Correcting Free Surface Error in Smoothed Particle Hydrodynamics

Wojciech Karas (UCR), Tamar Shinar (UCR)

July 13, 2015

Abstract

Smoothed Particle Hydrodynamics(SPH) is a popular fluid simulation method in computer graphics; however, this method still suffers from surface-tension-like effects caused by incorrect density values near the free surface. Current solutions to this problem either require supplementary particle representation around the free surface or create integration errors. We propose a new approach that avoids the need for extra particles while attempting to maintain correctness and order of the particles at the surface. Based on particle data and the surface representation of known particle sets, we create a density correction function which attempts to make up for the incorrect density values near the free surface. Our tests with this correction indicate fewer surface-tension-like effects at the surface and make for smoother surfaces.

1 Introduction

Originally developed in the 1970s for use in astrophysics, SPH has become an important method used to simulate fluids in the computer graphics community. First adapted for this purpose in the 1990s, the last decade has seen much research effort expended in decreasing computation time and making it robust enough to tackle a variety of challenges. Although much progress has been made, some fundamental issues remain. One of the greatest problems with SPH has been its inability to correctly handle the free surface boundary. SPH was created with no built-in notion of a boundary. The result is a significant drop in the ability of the method to accurately model the fluid surface. As it is at the boundary of the fluid that much of the interesting physical phenomena and visual effects occur, this problem causes a large number of non-physical and visual artifacts that plague the quality of the simulation. Until good solutions to this problem exist, SPH will continue to be secondary to other methods when dealing with free surface flows.

There are currently two types of boundaries considered in most fluid simulations: solid and free surface. As of this time, there is much work available on the subject of solid boundaries. Most recently, [1] have provided a very good

solution for two-way coupling with rigid bodies that fixes many of the issues encountered with solids. In comparison, there has not been much work done in the computer graphics community on improving free surface boundaries. This is the boundary where the fluid and the surrounding air meet. Due to the enormous density ratio between the fluid and the air, the effect of the air is often considered negligible, and it is left out of the simulation. Without the air represented, there is a deficiency of particles around the free surface. Consequently, SPH simulations produce surface-tension-like artifacts that warp the fluid at the free surface and reduce stability. Our goal in this paper is to present a method which fixes these shortcomings without the use any additional particle representation around the free surface.

2 Related Works

An in-depth introduction to SPH is available in [10]. There is also a recent literature review detailing the various work that has been done within the graphics community[7]. Every part of the method is given consideration, but there is very little dealing with the particle deficiency at the free surface. When it is addressed, it is under the area of multiphase fluids and that is not being considered here.

[3] provides a thorough analysis into the role of the free surface in SPH before any discretization occurs. Their findings indicate that smoothed quantities computed at the free surface suffer from lower accuracy than those computed inside the fluid and require a correction based on the volume integral of the smoothing kernel. Although these are important considerations, their conclusions do not hold after discretization. The earliest work to adapt SPH to the free surface is [9]. Particle deficiency at the free surface is avoided by use of the continuity equation to evolve density through a first order differential equation in time. Using time integration introduces errors that accumulate and cause drift from the rest volume. For these reasons, this approach is avoided in most graphics papers.

The most common approach to density computation used is the summation approach[9]. This appraoch suffers greatly near the free surface because of the particle deficiency. Some simple fixes proposed have been the shepard filter[5] and to disallow negative pressures[6]. The former approach, while smoothing the density field and providing some improvement, does not completely eliminate the issue. The latter approach results in non-physical behavior by the fluid with particles lacking cohesion and moving like billiard balls. The only paper which has proposed a complete solution, without introducing a second phase of the fluid, is [12]. Their approach, inspired by the Ghost Fluid Method[4], seeds a band around the free surface with ghost particles. These ghost particles contribute to the fluid particles' density summation, solving the particle deficiency issue. They do not affect the fluid particles in any other way. Every X steps, these particles are resampled, ensuring they provide an accurate contribution to the density summation of the fluid particles. This approach provides the stabil-

ity and accuracy at the free surface needed in SPH; however, these extra ghost particles add extra, and non-trivial, computation time and memory overhead to the simulation. In addition, in their examples, the surfaces of their fluids appear to be noisier than is expected.

