible two-phase flow [25].

Our deformable surface model is also closely related to the general shallow wave model proposed by Wang *et al.* [26]. While their mesh-based height field representation allows them to efficiently generate water drop flowing on solid surfaces, their technique relies on solid surface discretization and it cannot handle concave or free-falling water drops. Alternatively, the hybrid method proposed by Thürey *et al.* [4] used surface meshes to represent fine details of small water drops. Our technique can achieve similar results in a significantly more efficient way. Under a small-scale assumption, we can use a set of mesh operators to directly animate surface meshes without underlying Eulerian-based fluid simulators.

Our approach to the approximation of surface tension on water drop boundaries is similar to [5]. Since surface tension can be treated as the force generated by a potential energy proportional to the surface area, its effect is often related to the mean curvature flow. Implicit integration methods proposed in [27] and [28] formulate the Laplace-Beltrami operator into a sparse linear system, so that the flow can be efficiently calculated using a large time step. While original mean curvature flow algorithms tend to cause volume loss over time, a volume-preserving mean curvature flow algorithm proposed in [29] uses the gradient flow to compensate volume loss in a local way. Our system is also formulated in this fashion.

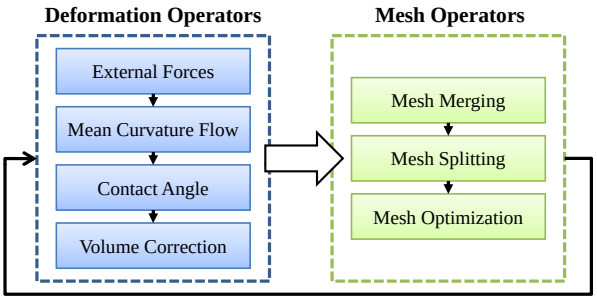


Fig. 1: *The pipeline of the deformable surface model for simulating water drops.*

3 Approach Overview

Our deformable surface model is based on the assumption that water drop motion is dominated by its boundary surface, due to the large viscosity and surface tension effects at a small scale. The model can be separated into two major components. The first component consists of deformation operators that evolve surface meshes over time in a physically plausible manner. These operators consider external forces, friction, surface tension and other physics phenomena relevant to water drops. The second component contains a set of mesh operators that process topological changes, including mesh merge, split and mesh optimization that improve the mesh quality. They make sure that water drop interactions are correctly handled so that deformation operators can be robustly processed. Since the whole model is based on surface meshes, they need to be efficiently performed over time.

The whole system pipeline is shown in Figure 1. we use four deformation operators to update the velocity and position for each vertex. External forces, including both gravity and solid collision forces, are first applied on the mesh shape and its velocities. The mean curvature flow is then used to mimic surface tension behaviors using an implicit integrator. To produce different hydrophilic effects when water drops flow on solid surfaces, we use a contact angle operator to explicitly modify the contact line according to the contact angle condition. Finally, volume loss is corrected using a volume correction operator. Once we finish evolving surface meshes, we use mesh operators to handle collisions and splits of water drops. We also optimize the mesh quality to enhance the robustness of the deformation operators.

Operators are processed independently and sequentially in our system. Each operator can be considered as a filter to velocities and positions of mesh vertices. After being processed by the previous operator, vertex velocities and positions become the input to the next operator. To simplify the notation, we will use $\mathrm{v_i^{old}}$ and $\mathrm{x_i^{old}}$ to denote the velocity and position of vertex i before it is processed by an operator, and $\mathrm{v_i^{new}}$ and $\mathrm{x_i^{new}}$ as vertex i's velocity and position after processing.

Since operators are processed independently, constraints enforced by one operator may be violated by other operators throughout the process. For example, volume conservation enforced by the volume correction operator may be violated again by mesh operators. Fortunately, we found from our experiment that we can safely ignore conflictions among different operators, as long as constraints can be iteratively satisfied in the system over time. The only exception is the solid-water collision constraint, which may cause noticeable volume loss and penetration issue if not properly handled. To solve this problem, we label out colliding vertices during the solid-water collision process in the external force operator. We then simply prevent them from moving in the solid surface normal direction, when we process other operators.

4 Deformation Operators

Similar to the method of characteristics that reduces the Navier-Stokes equations into a family of simple equations, we propose a set of deformation operators to evolve the velocity and position for each mesh vertex over time. Instead of formulating the problem over the whole water drop volume, we only consider the deformation of the surface boundary. This simplifies the problem and makes the deformation operator more efficient to process.

