EG UK Computer Graphics & Visual Computing (2019)
G. K. L. Tam and J. C. Roberts (Editors)

Accelerating Surface Tension Calculation in SPH via Particle Classification & Monte Carlo Integration

F. Zorrilla, J. Sappl, W. Rauch, M. Harders

Interactive Graphics and Simulation Group & Unit of Environmental Engineering
University of Innsbruck, Austria

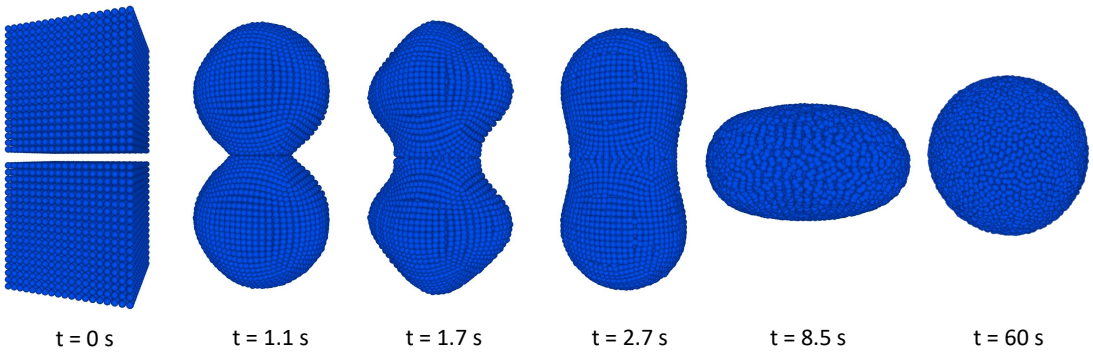

Figure 1: *SPH time evolution of droplets in 3D (initially cuboid), coalescing into a single spherical droplet. Surface tension calculation is accelerated using our method. Convex as well as concave regions can be robustly handled and shape evolution progresses as physically expected.*

Abstract

Surface tension has a strong influence on the shape of fluid interfaces. We propose a method to calculate the corresponding forces efficiently. In contrast to several previous approaches, we discriminate to this end between surface and non-surface SPH particles. Our method effectively smooths the fluid interface, minimizing its curvature. We make use of an approach inspired by Monte Carlo integration to estimate local normals as well as curvatures, based on which the force can be calculated. The technique is applicable, but not limited to 2D and 3D simulations, and can be coupled with any common SPH formulation. It outperforms prior approaches with regard to total computation time per time step, while being stable and avoiding artifacts.

Keywords: SPH fluid simulation, particle classification, estimation of surface normal/tension/curvature

1. Introduction

Surface tension is a phenomenon appearing at the interface of differing media, typically involving a liquid and a gas; such as, for instance, a water-air interface. It results from cohesive forces attracting the molecules of the liquid towards each other. Formally, surface tension is defined as the ratio between the surface force and the distance along which it acts. These forces lead, for instance, to smoothing of fluid surfaces, wherefore they play a vital role in the visual appearance. Accordingly, computational fluid simulations should include estimations of these processes.

In smoothed particle hydrodynamics (SPH) [Mon92], fluids are discretized into particles. Due to this, the interface, e.g. between water and air, is not exactly defined. Therefore, proper ways of approximating the surface tension forces are required. In SPH approaches, these forces are often computed per particle, based on an estimate of the local normal direction as well as of the local curvature. However, some state-of-the-art methods generalize such force calculations to all particles in the fluid, not taking into consideration whether they are located at the surface or not. Technically, this should not introduce any artifacts, since the forces obtained for non-surface particles would be calculated as zero. Nevertheless, computational resources are being spent in the process, without having any effect on the overall fluid behavior.

DOI: 10.2312/cgvc.20191260

Instead of the above, it would be advantageous to first classify particles into surface and non-surface ones, as for instance suggested in [HWZ*14, SCN*16]. Assuming this can be done efficiently, the subsequent surface tension calculation could be accelerated, leading in total to a reduced computation time per simulation step. Related to this notion, a method for SPH surface detection in 2D has been presented in [Dil00]. They classify a particle as part of the surface, if a circle centered at the particle position is not fully overlapped by circles associated with neighboring particles. Inspired by this idea, we propose an extension of the method, with which we first classify particles (in 2D or 3D). For this step, we employ linear regression, based on machine learning techniques. Once the particles are classified, the local normal and curvature have to be obtained. This is realized by a Monte Carlo approach, where the geometry is locally sampled to determine local coverage. The approach only requires the neighborhood geometry, wherefore it is applicable to any currently existing SPH algorithm for fluid simulation. Furthermore, we also suggest adaptive adjustment of the sample resolution, according to the time step. Comparing our approach to state-of-the-art methods for surface tension force estimation, the total simulation runtimes could be consistently reduced with our approach. As an initial example using our method, see Figure 1 – the evolution of two particle sets is depicted, arranged initially as two cubes, following our surface tension calculation. Note the coalescence of both parts, including oscillatory movement over time, while also exhibiting concavities. The final equilibrium, minimizing surface tension energy is, as expected, a spherical droplet.

2. Related Work

Various approaches to calculate surface tension forces in SPH fluids have been proposed in the past. In earlier work, it was attempted to represent surfaces with a smoothed color field, as seen in [Mor00], [MCG03], [KAG*05] and [Kel06]. The latter is a scalar field, which is initially set to one at particle locations, and zero everywhere else. This permits to obtain estimates of surface normals and curvatures. The latter are calculated as the gradient, as well as the divergence of the gradient of the field, respectively. However, the technique usually leads to a random assignment of normals for particles far from the surface. Moreover, errors in the curvature values result and conservation of the fluid momentum is not ensured. The local nature of our method will reduce problems of normal randomness at locations far from the surface, as well as reduce curvature error.