3 Smoothed Particle Hydrodynamics

A fluid is represented by a system of N particles. Each particle has a mass m_i, position x_i, velocity v_i, and acceleration a_i. Other quantities for particle i are computed using an interpolation of nearby particle values:

$$A(\mathbf{x}) = \sum_j A_j \frac{m_j}{\rho_j} W(\mathbf{x} - \mathbf{x}_j, h) \tag{1}$$

Derivatives of these smoothed quantities are calculated by taking the derivatives of the smoothing equation. These smoothed quantities are only dependent upon the values of the quantity at each particle. Each particle's contribution is weighed through the smoothing kernel W. This is a radially symmetric, monotonically decreasing, and normalized function; often with a finite support radius h. Here we use the spiky and poly_6 kernels of [11], The former for the gradient of the kernel and the latter for all other kernel evaluations.

In order to use the smoothing equation for a particle, the first quantity that must be computed is the density. We use the summation approach:

$$\rho_i(\mathbf{x}_i) = \sum_j m_j W(\mathbf{x}_i - \mathbf{x}_j, h) \tag{2}$$

The forces we consider on each particle are gravity, pressure, and viscosity. Gravity is a constant force. $\mathbf{F}_i^g = m_i \mathbf{g}$ where $\mathbf{g}$ is the constant acceleration due to gravity. This is set at $-9.8 \frac{m}{s^2}$ in the downward direction in our simulations. The pressure force is computed as:

$$\mathbf{F}_i^p = -\sum_j m_j (\frac{p_i}{\rho_j^2} + \frac{p_j}{\rho_i^2}) \nabla W(\mathbf{x}_i - \mathbf{x}_j, h) \tag{3}$$

The pressure of each particle must be computed first. We use the WCSPH [2] method:

$$p_i = k((\frac{\rho_i}{\rho_0})^7 - 1) \tag{4}$$

with $k = \frac{\rho_0 c_s^2}{7}$. ρ_0 is the user defined rest density which the pressure force attempts to enforce for each particle. We set it at $1002.66 \frac{kg}{m^3}$. c_s is the speed of sound in the fluid which we set to $10 \frac{m}{s}$. This assumes the fluid is weakly compressible with small variations in density. More recent developments, such as PCISPH [13] and IISPH [6], can also be used. The viscosity force, needed to keep the simulation stable, is taken from [2]:

$$\mathbf{F}_i^\nu = \sum_j m_j \nu (\frac{\mathbf{v}_{ij} \cdot \mathbf{x}_{ij}}{|\mathbf{x}_{ij}|^2 + \epsilon h^2}) \nabla W(\mathbf{x}_i - \mathbf{x}_j, h) \tag{5}$$

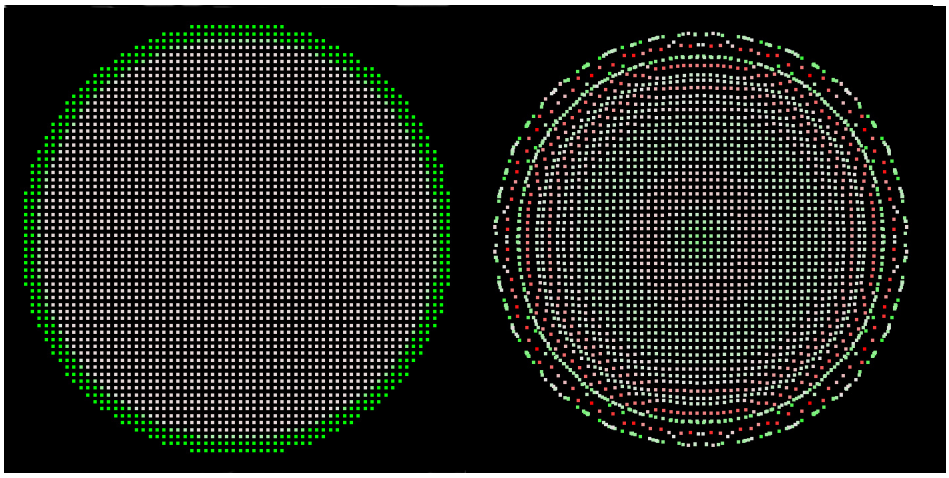


Figure 1: Particles on the left represent initial state of a fluid ball. Particles on the right show state of ball after two frames. The color and magnitude of a particle here, and in all other similar figures, correspond to how close to reference density a particle's density is within a 5% tolerance. Green means below, white means at, and red means over.

where $\nu = \frac{2\alpha h c_s}{\rho_i + \rho_j}$. . We set $\alpha = 0.01$ and $\epsilon = 0.01$. $\mathbf{v}_{ij}$ and $\mathbf{x}_{ij}$ are the relative velocity and position, respectively. This force is only used if $\mathbf{v}_{ij} \cdot \mathbf{x}_{ij} < 0$.