4.1 External Forces

We consider external forces, including gravity, solid collision, friction and viscosity in this subsection. Let $\mathrm{v_i^{old}}$ be the velocity of vertex i before this operator, the gravity force can be simply applied as: $\mathrm{v_i^{new}} = \mathrm{v_i^{old}} + \mathrm{g}\Delta t$, in which $\mathrm{v_i^{new}}$ is the new velocity, g is the gravity acceleration and Δt is the time step. The position $\mathrm{x_i}$ of vertex i is then evolved using the forward Euler method as: $\mathrm{x_i^{new}} = \mathrm{x_i^{old}} + \mathrm{v_i^{new}}\Delta t$. By using the updated vertex positions, we find solid-water collisions either analytically or numerically by detecting whether the vertex penetrates the surface of a solid object. If that happens, the vertex position $\mathrm{x_i^{new}}$ will be projected to the nearest location on the solid surface, and the velocity will also be adjusted in an inelastic way:

$$\mathrm{v_i^{new}} = \mathrm{v_i^{old}} - \left((\mathrm{v_i^{old}} - \mathrm{v_s}) \cdot \mathrm{n_i}\right) \mathrm{n_i}, \tag{1}$$

in which $\mathrm{v_s}$ is the velocity of the solid object at $\mathrm{x_i^{new}}$ and $\mathrm{n_i}$ is the solid surface normal at $\mathrm{x_i^{new}}$. Intuitively, Equation 1 removes any relative velocity along the normal direction between the vertex and the solid object. Once colliding vertices are found and processed, we apply friction forces on them to account for the slipping condition between water and the solid object:

$$\mathrm{v_i^{new}} = \begin{cases} 0, & \left|\mathrm{v_i^{old}}\right| < \varepsilon \\ \mathrm{v_i^{old}} - \varepsilon \mathrm{v_i^{old}} \left|\mathrm{v_i^{old}}\right|^{-1}, & \text{otherwise} \end{cases} \tag{2}$$

in which ε is a friction magnitude coefficient. We then use the difference between $\mathrm{v_i^{new}}$ and $\mathrm{v_i^{old}}$ to immediately adjust the vertex position: $\mathrm{x_i^{new}} = \mathrm{x_i^{old}} + (\mathrm{v_i^{new}} - \mathrm{v_i^{old}})\Delta t$.

Although the viscosity is not considered as an external force in volumetric fluid simulators, we apply a damping force on the surface model to achieve similar viscosity effects. Similar to the damping force used in a mass-spring system, our damping effect is made of two terms:

$$\mathrm{v_i^{new}} = (1 - \mu\Delta t)\mathrm{v_i^{old}} + \eta\Delta t \boldsymbol{\Delta} \mathrm{v_i^{old}} \tag{3}$$

in which the first term is caused by a regular damping force that gradually reduces the velocity magnitude in each time step using a constant ratio μ, and the second term is caused by an explicit Laplacian-Beltrami operator using a viscosity coefficient η. The second term is especially similar to the viscosity term in the Navier-Stokes equations, except that it is defined over the surface mesh only. μ is typically chosen from 0.3 to 0.5, and η is usually between 0 and 0.1. Since our experiment shows that using a large η can further avoid certain mesh quality issues, we allow the system to apply extra viscosity damping effect if necessary (i.e., when the mesh quality is unsatisfactory even after geometric operations, which will be described in Section 5).

4.2 Mean Curvature Flow

We use the mean curvature flow to generate surface tension behaviors as Sussman *et al.* [25] suggested. Let x be any point on the water surface S, γ be the coefficient of the curvature flow, and ∇_S^2 be the Laplace-Beltrami operator for S. The mean curvature flow in a continuous setting is defined as:

$$\frac{\partial \mathrm{x}}{\partial t} = -2\gamma\kappa\mathrm{n} = \gamma\nabla_S^2 \mathrm{x}, \tag{4}$$

in which $\kappa\mathrm{n}$ is the mean curvature defined in the normal direction n. A semi-implicit method proposed by Desbrun *et al.* [27] discretizes this function over a triangle mesh:

$$\left(\mathbf{I} + \gamma\Delta t \mathbf{M^{-1}L}\right) \mathbf{X}^{\mathrm{new}} = \mathbf{X}^{\mathrm{old}}, \tag{5}$$

in which $\mathbf{X}$ is the vertex position vector, Δt is the time step, $\mathbf{M}$ is the lumped mass matrix and $\mathbf{L}$ is a symmetric matrix consisting of cotangent coefficients from the discrete Laplace-Beltrami operator. Let $N(i)$ be the one-ring neighborhood of vertex i.

$$l_{ij} = -\frac{1}{2}(\cot\alpha_{ij} + \cot\beta_{ij}), \qquad \text{for } j \in N(i) \tag{6}$$

and,

$$l_{ii} = -\sum_{j \in N(i)} l_{ij}, \tag{7}$$

Fig. 2: *Droplets on the solid surfaces with different contact angles. They are 60, 90, and 120 degrees form left to right.*

in which α_{ij} and β_{ij} are the two angles opposite to the edge between i and j. Since both $\mathbf{I}$ and $\mathbf{L}$ are symmetric, Equation 5 can be transformed into a symmetric positive definite system by multiplying $\mathbf{M}$ on both sides. We use a conjugate gradient solver with an incomplete Cholesky preconditioner to solve this system. The solution is a new vertex position vector. Once we obtain the new intermediate positions, the velocity at each vertex is updated again: $\mathrm{v_i^{new}} = \mathrm{v_i^{old}} + (\mathrm{x_i^{new}} - \mathrm{x_i^{old}})/\Delta t$.

