Pacific Graphics 2020
E. Eisemann, A. Jacobson, and F.-L Zhang
(Guest Editors)

Volume 39 (2020), Number 7

Semi-analytical Solid Boundary Conditions for Free Surface Flows

Yue Chang[1], Shusen Liu[2,3], Xiaowei He[†2], Sheng Li[‡4,5] and Guoping Wang[4,5]

[1]School of Software & Microelectronics, Peking University, China
[2]State Key Lab. of CS, ISCAS, China
[3]University of Chinese Academy of Sciences, China
[4]Dept.of computer science and technology, Peking University, China
[5]Beijing Engineering Center of Virtual Simulation and Visualization, China

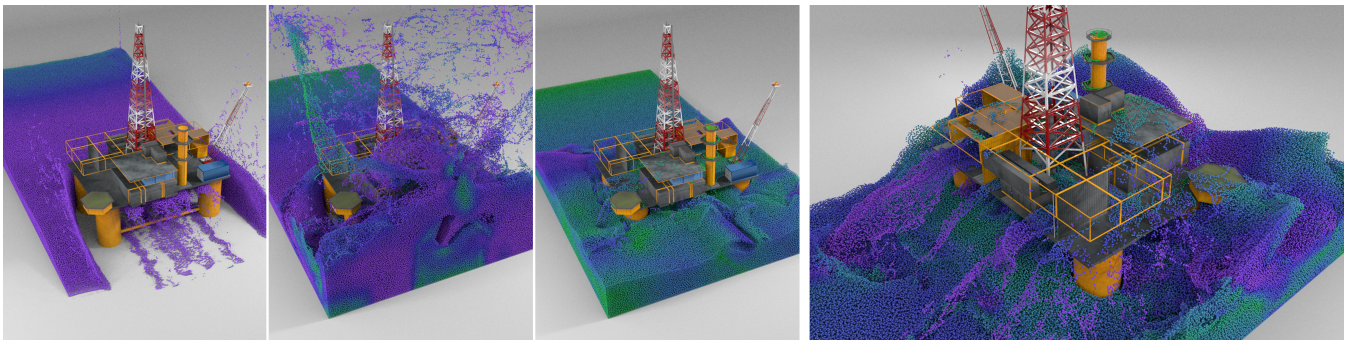


Figure 1: *Dam breaking with a drilling platform. The proposed semi-analytical solid boundary handling approach is integrated into a projection-based method to support coupling between particles and boundary triangles. Velocities of particles are color coded.*

Abstract

*The treatment of solid boundary conditions remains one of the most challenging parts in the SPH method. We present a semi-analytical approach to handle complex solid boundaries of arbitrary shape. Instead of calculating a renormalizing factor for the particle near the boundary, we propose to calculate the volume integral inside the solid boundary under the local spherical frame of a particle. By converting the volume integral into a surface integral, a computer aided design (CAD) mesh file representing the boundary can be naturally integrated for particle simulations. To accelerate the search for a particle's neighboring triangles, a uniform grid is applied to store indices of intersecting triangles. The new semi-analytical solid boundary handling approach is integrated into a position-based method [MM13] as well as a projection-based [HWW*20] to demonstrate its effectiveness in handling complex boundaries. Experiments show that our method is able to achieve comparable results with those simulated using ghost particles. In addition, since our method requires no boundary particles for deforming surfaces, our method is flexible enough to handle complex solid boundaries, including sharp corners and shells.*

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation;***

1. Introduction

Smoothed particle hydrodynamics (SPH) is well suited for simulating various phenomena including water splashes [SP09, ICS*14, BK15], multiphase flows [YCR*15, YML*17] as well as viscous fluids [TDF*15, PICT15, WKBB18]. For a thorough review of its application in computer graphics, we refer to a recent tutorial provided by Koschier et al. [KBST19]. However, treating solid wall boundaries still remains one of the most challenging parts in SPH. Over the past three decades, a variety of different strategies (e.g., ghost particles [AIA*12], density maps [KB17], semi-analytical boundaries [FLR*13], etc) have been developed for implementing solid walls in SPH , each with its advantages and disadvantages.

† xiaowei@iscas.ac.cn
‡ lisheng@pku.edu.cn

According to the standard SPH formulation, if a particle is located near the solid boundary, only particles inside the fluid domain contribute to the summation of the particle interaction. This one-sided particle approximation does not give correct solutions, because the field variables inside the solid boundary usually cannot be reduced to zero, e.g., when calculating the particle density. To develop a practical solid boundary handling technique, both accuracy and efficiency should be considered in priority. Among all solid boundary handling techniques, ghost particles are most commonly used. However, applying ghost particles is not an elegant way to deal with large planar regions. To fully remove the particle deficiency problem, one or multiple layers of ghost particles are necessary to be sampled [AIA*12, HLW*12]. Unfortunately, the large number of ghost particles has a great side effect to the simulation performance. To reduce the computational cost, semi-analytical methods aim to convert the volume integral over the boundary into a surface integral. However, existing methods either cannot handle complex boundary geometries [KBLP04,FM15] or are only suited to take the boundary integral for the gradient term [FLR*13, MFK*15].

In this paper, we propose to take the semi-analytical approach of [KBLP04, FM15] and extend it to support complex solid boundaries of arbitrary shape. To compute the kernel correction, Fujisawa and Miura [FM15] have simplified solid boundaries into planar ones, which works well for simple boundaries. However, the results are unpredictable for boundaries of complex geometries, e.g., those with sharp bumpy corners. We instead propose to take the volume integral inside the solid boundary. We reformulate the integral under the local spherical frame of each particle and then convert it into a surface integral. To avoid introducing a discontinuity between adjacent triangles, we propose to cluster all triangles that belong to the same plane together and apply the same integration method for each cluster. To accelerate the query for a particle's neighboring triangles, we propose to use a uniform grid to store triangle indices. We do the intersection test between a triangle and a grid cell, then store the triangle index into the corresponding grid cell if intersection occurs. During the particle simulation, the query for neighboring triangles can then be executed in parallel on modern GPUs. Finally, we integrate our semi-analytical approach into a position-based method [MM13] as well as a projection-based incompressible SPH solver [HWW*20].

Experiments show that our method is able to achieve comparable results with those simulated with ghost particles. No boundary particles are required for deforming surfaces, therefore, our method is flexible enough to handle complex solid boundaries, including sharp corners and shells.

2. Related Work

Over the past three decades, many boundary handling techniques have been developed in SPH. We review the most popular and commonly used ones in this section.

Particle-based methods. The idea of introducing boundary particles for incompressible fluid surface flows can be traced back to the repellent-particle approach [Mon94, MK09]. This method is easy to implement as the Lennard-Jones potential force in its original form is typically modeled as the repulsion between a fluid particle and a boundary particle. Due to its flexibility, Becker and Teschner [BT07] applied repellent boundary particles in a weakly compressible SPH solver for free surface flows. Becker et al. [BTT09] extended the repellent-particle approach by applying direct forcing based on a predictor-corrector scheme for rigid-fluid coupling, thus a large range of slip and Neumann boundary conditions can be imposed for arbitrarily boundaries. Ihmsen et al. [IAGT10] presented a new boundary method by combining the idea of direct forcing with the pressure-based frozen-particles method [SSP07] to enforce non-penetration of rigid objects for large time steps. Nevertheless, the repellent-particle approach still leads to spurious behaviours. For example, it is impossible to maintain particles stationary next to a vertical wall in the presence of gravity with hydrostatic conditions. Libersky and Petschek [PL93] therefore introduced ghost particles to reflect a symmetrical surface boundary condition. Randies and Libersky [RL96] extended the ghost particle method to more general boundary conditions by assigning the same boundary value of a field variable to all the ghost particles. Compared to the repellent-particle approach, the ghost-particle approach shows smoother behavior of the particles in proximity of the solid boundary [CL03]. Marrone et al. [MAC*11] proposed to use fixed ghost particles to enhance treatment of solid boundaries. Unlike the standard ghost particle technique which mirrors each particle near the solid boundary into a ghost at each time step, fixed ghost particles are created only once at the beginning of the simulation. Therefore it is possible to enforce both Dirichlet and Neumann conditions through a moving least-square interpolation. Based on fixed ghost particles, He et al. [HLW*12] introduced staggered particles to flexibly control diverse solid boundary conditions. Akinci et al. [AIA*12] proposed to sample boundary particles only on the surface of rigid objects, which has the advantage of allowing us to deal with lower-dimensional rigid bodies including shells and rods. Their work was later extended to support elastic solids [ACAT13]. To resolve perceivable oscillations of particles in planar regions due to erroneous computation of boundary normals, Band et al. [BGT17] applied a moving least square method [ABC*03] to improve the accuracy in computing surface normals and the distance information. Gissler et al. [GPB*19] presented a strong fluid-rigid coupling for SPH fluids and rigid bodies with particle-sampled surfaces.

