Pacific Graphics 2015
N. J. Mitra, J. Stam, and K. Xu
(Guest Editors)

*Volume 34 (2015), Number 7*

# An Efficient Boundary Handling with a Modified Density Calculation for SPH

Makoto Fujisawa[1] and Kenjiro T. Miura[2]

[1]University of Tsukuba, Japan
[2]Shizuoka University, Japan

**Abstract**

*We propose a new boundary handling method for smoothed particle hydrodynamics (SPH). Previous approaches required the use of boundary particles to prevent particles from sticking to the boundary. We address this issue by correcting the fundamental equations of SPH with the integration of a kernel function. Our approach is able to directly handle triangle mesh boundaries without the need for boundary particles. We also show how our approach can be integrated into a position-based fluid framework.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

## 1. Introduction

Particle-based fluid simulation is widely used to model complex fluid phenomena for computer graphics animation because it offers attractive features such as mass conservation, ease of extending the simulation space, and simplicity. One popular particle-based fluid simulation technique is smoothed particle hydrodynamics (SPH). SPH simulation has two main problems. The first is how to enforce incompressibility. SPH was originally designed to model compressible flows, and thus allows the density to change easily. Many methods, such as WCSPH [BT07], PCISPH [SP09], position-based fluids [MM13], and IISPH [ICS*14] have been developed in order to solve this problem.

The second problem is how to handle solid boundaries. Almost all fluid scenes in computer graphics include both solids and fluids. Although we can simulate the interaction between the fluid particles and boundary by assigning some force or pressure to the particles near the boundary, the problem of particle stacking remains. The techniques for incompressibility mentioned above tend to stabilize the density of the particles, in other word, the distribution of particles within the effective sphere. This can cause particles to cluster along the boundary shape due to the absence of particles inside the boundary. Many methods use boundary particles, which are placed inside the boundary. Boundary particles are treated somewhat like fluid particles except that they have a static position. The particle representation restricts the details of the boundary since the shape of the boundary needs to be approximated by finite-sized particles even though the solid boundary is represented by a flat plane, a collection of triangles, or other such shapes. This also dramatically increases the number of particles in the simulation space. As a result, the computational cost for a neighborhood particle search is high.

In this paper, we present the theoretical problem of SPH for the case where the scene has a solid boundary and show how to solve this problem. We propose a fast calculation method of integration of kernel functions, which are used to correct the problems for a triangle mesh. We treat the mesh as a collection of flat planes. Our method is based on the basic formulation of SPH, meaning that it can be applied to various different SPH algorithms, even algorithms that do not use pressure or force. As an example, we show that our method can be applied to PBF [MM13].

## 2. Related Work

SPH was independently introduced by Gingold and Monaghan [GM77] and Lucy [Luc77], and has been extensively used for physics research. Müller et al. [MCG03] introduced SPH for interactive viscosity fluid animation. Explicit calculation of pressure using an equation of state allows for very fast computation, but also causes the incompressibility problem. To solve this problem, various algorithms have been developed [CBP05, BT07, SP09]. These algorithms require very small time-step sizes for stable simulation. PBF [MM13] relaxes the time-step size with some iterations. PBF achieves a constant density fluid by solving a set of positional constraints using an iterative solver integrated into a position-based dynamics [MHHR07] framework. IISPH [ICS*14] also relaxes the time-step size by solving a linear system. We refer to [IOS*14] for more complete survey on SPH for computer graphics.

A distance-based penalty method is commonly used to solve the boundary handling problem for particle methods [MKA*04, Mon05, DTM*12]. An impulse-based force is also used to achieve two-way coupling between SPH fluid and rigid bodies [CBP05, OKR09, HEW15]. These methods only focus on the forces acting on the particle, so that the particles near the boundary tend to stack to enforce uniform density. Harada et al. [HKK07] solved this problem by using a wall weight function for planar boundaries. Boundary particles are commonly used to produce a more smooth density distribution on the boundary [BYM05, BIT09, BTT09]. The boundaries are sampled from the frozen fluid particles, with the boundary particles treated the same as the fluid particles. Ihmsen et al. [IABT11] corrected the distribution of particles on the boundary by using the pressure force. Schechter and Bridson [SB12] used ghost particles instead of frozen particles. Ghost particles are placed during the simulation depending on the position of the fluid particles on the boundary. These methods required multiple layers of boundary particles for smooth density or pressure distribution. Akinci et al. [AIA*12] only used one layer of particles by taking into account a virtual volume calculated from the neighboring boundary particles. They extended their work to realize two-way coupling with thin, deformable objects [ACAT13].