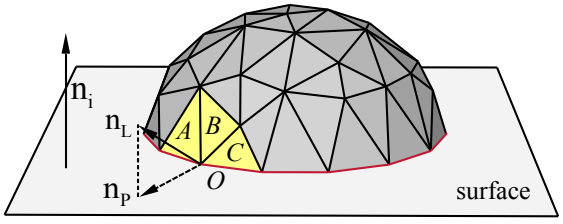


Fig. 3: *The contact line and its neighborhood.*

4.3 Contact Angle Operator

The mean curvature flow allows us to generate surface tension effects on water surface boundaries between water and air. When a water drop flows on a solid surface, its contact line will be further affected by the hydrophobicity of the solid material. Different hydrophobic properties cause water drops to form different shapes. A characteristic way to describe the hydrophilic property is the stable contact angle, defined as the angle between the water/air surface and the solid surface when a water drop reaches equilibrium. We propose a novel contact angle operator to achieve the hydrophilic effect by explicitly finding the current contact angle and using it to drive the motion of the contact line. This allows us to realistically generate various water drops on different solid surfaces as Figure 2 shows. It also causes small water drops to stay on a window panel in Figure 10, because of a larger stable contact angle when the contact line moves forward (advancing contact angle) and a smaller stable contact angle when the contact line retreats (receding contact angle).

The first step in this operator is to locate the contact line on the surface mesh. Supposing that vertices that collide with solid objects have been labeled during the solid collision step in Section 4.1, we define contact vertices as those colliding vertices that are neighboring non-colliding vertices (i.e., they are neighboring three interfaces: water-air, water-solid and air-solid). Contact lines are then simply defined as a set of edges connecting contact vertices, as the red curve shows in Figure 3.

Once we find the contact line, we move the contact line in order to reach a stable contact angle. Inspired by the virtual surface method proposed by Wang *et al.* [1], we first try to adjust the mean curvature estimator at the contact line so that the mean curvature flow drives the contact line to move either forward or backward automatically. Unfortunately, our experiment shows that this approach becomes problematic when water drops are represented as surface meshes, since the mean curvature flow relies heavily on the mesh quality and the mesh quality around the contact line is typically not comparable to other places on the surface mesh due to their sharp curvatures.

Instead, we propose an explicit way to calculate the current contact angle and then use it to move each vertex on the contact line. Let O be a contact vertex, and, *e.g.*, A, B and C be its three neighboring triangles between water and air as Figure 3 shows, we find the water-air surface normal $\mathrm{n_L}$ at vertex O by the area-weighted average of triangle normals of A, B and C. The angle between $\mathrm{n_i}$ and $\mathrm{n_L}$ is a good approximation of the contact angle for vertex O, as it measures the dihedral angle between the two tangent planes.

Based on whether the current contact angle is larger or smaller than the stable contact angle, the contact vertex should move either forward or backward. In practice, its more complicated due to the water hysteresis phenomena mentioned in [1]. So we use a dynamic contact angle scheme to guide contact line motions by applying a boundary force on each contact vertex:

$$\mathrm{f_{bnd}} = \begin{cases} 0, & \theta_c^r < \theta < \theta_c^a \\ \alpha(\theta - \theta_c^r)\mathrm{n_p}|\mathrm{n_p}|^{-1}, & \theta < \theta_c^r \\ \alpha(\theta - \theta_c^a)\mathrm{n_p}|\mathrm{n_p}|^{-1}, & \theta > \theta_c^a \end{cases} \quad (8)$$

in which θ_c^r and θ_c^a are the receding and advancing contact angles, α is a magnitude coefficient for the boundary force, and $\mathrm{n_p}$ is the projection of water surface normal $\mathrm{n_L}$ on the solid surfaces as Figure 3 shows. Intuitively, Equation 8 implies that the contact vertex is reluctant to move and tries to minimize the boundary force magnitude. This model is important for us to make small water drops stay on a window panel as Figure 10 shows. Figure 11 also shows that some water drops are even able to rest underneath the tweety model, since the receding contact angle causes a strong surface tension force to counter the gravity.

4.4 Volume Correction

Water drop volumes can be changed by either deformation operators or mesh operators, and it can be easily accumulated to cause obvious visual artifacts over time. In this subsection, we propose a volume correction operator to minimize volume changes.

Inspired by volume correction techniques for the mean curvature flow problem proposed in [5] and [29], we develop a volume correction method to handle volume changes caused by all possible operations, not just by the mean curvature flow operator. We first calculate the rigid velocity of each water drop, including both translational velocity and rotational velocity, and then remove them from the velocity field over the mesh. The remaining velocity $\mathrm{u_i}$ deforms the water drop and causes its volume to change. Let a_i be the volume change rate per unit area, we estimate a_i as:

$$a_i = \mathrm{u_i} \cdot \mathrm{n_i}, \tag{9}$$

in which $\mathrm{n_i}$ is the water surface normal at vertex i. We further calculate the local average of a_i over i's neighborhood $N(i)$ as $\bar{a}_i$:

$$\bar{a}_i = \frac{\sum_{j \in N(i)} A_j a_j}{\sum_{j \in N(i)} A_j}, \tag{10}$$

where A_i is the lumped area of vertex i. We then subtract $\bar{a}_i$ from $\mathrm{u_i}$:

$$\mathrm{u_i} = \mathrm{u_i} - \bar{a}_i \mathrm{n_i}, \tag{11}$$

This is similar to the local volume correction method proposed by Eckstein *et al.* [29], except that we correct volumes caused by the deformation velocity $\mathrm{u_i}$, rather than the mean curvature flow.