In [BT07], the surface tension force is modeled as a sum of cohesion forces between particles in the same fluid phase. However, the equilibrium of these cohesion forces, as found in simulations, does not always correspond to the correct minimal surface area, as one would expect from a surface tension dominated fluid. The method is also prone to clustering of particles on the fluid surface. To avoid such particle clustering, it was suggested in [TM05] to introduce a repulsion force when particles are too close to each other. This was achieved by manually tuning a force profile according to particle separation distance. In our method particle clustering is avoided since we do not use cohesion forces. Related to this, it was stated in [AAT13] that the surface tension force cannot be estimated as a summation of cohesion forces alone, as observed in nature, since SPH particles represent a fluid on a macroscopic level. Instead, they suggest to combine a cohesion with a surface minimization term. Thus, their force term minimizes fluid surface area, conserves momentum, and prevents clustering. However, forces are manually tuned to attract particles in a certain distance range, while repulsing particles that are too close. In contrast, in [YWTY12] the curvature minimization problem is first solved on a mesh that is reconstructed from the SPH particles; and later the results are transferred back to the particles. The authors encountered surface waves that could appear due to a mismatch between mesh vertices and underlying SPH particles. The effect could be reduced in a post-processing step.

All the methods above treat all particles equally. However, for non-surface particles the resulting force will be zero. Thus, time is spent on calculations that do not have an effect on the simulation. Thus, it may be beneficial to classify particles initially, and then compute forces only for surface particles. One of the first methods that distinguishes between surface and non-surface particles is [HWZ*14]. The force is modeled based on the asymmetric neighborhood of particles close to the surface, which leads to asymmetries in the summation of Van der Waals interactions. This yields a force acting on surface particles, proportional to surface curvature. The work in [SCN*16] introduces a method for surface particle classification based on visual occlusion of particles from different viewpoints. However, the method is computationally intensive and cannot accommodate false positives. In [ZQC*14] surface tension was computed for long, thin objects.

In addition to the above, curvature estimation in general point clouds is also a widely studied topic. In related work, magnitudes proportional to the surface curvature are sometimes computed, but not the exact value itself. This suffers from the similar problem, that also in general point clouds surface curvature may not be exactly defined. Moreover, most existing work already assumes the availability of a surface-only point cloud, e.g. [FK14], [MOG09]. In contrast, our work starts with particle locations in a volume. Finally, also note the relation of the problem to SPH surface reconstruction, e.g via distance fields, such as [ZB05, YT13].

3. Methods

Following the idea of modeling surface tension with a continuum method in [BKZ92], we calculate the surface tension force via $\mathbf{f}_{st}^i = -\sigma\kappa^i\hat{\mathbf{n}}^i$, where κ^i and $\hat{\mathbf{n}}^i$ are surface curvature as well as normal at SPH particle i (note that we employ superscripts to denote particle indices). Further, σ is a constant surface tension parameter, measured in N/m, that depends on the simulated fluid. As mentioned above, we thus have to approximate curvature as well as normal direction per particle.

Our proposed method is organized in three major algorithmic steps. First, particles in an SPH simulation are classified into two groups – surface and non-surface particles. Second, the normal vector is estimated for all surface particles. This makes use of a Monte Carlo technique to locally estimate an integral, taking into account neighboring particles. Due to the probabilistic nature of Monte Carlo computations, the resulting normal vectors are additionally smoothed. Thirdly, following a similar Monte Carlo strategy, we estimate local curvature, again only for the classified surface particles, and again with a subsequent smoothing step. The described

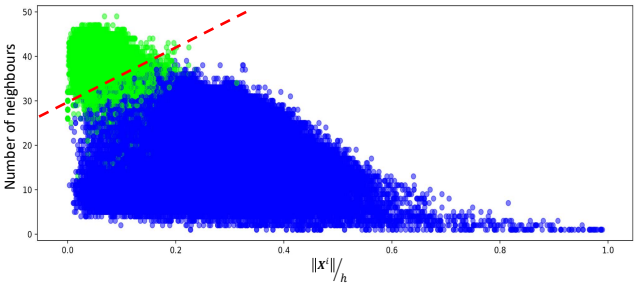

Figure 2: *Setup of 2D feature space for particle classification. Each point represents a surface (blue) or non-surface particle (green), for the test simulation data. The dashed red line indicates the linear classifier, which was shifted such as to result in no false negatives. Note that some false positives are still encountered.*

process can be applied to 2D or 3D scenes. In addition, the number of random samples, and thus the accuracy, can automatically be adjusted according to the simulation time step. Employing the computed data, the surface tension forces per particle can be computed. The individual steps are described in detail in the following.

3.1. Particle classification

The main idea of classifying particles is to reduce the computation time of the surface tension calculation. We aim to achieve this by excluding (ideally all) non-surface particles from the calculation. Doing so should not affect the overall result since for the latter the surface tension force should be zero anyhow. In contrast, it is crucial for the correctness of the result that no surface particle be misclassified (i.e., there should be no *false negatives*). Incorrect classification of non-surface particles as surface ones (i.e., *false positives*) should be minimized, but does not affect the correctness of the fluid dynamics.

In order to properly classify the particles, we experimented with defining various feature spaces. Optimally, this should only depend on the local geometry. As possible features, we examined, for instance, the summation of neighborhood masses, using various weighting kernels. However, it turned out that good results could already be achieved by mapping fluid particles into a simple 2-dimensional feature space. The first component of this space is given by the mass-weighted average distance of particles in a local neighborhood:

$$\left\| \frac{\mathbf{X}^i}{h} \right\| = \frac{1}{h} \frac{\left\| \sum_j m^i \mathbf{x}^j - m^j \mathbf{x}^i \right\|}{\sum_j m^j}, \tag{1}$$

where m and $\mathbf{x}$ are masses and positions of particles i and j, respectively. The neighborhood is defined by the (user-selected) SPH kernel radius h; thus each particle i has associated neighbor particles j, at distances smaller than h. Note that we normalize by dividing the mass-weighted average by h, thus making the feature independent of kernel size. For the second feature dimension, we just employ the number of neighbors per particle N^i.