In order to correct density fluctuation on interface, many kernel correction techniques are proposed. Li and Liu [LL96] and Liu et al. [LLB97] proposed to use a moving least-square kernel to reproduce any order polynomial with the irregular particle distribution. Colagrossi and Landrini [CL03] also used the MLS kernel to re-initialize the distribution of the density. Panizzo [Pan04] introduced a Shepard filter to correct the kernel. These correction techniques are only considering a distribution of neighboring particles. Kulasegaram et al. [KBLP04] proposed an improvement to the equation for the density in SPH calculations, and was able to prevent particle stacking without using boundary particles. Their method introduced a correction term to avoid density underestimation at the boundary by considering the area occupied by the solid boundary in the effective radius. The correction term was calculated approximately by spline curve fitting and only planar boundaries were assumed. Feldman and Bonet [FB07] calculated the correction term by integrating the density function in the sphere with a boundary represented by several line segments in two-dimensional space. Ferrand et al. [FLR*13] proposed a calculation method of the gradient of the correction term using the Gauss theorem and Mayrhofer et al. [MFK*15] extended their work for a boundary represented by triangles in three-dimensional space. They only considered the gradient of the correction term while SPH approximation of the equation of continuity and the momentum equation only includes the gradient term. However the position-based approach uses the density itself to evaluate an incompressibility. He et al. [HWZ*14] proposed a similar approach for avoiding the ghost particles to calculate a surface tension force acting on a free surface. They used two kernels to avoid density underestimation while the method depends on user-specified parameter and they did not address solid boundary conditions.

In this paper, we present a theoretical solution for the correction term itself for the case of a planar boundary. We also propose an approximate calculation method for triangle mesh boundaries. Since this method improves the fundamental equation of SPH, it can be adapted for use with a variety of algorithms. We integrate our method with PBF that does not use force or pressure to correct density fluctuations.

## 3. Method

### 3.1. SPH algorithm

In SPH method, a general quantity $A$ at $x$ is approximated by a weighted sum of neighbor particles.

$$A(x) = \sum_{j \in N} m_j \frac{A_j}{\rho_j} W(|x_j - x|, h), \tag{1}$$

where $N$ is the number of neighbor particles, $m_j$ is the mass of a particle $j$, $\rho_j$ is the density and the $W(r,h)$ is a smoothing kernel with effective radius $h$. For example, the density of the fluid can be calculated by

$$\rho(x) = \sum_{j \in N} m_j W(|x - x_j|, h). \tag{2}$$

The derivative of the quantity $\nabla A$ is easily evaluated from the derivative of the kernel function. Equation (1) is also used to approximate other quantities in the governing equations.

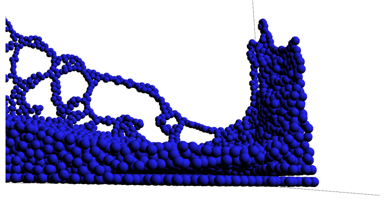
### 3.2. Modified Density Calculation

The density calculation using equation (2) can also be derived from a discrete density approximation. If we define the density as a sum of masses around the position $x$, the discrete density can be defined as follows.

$$\hat{\rho}(x) = \sum_{j \in N} m_j \delta(|x - x_j|), \tag{3}$$

where $\delta$ is the Dirac delta function. This definition creates a discontinuity in the density field and is not suitable for numerical calculations. In order to get a continuous field, a Gaussian-like smoothing function $W(r,h)$ is used.

$$\rho(x) = \frac{\int \hat{\rho}(x') W(|x - x'|, h) dx'}{\int W(|x - x'|, h) dx'} \tag{4}$$


(a) Result without $\gamma$

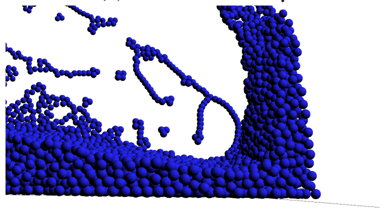

(b) Result with $\gamma$

**Figure 1:** *Examples of particle stacking.*

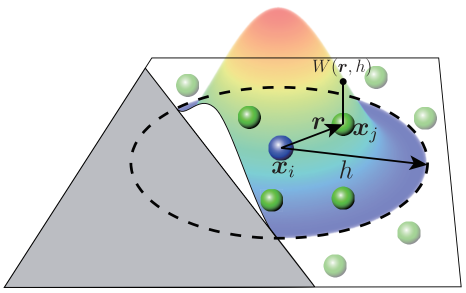


**Figure 2:** *Density calculation in two-dimensional space using the kernel function $W$.*

By substituting equation (3) into (4), we can obtain a continuous density for particle $i$ [KBLP04].