Grid-based methods. Solid wall boundaries can alternatively be sampled into Eulerian grids, among which the signed distance field is most commonly used to prevent particles from penetrating into the wall [HKK07, APKG07]. However, without solving the particle deficiency problem, numerical artifacts can arise near the solid boundaries. Koschier and Bender [KB17] proposed to extend the fluid's density field into the boundary geometry and discretize the function using cubic polynomials on a sparse grid without any dependence on particle sampling. To further improve the performance, Bender et al. [BKWK19] suggested not to precompute the density field on the grid, but instead determine the intersection volume between a particle's support domain and the boundary on a coarse grid. Compared to particle-based boundary handling techniques, the grid-based methods shows better performance of neighborhood searches. However, the grid-based methods are not flexible enough to handle solid boundaries of arbitrary shape. For example,

it is impossible to discretize lower-dimensional shells or rods with Eulerian grids.

Semi-analytical methods. Since the computational efforts required for both particle-based and grid-based boundary techniques are not negligible, Kulasegaram et al. [KBLP04] introduced a wall renormalization factor by using a semi-analytical formulation to account for the missing area of the kernel support. Unfortunately, this attempt did not present a clear and simple way to compute renormalization terms for all geometries. To handle solid boundaries of arbitrary shape, Ferrand et al [FLR*13] proposed a new approach to calculate the renormalization factor in 2D space by considering the local shape of a wall as well as the position of a particle relative to the wall. Mayrhofer et al. [MFK*15] extended the proposed semi-analytical boundary conditions to 3D. Chiron et al. [CdOL19] further adapted the Laplacian operator to the fully discrete approach of the walls. However, their method only considers how to take the boundary integral for the gradient term. For an arbitrary function, its integral over the boundary volume usually cannot be converted into a surface integral with the divergence theorem. Alternatively, Monaco et al. [MMG*11] proposed to take the boundary integration under the local spherical frame of each particle. Fujisawa et al. [FM15] follow the same principle and improved the density calculation for PBF [MM13]. Tang et al. [TCJ20] have recently integrated the semi-analytical wall boundary conditions into a projection method to model incompressible and divergence-free flow with a free surface.

3. Semi-analytical Boundary Handling

In the standard SPH methodology, a continuum is represented with a set of particles denoted by the subscript i in a domain $\mathcal{F}$. Consider a smooth function f defined over the domain $\mathcal{F}$, the standard SPH interpolation of f at particle i is given by

$$f_i = \sum_j V_j f_j W_{ij}, \tag{1}$$

where j denotes all neighboring particles, V_j is the volume of particle j, $f_j = f(\mathbf{x}_j)$ and $W_{ij} = W(\|\mathbf{x}_i - \mathbf{x}_j\|, h)$ with h being the support radius. In general, the above particle approximation of f is of second order accuracy [LL03]. However, if a particle is located near the boundary, where the normalization condition is not satisfied, the value of f will be underestimated due to the particle deficiency. A feasible solution to solve this problem is to renormalize f_i as follows [RL96]

$$f_i = \frac{1}{\gamma_i} \sum_j V_j f_j W_{ij}, \tag{2}$$

where $\gamma_i = \sum_j V_j W_{ij}$ is the integral of the SPH kernel W inside the computational domain $\mathcal{F}$. The advantage of applying Equation 2 is its simplicity and improved accuracy over standard SPH [HLL*12]. However, in simulating incompressible free surface flows, if the solid is not explicitly modeled, it is not easy to distinguish numerical errors caused by solid wall boundaries from those caused by free surface boundaries. As demonstrated in Figure 2, although the two particles are located at different places, they will be treated the same if Equation 2 is applied to compute the corresponding physical quantities.

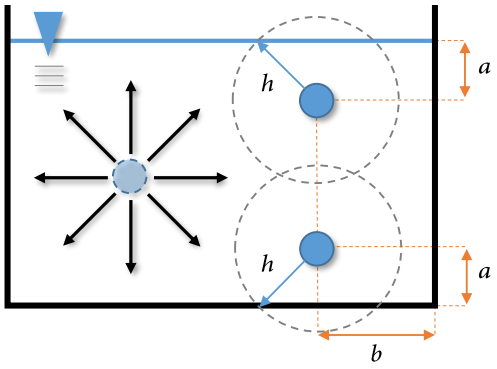


Figure 2: *Sketch of a static water.*

Before diving into the details of our semi-analytical solid boundary handling approach, let us revisit Equation 1 from a continuum approach. If a particle is located near the solid wall boundary, the integral representation of the function f has the following identity

$$\begin{aligned} f(\mathbf{x}) &= \int_{\mathcal{D}} f(\mathbf{x}')\, \omega(\mathbf{x}-\mathbf{x}', h)\, d\mathbf{x}' \\ &= \int_{\mathcal{F}} f(\mathbf{x}')\, \omega(\mathbf{x}-\mathbf{x}', h)\, d\mathbf{x}' + \int_{\mathcal{B}} f(\mathbf{x}')\, \omega(\mathbf{x}-\mathbf{x}', h)\, d\mathbf{x}' \\ &= f^{\mathcal{F}}(\mathbf{x}) + f^{\mathcal{B}}(\mathbf{x}), \end{aligned} \tag{3}$$

where $\mathcal{D}$ denotes the support domain of a particle, $f^{\mathcal{F}}(\mathbf{x})$ denotes the weighted integral inside the computational domain $\mathcal{F}$ and $f^{\mathcal{B}}(\mathbf{x})$ denotes the weighted integral inside the solid boundary $\mathcal{B}$. The value of $f^{\mathcal{F}}(\mathbf{x})$ can be simply calculated according to Equation 1. The difficulty lies in how to calculate $f^{\mathcal{B}}(\mathbf{x})$ efficiently and accurately.

For the convenience of the following discussion, let us denote

$$g(\mathbf{x}, \mathbf{x}') \equiv f(\mathbf{x}')\, \omega(\mathbf{x}' - \mathbf{x}). \tag{4}$$