Next, in order to train a classifier, we have to generate fluid simulation data, and determine for each particle which class it belongs to. The latter training data classification step is done employing a similar strategy as followed for our normal and curvature estimation, as outlined in Sections 3.2 and 3.3. We randomly generate samples on a sphere enclosing a particle and determine coverage of these by neighboring spheres. In order to achieve high accuracy in this classification, we employ a very large number of samples. Further details of the underlying Monte Carlo strategy will be presented below. Simulations to create the training data have been performed using the SPlisHSPlasH framework [Ben17]. Scenes with particle numbers between 2K and 30K are employed. Different particle configurations, obstacles, boundaries, and gravity forces are used, to ensure broad coverage. Thus generated, and initially classified, particles are plotted in our 2-dimensional feature space in Figure 2. Note that using the described features, the two particle classes already exhibit a reasonable separation. It becomes apparent that a linear classifier may already suffice for the classification task.

For the classification step machine learning strategies can be employed. Since we initially worked in higher-dimensional feature spaces, we decided to employ a neural network classifier. However, as discussed above, moving to a 2-dimensional feature space turned out to be sufficient for our purpose. We still employ a neural network as linear classifier, however, using a simpler approach, such as for instance support vector machines would also be adequate. In this context, it should be noted that some recent work explored the use of machine learning in fluid simulation, however, only for obtaining solutions to the Navier–Stokes equations, instead of performing the task of classifying particles (e.g. [TSSP16, CT17, WBT18, JSP*15]).

In our technique, we effectively obtain a line separating the two classes in the feature space. However, since we strive to minimize (i.e., optimally avoid) false negatives, we opted to shift the line in normal direction, such that no false positives remain (i.e., with respect to the training data). The obtained linear classifier is then applied in any new fluid simulation, dividing particles into surface and non-surface ones, progressively per time step. Applying this approach in our tests, we did not encounter any false negatives in these simulations, also with different particle configurations and geometries. Still, false positives do result. In the experiments outlined in Section 4 the method yielded on average 0.014% false positives in the Droplet, 5.66% in the Dambreak, and 4.75% in the Crown splash test scenario, respectively. Still, the method proved to work fast and be robust with regard to false negatives. The performance of the method is three orders of magnitudes faster than the timings reported by [SCN*16] for a double dambreak scenario. Finally, note that if the classifier is not shifted, then surface particles would be falsely classified as non-surface particles. This leads to computational errors, which become visible e.g. as bumps on sphere surfaces; in the end unwanted oscillations and particle motions would result. In future work, we will explore the performance of the method also in the context of multi-scale SPH models, i.e. when quite different sampling densities are employed.

3.2. Normal calculation

Once the classification has been finalized, we have to compute the surface normals, as well as the curvatures, per particle. Since we do not make use of any smoothed field in the fluid we have to calculate both values using only the geometry as input. Both calculations follow a similar notion, wherefore, the general idea of both will be outlined first. The following will address the 3D case, but the concept applies analogously to 2D.

The key idea in both cases is to first assume a sphere S_1 of radius r_1 around a considered surface particle at position $\mathbf{x}^i$. The radius will always be selected equal to the SPH kernel size h. Next, additional spheres S_2^j of radius r_2 are considered, with their respective centers given by all neighboring particles at position $\mathbf{x}^j$ (i.e., all surface and non-surface ones combined). For this, the neighborhood of a particle is again given according to kernel radius h. Also, note that r_2 is usually smaller than r_1. The neighboring spheres S_2^j will overlap the initial sphere S_1, located at particle i, thus leaving a smaller spherical area A_1 that is not overlapped, i.e., not within the neighbor spheres.

Since we work with incompressible or weakly compressible SPH particle distributions the density of the point cloud has to be nearly constant. Thus, it can be conjectured that the surface normal $\mathbf{n}^i$ at the particle will point towards the centroid of the non-overlapped spherical area on the sphere. In addition, as will be discussed in more detail below, we also found that the fraction of the sphere that has not been covered is related to the surface curvature at that point. The area of the sphere that is not overlapped by the neighboring ones can be calculated with a spherical integral. However, this integral can be computationally very expensive to determine exactly, wherefore we propose to estimate it using a Monte Carlo integration strategy.

With regard to the normal computation, we first generate N^i uniformly distributed random sample points $\mathbf{p}^k$ on the surface of sphere S_1 of particle i. Next, we will only consider those that are not inside of any neighboring sphere S_2^j. For our following derivations, we will represent this with a binary function:

$$S(\mathbf{p}^k) = \begin{cases} 0 & \text{if } \mathbf{p}^k \text{ is overlapped,} \\ 1 & \text{if } \mathbf{p}^k \text{ is not overlapped.} \end{cases} \tag{2}$$

Based on this, we obtain a first estimated surface normal:

$$\tilde{\mathbf{n}}^i = \text{nrm}\left(\sum_{k=1}^{N^i} (\mathbf{p}^k - \mathbf{x}^i)\, S(\mathbf{p}^k) \right), \tag{3}$$

where $\text{nrm}(\cdot)$ is a normalization operator returning a unit vector. Elements in this computation process are visualized in Figure 3(b). Also note that non-surface particles will be assigned with a zero vector. Due to the probabilistic nature of our method, discontinuities in the estimated normal field may be encountered; especially, at lower sampling numbers. However, the normal field should be as smooth as possible on the surface of the fluid. Therefore, we propose to carry out an additional smoothing step. First, we compute a weighted average of all neighboring surface particle normals, based on the results obtained in the previous step:

$$\tilde{\mathbf{n}}^i_{Nei} = \sum_{j=1}^{N^i} W(||\mathbf{x}^j - \mathbf{x}^i||)\tilde{\mathbf{n}}^j, \tag{4}$$

employing weight kernel W, again with kernel radius h:

$$W(x) = \begin{cases} 0 & \text{if } x > h, \\ 1 - |x|/h & \text{if } x \leq h. \end{cases} \tag{5}$$