$$\rho_i = \frac{1}{\gamma_i} \sum_{j \in N} m_j W(|x_i - x_j|, h) \tag{5}$$

where

$$\gamma_i = \int_V W(|x_i - x|, h) dx \tag{6}$$

The kernel function is designed to be $\gamma_i = 1$ when a spherical region with effective radius $h$ and center position $x_i$ is filled with neighboring particles. This assumption is not true if the particle $i$ is on the boundary as shown in Figure 2. The gray area represents a rigid boundary. In this case, integration of the kernel function would give $\gamma_i < 1$. As a result, equation (2) under the assumption of $\gamma_i = 1$ underestimates the density. This is why particle stacking occurs. Figure 1(a) shows the effect that particle stacking has on simulations. The fluid particles form a layer that causes a void region because of the high density of the particles. In order to relax this, most methods sample the boundary particles in the boundary region to fill the neighbor particles in the spherical region.

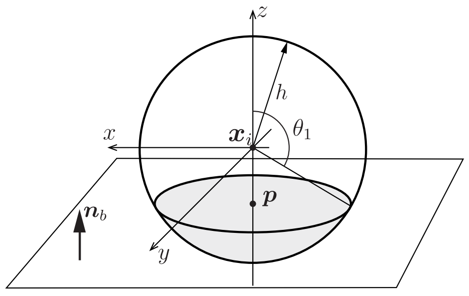


**Figure 3:** *Coordinates for $\gamma$ calculation.*

In this paper, we use equation (5) instead of equation (2). We are able to prevent particle stacking by using $\gamma$ as shown in Figure 1(b) without placing boundary particles. In [KBLP04], $\gamma$ is approximated by a spline function that varies depending on the distance to the planar boundary. We show that $\gamma$ can be analytically calculated if we can assume that the boundary is a plane.

### 3.3. $\gamma$ for Planar Boundary

We assume the coordinate system as shown in Figure 3. The origin is at the center position $x_i$ of the particle $i$ and the $z$ points in the direction of $n_b$ which is the normal of the closest boundary plane. Equation (6) can be rewritten in polar coordinates $(r, \theta, \phi)$ centered at $x_i$ as follows.

$$\gamma_i = \int_0^{2\pi} \int_0^{\pi} \int_0^{g(\theta)} W(r, h) r^2 \sin\theta dr d\theta d\phi \tag{7}$$

where $g(\theta) \in [0, h]$ is a distance function from the center $x_i$ to the boundary. $g(\theta)$ is a constant with respect to $\phi$ under the assumption that the $z$ axis is equal to the normal $n_b$. Thus, if the boundary is represented by a plane $z = c$, then $g(\theta)$ can be calculated by

$$g(\theta) = \begin{cases} \dfrac{c}{\cos\theta} & \theta > \theta_1 \\ h & \theta \leqq \theta_1 \end{cases} \tag{8}$$

where $\theta_1 = \pi - \cos^{-1}(c/h)$ is the angle between the $z$ axis and a vector from the center to the boundary circle (see Figure 3). We define $c$ as the signed distance from the center to the boundary in these coordinates ($c < 0$ in case of Figure 3).

By assuming a linear polynomial for the boundary, the distance function $g$ can be defined in a simpler form. There are many definitions of the kernel function $W(r, h)$, but we can calculate $\gamma$ by substituting any kernel function into equation (7). In this paper, we use a Poly6 kernel [MCG03] for the density calculation (see Appendix A) and Mathematica (Wolfram Research) was used to solve equation (7).