According to the discussion in [FLR*13], if there exists a function $\mathbf{G}(\mathbf{x}, \mathbf{x}')$ satisfying

$$g(\mathbf{x}, \mathbf{x}') = \nabla_{\mathbf{x}'} \cdot \mathbf{G}(\mathbf{x}, \mathbf{x}') \tag{5}$$

with $\nabla_{\mathbf{x}'}\cdot$ representing the divergence operator with respect to $\mathbf{x}'$, the volume integral of $g(\mathbf{x}, \mathbf{x}')$ over the boundary $\mathcal{B}$ can be transferred to a surface integral of $\mathbf{G}(\mathbf{x}, \mathbf{x}')$ over $\partial\mathcal{B}$. For an arbitrary function $g(\mathbf{x}, \mathbf{x}')$, finding such a function $\mathbf{G}(\mathbf{x}, \mathbf{x}')$ is never an easy task. However, if f belongs to a radial basis function (RBF), i.e., $f(\mathbf{x}') = f(\|\mathbf{x}' - \mathbf{x}\|)$, the derivation can be greatly simplified. By invoking that the SPH kernel function ω is a radial basis function as well, it is possible to rewrite the integral of f inside $\mathcal{B}$ as

$$\begin{aligned} f^{\mathcal{B}}(\mathbf{x}) &= \int_{\mathcal{B}} g(\mathbf{x}, \mathbf{x}')\, dV \\ &= \int_{\Omega} \left(\int_{r(\theta,\varphi)}^{h} g(r)\, r^2 dr \right) \sin\theta d\theta d\varphi \\ &= \int_{\Omega} G(r)|_{r(\theta,\varphi)}^{h}\, d\Omega \end{aligned}, \tag{6}$$

where Ω is the solid angle that measures the amount of field of view

that the boundary $\mathcal{B}$ covers from $\mathbf{x}$, $G(r)$ is a single variable function satisfying $G'(r) = g(r)r^2$ with $r = \|\mathbf{x}' - \mathbf{x}\|$, $G(r)|^h_{r(\theta,\varphi)} = G(h) - G(r(\theta,\varphi))$ and $d\Omega = \sin\theta d\theta d\varphi$ is a formula for the differential in spherical coordinates. The advantage of applying Equation 6 is that the boundary integral is taken over a two-dimensional space, therefore, a computer aided design (CAD) mesh file representing the boundary can be integrated for the boundary integral.

Comparison to [FLR*13]. The question is whether we can still apply the method in [FLR*13] to take the integration for an arbitrary radial basis function $g(r)$. To answer this question, let us take $g(r) = 1 - r$ $(r \leq 1)$ for example. By invoking that $\mathbf{G}(\mathbf{x},\mathbf{x}') = h(r)\frac{(\mathbf{x}-\mathbf{x}')}{r^3}$ meets $\nabla_{\mathbf{x}'} \cdot \mathbf{G}(\mathbf{x},\mathbf{x}') = g(r)$ if there exists a function $h(r)$ satisfying $h'(r) = g(r)r^2$, the formulation of $\mathbf{G}(\mathbf{x},\mathbf{x}')$ for $g(r) = 1 - r$ can be written as

$$\mathbf{G}(\mathbf{x},\mathbf{x}') = (\frac{1}{3} - \frac{r}{4} + \frac{C}{r^3}) * (\mathbf{x} - \mathbf{x}'), \tag{7}$$

where C is a constant. To integrate 7 over the surface boundary, we actually fall into a dilemma. If we set $C = 0$, the value of $\mathbf{G}(\mathbf{x},\mathbf{x}')$ is not zero on the boundary of the support domain. Otherwise, if we set $C = -1/12$ (which guarantees $\mathbf{G}(\mathbf{x},\mathbf{x}') = 0$ for $r = 1$), the surface integral may involves singular values. Since no feasible solutions are available in Ferrand et al. [FLR*13]'s original work to handle above two cases, it is not possible to directly apply their method to take the integration for arbitrary radial basis functions.

4. Numerical Computation of the Boundary Integral

Since triangular meshes are most widely used in computer graphics, we only discuss how to compute $f^{\mathcal{B}}$ for boundary representation with triangular meshes. We believe other types of surface meshes are compatible with our method as well. We will first present general principles before then describing the details for special cases. For simplicity, let us consider an infinitesimal triangle s that is located completely inside the support domain of a particle i, as shown in Figure 3. By applying an one-point Gaussian quadrature rule [GW69], the boundary integral $f^{\mathcal{B}}_{is}(\mathbf{x})$ over the infinitesimal triangle s results in the following approximation

$$f^{\mathcal{B}}_{is} = G(r)|^h_{r(\|\mathbf{x}_s - \mathbf{x}_i\|)}\,\Omega_s, \tag{8}$$

where $\mathbf{x}_s$ represents a sampling point for the one-point Gauss quadrature, Ω_s represents the solid angle of triangle s with respect to $\mathbf{x}_i$. From Equation 8, it can be noted that the accuracy of $f^{\mathcal{B}}_{is}$ depends on how we select the sampling point $\mathbf{x}_s$ as well as on how we compute Ω_s. For the special case when all three vertices are inside the support domain, Oosterom and Strackee [OS83] provided a useful formula for calculating the solid angle subtended by a triangular surface. However, their method cannot be easily extended to deal with more complex cases, e.g., when a triangle intersects with the boundary of particle i's support domain. To uniformly compute Ω_s for all possible intersections between a triangle and a particle's support domain, let us present the standard formula which calculates Ω_s as follows

$$\Omega_s = \frac{(\mathbf{n}_s \cdot \mathbf{d}_s)A_s}{l_s^2}, \tag{9}$$

where $\mathbf{n}_s$ is the normal vector of triangle s, A_s denotes the triangle area, l_s denotes the distance from $\mathbf{x}_i$ to triangle s and $\mathbf{d}_s$ is a

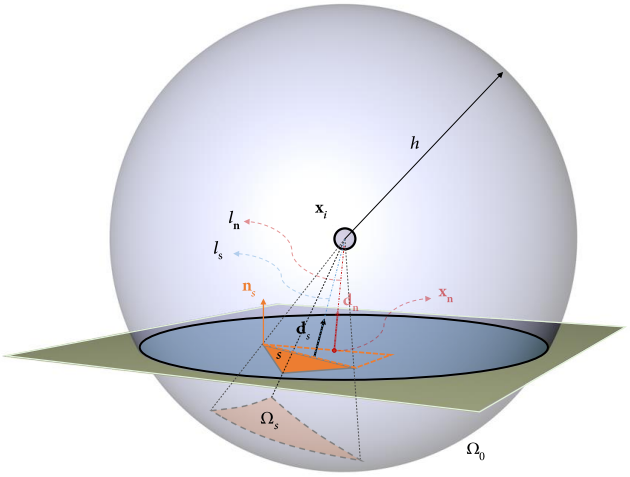


Figure 3: *Sketch of the intersection between particle i's support domain and the solid boundary.*

normalized vector pointing from the closest point on triangle s to $\mathbf{x}_i$. Note that the term $(\mathbf{n}_s \cdot \mathbf{d}_s)A_s$ is an approximation of the spherical surface area, whose value is usually underestimated. Since the value of l_s^2 is also underestimated, the final result is mostly overestimated compared to an analytical solution. To demonstrate this problem, we apply Equation 9 to compute the solid angle of the intersection of a plane with a particle's support domain. As shown in Figure 4, if the plane is discretized into a low-resolution triangular mesh, the numerical result is overestimated. Its accuracy can be improved by increasing the mesh resolution. According to our test, the numerical value matches the exact value when the mesh resolution is increased by an order of 64, which means around 1000 triangles will be located inside the support domain of the particle. The large number of neighboring triangles makes it impractical for implementing a particle simulation.

Ideally, it is preferred to use a coarse mesh to reduce the computational cost, we therefore propose to calculate the solid angle as follows

$$\Omega_s = \frac{(\mathbf{n}_s \cdot \mathbf{d}_\mathbf{n})A_s}{A_0}\Omega_0, \tag{10}$$

where A_0 is the area of the intersection between the plane and particle i's support domain, Ω_0 denotes the solid angle of A_0, $\mathbf{d}_\mathbf{n}$ denotes the normalized vector from the closest point $\mathbf{x}_\mathbf{n}$ on coplanar triangles to $\mathbf{x}_i$. The formulae which compute A_0 and Ω_0 are written as

$$A_0 = \pi\left(h^2 - l_\mathbf{n}^2\right) \quad and \quad \Omega_0 = 2\pi\left(1 - \frac{l_\mathbf{n}}{h}\right) \tag{11}$$

with $l_\mathbf{n}$ representing the distance from $\mathbf{x}_i$ to coplanar triangles. Note that the major difference in Equation 10 is that we use $\mathbf{d}_\mathbf{n}$ instead of $\mathbf{d}_s$. In other words, if two triangles are located on the same plane, the same vector $\mathbf{d}_\mathbf{n}$ will be used to compute solid angles. This triangle clustering algorithm also helps remove the discontinuity between adjacent triangles, as demonstrated in Figure 5. Figure 4 shows Equation 10 gets the exact solution for all three different resolution of triangular meshes.