Also note again that the normals of non-surface particles have been set to zero in the previous step. The final smoothed surface particle normal is then obtained by a weighted average of both computed temporary normals:

$$\hat{\mathbf{n}}^i = \text{nrm}\left((1-\tau)\tilde{\mathbf{n}}^i + \tau\tilde{\mathbf{n}}^i_{Nei}\right), \tag{6}$$

where τ is a user selected interpolation parameter. For all our computations we have set it to 0.5. The outcome of the normal smoothing process is also illustrated in Figure 3. Further note that this smoothing step could potentially be repeated several times.

In order to evaluate the accuracy of the proposed normal estimation process we carried out a comparison between analytically defined and our estimated normals. As error measure we determine $\|\hat{\mathbf{n}}^i - \mathbf{n}^a\|$, where $\hat{\mathbf{n}}^i$ is the estimated normal and $\mathbf{n}^a$ the analytically correct one. The latter were both obtained for the geometry of a 2D circle; a random 2D point cloud is generated by sampling the geometry. Next, due to the random nature of our estimation process, we determine as final error value the average of 100 computations. The results of this study are summarized in Table 1. We show both the dependency of the average error on the number of samples as well as on the number of smoothing steps. As can be seen, the higher the number of sampling points, the more accurate the approximation becomes. Also, additional smoothing improves the estimates, by filtering out noise incurred by the representation as a point cloud.

3.3. Curvature calculation

The surface curvature of a 3D surface is locally given by two values, also known as principal curvatures [Gol05]. These are defined as the eigenvalues of the shape operator at a point on the surface. By averaging the two we obtain the mean curvature κ. Gaussian and mean curvature estimation fail with point clouds including noise.

Table 1: *Normal estimation error for 2D circle test case. The average error depends on the number of samples in the Monte-Carlo integration as well as on the number of times the smoothing algorithm is applied.*

	Smoothing Steps		
Samples	0	1	2
10	0.290	0.171	0.131
20	0.201	0.115	0.085
50	0.120	0.068	0.051
100	0.088	0.049	0.035
500	0.038	0.021	0.017

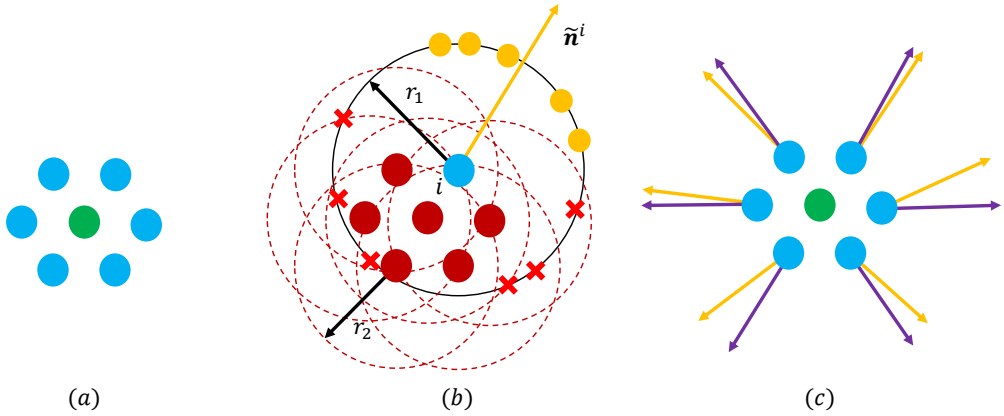

Figure 3: *Visualization of normal estimation and smoothing in 2D. (a): point cloud with surface (blue) and non-surface (green) particles. (b): samples $\mathbf{p}^k$ on circle around surface particle i; red crosses are overlapped by neighbor circles; yellow dots not, thus these are used for normal estimation. (c): initially estimated normals (yellow) and normals after smoothing (purple).*

We have found that it is possible to establish a direct relation between the mean curvature and the fraction of a sphere that is not covered by neighboring ones, via the process outlined above. We begin by noting that the fraction of the uncovered surface area of a sphere, using the mapping function (2), is given by:

$$\lambda = \frac{1}{4\pi r^2} \iint S(\mathbf{x}(\theta,\varphi)) r^2 \sin(\varphi)\, \mathrm{d}\theta \mathrm{d}\varphi\,, \tag{7}$$

where $\mathbf{x}(\theta,\varphi)$ is a sphere surface location with spherical coordinates θ and φ. As before, instead of attempting to compute this value exactly, we will approximate it, for a particle i, based on random samples following a Monte Carlo integration strategy:

$$\lambda \approx \frac{1}{N^i} \sum_{k=1}^{N^i} S(\mathbf{p}^k)\,. \tag{8}$$

As will be seen later, it is possible to estimate κ from λ, which itself can be determined from the randomly sampled points $\mathbf{p}^k$. Note that $\lambda \in [0,1]$. We will first derive the underlying relationship in 2D, and later extend to 3D.

3.3.1. Relationship in 2D

The following derivation is explained while closely referring to the illustration and notation in Figure 4. We start with assuming in 2D a circular outline (shown on the bottom in blue), representing a shape for which the curvature should be determined. The circle has a radius of R, and thus the sought curvature κ is given in this case analytically by the reciprocal $1/R$. However, later the formalism should be applied to any arbitrary shape or curve, based on randomly sampled locations.