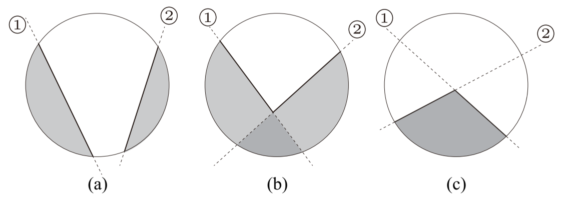


**Figure 4:** *Combination of planar boundaries for $\gamma$ calculation.*

### 3.4. $\gamma$ for Triangle Mesh

Triangle meshes are composed of several flat planes. We extend the $\gamma$ calculation method for planar boundaries to mesh boundaries. If there are $n$ triangles in the effective sphere whose radius equals $h$, we can calculate the $\gamma_i$ $(i = 0, \ldots, n)$ for each triangle as a plane. For the case where $W(r,h)$ is constant, the $\gamma$ value for the mesh boundary is easily calculated by multiplying all $\gamma_i$ because $\gamma$ equals 1 when there are no boundaries in the computational region. However, we have to consider another strategy since $W(r,h)$ is commonly a Gaussian-like function. The simple solution is to use a numerical integration method like Monte Carlo. However, this would take a huge amount of computational time. We thus only use the numerical integration method for vertices and edges. These $\gamma$ values are precomputed and stored in each element.

Figure 4 shows three patterns for the case where two polygons exist within the effective sphere. In the case shown in Figure 4(a), $\gamma$ can be calculated by simple addition and subtraction since the two polygons (planes) do not intersect within the sphere.

$$\gamma = 1 - \sum_i (1 - \gamma_i) \tag{9}$$

where $1 - \gamma_i$ is the integrated value of $W(r,h)$ in the gray region in Figure 4(a).

In the case of Figure 4(b), the two planes intersect and form a convex shape. We have to subtract the overlapping region from the second term of equation (9).

$$\gamma = 1 - \left( \sum_i (1 - \gamma_i) - \sum_i \sum_{j>i} \gamma'_{ij} \right) \tag{10}$$

where $\gamma'_{ij}$ is the integral of $W(r,h)$ in the overlapping region represented by the dark gray in Figure 4(b). When the center of the particle is located at the vertex (intersection point) shared by the two planes in a two dimensional space, $\gamma'_{ij}$ can be approximated by a function of $\gamma'_v$ and $\tilde{W}(r,h) \in [0,1]$, which is a normalized $W(r,h)$. This follows because $W(r,h)$ is a point-spread function. $\gamma'_{ij}$ will change depending on $\tilde{W}(d,h)$ in $0 < d < h$ while $d$ is the signed distance between the vertex and the center of the particle ($d < 0$ means the particle is inside the boundary).

$$\gamma'_{ij} = \begin{cases} \gamma'_v & d \leq 0 \\ f_d(\tilde{W})\gamma'_v & 0 < d < h \\ 0 & d \geq h \end{cases} \tag{11}$$

where $f_d(\tilde{W}) \in [0,1]$ is a function that approximates the change in $\gamma'_{ij}$. In this paper, we use $f_d(\tilde{W}) = \tilde{W}^3$, which is an empirically derived function. $\gamma'_v$ is precomputed and stored in each vertex.

In the case of three dimensions, we precompute $\gamma'_{ij}$ not only for the vertices but also for the edges. $\gamma'_v$ in equation (11) is replaced by the gamma of the edge $\gamma'_e$. If one of the vertices of the edge is also included in the sphere, we subtract $f_d(\tilde{W})\gamma'_v$ from $\gamma'_e$. Finally, $f_d(\tilde{W})\gamma'_v$ for all vertices in the sphere are subtracted from $\gamma$. When the edge or the vertex is convex such as in Figure 4(c), we can calculate using $\gamma = 1 - \gamma'_{ij}$.

In order to treat more complex shapes, we have to consider cases in which many more planes are located in the sphere and form a bumpy surface. Our calculation procedure considering these situations is outlined in Algorithm 1. The edge and vertex in the sphere are called the shared edge and the shared vertex respectively in Algorithm 1. Each edge is processed sequentially and marked as processed to avoid double addition or subtraction. If the calculated $\gamma$ is less than $\gamma_{min}$, we use $\gamma = \gamma_{min}$ to avoid zero division.