Alternatively, a global volume correction method can also be used to explicitly fix volume changes. This is done by calculating the volume change between current and the initial volume and then offsetting the water mesh uniformly with a distance $d = \Delta V / A$, in which ΔV is the volume change and A is the total surface area. The offset is performed by moving each vertex along its normal direction: $\mathrm{u_i} = \mathrm{u_i} + d\mathrm{n_i}$.

We noticed that the local method usually preserves capillary waves better during the simulation, as the bar example shows in Figure 4. On the other hand, it does not guarantee exact volume conservation like the global method, since $\bar{a}_i$ in Equation 10 only approximates how fast the volume changes at vertex i numerically. Therefore, we choose different volume correction operators under different situations. When simulating multiple water drops from a distant view, we choose the global correction operator for efficiency since capillary behaviors are hardly noticeable. But when those details are important in the animation, we choose the local correction operator, followed by a global operator to remove remaining volume changes if necessary. Once $\mathrm{u_i}$ gets processed, we add the rigid velocity component back to $\mathrm{u_i}$ and the result becomes a volume-preserving velocity field.

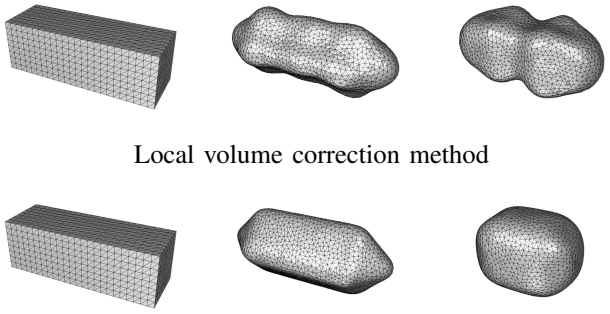


Fig. 4: *A bar-shaped water drop deforms under the surface tension force. Using the local volume correction method (top), the system is able to preserve more surface details than the global volume correction method (bottom).*

5 Mesh Connectivity Operators

So far we have described how to update vertex positions of the mesh through a set of deformation operators. We now elaborate on the adaptive remeshing stage, which handles topological changes such as merging and splitting, adjusts sampling density, and improves mesh quality. The whole method completely relies on mesh operations and it does not use any volumetric representation for topological checks, as Wojtan and his collaborators did in [2], [3].

During each time step, we perform local topology repairs. The main steps include edge collapse, edge split and edge flip, as Hoppe *et al. proposed in [30]*. We split edges if their lengths are above a maximum threshold, and collapse edges when their lengths are below a minimum threshold. We set the minimal threshold 0.04cm, while the maximum threshold is set to be three times of the minimum, as mentioned by Brochu in [31]. In order to speed up the process, we use two priority queues (implemented as a min-heap and a max-heap) to sort the edges according to their lengths. We flip an edge if it has a negative cotangent weight, which intuitively means that the edge may violate the Delaunay condition. Mesh operators improve mesh

quality and they have direct influence on the performance of deformation operators, as we found in our experiment.

We apply a weighting factor to the edge length so that the mesh can be adaptively sampled during the simulation. For example, when an edge is defined over the water-solid interface, we assign a small weight to its length so that the edge can collapse even if its actual length is above the minimum threshold. This allows us to have fewer vertices over the water-solid interface, as their positions do not affect water drop motions.

Due to the mean curvature flow and the contact angle model proposed in Section 4.2 and 4.3, vertices along the contact line are sensitive to topological changes. To avoid potential stability issues near the contact line, we do not flip edges that are part of the contact line. However, edge split and collapse are still allowed since they have less influence on the contact line shape.

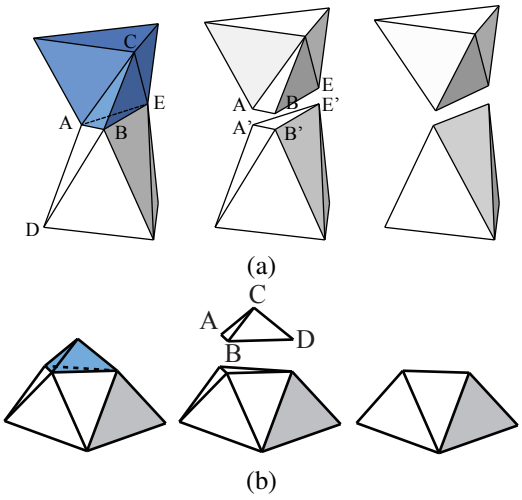


Fig. 5: *Mesh split scheme: (a) We separate the mesh if it has a thin tube-like shape. (b) If one connected component becomes a single tetrahedron after mesh split, we simply delete it.*

5.1 Mesh Merge

In order to handle water drop collisions, we maintain a bounding box for each water drop. For each pair of water drops that have intersecting bounding boxes, we apply mesh-mesh collision detection by testing edge-face collisions. We use an AABB tree to accelerate this detection process. When collision happens, we perform a Boolean union operation on the two volume sets as follows. First, we identify the intersection curve by using the intersection method proposed in [32]. We then re-triangulate the whole intersection neighborhood and we perform a mesh coloring method proposed in [32] to decide whether a face is inside or outside of the other mesh. Once the inside and outside are recognized, we remove all triangles inside and stitch outside triangles together, so that the water surface mesh becomes closed again. Any volume loss during the merging process will be corrected later using the global volume correction method.