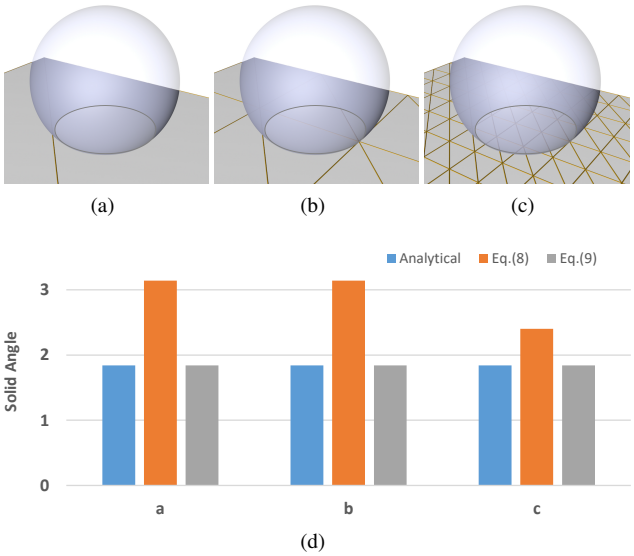


Figure 4: *Evaluation of our method in calculating the solid angle for different resolution of surface boundaries.*

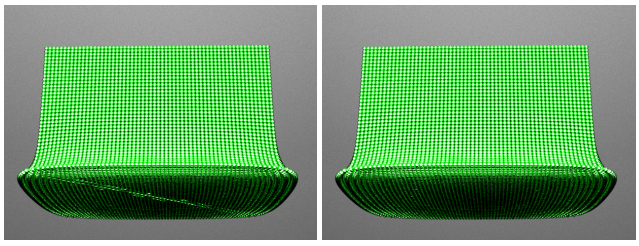


Figure 5: *(a)Without triangle clustering, discontinuous artifacts may arise between two adjacent triangles; (b)With triangle clustering, the discontinuity is removed.*

The final unresolved issue is how to select the sampling point $\mathbf{x}_s$ for $G(\cdot)$. The most obvious way is to select the closest point on triangle s. However, the above-mentioned discontinuity problem still exists. We therefore use the same strategy as how we calculate $\mathbf{d_n}$ and set $\mathbf{x}_s = \mathbf{x_n}$. For completeness, the final formula for computing the boundary integral of a triangle s is written as follows

$$f_{is}^{\mathcal{B}} = G(r)|_{r(\|\mathbf{x_n}-\mathbf{x}_i\|)}^{h} \frac{(\mathbf{n}_s \cdot \mathbf{d_n})A_s}{A_0}\Omega_0 \qquad (12)$$

One side effect with this strategy is that the boundary integral can be overestimated, as demonstrated in Figure 6(a). It explains the mismatch between the numerical results and exact values in computing the volume fraction, as shown in Figure 6(b). Nevertheless, Equation 12 still captures the increasing trend of the volume fraction. In particular, it has the advantage of being robust to handle complex boundaries, such as the boundary with a small dihedral angle.

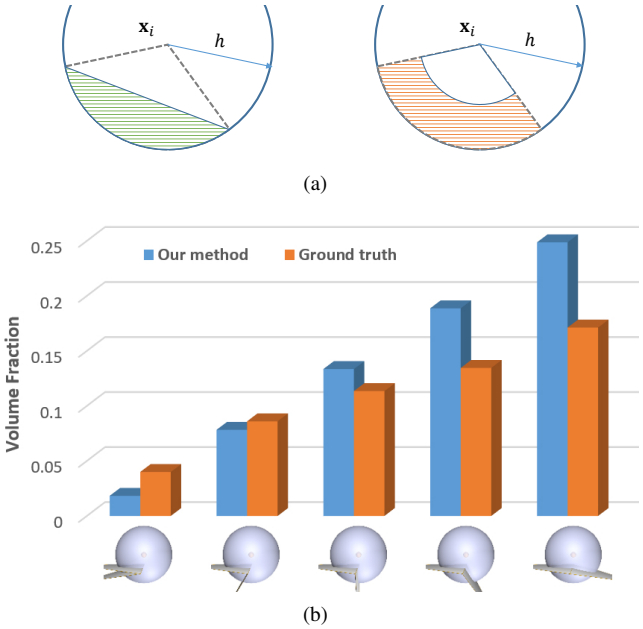


Figure 6: *(a) Compared to the exact value(left), our semi-analytical boundary integral results in a slightly overestimated value(right); (b) Evaluation of our method in calculating volume fractions for boundaries with different geometries.*

4.1. Domain Decomposition

In our previous discussion, we only discussed a special case when all three triangle vertices are inside a particle's support domain $\mathcal{S}$. For this special case, the intersection area A_s can be easily computed. To extend to other cases, we first do an intersection test between particle i's support domain and the plane containing triangle s. If the distance from particle i to the plane is larger than or equal to h, no intersection should happen between particle i's support domain and triangle s. Otherwise, the intersection problem can be simplified into a 2D intersection test between a circle and a triangle. Our principle to calculate the intersection area is to decompose the intersection region into a combination of triangles and circular sectors. Figure 7 demonstrates all 9 different cases that can result from the triangle / circle intersection. To distinguish each case, we check how many vertices are inside the support domain as well as how many edges intersect the support domain. By denoting the number of vertexes inside the support domain as n_v and the number of edges intersecting $\mathcal{S}$ as n_e, Algorithm 1 shows a pseudo algorithm to distinguish between different cases according to the values of n_v and n_e. For the special case of $n_v = 0$ and $n_e = 0$, we do an additional test to distinguish case ① from case ② by checking whether the center of the circle is located inside the triangle. For more details on how to do the segment/sphere intersection as well as how to calculate the areas for circular sectors and triangles, we refer to Schneider's book [SE02].

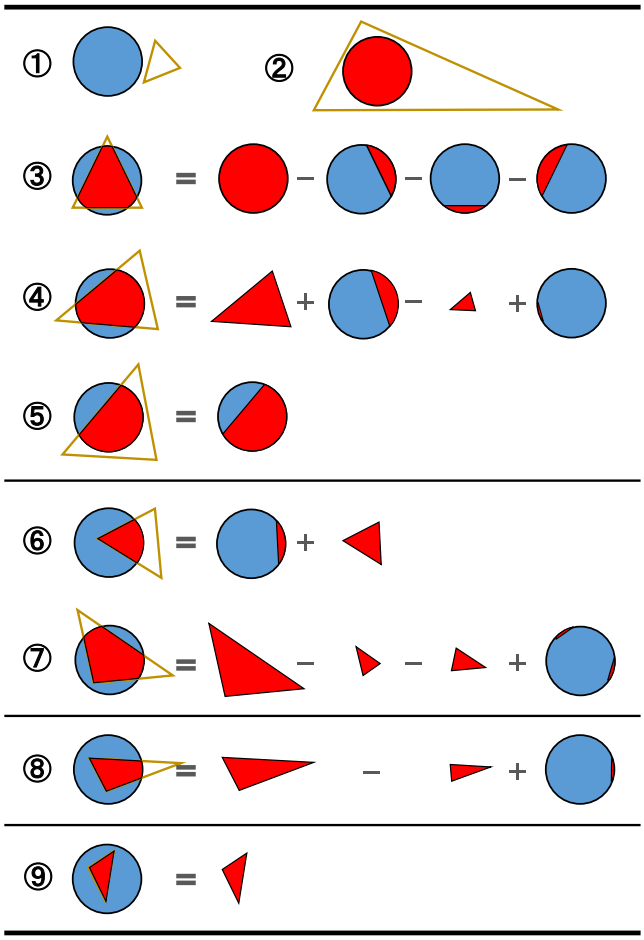


Figure 7: *An intuitive illustration for the domain decomposition algorithm.*

4.2. Nearest-Triangle Search

In querying neighboring triangles, a naive brute-force search for a scene that consists of N particles and M triangles has a computational complexity of $O(MN)$, which is too heavy for large-scale simulations. We therefore use a strategy similar to the nearest-particle search to accelerate the nearest-triangle search. For dynamic boundaries, the implementation on GPU can be divided into the following stages.