First, in order to render our derivation independent of particle size, we will attempt to estimate an adjusted curvature parameter $\tilde{\kappa} = h\kappa$, considering correspondingly also an adapted $\tilde{R} = R/h$. With this in mind, as a starting point for examining in this case the relationship between λ and $\tilde{\kappa}$, we will begin with deriving a lower bound λ_{min}, i.e., in 2D the minimal arc that would not be covered by neighboring circles. For this, first consider a particle i on the circular outline. We associate with this particle again a circle C_1, with radius r_1, and center $\mathbf{x}^i$. Next, consider additional neighboring particles j, akin to what was discussed above; to these again correspond circles C_2^j with centers $\mathbf{x}^j$, all with the same radius $r_2 < r_1$, overlapping circle C_1. Note that the maximal overlap will result for those particles j that are also located on the circular outline; in 2D there would be two of these, next to particle i. Thus, we have to find the geometrical relationship at which circle C_2 around such a particle j would cover a maximal arc on C_1. When the circles overlap, we can find two intersection points; denoting the outer one as $\mathbf{x}^I$, the maximum coverage will result when the vector between $\mathbf{x}^I$ and $\mathbf{x}^j$ is perpendicular to the vector between $\mathbf{x}^i$ and $\mathbf{x}^j$ (see also Figure 4). In this situation, the angle between the normal at particle i and the vector between $\mathbf{x}^i$ and $\mathbf{x}^I$ is given as φ. Also note that this angle can be obtained via:

$$\varphi = \begin{cases} \pi - \alpha - \beta & \text{if } \tilde{\kappa} > 0\,, \\ \beta - \alpha & \text{if } \tilde{\kappa} \leq 0\,, \end{cases} \tag{9}$$

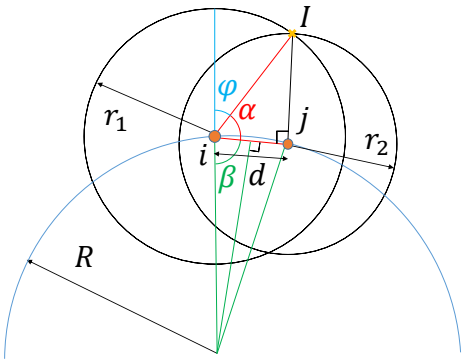

Figure 4: *Configuration for maximal arc coverage of circle around neighbor particle j on circle around particle i.*

where angles α and β are defined based on the chord between the particle positions, as depicted. Also note that the distance between the latter is given as:

$$d = \sqrt{r_1^2 - r_2^2} = h\sqrt{1-(r_2/r_1)^2}. \tag{10}$$

According to the geometric configuration, both angles can be obtained via:

$$\alpha = \sin^{-1}(r_2/r_1), \tag{11}$$

$$\beta = \cos^{-1}\left(\frac{d/2}{R}\right) = \cos^{-1}\left(\tilde{\kappa}/2\cdot\sqrt{1-(r_2/r_1)^2}\right). \tag{12}$$

Finally, due to having two neighboring particles in symmetric configuration, we have to consider 2φ for the non-covered arc. Overall, we obtain $\lambda_{min} = 2\varphi/2\pi$. Using the previous equations, we obtain a closed form solution, independent of the sign of the curvature:

$$\tilde{\kappa} = -2\left(1-(r_2/r_1)^2\right)^{-1/2}\cos\left(\lambda_{min}\pi+\alpha\right), \tag{13}$$

where the adjusted curvature is related to the ratio of the radii r_2/r_1 and the minimal covered fraction λ_{min}, which we approximate via random sampling.

3.3.2. Relationship in 3D

In 3D we follow a similar derivation. As before, we attempt to do this via estimating the ratio of a minimal, uncovered spherical surface area to the complete surface of a sphere. Again, we assume a particle i on this surface, surrounded by several particles j, for which again local spheres of radius r_1 and r_2 are defined. In 3D the uncovered spherical surface area A will be a spherical cap, which is given analytically. A cap on a sphere with radius R, defined by a projected solid angle φ is given as:

$$A = \int_0^{\phi}\int_0^{2\pi} R^2\sin(\varphi)\mathrm{d}\varphi\,\mathrm{d}\theta = 2\pi R^2\left(1-\cos(\varphi)\right). \tag{14}$$

As in the 2D case, we compute λ_{min} based on the non-covered surface area:

$$\lambda_{min} = \frac{2\pi R^2(1-\cos(\varphi))}{4\pi R^2} = \frac{1-\cos(\varphi)}{2}. \tag{15}$$

Thus, rearranging the terms we can also in 3D express the adjusted curvature analytically:

$$\tilde{\kappa} = -2\left(1-(r_2/r_1)^2\right)^{-1/2}\cos\left(\cos^{-1}(1-2\lambda_{min})+\alpha\right), \tag{16}$$

again depending on the ratio of r_2/r_1 and λ_{min}, which we can estimate. For our implementation and tests we employed the ratio $r_2/r_1 = 0.8$, which yielded optimal performance.

The described approach can be applied to estimate the surface curvature of any given point cloud. In Figure 5, we provide examples of curvature calculations, based on our method. Curvatures of a simple cube and a Stanford bunny point cloud are visualized, color-coded from negative (dark blue) to positive (red) values.