We process mesh self-collisions in a similar fashion. Although we did implement it in our system, we usually turn it off for animation production in practice, since the additional step will greatly increase the computational cost and have minimal impact visually. This is based on the assumption that self-collision rarely happens for small water drops, because of the strong surface tension effects.

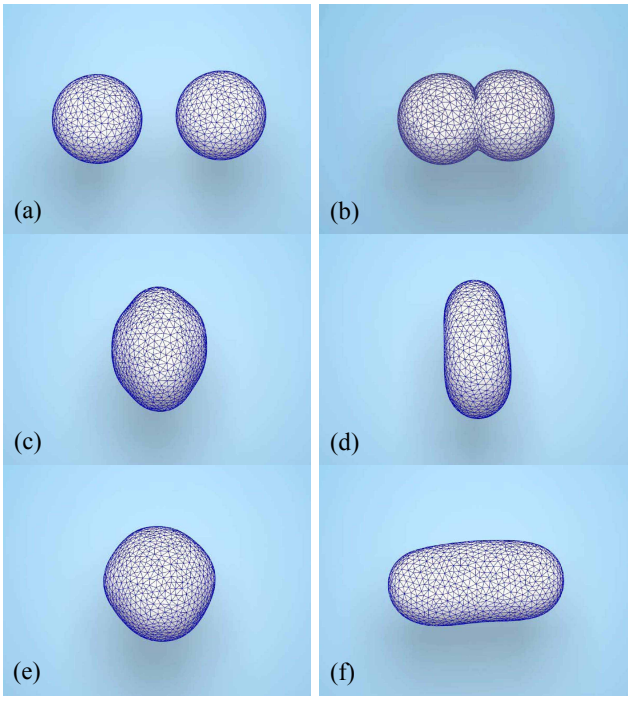


Fig. 6: *Two droplets colliding into each other. The topology of the mesh is refined dynamically over time.*

5.2 Mesh Split

We perform water drop splits at the edge collapse stage. Whenever edge collapse causes the mesh to be non-manifold, we separate the mesh into two meshes (connected components) in order to maintain the manifold property. Given A and B as two vertices of the edge that is going to be collapsed as in Figure 5a, we detect this situation in practice by testing whether A and B share a common neighboring vertex E, which is not incident to either of their common adjacent triangles. If such an E exists, we split the mesh into two meshes at A, B and E, by inserting new vertices and triangles into corresponding meshes. In the extreme case when a new mesh is a single tetrahedron, we simply delete it as shown in Figure 5b. After mesh split, new meshes are treated as two individual water drops. Any volume loss in the process is evenly distributed to two new meshes, and compensated by the next global volume correction operator.

6 Application and Results

We have created animations for various scenarios involving water drops using the deformable surface model (please refer to the attached video for animation results). Our system, implemented using multi-threading, runs on an 8-core Xeon 2.4GHZ, 16GB RAM workstation.

The mesh resolution varies from 10K to 50K in our experiment and the simulation system typically runs at 10 to 50 FPS. On average, 58 percent of the computational cost is spent on solving the mean curvature flow and evolving the surface mesh. Collision detection uses 30 percent of the computational time. The rest is used to process mesh connectivity operators, including mesh merge, split and flip.

6.1 Validation

6.1.1 Colliding Droplets

In this example, we compare our animation results with a real water drop collision experiment[1]. Figure 6 shows that our method can accurately simulate the capillary waves over the water drop surface, even though this is only a surface deformable model.

6.1.2 Droplet Hitting the Ground

We compare our result with a real droplet hitting the ground experiment[2], in order to test how the contact angle operator performs in an extreme condition. Specifically, the ground plane is set to be highly hydrophilic. Figure 7 shows that the water drop quickly spread out on the solid surface due to the strong hydrophilic behavior.

6.1.3 Resting Droplets

Figure 8 shows different shapes of a water drop when the surface tension coefficient varies. Adjusting the surface tension coefficient as shown in Figure 8 will produce effects similar to modifying the contact angle condition, as shown in Figure 2. While both examples look physically plausible as static water drops, they are simulated at different physics conditions and their dynamic behaviors are different.

6.2 Animation

6.2.1 Water Tap

Figure 9 shows an animation of water streaming from a tap. The stream splits into small water drops due to the Rayleigh-Plateau instability [4]. In general, a large surface tension coefficient is more likely to cause the water streams to split. This example runs at 50FPS.