### 3.5. Integration into PBF

PBF [MM13] is a position-based method for particle-based fluid simulation. PBF can enforce the incompressibility of the fluid by imposing a density constraint $c_i(x_1, \ldots x_N) = \rho_i/\rho_0 - 1 = 0$ on all particles, where $\rho_0$ is the rest density. After particles have been moved to the predicted positions by forces such as gravity, the position of particle $x_i$ is corrected to satisfy the constraint (i.e. $c(x + \Delta x) = 0$) by the following equation.

$$\Delta x_i = \frac{1}{\rho_0} \sum_{j \in N} (\lambda_i + \lambda_j) \nabla W(x_i - x_j, h). \tag{12}$$

where $\Delta x$ is the particle displacement and $\lambda$ is a scaling factor expressed by

$$\lambda = -\frac{c_i(x_1, \ldots x_N)}{\sum_k |\nabla_{x_k} c_i|^2 + \varepsilon}. \tag{13}$$

The numerator $c_i$ can be computed from equation (5). The denominator includes $\nabla \rho_i$ term because $\nabla_{x_k} c_i$ is defined as $\frac{1}{\rho_0} \nabla_{x_k} \rho_i$. We assume that $\gamma_i$ is a function of the distance $e = d/h$, which is the distance normalized by the effective radius $h$. From this assumption, $\nabla \rho_i$ is given by equation (5).

$$\nabla \rho_i = \frac{1}{\gamma_i} \sum_{j \in N} m_j \nabla W(x_i - x_j, h) - \frac{\rho_i}{\gamma_i h} \frac{\partial \gamma_i}{\partial e} n_b. \tag{14}$$

**Algorithm 1** $\gamma$ calculation.

```
$\gamma' = 0$
for each plane $i$ without shared edge do
  $\gamma' += 1 - \gamma_i$
end for
for each shared edge $e$ formed by planes $i$ and $j$ do
  if edge is concave then
    $\gamma' -= \gamma'_{ij}$
    for $p \in i, j$ do
      if plane $p$ includes another shared edge then
        if another shared edge is convex
            or unprocessed concave edge then
          $\gamma' += 1 - \gamma_p$
        end if
      else
        $\gamma' += 1 - \gamma_p$
      end if
    end for
    if edge includes shared vertex $v$ then
      $\gamma' -= f_d(\tilde{W})\gamma'_v$
    end if
  else
    $\gamma' += \gamma'_{ij}$
    for $p \in i, j$ do
      if plane $p$ includes another shared edge then
        if another shared edge is concave
            or unprocessed convex edge then
          $\gamma' -= 1 - \gamma_p$
        end if
      end if
    end for
  end if
  mark $e$ as processed
end for
for each shared vertex $v$ do
  $\gamma' += f_d(\tilde{W})\gamma'_v$
end for
return $1 - \gamma'$
```

where $n_b$ is the normal to the boundary. We have to consider the two cases $k = i$ and $k \neq i$, which then gives the final equation:

$$\nabla_{x_k} c_i = \frac{1}{\rho_0 \gamma_i} \begin{cases} \sum_{j \in N} m_j \nabla W(x_i - x_j, h) - \frac{\rho_i}{h} N_b & k = i, \\ -m_i \nabla W(x_i - x_k, h) & k \neq i, \end{cases} \tag{15}$$

where $N_b = \frac{\partial \gamma_i}{\partial e} n_b$ and $\partial\gamma/\partial e$ are the derivatives of $\gamma$ along the normal direction. We use the central difference to calculate $\partial\gamma/\partial e$.

## 4. Results

This section describes the results of applying the proposed method to several test cases. All results were generated on a computer equipped with a 3.7 GHz Intel Core i7 CPU and an NVIDIA GeForce GTX TITAN GPU. The algorithm was predominantly implemented on a GPU by using NVIDIA CUDA. We also used a SOR solver [MMCK14] to process the PBF in parallel.

Figure 1 shows a breaking dam scene with approximately 15,000 particles and 6 planar boundaries. Our method could prevent particle stacking. Figure 5 shows a breaking dam scene with bunny meshes. The number of particles is approximately 105,000 and the bunnies are composed of around 15,000 triangles. Our approach can handle not only the planner boundary surrounding the scene but also the mesh boundary. Figure 6 shows a water drop scene with a bowl shaped boundary. The number of particles is approximately 70,000 and the bowl is composed of around 4,000 triangles. Figure 7 shows a cross-section of the Figure 6. Only the fluid particles are shown. We do not observe particle stacking in the case of the mesh boundary. Figure 8 shows the results for a water flow scene on a bumpy downslope. If we want to accurately represent this kind of boundary, a large number of very small sized boundary particles is required. The maximum number of particles is around 35,000 and there are 2,000 triangles in the boundary. All the meshes used to the experiments are large enough compared to the kernel size. We have to carefully set the kernel size because Algorithm 1 might cause a problem in case of the mesh is very fine.