1. Allocate a uniform grid that is large enough to cover all boundary triangles as well as an array of the same size to store counters indicating how many triangles intersect with each grid cell.
2. For each triangle, find all grid cells that have intersection with the triangle and increase the corresponding intersection counter by one. To avoid write conflicts when two triangles have intersections with the same grid cell, use the atomic add function in CUDA to do the addition.
3. Take a parallel prefix sum on the counter array and return the total number of intersections as I.
4. Allocate an additional array of size I. For each triangle, do step 2 again and store the triangle index at a position demonstrated in Figure 8.
5. For each particle, find the grid cell it belongs to, iterate over all triangle indices stored at the nearest 27 (9 in 2d) grid cells and remove duplicate triangle indices.

For static boundaries, note that the first four steps are taken only once and can be precomputed at the beginning of simulation.

Algorithm 1 Domain Decomposition Algorithm

```
if n_v = 0 then
    switch n_e do
        case 0
            Calculate A_s according to ① or ②;
        case 1
            Calculate A_s according to ⑤
        case 2
            Calculate A_s according to ④
        case 3
            Calculate A_s according to ③
else if n_v = 1 then
    switch n_e do
        case 2
            Calculate A_s according to ⑥
        case 3
            Calculate A_s according to ⑦
else if n_v = 2 then
    Calculate A_s according to ⑧
else
    Calculate A_s according to ⑨
```

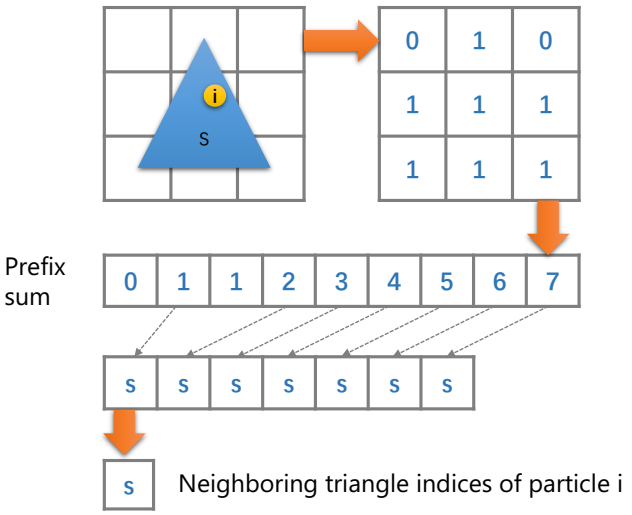
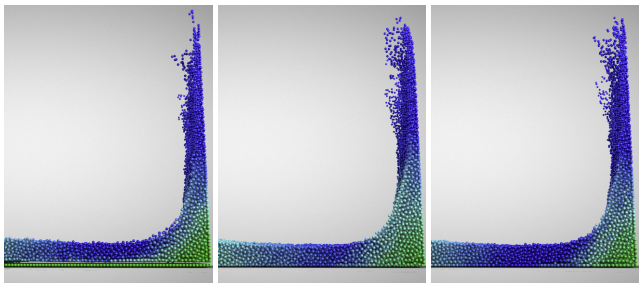


Figure 8: *An illustration of the nearest-triangle search.*

5. Incompressible Free Surface Flows

In this section, we evaluate the accuracy and efficiency of our semi-analytical solid boundary handling technique with two common in-



(a) signed distance field (b) ghost particle (c) our method

Figure 9: *A comparison of different boundary handling techniques. The total fluid particle is 89k for all three simulations while (a) uses a signed distance field as the boundary and take an average of* 19.7*ms for each timestep, (b) uses 122k ghost particles as the boundary and takes an average of* 26.7*ms per timestep, (c) use 12 triangles as the boundary and takes an average of* 23.8*ms per timestep.*

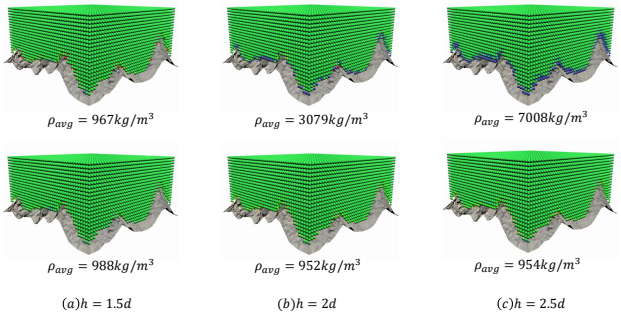


Figure 10: *A comparison between our method (bottom) and Fujisawa and Miura [FM15](top). By assuming the rest density to be* $1000kg/m^3$*, we calculate the average density for the one layer of particles that are closest to the boundary. Note Fujisawa and Miura's method suffers a large error when the kernel size is large while our method gets consistent results for all kernel sizes. The density field is color coded.*

compressible fluid solvers, include both a position-based solver and an projection-based solver. We only list key equations here and refer readers to their original work for detailed discussions.

5.1. Position-based Fluids

The core idea of the position-based incompressible fluid solver is to enforce the following constraint [MM13] for each particle i

$$C_i = \left(\frac{\rho_i}{\rho_0}\right) - 1 = 0 \tag{13}$$

where ρ_0 is the rest density and ρ_i is originally given by the standard SPH density estimator

$$\rho_i = \sum_j m_j \omega_{ij}, \tag{14}$$

which only considers fluid particles inside the computational domain. Due to an underestimation of density near the solid boundary, Figure 9(a) shows obvious artifacts at the bottom of a tank for the dambreak test.

To remove those artifacts, we reformulate ρ_i as

$$\rho_i = \rho_i^{\mathcal{F}} + \rho_i^{\mathcal{B}} \tag{15}$$

where $\rho_i^{\mathcal{F}}$ is calculated with Equation 14. By assuming the density inside the solid equals to ρ_0, the boundary integral $\rho_i^{\mathcal{B}}$ is written as

$$\rho_{is}^{\mathcal{B}} = \rho_0 \, W|^h_{r(\|\mathbf{x_n}-\mathbf{x}_i\|)} \frac{(\mathbf{n}_s \cdot \mathbf{d_n}) A_s}{A_0} \Omega_0, \tag{16}$$

where W is a function satisfying $dW/dr = \omega$. For the following derivation, we additionally assume $\rho_{is}^{\mathcal{B}}$ is constant. Therefore, the position update $\Delta\mathbf{x}_i$ for each particle i is just the same as its original form [MM13]

$$\Delta\mathbf{x}_i = \frac{1}{\rho_0} \sum_j \left(\lambda_i + \lambda_j\right) \nabla_i \omega_{ij}, \tag{17}$$

where the scaling factor λ_i is written as

$$\lambda_i = -\frac{C_i}{\sum_k \|\nabla C_i\|^2 + \varepsilon} \tag{18}$$

with ε representing a user specified relaxation parameter.

Evaluation. Figure 9(c) shows the effectiveness of the proposed semi-analytical boundary technique in removing the artifacts near the boundary. Compared to the one using a signed distance field, our method only introduces 26% additional computational cost while the simulation with ghost particles introduce 52%. In addition, Figure 10 shows a comparison between our method and the one proposed by Fujisawa and Miura [FM15] in calculating the density for particles near the boundary. Note the unsatisfactory results Fujisawa and Miura's method gets when a large kernel size is set for the density calculation. In contrast, our method is not sensitive to the kernel size.