1. http://www.youtube.com/watch?v=uVQS2W0_r7U
2. http://www.youtube.com/watch?v=m1KKbJo4nYk&feature=related

6.2.2 Window Panel

Figure 10 shows an animation result made of hundreds of water drops flowing on a glass window panel. In this example, droplets may adhere on the window panel because the surface tension effects dominate. However, when the volumes of the water drops grow, they become more susceptible to gravity and start flowing down the window panel with different terminal speeds. This example also demonstrates user interactions by allowing the user to manually move a button over the window panel. Water drops are pushed away from its trajectory. While the frame rate varies with the total number of triangles used to define water drops, we found that the system can still run at 30FPS even with 50K triangles. The rendering part of this example is implemented through OpenGL with GPU acceleration.

6.2.3 Lotus Leaves

Figure 12 shows water drops falling onto lotus leaves, which are considered to be highly hydrophobic surfaces. During the simulation, there are approximately 180 water drops containing 30K triangles. The whole system runs at 35 FPS in this example.

6.2.4 Tweety

Our third example, shown in Figure 11, demonstrates how water drops behave on a curved surface. Since the Tweety model is significantly more complicated than the previous two examples, we use an AABB tree to accelerate water-solid collision detection tests. By assuming that the solid surface is smooth, the contact angle of each contact vertex can be calculated in the same way as in flat surface examples. The minimal contact angle is set to 40°. When the surface tension is not strong enough to hold water drops, they will drip off from the bottom and split into multiple water drops. This scene contains approximately 10K triangles. The simulation runs at 10 FPS approximately.

6.3 Limitations

While a deformable surface model brings efficiency to our system, it is less accurate compared with a volumetric representation since it ignores the interior. Thus, it is not suitable for handling large bodies of water, when both viscosity and surface tension effects become less signficant. Although the mean curvature flow operator in our system is solved by an implicit time integrator, it is not unconditionally stable since its stability also relies on the mesh quality. This issue can be worsened with increased time steps. When simulating water drops on curved solid surfaces, we assume that the solid surface is smooth. This assumption becomes invalid when the solid surface has very sharp features, which may cause instability in the contact angle model.

Fig. 7: *A droplet falling onto a hydrophilic surface.*

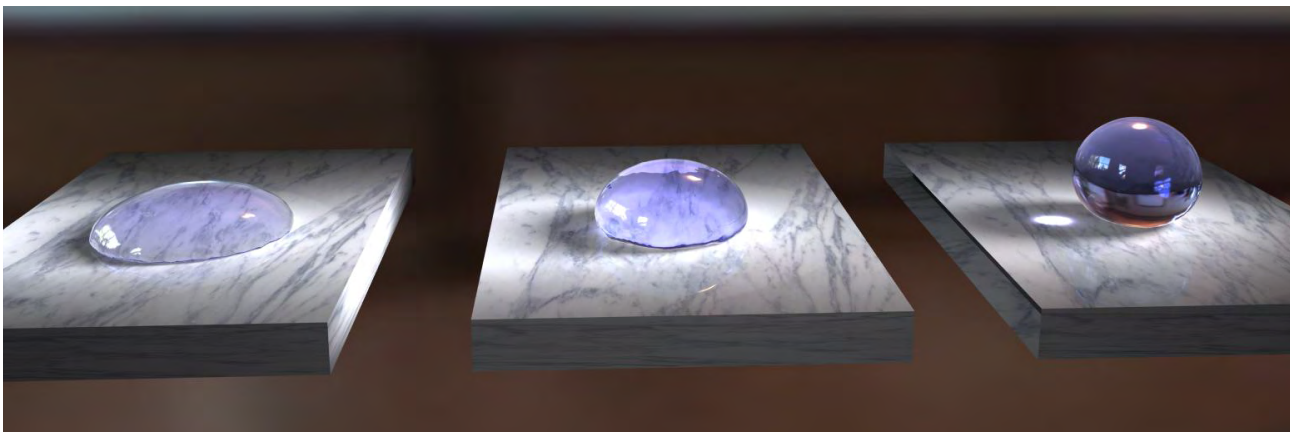


Fig. 8: *Droplets on ideal non-wetting surfaces with different tension coefficients 0.05, 0.2 and 1.0 from left to right.*

Fig. 9: *A water tap example. Surface tension coefficients are 0.01, 0.2, and 1.0, from left to right.*

7 Conclusion and Future Work

In this paper, we proposed a deformable surface model to animate small-scale fluid motion of water drops. The model is separated into two components: deformation operators that evolve the shapes of water drops over time, and mesh operators that handle topological changes and improve mesh quality. Using this model, we can efficiently generate various water drop behaviors. We can also easily combine it with existing fluid simulators to provide detailed water drop animations in generic fluid simulations.

In the future, we wish to explore further improvements on the system performance by a full GPU implemention of the system. We are also interested in finding better ways to adaptively re-sample meshes, so that the computational cost can be further reduced. How to improve the system stability is another problem that we would like to investigate.

Acknowledgments

We would like to thank Mingming He and Chen Cao for their help during video production, and Steve Lin for proofreading this paper. The project is partially supported by NSFC (No. 60825201), the 973 program of China (No. 2009CB320801), NSF grants IIS-0953096, CCF 0936830, CMMI 0757123, and CCF 0811313.