The sum of the forces on each particle are then related to its acceleration through Newton's Second Law. From there, the velocity and position can be computed through time integration. We used an euler step for both integrations. In addition, before integrating the velocity, we applied the XSPH method [8] by blending in a $\epsilon_X \sum_j \frac{m_j}{\rho_j}(\mathbf{v}_j - \mathbf{v}_i)$ term to each particle velocity. We set $\epsilon_X = 0.5$. This filters out noise in the particle velocities giving them a more cohesive movement.

The only interaction with solids we currently consider is that of the stationary square domain. We are currently using Reflective Boundary Conditions where particles coming in contact with a wall of the domain are reflected as in an elastic collision. This does not accurately model the interaction we want of the fluid with the domain and would like to integrate our approach with [1] in the future.

4 Density Correction

Although SPH works well in many domains, it has some major problems at the free surface. If the pressures for a pair of particles are too negative, the pressure force between these two particles will cause them to attract each other. This is fine inside the fluid where a particle has plenty of neighbors and variation from

the rest density is small. The ability of the pressure force to both repel and attract particles is important in maintaining particle distribution and keeping particles near rest density; however, at the free surface, where there are fewer fluid particle neighbors, this becomes an issue. The summation approach to computing density requires that there be sufficient particle neighbors in order to get a correct density value. With too few neighbors, the density of particles at the free surface will be too far below rest density. This causes an attractive pressure force between particles which results in the surface-tension-like artifacts at the free surface. This effect can be seen in Figure 1.

Instead of introducing new particles around the free surface, we propose a density correction scheme that will help to make up for the lost particle neighbors in the density summation and aim to keep the particles well ordered. The main idea is to correct the computed density of a particle near the free surface through a correction function $f(d_i)$ with d_i being the particle's distance to the free surface. This function is built ahead of the simulation and is based on density value data gathered from a variety of surfaces. These surfaces are represented by different sets of regularly spaced particles. The surfaces we used are shown in Figure 2. They were chosen to cover a variety of ways that the density can change with distance to the surface. With a given set of particles, we build the corresponding levelset surface representation $\phi(\mathbf{x}_i)$ using the surface definition given in [15]. After building the surface, we also run the fast marching method to make sure $\phi(\mathbf{x}_i)$ is a signed distance function. From this we can set $d_i = \phi(\mathbf{x}_i)$. For each surface, every particle within smoothing kernel radius h distance to the free surface has its distance d_i and the value of $\frac{\rho_i}{\rho_0}$, via the density summation approach, collected. This data is fit by the user to a chosen curve $g(d_i)$. From this we define $f(x_i)$ as:

$$f(d_i) = \begin{cases} g(d_i), & \text{if } |d_i| \leq h \text{ and } x_i \text{ is inside the surface} \\ g(0), & \text{if } x_i \text{ is at or outside the surface} \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

The second condition is used because early tests we ran showed incorrect particle movement that caused particles at the surface to drift out and away from it. The curve we used is shown in Figure 3. A fourth order polynomial was used:

$$g(d_i) = 191182d_i^4 + 8350.55d_i^3 - 515.42d_i^2 - 33.8543d_i + 0.516289 \tag{7}$$

In order to apply this during the simulation itself, we currently recreate the levelset representation of the surface at the beginning of each timestep. Then, as soon as the density is computed as normal, we divide each particle near the free surface by $f(d_i)$. We currently do not do the same for a particle whose smoothing kernel also intersects a solid domain wall.

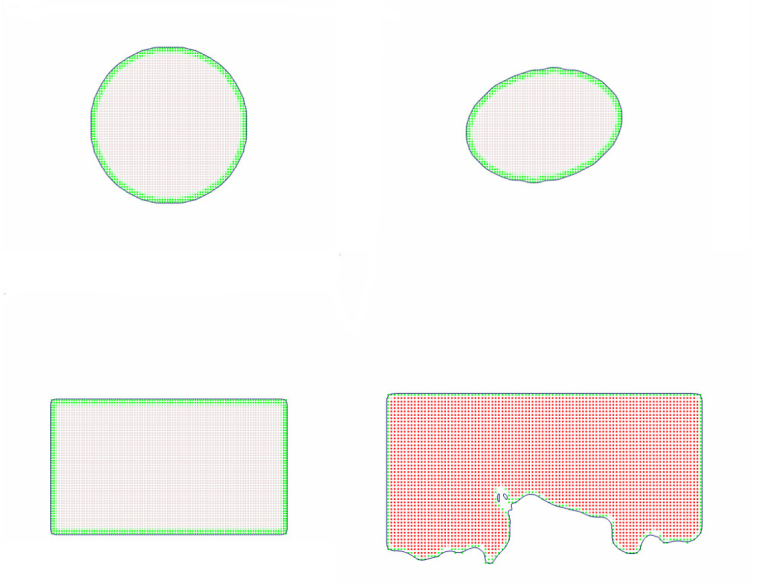


Figure 2: Four surfaces used in creation of correction function.