In order to further evaluate our curvature estimation method, we compare our approximations with analytically defined mean curvature values. The latter can, for instance, be obtained in closed form for any point on an ellipsoid [Bek16]. Thus, we create an ellipsoidal point cloud, for which we obtain our estimate, and compare to the ground truth. Due to the stochastic nature of our method, we determine the mean and the standard deviation for 40 measurements. Moreover, note that accuracy again depends on the number of samples, wherefore we also tested our method for different amounts of such samples. The results of this validation are compiled in Table 2. As can be seen, for smaller curvatures our estimation approaches the correct solution, independent of the curvature sign. Moreover, even for a small number of samples our estimated average curvature is close to the correct solution. Nevertheless, the standard deviation is large for small sample numbers, but can be reduced by increasing the number of samples. In addition, we found that for larger curvatures, also larger errors in the mean curvature resulted. This is due to radius r_1 becoming closer to actual surface features.

3.3.3. Curvature smoothing and adaptive sampling

Similar as for the normal field, the curvature field should also be smooth along the surface. The probabilistic nature of the estimation process may also introduce artifacts. Thus, we again suggest to apply one or more smoothing steps, averaging computed curvatures in a local neighborhood.

Furthermore, as already seen in Table 1, the accuracy of Monte Carlo approaches will depend on the number of samples. A straightforward approach could be to employ a constant number at all times; however, we have found that adapting the number according to the underlying numerical simulation is advantageous. The idea is inspired by the Courant–Friedrichs–Lewy (CFL) condition [Mon92], which relates numerical time step, spatial discretization, and propagation velocity. According to this, solution time

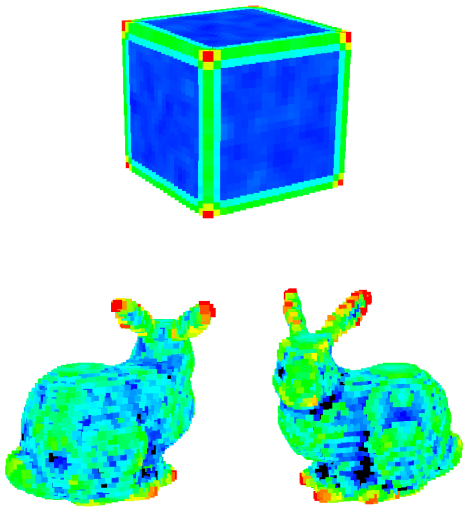

Figure 5: *Computed curvature (dark blue = negative) of test point clouds – cube (top); Stanford bunny (bottom).*

Analytical		0.213	−0.213	0.031	−0.031	0.5	−0.5
N	**50**	0.145 ± 0.365	-0.273 ± 0.332	0.038 ± 0.325	-0.049 ± 0.322	0.424 ± 0.275	-0.634 ± 0.345
	200	0.199 ± 0.125	-0.241 ± 0.159	0.052 ± 0.142	-0.014 ± 0.142	0.431 ± 0.173	-0.584 ± 0.150
	1000	0.196 ± 0.067	-0.238 ± 0.070	0.046 ± 0.070	-0.033 ± 0.068	0.426 ± 0.063	-0.583 ± 0.085
	10000	0.194 ± 0.021	-0.238 ± 0.023	0.030 ± 0.019	-0.034 ± 0.021	0.428 ± 0.019	-0.589 ± 0.021

Table 2: *Comparison of analytically defined (top row) with our estimated (bottom four rows) curvatures, obtained for an ellipsoid with semi-axes ($a = 100$, $b = 200$, $c = 400$); the latter is approximated for our method with a random point cloud. Measurements given both for negative (inside) and positive (outside) curvature. Average estimated curvatures and standard deviations are provided, based on 40 measurements, for different numbers of random samples N.*

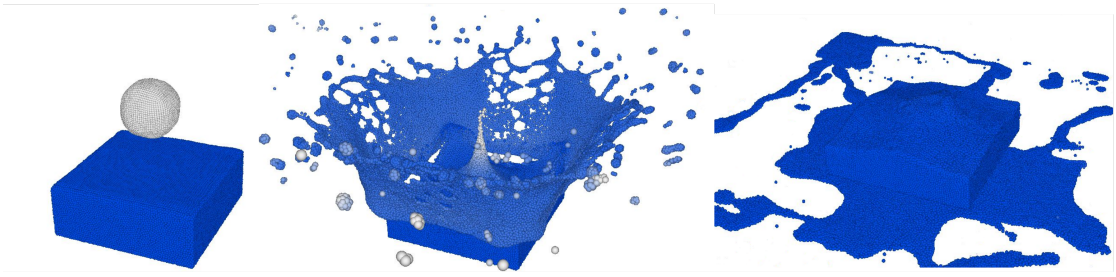

Figure 6: *Example of crown splash simulation using DFSPH and our surface tension force estimation (adaptive sampling with $C_{SD} = 10,000$, adaptive time step 0.1–1 ms).*

steps in SPH algorithms are often adaptively adjusted; commonly based on forces or velocities of the fluid particles. Along this line, we propose to adjust the number of random sampling points used per time step as $N = t_s C_{SD}$, with time step t_s and user-selected proportionality constant C_{SD}. The latter can be considered as a sampling density factor, its value representing a trade-off between accuracy and computation speed. We have achieved good results with setting this parameter to 10,000–100,000. Our adaptive sampling makes the total number of samples per particle over a simulation time period independent of the numerical time step size.