References

[1] H. Wang, P. J. Mucha, and G. Turk, "Water drops on surfaces," *ACM Transactions on Graphics (SIGGRAPH 2005)*, vol. 24, no. 3, pp. 921–929, July 2005.

Fig. 10: *Hundreds of droplets flowing on a window panel. The advancing contact angle is* 90° *and the receding contact angle is* 30°*. These images are taken from an interactive demo.*

Fig. 11: *Water drops flow on a tweety model. A large surface tension effect causes small water drops to rest at the bottom of the model.*

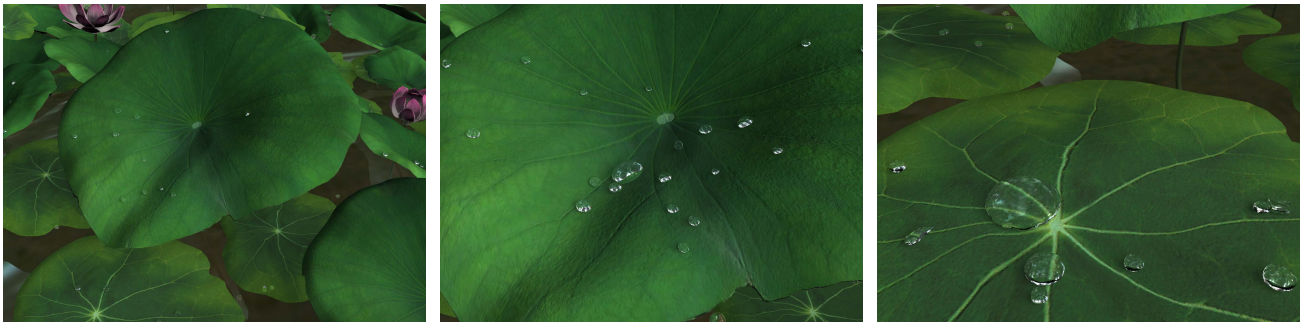


Fig. 12: *Water drops flowing on lotus leaves.*

[2] C. Wojtan, N. Thürey, M. Gross, and G. Turk, "Deforming meshes that split and merge," *ACM Transactions on Graphics (SIGGRAPH 2009)*, vol. 28, no. 3, pp. 76:1–76:10, July 2009.

[3] C. Wojtan, N. Thurey, M. Gross, and G. Turk, "Physics-inspired topology changes for thin fluid features," *ACM Transactions on Graphics (SIGGRAPH 2010)*, vol. 29, no. 4, pp. 50:1–50:8, July 2010.

[4] N. Thürey, C. Wojtan, M. Gross, and G. Turk, "A multiscale approach to mesh-based surface tension flows," *ACM Transactions on Graphics (SIGGRAPH 2010)*, vol. 29, no. 4, pp. 48:1–48:10, July 2010.

[5] M. Müller, "Fast and robust tracking of fluid surfaces," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2009)*, 2009, pp. 237–245.

[6] T. Brochu, "Fluid animation with explicit surface meshes and boundary-only dynamics," Master's thesis, University of British Columbia, 2006.

[7] M. Kass and G. Miller, "Rapid, stable fluid dynamics for computer graphics," in *Proceedings of the 17th annual conference on Computer graphics and interactive techniques (SIGGRAPH '90)*, 1990, pp. 49–57.

[8] N. Foster and D. Metaxas, "Realistic animation of liquids," in *Proc. of Graphics interface 1996*, 1996, pp. 204–212.

[9] J. Stam, "Stable fluids," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99)*, 1999, pp. 121–128.

[10] N. Foster and R. Fedkiw, "Practical animation of liquids," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH 2001)*, 2001, pp. 23–30.

[11] D. Enright, S. Marschner, and R. Fedkiw, "Animation and rendering of complex water surfaces," *ACM Transactions on Graphics (SIGGRAPH 2002)*, vol. 21, no. 3, pp. 736–744, July 2002.

[12] D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw, "Using the particle level set method and a second order accurate pressure boundary condition for free surface flows," in *In Proc. of 4th ASME-JSME Joint Fluids Engineering Conference*, 2003, pp. 2003–45 144.

[13] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating water and smoke

with an octree data structure," *ACM Transactions on Graphics (SIGGRAPH 2004)*, vol. 23, no. 3, pp. 457–462, August 2004.

[14] A. W. Bargteil, T. G. Goktekin, J. F. O'brien, and J. A. Strain, "A semi-lagrangian contouring method for fluid simulation," *ACM Transactions on Graphics*, vol. 25, no. 1, pp. 19–38, January 2006.

[15] L. B. Lucy, "A numerical approach to the testing of the fission hypothesis," *The Astronomical Journal*, vol. 82, pp. 1013–1024, 1977.

[16] R. A. Gingold and J. J. Monaghan, "Smoothed particle hydrodynamics: theory and application to non-spherical stars," vol. 181, pp. 375–398, 1977.

[17] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA 2003)*, 2003, pp. 154–159.

[18] R. Bridson and M. Müller-Fischer, "Fluid simulation," in *ACM SIGGRAPH 2007 courses*, ser. SIGGRAPH '07, 2007, pp. 1–81.