5.2. Projection-based Fluids

In a projection-based method, the objective is to solve the following governing equations

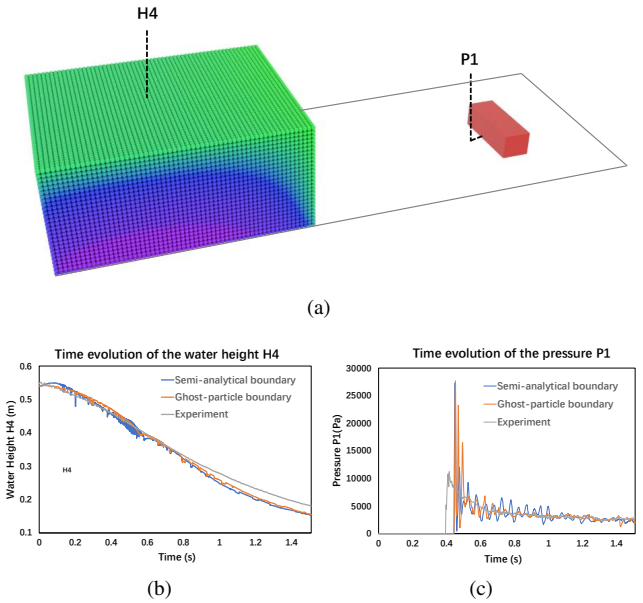
$$\begin{aligned} \nabla \cdot \left(\frac{\Delta t}{\rho} \nabla p\right) &= \nabla \cdot \mathbf{v}^*, \qquad && \textit{inside } \Omega, \\ p &= 0, && \textit{on } \partial\Omega, \end{aligned} \tag{19}$$

where p is the pressure and $\mathbf{v}^*$ is the intermediate velocity. Following the derivation in [HWW*20], the pressure Poisson equation can be discretized into

$$\mathcal{L}_i = \mathcal{D}_i. \tag{20}$$

The Laplacian of pressure $\mathcal{L}_i$ is written as

$$\mathcal{L}_i = \frac{\hat{A}_i}{\rho_0} p_i - \frac{1}{\rho_0} \sum_{j \in \mathcal{F}} \left(\frac{1}{\hat{\alpha}_i} + \frac{1}{\hat{\alpha}_j}\right) \frac{\omega_{ij}}{r_{ij}^2} p_j, \tag{21}$$



(a) (b) (c)

Figure 11: *3D dambreak with a squared obstacle. (a) Sketch of the 3D dambreak, please refer to [IV06] for the detailed configuration; (b) Time evolution of the water height H4; (c)Time evolution of the pressure P1.*

where $\hat{\alpha}_i$ represents the total weight whose value is computed as

$$\hat{\alpha}_i = \max\left(\alpha_0, \alpha_i^{\mathcal{F}\cup\mathcal{B}}\right), \quad \alpha_i^{\mathcal{F}\cup\mathcal{B}} = \sum_{j\in\mathcal{F}\cup\mathcal{B}} \omega_{ij}, \tag{22}$$

$\hat{A}_i$ represents the diagonal element of the coefficient matrix

$$\hat{A}_i = \max(A_0 - A_i^{\mathcal{B}}, A_i^{\mathcal{F}}), \tag{23}$$

with $A_i^{\mathcal{B}}$ and $A_i^{\mathcal{F}}$ denoting two terms as follows

$$A_i^{\mathcal{F}} = \sum_{j\in\mathcal{F}} \left(\frac{1}{\hat{\alpha}_i} + \frac{1}{\hat{\alpha}_j}\right)\frac{\omega_{ij}}{r_{ij}^2}, \quad A_i^{\mathcal{B}} = \sum_{j\in\mathcal{B}} \left(\frac{1}{\hat{\alpha}_i} + \frac{1}{\hat{\alpha}_j}\right)\frac{\omega_{ij}}{r_{ij}^2}. \tag{24}$$

Both α_0 and A_0 are precomputed values from an interior prototype particle with full fluid neighbors at the beginning of simulation. For more details on how we get Equation 22 and 23, please refer to [HWW*20] for the four different cases resulted from possible intersections between a particle's support domain and the fluid boundary.

The divergence of velocity is written as

$$\mathcal{D}_i = \frac{1}{\Delta t}\sum_{j\in\mathcal{F}} \left(\frac{1}{\hat{\alpha}_i} + \frac{1}{\hat{\alpha}_j}\right)\left(\frac{\mathbf{v}_j^* - \mathbf{v}_i^*}{2}\right)\cdot\mathbf{n}_{ij}\frac{\omega_{ij}}{r_{ij}} + \mathcal{D}_i^{\mathcal{B}} \quad . \tag{25}$$

where $\mathcal{D}_i^{\mathcal{B}}$ represents the source term introduced by ghost solid particles

$$\mathcal{D}_i^{\mathcal{B}} = \frac{1}{\Delta t}\sum_{j\in\mathcal{B}} \frac{2\Delta\mathbf{v}_{ij}^*}{\hat{\alpha}_i}\cdot\mathbf{n}_{ij}\frac{\omega_{ij}}{r_{ij}}, \tag{26}$$

with $\Delta\mathbf{v}_{ij}^*$ representing the relative velocity between particle i and the ghost particle, please refer to [HWW*20] for more details on how to apply the solid wall boundary condition.

By assuming $\hat{\alpha}_j = \hat{\alpha}_i$, $\mathbf{n}_{ij} = \mathbf{n}_s$ and $\Delta\mathbf{v}_{ij}^* = \Delta\mathbf{v}_s^*$ in the derivation, the integrands for $\mathcal{A}_i^{\mathcal{B}}$ and $\mathcal{D}_i^{\mathcal{B}}$ can be written as $\frac{2}{\hat{\alpha}_i}\frac{\omega}{r^2}$ and $\frac{2}{\hat{\alpha}_i\Delta t}(\Delta\mathbf{v}_s^*\cdot\mathbf{n}_s)\frac{\omega}{r}$, respectively. According to Equation 8, the values of $\alpha_i^{\mathcal{B}}$, $A_i^{\mathcal{B}}$ and $\mathcal{D}_i^{\mathcal{B}}$ can be reformulated as follows

$$\begin{aligned}
\alpha_i^{\mathcal{B}} &= \sum_s W\big|_{\|\mathbf{x}_s-\mathbf{x}_i\|}^h \Omega_s \\
A_i^{\mathcal{B}} &= \frac{2}{\hat{\alpha}_i}\sum_s W^{rr}\big|_{\|\mathbf{x}_s-\mathbf{x}_i\|}^h \Omega_s \\
\mathcal{D}_i^{\mathcal{B}} &= \frac{2}{\hat{\alpha}_i\Delta t}\sum_s \Delta\mathbf{v}_s^*\cdot\mathbf{n}_s W^r\big|_{\|\mathbf{x}_s-\mathbf{x}_i\|}^h \Omega_s
\end{aligned}, \tag{27}$$

in which W^r and W^{rr} are two functions satisfying the following identities

$$\frac{dW^r}{dr} = \frac{\omega}{r}, \quad \frac{dW^{rr}}{dr} = \frac{\omega}{r^2}. \tag{28}$$

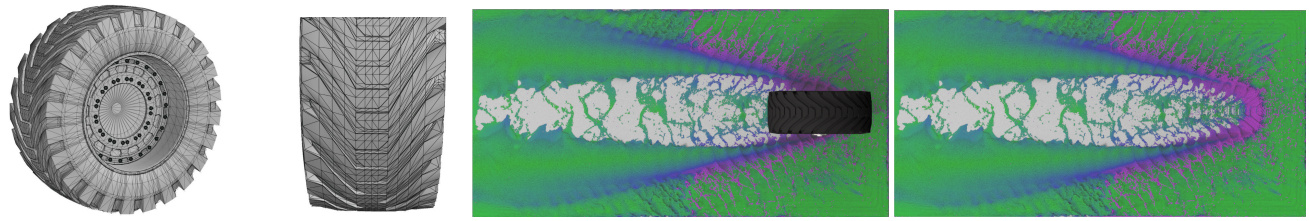
Evaluation. To validate our boundary handling technique in the projection-based method, we performed a 3D dam-break testcase as illustrated in Figure 11(a). The simulation results are compared to both the experimental results provided by Kleefsman et al. [KFV*05], as well as the projection method using fixed ghost particles. The water height measured at probe H4 as well as the pressure at P1 are plotted in Figure 11(b) and (c), respectively. It can be noted that the simulation result with our semi-analytical boundary conditions shows good agreement with the one using fixed ghost particles. Nevertheless, during the impact of the fluid against the obstacle, both methods predict the occurrence of the pressure peaks at around $t = 0.45s$, which is $0.05s$ later than the experimental data shows.