Table 1 sumarizes the performance of our method and existing boundary handling approach for particle-based boundary [AIA*12] for all results. The boundary particles are placed on not only the polygon objects but also the boundary of simulation space. The timestep is 0.005s for all scenes. The computational time of our method depends on the number of the polygons used to represent the boundary and it is not good results compared to previous method due to the complex algorithm. On the other hand, our method has a benefit in terms of a memory consumption. Therefore, the method is well suited for scenes with many fluid particles and simple boundary shapes, such as a flood in buildings or terrains.

## 5. Conclusions and Future Work

We proposed a new boundary handling method for SPH that does not sample particles inside the boundary. Fast calculation by integration of a kernel function allows for direct interactions between the fluid particles and mesh boundary without particle stacking. Our approach can easily be extended to various SPH algorithms and we showed the results of this for the PBF framework.

As future work, we plan to extend our approach to two-way interactions using impulse-based techniques, since cur-

**Table 1:** *Perfomance results for all examples*

| Scene | particles | polygons | Our method time/step[s] | Our method memory | [AIA*12] time/step[s] | [AIA*12] memory | [AIA*12] boudary particles |
|---|---|---|---|---|---|---|---|
| Dam Break | 15k | 6 | 0.032 | 51MB | 0.03 | 62MB | 20k |
| Drop into Bowl | 70k | 4k | 0.56 | 108MB | 0.06 | 148MB | 170k |
| Dam Break with Bunnies | 105k | 15k | 2.49 | 159MB | 0.3 | 202MB | 98k |
| Flow on Bumpy Slope | 35k | 2k | 0.25 | 100MB | 0.12 | 196MB | 156k |

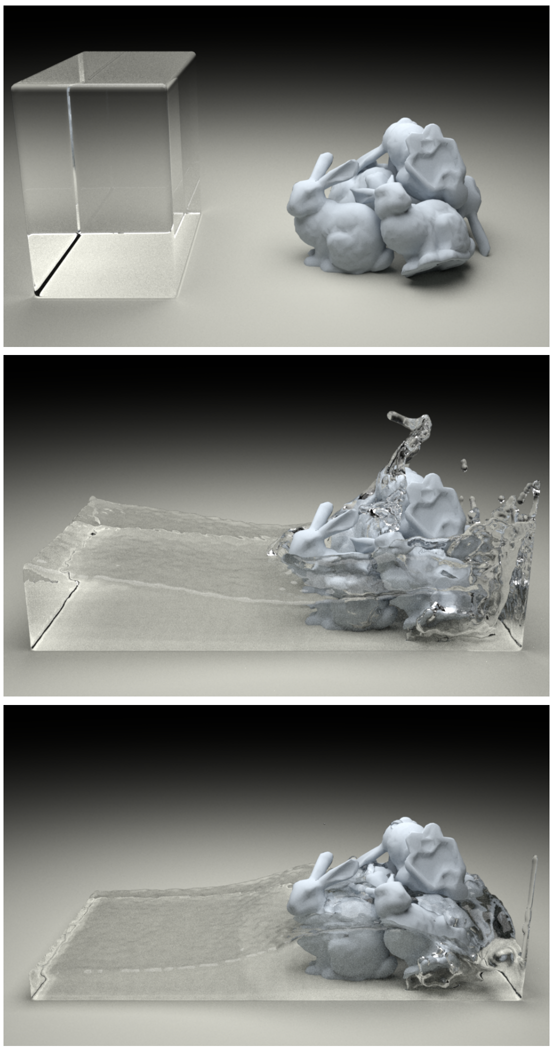


**Figure 5:** *Dam breaking scene with bunnies (105k particles and 15k polygon mesh).*

rent methods only support one-way interactions from the solid boundary to the fluid particles. Also the method would easily extend for the moving rigid obstacles because the cached $\gamma$ value is independent to a rotation and a translation. Our algorithm for the polygon boundary needs to be improved to handle boundaries in which the planes intersect in a more complex fashion in the effective sphere. We are also interested in extending to other kinds of surface representations, such as quadratic polynomials and spline surfaces. Moreover, we have to improve the performance of the method using more precomputation like a precomputated radial transfer [SKS02] or a combination with particle-based boundary handling approach for only complex boundary.

## Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 25730069 and 25289021. We also thank to anonymous reviewers for their constructive comments. The images in this paper were rendered using Mitsuba renderer[†].