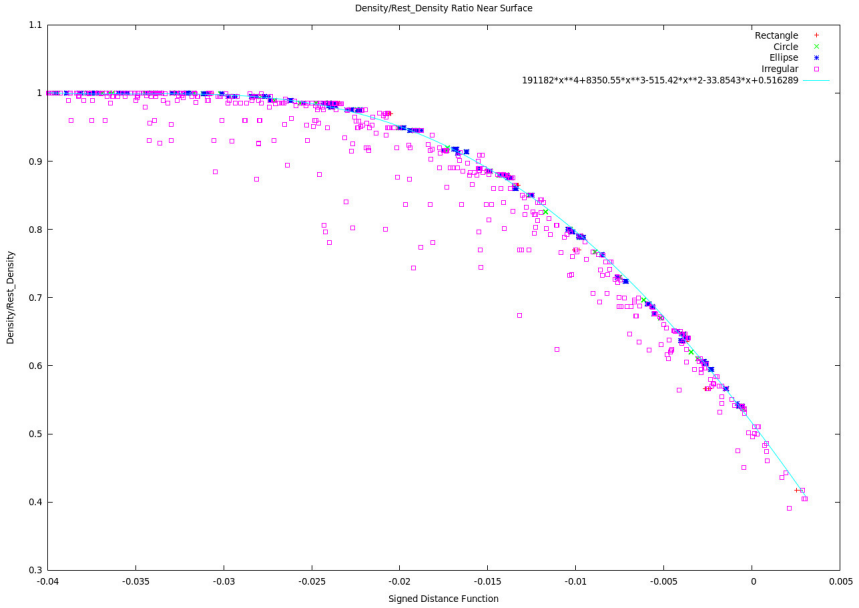


Figure 3: Data taken from all four surfaces and the quartic polynomial fit used.

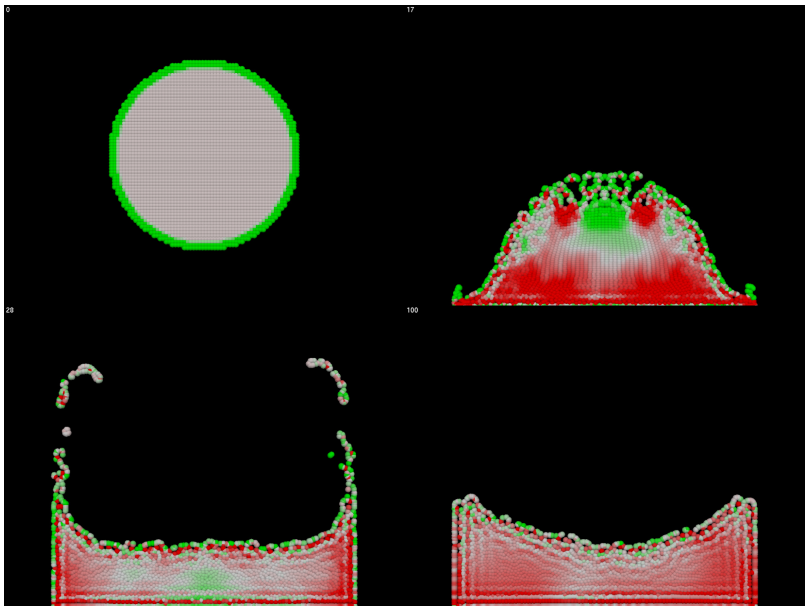


Figure 4: Frames from 2D simulation without density correction.