[19] B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'Brien, "Fluid animation with dynamic meshes," *ACM Transactions on Graphics (SIGGRAPH 2006)*, vol. 25, no. 3, pp. 820–825, July 2006.

[20] M. Wicke, D. Ritchie, B. M. Klingner, S. Burke, J. R. Shewchuk, and J. F. O'Brien, "Dynamic local remeshing for elastoplastic simulation," *ACM Transactions on Graphics (SIGGRAPH 2010)*, vol. 29, no. 4, pp. 49:1–49:11, July 2010.

[21] G. Irving, E. Guendelman, F. Losasso, and R. Fedkiw, "Efficient simulation of large bodies of water by coupling two and three dimensional techniques," *ACM Transactions on Graphics (SIGGRAPH 2006)*, vol. 25, no. 3, pp. 805–811, July 2006.

[22] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw, "Two-way coupled sph and particle level set fluid simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 797–804, July 2008.

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

[24] J.-M. Hong and C.-H. Kim, "Discontinuous fluids," *ACM Transactions on Graphics (2005)*, vol. 24, no. 3, pp. 915–920, July 2005.

[25] M. Sussman and M. Ohta, "A stable and efficient method for treating surface tension in incompressible two-phase flow," *Journal of Scientific Computing*, vol. 31, no. 4, pp. 2447–2471, June 2009.

[26] H. Wang, G. Miller, and G. Turk, "Solving general shallow wave equations on surfaces," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA 2007)*, 2007, pp. 229–238.

[27] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99)*, 1999, pp. 317–324.

[28] D. Mathieu, M. Mark, S. Peter, and A. H. Barr, "Discrete differential-geometry operators in nD." Springer-Verlag, 2000, pp. 35–57.

[29] I. Eckstein, J.-P. Pons, Y. Tong, C.-C. J. Kuo, and M. Desbrun, "Generalized surface flows for mesh processing," in *Proceedings of the fifth Eurographics symposium on Geometry processing (SGP 2007)*, 2007, pp. 183–192.

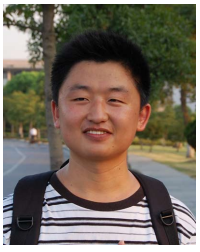
[30] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh optimization," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques (SIGGRAPH '93)*. ACM, 1993, pp. 19–26.

[31] T. Brochu and R. Bridson, "Robust topological operations for dynamic explicit surfaces," *SIAM J. Sci. Comput.*, vol. 31, pp. 2472–2493, 2009.

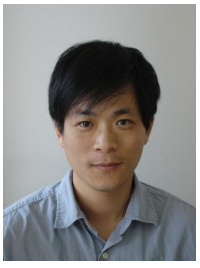
[32] D. Baraff, A. Witkin, and M. Kass, "Untangling cloth," *ACM Transactions on Graphics (SIGGRAPH 2003)*, vol. 22, no. 3, pp. 862–870, July 2003.

Yizhong Zhang is a Ph.D. student in the state key laboratory of CAD&CG, Zhejiang University, Hangzhou, China. He received his B.S. degree in Mechatronics Engineering from Zhejiang University in 2010. His research interests include physically based simulation, computer graphics and robotics.

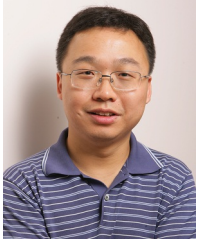
Huamin Wang is an assistant professor in the department of Computer Science and Engineering at the Ohio State University. Before that, he finished his postdoctoral study in the department of Electronic Engineering and Computer Sciences, at the University of California, Berkeley. He received a Ph.D. degree from Georgia Institute of Technology in 2009, a M.S. degree from Stanford University in 2004 and a B.Eng. degree from Zhejiang University (Mixed Class 1998) in 2002. He was a recipient of the NVIDIA fellowship award in 2005.



Shuai Wang is a postdoctoral researcher in the state key laboratory of CAD&CG, Zhejiang University, Hangzhou, China. He received his Ph.D. degree in condensed matter physics from Institute of Physics, Chinese Academy of Sciences in 2010. His research interests include physically based simulation and computer graphics.



Yiying Tong is an assistant professor at Michigan State University. Prior to joining MSU, He worked as a postdoctoral scholar at Caltech. He received his Ph.D. degree from University of Southern California in 2004. His research interests include discrete geometric modeling, physically-based simulation/animation, and discrete differential geometry. He received the U.S. National Science Foundation (NSF) Career Award in 2010.



Kun Zhou is a Cheung Kong distinguished professor in the Computer Science Department of Zhejiang University, and a member of the State Key Lab of CAD&CG, where he leads the Graphics and Parallel Systems Group. Prior to joining Zhejiang University in 2008, Dr. Zhou was a Leader Researcher of the Internet Graphics Group at Microsoft Research Asia. He received his BS degree and PhD degree in computer science from Zhejiang University in 1997 and 2002, respectively. His research interests include shape modeling/editing, texture mapping/synthesis, real-time rendering and GPU parallel computing.