6. More Results

In this section, we demonstrate more examples simulated with both the position-based and projection-based methods. All implementations are based on the open source project PhysIKA (https://github.com/PhysikaTeam/PhysIKA) and parallelized with CUDA. Continuous collision detection (CCD) [WTTM15] between fluid particles and triangle mesh elements is used to prevent fluid particles from penetrating into the solid. Performance measurements are given for an Nvidia GeForce GTX 1060 and a 4-core 3.46 GHz Intel i7.

Industrial automotive application. Figure 12 shows a simplified simulation of the aquaplaning phenomenon between a tyre and a thin layer of water. The tyre, which composed of 29414 vertices and 57204 triangles as shown in Figure 12(a), is moved with a horizontal velocity of $4.7m/s$ as well as a rotational velocity of $9.4m/s$. The water, which has a thickness of 25mm, is composed of 1.58 million particles. A one-way solid-fluid coupling is taken to simulate the complex interactions between the water and the tyre. Figure 12(b) and (c) show the simulation results at $t = 0.8s$ with and without the tyre. Note that the fluid is highly fragmented and takes the shape of the grooves for particles near the contact region.

Dam-break with obstacles. Figure 1 shows the impact of ocean



(a) Tyre mesh(57204 triangles) (b) With the tyre rendered (c) Without the tyre rendered

Figure 12: *A one-way coupling example between a tyre and a thin layer of water. Velocities are color coded and the projection-based method takes* 9.75*s per timestep on average.*

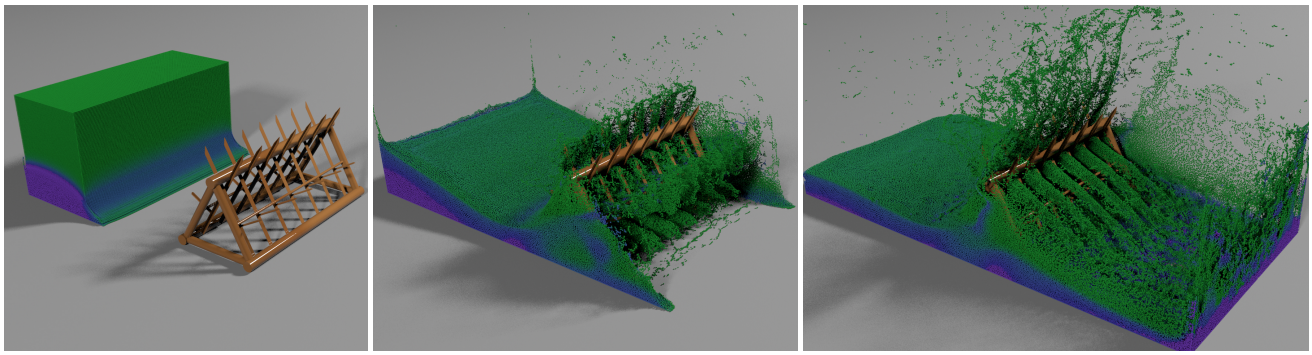


Figure 13: *3D Dambreak with a complex obstacle. The pressure field is color coded and the projection-based solver takes* 8.99*s per timestep on average.*

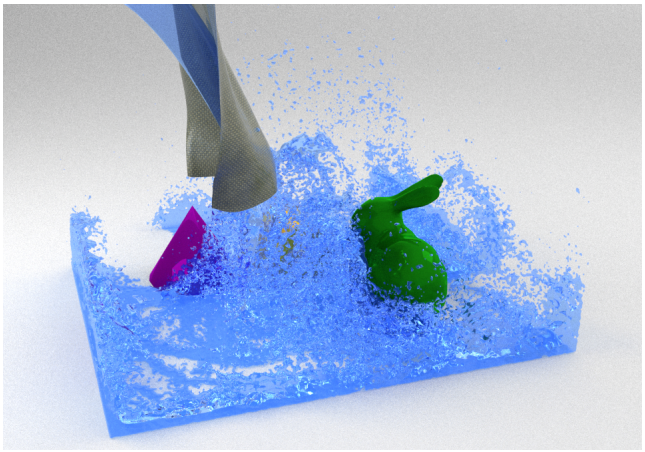


Figure 14: *A two-way coupling example consisting of* 2.09 *millions of particles and* 91*k triangles. The position-based solver takes* 2.73*s per timestep on average.*

waves on the drilling platform. The water is composed of 1.86 million particles and the drilling platform is composed of 95k triangles. Figure 13 show another impact of water on a complex barricade, where the water is composed of 1.86 million particles and the barricade is composed of 3.6k triangles. Both the drilling platform and the barricade contain irregular triangles. Note that the correct behaviors are still captured between fluid particles and solid triangles.

Two-way coupling. Finally, we demonstrate the flexibility of our method in handling two-way solid-fluid couplings. The position-based solver is applied to the fluid simulation. Since no particle resampling is required for the solid boundary, a cloth modeled with the projective peridynamics [HWW17] can be easily integrated to realize a two-way coupling between the cloth and the water, as shown in Figure 14. In this example, a maximum of 2.09 million particles and 91k triangles are used to simulate the complex interaction between fluid and static/dynamic solid boundaries. For the interaction between particles and the cloth, if a particle is approaching from the opposite of the cloth, the triangle normal should be flipped to avoid negative boundary integrals.

7. Conclusions

We presented a semi-analytical approach to handle complex solid boundaries of arbitrary shape for particle simulations. By converting the volume integral into a semi-analytical surface integral, our method allows a computer aided design (CAD) mesh file representing the boundary to be integrated for particle simulations. Experiments show that our method is able to achieve comparable results with those simulated using ghost particles. In addition, our method shows better performance, especially for large-scale boundaries, and is flexible enough to handle complex solid boundaries.

Our method also has some limitations. First, with the one-point

Gauss quadrature, our semi-analytical boundary integration results in slightly overestimated values. Higher-order Gaussian quadrature should be applied to improve the accuracy. Second, in simulating interactions between the fluid and planar structures (e.g., a cloth), we need to flip the normal of the boundary if a particle is approaching the solid from the other side of the surface. Finally, our method cannot handle interactions between fluids and linear structures (e.g., hairs) yet. A possible remedy is to treat linear structures as connected cylinders and calculate the boundary integral over the cylinders. We will consider extending our method to handle linear structures in our future work.

8. Acknowledgement

We would like to thank anonymous reviewers for their valuable comments. The project was supported by the National Key R&D Program of China (No.2017YFB1002700, No.2017YFB0203000), the National Natural Science Foundation of China (No.6187070657, No.61632003), Youth Innovation Promotion Association, CAS (No.2019109).