5 Results

Our simulations were run on a quadcore Intel i7-2600K with 16GB RAM. We have run tests in both 2D and 3D with a focus on the simulation of a ball of fluid hitting the bottom of a closed container. Background grids were used to speed up the neighborhood search and surface reconstruction. Surface reconstruction used a uniform grid with 128 cells in each direction. The fluid balls were sampled by taking a uniform grid of particles and deleting those which were outside of the ball. Each example was run for 100 frames with a general timestep of 0.000123077s. For the 2D example, we used 3,088 particles with a total running time of about two minutes without the density correction and about five and a half minutes with the correction. In the 3D example we had 129,648 particles with a total running time of about nine hours without the density correction and about thirty three hours with the correction. Our timesteps were chosen based on the definition of [2]. Our tests have not indicated any need to further limit the timestep based on our correction function. If the function is accurate enough, it should not destabilise the simulation. We have used the parameters and density correction function as mentioned in sections 3 and 4. Although the function we created was done using 2D surfaces, we have also found it working well enough for the 3D example. In the future we would like to try other functions made specifically for our 3D examples. Figure 4, 5, 6, and 7 will show some side by side comparisons of the examples with one simulation running without the density correction and the other running with the density

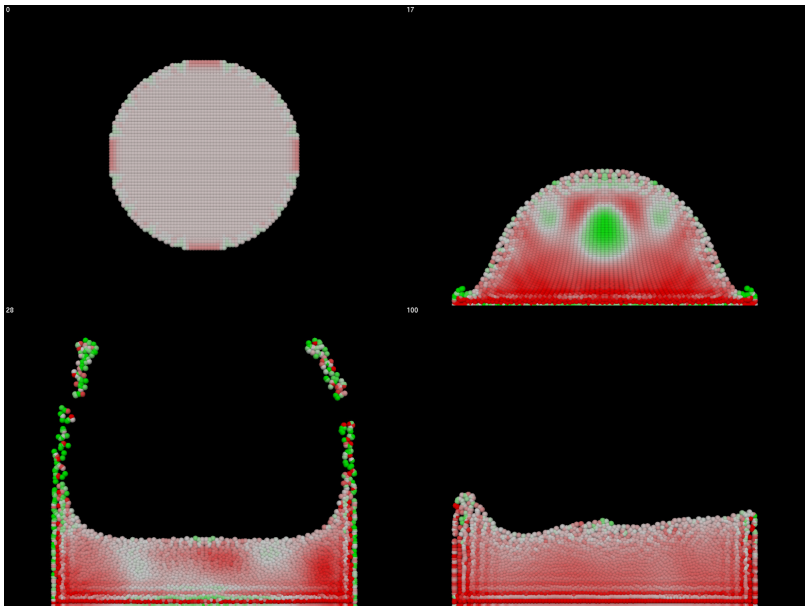


Figure 5: Frames from 2D simulation with density correction.

correction. Based on the particle colors, it can be seen that the particles begin in an underdense state without the correction but are very close to rest density with the correction. As the ball hits the bottom of the container, there is less distortion in the frame with the correction function applied. It can be seen that the surface in the corrected simulation has less noise than the one without the correction. Videos of these examples can be provided upon request.

6 Conclusion and Future Work

In this paper we presented a new correction method for SPH particle density near the free surface of a fluid. Instead of creating new particles, we have used the surface representation of the particles to create a density correction function ahead of the simulation that can be applied during a simulation. Although we have attempted to keep computational costs down by not introducing any new structures, our current model does incur some significant extra cost. Normally, the surface representation of the particles is computed only once per visual frame of the simulation. So far, our tests have computed a new surface representation at every new simulation step. This can incur plenty of extra cost. We would like to try and reduce the number times the surface representation is recreated. Like reseeding the air particles every X steps in Ghost SPH, the next step would be to try using the same surface representation for Y steps and see the effect on the stability and accuracy of the method. In addition to this, there are several other issues that need to be explored. Our density correction