4. Results

In order to further evaluate our method we compare it to approaches employed in prior work for the computation of surface tension, specifically the work by Becker and Teschner (2007) [BT07] and by Akinci, Akinci, and Teschner (2013) [AAT13]. Further, the surface tension calculations are integrated into different full SPH solvers, specifically weakly compressible SPH (WCSPH) [BT07], predictive corrective incompressible SPH (PCISPH) [SP09], and divergence free SPH (DFSPH) [BK17, BK15], in a reference implementation. As framework for the comparisons we employ "SPlisHSPlasH" [Ben17], an open source environment for physically-based simulation of fluids, which provides the implementations for the mentioned comparison algorithms. All computations were performed using only the CPU; i.e., no optimizations, such as GPU calculations were employed. Further, we obtain computation times for three different common test scenes, as also suggested by [HRWE15] – Droplet, Crown splash, and Dambreak. These scenes cover various dynamic behaviors, and also require different time step intervals, according to the CFL condition. That is, for Droplet, WCSPH: 1 ms, DFSPH/PCISPH: 5 ms; Crown splash, all 0.1–1 ms; Dambreak, all 0.5–2 ms. Finally, our proposed adaptive adjustment of sample numbers is employed with $C_{SD} = 10,000$. Snapshots of two example simulations of these experiments are shown in Figures 6 and 7.

The timing results of the comparison experiment are provided in Table 3. As can be seen, in all cases our method resulted in reduced average total computation times. Note that in all cases visually highly similar fluid simulation results were obtained, and no instabilities were encountered. Moreover, the smaller the time step, the better our method performed compared to the other two surface tension calculation methods, when computing $\mathbf{f}_{st}$; this becomes especially evident for the Crown splash scene, which employed the smallest time step, where significant improvements resulted for this step. However, for larger time steps, the advantage of employing our proposed approach for calculating $\mathbf{f}_{st}$ is reduced; for instance, in the droplet scene, for both DFSPH and PCISPH, the surface tension calculation times turned out to be slower for our method; nevertheless, the total computation time per time step still remained better. We assume this to be due to non-surface particles also experiencing non-zero surface tension forces in the other methods, which requires additional iterations of the pressure solver to achieve the correct fluid density.

	Scene	N	Becker 2007 [BT07] $\mathbf{f}_{st}$ [ms]	Becker 2007 [BT07] Total [ms]	Akinci 2013 [AAT13] $\mathbf{f}_{st}$ [ms]	Akinci 2013 [AAT13] Total [ms]	Our method $\mathbf{f}_{st}$ [ms]	Our method Total [ms]
DFSPH	Droplet	10100	**6.60**	41.89	10.20	49.22	8.95	**38.43**
	Crown	145000	62.19	478.44	103.89	508.48	**17.25**	**424.81**
	Dam Break	26100	16.13	116.05	19.49	111.48	**5.44**	**74.17**
PCISPH	Droplet	10100	**7.15**	98.91	11.10	129.17	9.62	**51.09**
	Crown	145000	87.18	1089.16	115.36	961.35	**19.61**	**859.49**
	Dam Break	26100	18.01	435.23	22.11	353.40	**6.86**	**337.70**
WCSPH	Droplet	10100	7.46	26.64	11.28	30.22	**3.96**	**23.82**
	Crown	145000	83.93	312.13	112.73	349.34	**18.28**	**258.49**
	Dam Break	26100	15.67	61.90	22.38	69.13	**9.67**	**56.71**

Table 3: *Average computation times in milliseconds per simulation time step, for surface tension calculation as well as the complete SPH solutions (lowest values in bold font). Three different SPH solvers were employed (DFSPH, PCISPH and WCSPH), as well as three different test scenes (Droplet, Crown, Dam Break), all computed in SPlisHSPlasH. Our proposed adaptive adjustment of sample numbers is employed; also time steps are adapted dynamically according to the CFL condition.*

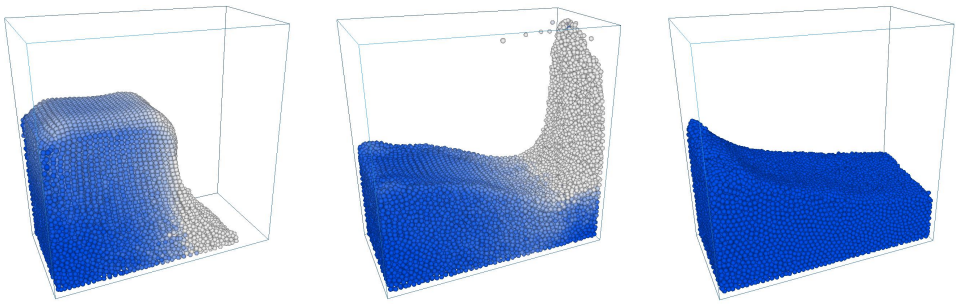

Figure 7: *Example of dam break simulation using DFSPH and our surface tension force estimation (adaptive sampling with $C_{SD} = 10,000$, adaptive time step 0.5–2 ms).*

5. Discussion & Conclusion

We have presented a method to accelerate the calculation of surface tension forces in SPH fluid animations. In contrast to other approaches, we discriminate between surface and non-surface particles. This leads to an improvement in the computation time, since the forces are calculated for just a fraction of the particles. Based on this, we can effectively smooth and minimize the surface of the fluid. The accuracy of our method can be tuned by adjusting the value of C_{SD}. When the time step is reduced, e.g. according to the CFL condition, the number of sampling points N^i is also adjusted. Surprisingly, we found that even for a low number, $N^i \approx 10$ the simulation remains stable. It is in cases when the time step is small (< 1 ms) that our method offers a considerably improved performance over the other tested methods. However, when the simulation runs slower ($t_s \approx 5$ ms) the advantage is diminished; still the computation of our method remains comparable to other algorithms.

A disadvantage we encountered when using a very low resolution in the sampling is the possible creation of incorrect momentum. The sum of the forces around a closed fluid surface should vanish, but for low resolution sampling of the integral this is not ensured. We will examine in future work possible strategies to avoid this artifact. Our approach also includes an estimation of the local mean curvature at the fluid particles; here, the latter could be considered as a general point cloud. Since the surface interface in any point cloud is not clearly defined, the curvature is neither. Related works, e.g. [Mor00, MCG03, KAG*05], compute the divergence of the gradient of the color field to estimate the curvature. This leads to a quantity that is only proportional to the exact curvature. In our method we obtain an approximation of the curvature based on a spherical integral. The procedure is similar to searching for a spherical surface, locally best fitting to a point cloud. The mean curvature would be calculated from the radius of such a sphere.