References

[ABC*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (2003), 3–15. 2

[ACAT13] AKINCI N., CORNELIS J., AKINCI G., TESCHNER M.: Coupling elastic solids with smoothed particle hydrodynamics fluids. *Computer Animation and Virtual Worlds 24*, 3-4 (2013), 195–203. 2

[AIA*12] AKINCI N., IHMSEN M., AKINCI G., SOLENTHALER B., TESCHNER M.: Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph. 31*, 4 (2012), 62. 1, 2

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

[BGT17] BAND S., GISSLER C., TESCHNER M.: Moving least squares boundaries for sph fluids. In *VRIPHYS* (2017), pp. 21–28. 2

[BK15] BENDER J., KOSCHIER D.: Divergence-free smoothed particle hydrodynamics. In *Proceedings of SCA* (2015), ACM, pp. 147–155. 1

[BKWK19] BENDER J., KUGELSTADT T., WEILER M., KOSCHIER D.: Volume maps: An implicit boundary representation for sph. In *Motion, Interaction and Games*. 2019, pp. 1–10. 2

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *Proceedings of SCA* (2007), pp. 209–217. 2

[BTT09] BECKER M., TESSENDORF H., TESCHNER M.: Direct forcing for lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics 15*, 3 (2009), 493–503. 2

[CdOL19] CHIRON L., DE LEFFE M., OGER G., LE TOUZÉ D.: Fast and accurate sph modelling of 3d complex wall boundaries in viscous and non viscous flows. *Computer Physics Communications 234* (2019), 93 – 111. 3

[CL03] COLAGROSSI A., LANDRINI M.: Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics 191*, 2 (2003), 448–475. 2

[FLR*13] FERRAND M., LAURENCE D., ROGERS B. D., VIOLEAU D., KASSIOTIS C.: Unified semi-analytical wall boundary conditions for inviscid, laminar or turbulent flows in the meshless sph method. *International Journal for Numerical Methods in Fluids 71*, 4 (2013), 446–472. 1, 2, 3, 4

[FM15] FUJISAWA M., MIURA K. T.: An efficient boundary handling with a modified density calculation for sph. *Computer Graphics Forum 34*, 7 (2015), 155–162. doi:10.1111/cgf.12754. 2, 3, 7

[GPB*19] GISSLER C., PEER A., BAND S., BENDER J., TESCHNER M.: Interlinked sph pressure solvers for strong fluid-rigid coupling. *ACM Trans. Graph. 38*, 1 (Jan. 2019). 2

[GW69] GOLUB G. H., WELSCH J. H.: Calculation of gauss quadrature rules. *Mathematics of Computation 23*, 106 (1969), 221–221. 4

[HKK07] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics on gpus. *Proc.computer Graphics Int.rio De Janeiro Brazil May Jun 4*, 4 (2007), 671–691. 2

[HLL*12] HE X., LIU N., LI S., WANG H., WANG G.: Local poisson SPH for viscous incompressible fluids. *Computer Graphics Forum 31*, 6 (2012), 1948–1958. 3

[HLW*12] HE X., LIU N., WANG G., ZHANG F., LI S., SHAO S., WANG H.: Staggered meshless solid-fluid coupling. *ACM Trans. Graph. 31*, 6 (2012), 149. 2

[HWW17] HE X., WANG H., WU E.: Projective peridynamics for modeling versatile elastoplastic materials. *IEEE Transactions on Visualization and Computer Graphics* (2017), 2589 – 2599. 9

[HWW*20] HE X., WANG H., WANG G., WANG H., WU E.: A variational staggered particle framework for incompressible free-surface flows, 2020. arXiv:2001.09421. 1, 2, 7, 8

[IAGT10] IHMSEN M., AKINCI N., GISSLER M., TESCHNER M.: Boundary handling and adaptive time-stepping for pcisph. In *Workshop on virtual reality interaction and physical simulation VRIPHYS* (2010), vol. 6. 2

[ICS*14] IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 426–435. 1

[IV06] ISSA R., VIOLEAU D.: 3d dambreaking. *https://spheric-sph.org/tests/test-2* (2006). 8

[KB17] KOSCHIER D., BENDER J.: Density maps for improved sph boundary handling. In *Proceedings of SCA* (2017), pp. 1–10. 1, 2

[KBLP04] KULASEGARAM S., BONET J., LEWIS R., PROFIT M.: A variational formulation based contact algorithm for rigid boundaries in two-dimensional sph applications. *Computational Mechanics 33*, 4 (2004), 316–325. 2, 3

[KBST19] KOSCHIER D., BENDER J., SOLENTHALER B., TESCHNER M.: Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids. 1

[KFV*05] KLEEFSMAN K. M. T., FEKKEN G., VELDMAN A. E. P., IWANOWSKI B., BUCHNER B.: A volume-of-fluid based simulation method for wave impact problems. *Journal of Computational Physics 206*, 1 (2005), 363–393. 8

[LL03] LIU G.-R., LIU M. B.: *Smoothed particle hydrodynamics: a meshfree particle method*. World scientific, 2003. 3

[MAC*11] MARRONE S., ANTUONO M., COLAGROSSI A., COLICCHIO G., LE TOUZÉ D., GRAZIANI G.: δ-sph model for simulating violent impact flows. *Computer Methods in Applied Mechanics and Engineering 200*, 13-16 (2011), 1526–1542. 2

[MFK*15] MAYRHOFER A., FERRAND M., KASSIOTIS C., VIOLEAU D., MOREL F.-X.: Unified semi-analytical wall boundary conditions in sph: analytical extension to 3-d. *Numerical Algorithms 68*, 1 (2015), 15–34. 2, 3

[MK09] MONAGHAN J. J., KAJTAR J. B.: Sph particle boundary forces for arbitrary boundaries. *Computer physics communications 180*, 10 (2009), 1811–1820. 2

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (2013), 104. 1, 2, 3, 7

[MMG*11] MONACO A. D., MANENTI S., GALLATI M., SIBILLA S., AGATE G., GUANDALINI R.: Sph modeling of solid boundaries through a semi-analytic approach. *Engineering Applications of Computational Fluid Mechanics 5*, 1 (2011), 1–15. 3

[Mon94] MONAGHAN J. J.: Simulating free surface flows with sph. *Journal of computational physics 110*, 2 (1994), 399–406. 2

[OS83] OOSTEROM A. V., STRACKEE J.: The solid angle of a plane triangle. *IEEE Transactions on Biomedical Engineering 30*, 2 (1983), 125–126. 4

[PICT15] PEER A., IHMSEN M., CORNELIS J., TESCHNER M.: An implicit viscosity formulation for sph fluids. *ACM Trans. Graph. 34*, 4 (July 2015), 114:1–114:10. 1

[PL93] PETSCHEK A. G., LIBERSKY L. D.: Cylindrical smoothed particle hydrodynamics. *Journal of Computational Physics 109*, 1 (1993), 76–83. 2

[RL96] RANDLES P., LIBERSKY L. D.: Smoothed particle hydrodynamics: some recent improvements and applications. *Computer methods in applied mechanics and engineering 139*, 1-4 (1996), 375–408. 2, 3

[SE02] SCHNEIDER P., EBERLY D. H.: *Geometric tools for computer graphics*. Elsevier, 2002. 5

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Trans. Graph. (SIGGRAPH) 28*, 3 (July 2009), 40:1–40:6. 1

[SSP07] SOLENTHALER B., SCHLÄFLI J., PAJAROLA R.: A unified particle model for fluid–solid interactions. *Computer Animation and Virtual Worlds 18*, 1 (2007), 69–82. 2

[TCJ20] TANG Y., CHEN S., JIANG Q.: A conservative sph scheme using exact projection with semi-analytical boundary method for free-surface flows. *Applied Mathematical Modelling 82* (2020), 607–635. 3

[TDF*15] TAKAHASHI T., DOBASHI Y., FUJISHIRO I., NISHITA T., LIN M. C.: Implicit formulation for sph-based viscous fluids. *Wiley Online Library 34*, 2 (2015), 493–502. 1

[WKBB18] WEILER M., KOSCHIER D., BRAND M., BENDER J.: A physically consistent implicit viscosity solver for sph fluids. *Computer Graphics Forum 37*, 2 (2018), 145–155. 1

[WTTM15] WANG Z., TANG M., TONG R., MANOCHA D.: Tightccd: Efficient and robust continuous collision detection using tight error bounds. *Computer Graphics Forum* (2015). 8

[YCR*15] YANG T., CHANG J., REN B., LIN M. C., ZHANG J. J., HU S.-M.: Fast multiple-fluid simulation using helmholtz free energy. *ACM Trans. Graph. 34*, 6 (2015), 1–11. 1

[YML*17] YANG T., MARTIN R. R., LIN M. C., CHANG J., HU S.: Pairwise force sph model for real-time multi-interaction applications. *IEEE Transactions on Visualization and Computer Graphics 23*, 10 (2017), 2235–2247. 1