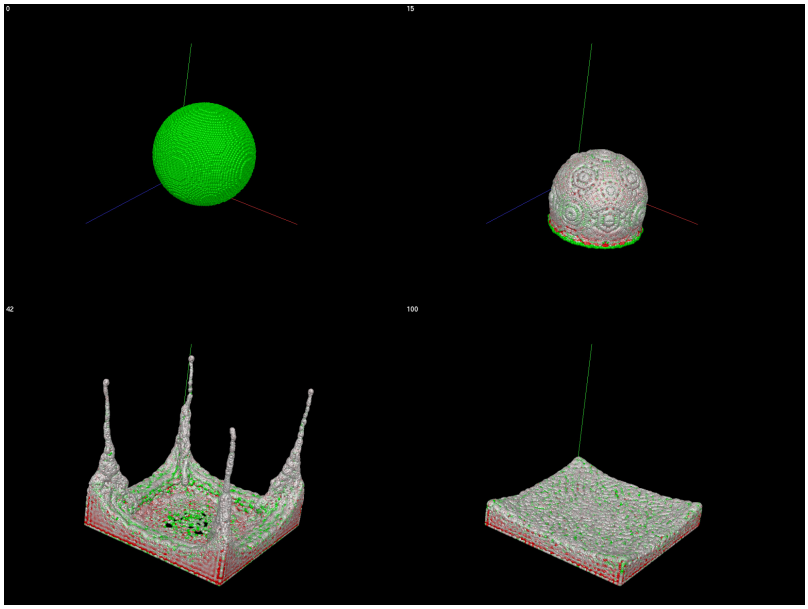


Figure 6: Frames from 3D simulation without density correction.

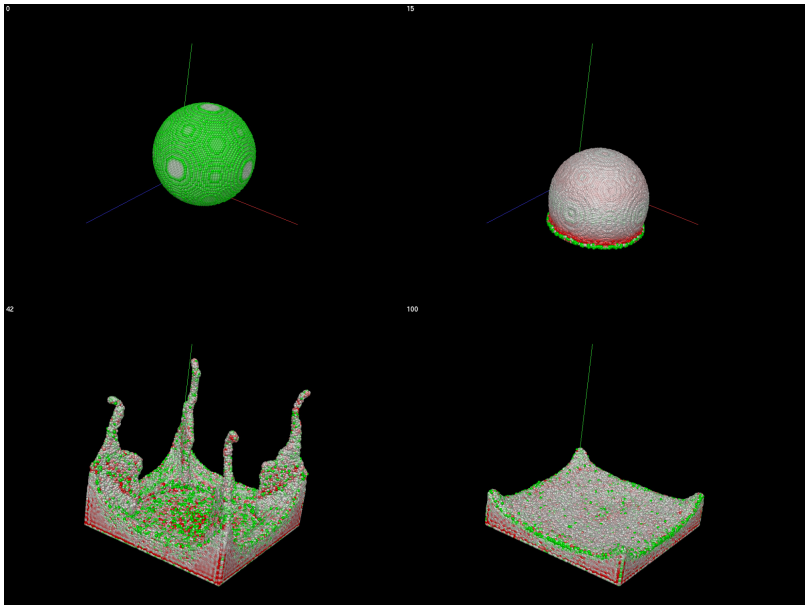


Figure 7: Frames from 3D simulation with density correction.

function currently only depends on the particle distance to the surface. As the distance to the surface can not determine a unique surface, our current function is a crude approximation. A particle near a surface with a sharp corner or thin filament will need a correction that is different than a particle near a flat surface. A new function taking into account the nearby surface shape would be needed for further improvement. Another area to improve in is the choice of surface representation. Our current choice was designed to be used with the method of [15] where particles do not carry variable notion of volume and density that SPH particles do in the weakly compressible approach. Using a surface representation that does take this into account, such as [11] and [14], might improve our method. Finally, as mentioned in section 3, we would like to extend our method to be compatible with solid surface boundary methods.

References

[1] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph.*, 31(4):62:1–62:8, July 2012.

[2] Markus Becker and Matthias Teschner. Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 209–217, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[3] Andrea Colagrossi, Matteo Antuono, and David Le Touzé. Theoretical considerations on the free-surface role in the smoothed-particle-hydrodynamics model. *Phys. Rev. E*, 79:056701, May 2009.

[4] Ronald P. Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method), 1999.

[5] Moncho Gomez-Gesteira, Benedict D. Rogers, Robert A. Dalrymple, and Alex J.C. Crespo. State-of-the-art of classical sph for free-surface flows. *Journal of Hydraulic Research*, 48(sup1):6–27, 2010.

[6] Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. Implicit incompressible sph. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):426–435, March 2014.

[7] Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. SPH Fluids in Computer Graphics. In Sylvain Lefebvre and Michela Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014.

[8] J. J. Monaghan. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30(1):543–574, 1992.

[9] J. J. Monaghan. Simulating free surface flows with sph. *J. Comput. Phys.*, 110(2):399–406, February 1994.

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

[11] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[12] Hagit Schechter and Robert Bridson. Ghost sph for animating water. *ACM Trans. Graph.*, 31(4):61:1–61:8, July 2012.

[13] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible sph. *ACM Trans. Graph.*, 28(3):40:1–40:6, July 2009.

[14] Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Trans. Graph.*, 32(1):5:1–5:12, February 2013.

[15] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Trans. Graph.*, 24(3):965–972, July 2005.