Further, note that instead of employing randomly generated uniform samples, we also explored the use of pseudo-random Halton sequences [BS91]. Due to the deterministic nature of the latter, it may be possible to save further computation time. This will be explored in more detail in the future. Finally, our method can effectively be coupled with any other SPH algorithm, since it only takes the geometry as input for the computation. It can be employed to improve the overall SPH computation time, when smoothing fluid surfaces in computer graphics applications. It is left for future work to explore the possibility of applying this type of procedure in other contexts of fluid simulations.

References

[AAT13] AKINCI N., AKINCI G., TESCHNER M.: Versatile surface tension and adhesion for SPH fluids. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 182. 2, 7, 8

[Bek16] BEKTAS S.: Generalized Euler formula for curvature. *Intl. Journal of Research in Engineering and Applied Sciences 6*, 3 (2016). 6

[Ben17] BENDER, JAN: SPlisHSPlasH. https://github.com/InteractiveComputerGraphics/SPlisHSPlasH, 2017. 3, 7

[BK15] BENDER J., KOSCHIER D.: Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics symposium on computer animation* (2015), ACM, pp. 147–155. 7

[BK17] BENDER J., KOSCHIER D.: Divergence-free SPH for incompressible and viscous fluids. *IEEE Trans. on Visualization and Computer Graphics 23*, 3 (2017), 1193–1206. 7

[BKZ92] BRACKBILL J. U., KOTHE D. B., ZEMACH C.: A continuum method for modeling surface tension. *Journal of computational physics 100*, 2 (1992), 335–354. 2

[BS91] BERBLINGER M., SCHLIER C.: Monte Carlo integration with quasi-random numbers: some experience. *Computer physics communications 66*, 2-3 (1991), 157–166. 8

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), Eurographics Association, pp. 209–217. 2, 7, 8

[CT17] CHU M., THUEREY N.: Data-driven synthesis of smoke flows with CNN-based feature descriptors. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 69. 3

[Dil00] DILTS G. A.: Moving least-squares particle hydrodynamics ii: conservation and boundaries. *Intl. Journal for numerical methods in engineering 48*, 10 (2000), 1503–1524. 2

[FK14] FOORGINEJAD A., KHALILI K.: Umbrella curvature: a new curvature estimation method for point clouds. *Procedia Technology 12* (2014), 347–352. 2

[Gol05] GOLDMAN R.: Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design 22*, 7 (2005), 632–658. 4

[HRWE15] HUBER M., REINHARDT S., WEISKOPF D., EBERHARDT B.: Evaluation of surface tension models for sph-based fluid animations using a benchmark test. In *VRIPHYS* (2015), pp. 41–50. 7

[HWZ*14] HE X., WANG H., ZHANG F., WANG H., WANG G., ZHOU K.: Robust simulation of sparsely sampled thin features in SPH-based free surface flows. *ACM Transactions on Graphics (TOG) 34*, 1 (2014), 7. 2

[JSP*15] JEONG S., SOLENTHALER B., POLLEFEYS M., GROSS M., ET AL.: Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics 34*, 6 (2015), 199. 3

[KAG*05] KEISER R., ADAMS B., GASSER D., BAZZI P., DUTRÉ P., GROSS M.: A unified Lagrangian approach to solid-fluid animation. In *Symposium Point-Based Graphics Proceedings* (2005), IEEE, pp. 125–148. 2, 8

[Kel06] KELAGER M.: Lagrangian fluid dynamics using smoothed particle hydrodynamics. *University of Copenhagen: Department of Computer Science* (2006). 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), Eurographics Association, pp. 154–159. 2, 8

[MOG09] MÉRIGOT Q., OVSJANIKOV M., GUIBAS L.: Robust voronoi-based curvature and feature estimation. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling* (2009), ACM, pp. 1–12. 2

[Mon92] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics 30*, 1 (1992), 543–574. 1, 6

[Mor00] MORRIS J. P.: Simulating surface tension with smoothed particle hydrodynamics. *International journal for numerical methods in fluids 33*, 3 (2000), 333–353. 2, 8

[SCN*16] SANDIM M., CEDRIM D., NONATO L. G., PAGLIOSA P., PAIVA A.: Boundary detection in particle-based fluids. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 215–224. 2, 3

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. In *ACM transactions on graphics (TOG)* (2009), vol. 28, ACM, p. 40. 7

[TM05] TARTAKOVSKY A., MEAKIN P.: Modeling of surface tension and contact angles with smoothed particle hydrodynamics. *Physical Review E 72*, 2 (2005), 026301. 2

[TSSP16] TOMPSON J., SCHLACHTER K., SPRECHMANN P., PERLIN K.: Accelerating Eulerian fluid simulation with convolutional networks. *preprint arXiv:1607.03597* (2016). 3

[WBT18] WIEWEL S., BECHER M., THUEREY N.: Latent-space Physics: Towards learning the temporal evolution of fluid flow. *arXiv preprint arXiv:1802.10123* (2018). 3

[YT13] YU J., TURK G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Trans. Graph. 32*, 1 (Feb. 2013), 5:1–5:12. 2

[YWTY12] YU J., WOJTAN C., TURK G., YAP C.: Explicit mesh surfaces for particle based fluids. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 815–824. 2

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (July 2005), 965–972. 2

[ZQC*14] ZHU B., QUIGLEY E., CONG M., SOLOMON J., FEDKIW R.: Codimensional surface tension flow on simplicial complexes. *ACM Trans. Graph. 33*, 4 (July 2014), 111:1–111:11. 2