## References

[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 (July 2012), 62:1–62:8. 2, 5, 6

[BIT09] Becker M., Ihmsen M., Teschner M.: Corotated sph for deformable solids. In *Proc. Eurographics Workshop on Natural Phenomena* (2009). 2

[BT07] Becker M., Teschner M.: Weakly compressible sph for free surface flows. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), pp. 209–217. 1

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

[BYM05] Bell N., Yu Y., Mucha P. J.: Particle-based simulation of granular materials. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM, pp. 77–86. 2

[†] http://www.mitsuba-renderer.org

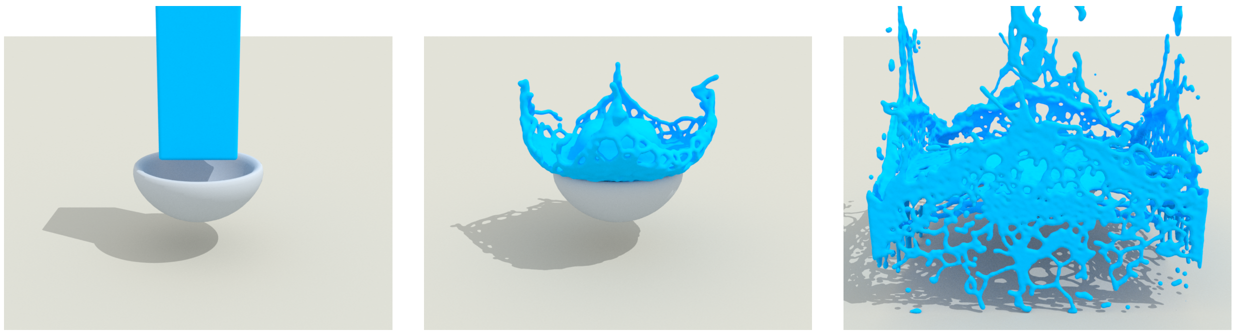


**Figure 6:** *Water dropping into a bowl-shaped mesh boundary (70k particles and 4k polygon mesh).*

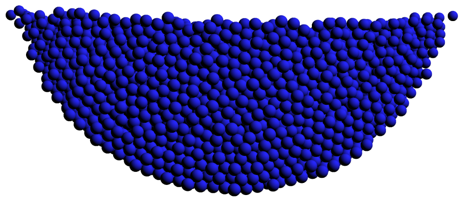


**Figure 7:** *Underlying simulation particles of the water drop scene.*

[CBP05] Clavet S., Beaudoin P., Poulin P.: Particle-based viscoelastic fluid simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (July 2005), pp. 219–228. 1, 2

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

[DTM*12] Du P., Tang M., Meng C., Tong R., Lin L.: A fluid/cloth coupling method for high velocity collision simulation. In *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry* (New York, NY, USA, 2012), VRCAI '12, ACM, pp. 309–314. 2

[FB07] Feldman J., Bonet J.: Dynamic refinement and boundary contact forces in sph with applications in fluid flow problems. *International Journal for Numerical Methods in Engineering 72* (2007), 295–324. 2

[FLR*13] Ferrand M., Laurence D., Rogers B., 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* (2013), 446–472. 2

[GM77] Gingold R. A., Monaghan J. J.: Smoothed particle hydrodynamics - theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society 181* (1977), 375–389. 1

[HEW15] Huber M., Eberhardt B., Weiskopf D.: Boundary handling at cloth-fluid contact. *Computer Graphics Forum 34*, 1 (2015), 14–25. 2

[HKK07] Harada T., Koshizuka S., Kawaguchi Y.: Smoothed particle hydrodynamics in complex shapes. In *Proc. Spring Conference on Computer Graphics* (2007), pp. 235–241. 2

[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 Trans. Graph. 34*, 1 (Dec. 2014), 7:1–7:9. 2

[IABT11] Ihmsen M., Akinci N., Becker M., Teschner M.: A parallel sph implementation on multi-core cpus. *Computer Graphics Forum 30*, 1 (March 2011), 99–112. 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 (Mar. 2014), 426–435. 1, 2

[IOS*14] Ihmsen M., Orthmann J., Solenthaler B., Kolb A., Teschner M.: Sph fluids in computer graphics. In *Eurographics 2014 - State of the Art Reports* (2014), pp. 21–42. 2

[KBLP04] Kulasegaram S., Bonet J., Lewis R. W., 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

[LL96] Li S., Liu W. K.: Moving least-square reproducing kernel method part ii: Fourier analysis. *Computer Methods in Applied Mechanics and Engineering 139*, 1-4 (1996), 159 – 193. 2

[LLB97] Liu W.-K., Li S., Belytschko T.: Moving least-square reproducing kernel methods (i) methodology and convergence. *Computer Methods in Applied Mechanics and Engineering 143*, 1-2 (1997), 113 – 154. 2

[Luc77] Lucy L. B.: A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal 82*, 12 (1977), 1013–1024. 1

[MCG03] Müller M., Charypar D., Gross M.: Particle-based fluid simulation for interactive applications. In *Proc. ACM/Eurographics Symposium on Computer Animation 2003* (2003), pp. 154–159. 1, 3

[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* (2015), 15–34. 2

[MHHR07] Müller M., Heidelberger B., Hennix M., Ratcliff J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (2007), 109–118. 2

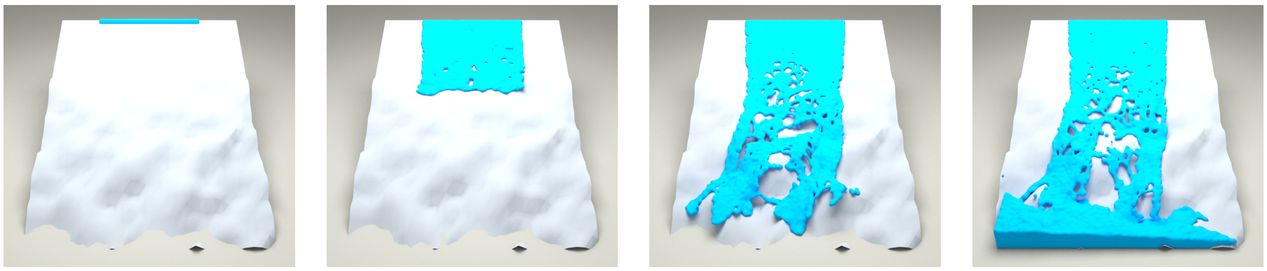


**Figure 8:** *Water flow on a bumpy downslope (maximum 35k particles and 2k polygon mesh).*

[MKA*04] MÜLLER M., KEISER R., A.NEALEN, PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. In *SCA2004* (2004). 2

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Trans. Graph. 32*, 4 (2013), 104:1–104:12. 1, 2, 4

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (July 2014), 153:1–153:12. 5

[Mon05] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Reports on Progress in Physics 68*, 8 (2005), 1703. 2

[OKR09] OH S., KIM Y., ROH B.-S.: Impulse-based rigid body interaction in sph. *Computer Animation and Virtual Worlds 20*, 2-3 (2009), 215–224. 2

[Pan04] PANIZZO A.: *Physical and numerical modelling of subaerial landslide generated waves*. PhD thesis, Universita degli studi di L ' Aquila, 2004. 2

[SB12] SCHECHTER H., BRIDSON R.: Ghost sph for animating water. *ACM Trans. Graph. 31*, 4 (July 2012), 61:1–61:8. 2

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 527–536. 6

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible sph. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers* (New York, NY, USA, 2009), ACM, pp. 1–6. 1

## Appendix A: Equation of $\gamma$ with Poly6 kernel

The Poly6 kernel for the density calculation in three-dimensional space is

$$W(r,h) = \frac{315}{64\pi h^9}\left(h^2 - r^2\right)^3.$$

Also, equation (7) can be rewritten as

$$\gamma_i = 2\pi\left(\int_0^{\theta_1} \sin\theta \int_0^{h} W(r,h) r^2 dr d\theta + \int_{\theta_1}^{\pi} \sin\theta \int_0^{\frac{c}{\cos\theta}} W(r,h) r^2 dr d\theta\right).$$

By substituting the Poly6 kernel function and equation (8) into above equation, we can obtain the following equation.

$$\begin{aligned}\gamma_i = & \frac{\cos\theta_1 + 1}{2} \\ & + \frac{315}{32h^9}\left(\frac{c^3h^6}{6}\left(1-\sec^2\theta_1\right) - \frac{3c^5h^4}{20}\left(1-\sec^4\theta_1\right)\right. \\ & \left. + \frac{c^7h^2}{14}\left(1-\sec^6\theta_1\right) - \frac{c^9}{72}\left(1-\sec^8\theta_1\right)\right).\end{aligned}$$