Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2017)
F. Jaillet and F. Zara (Editors)

Moving Least Squares Boundaries for SPH Fluids

Stefan Band Christoph Gissler Matthias Teschner

University of Freiburg, Germany

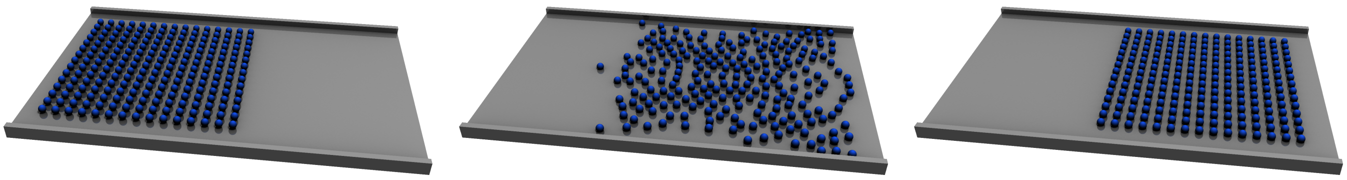


Figure 1: *Comparison of our proposed boundary handling scheme to Akinci et al. [AIA*12]. Left: initial setting. Middle: fluid particles slide down the inclined plane. The pattern is distorted due to erroneous oscillations in the density and pressure force computation. Right: our approach removes these erroneous oscillations.*

Abstract

*The paper shows that the SPH boundary handling of Akinci et al. [AIA*12] suffers from perceivable issues in planar regions due to deviations in the computed boundary normals and due to erroneous oscillations in the distance computation of fluid particles to the boundary. In order to resolve these issues, we propose a novel boundary handling that combines the SPH concept with Moving Least Squares. The proposed technique significantly improves the distance and normal computations in planar boundary regions, while its computational complexity is similar to Akinci's approach. We embed the proposed boundary handling into Implicit Incompressible SPH in a hybrid setting where it is applied at planar boundaries, while Akinci's technique is still being used for boundaries with complex shapes. Various benefits of the improved boundary handling are illustrated, in particular a reduced particle leakage and a reduced artificial boundary friction.*

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

Keywords: physically-based animation, fluid simulation, Smoothed Particle Hydrodynamics, Moving Least Squares, boundary handling

1. Introduction

Particle-based boundary representations in fluid simulations are flexible and easy to handle with Smoothed Particle Hydrodynamics (SPH) [Mon92]. As shown, e.g. in [IAGT10, AIA*12], particles can represent solid boundaries with arbitrary geometric complexity. Varying sample sizes and potentially missing kernel contributions can be handled, which makes it remarkably easy to generate particle boundaries from scratch or from alternative representations such as triangle meshes [ACAT13].

However, particle-based boundary representations also have issues. In particular, this paper shows that the boundary handling scheme of Akinci et al. [AIA*12] suffers from perceivable issues in planar regions due to deviations in the boundary normal computation and due to erroneous oscillations in the distance computation of fluid particles to the true boundary. To address these issues, we propose a combination of the versatile unified particle representation of boundaries [AIA*12] with a Moving Least Squares (MLS) technique [ABCO*03]. Thereby, we locally reconstruct the surface of the true boundary by fitting boundary particles to a plane. Hence, for planar boundaries, we get precise surface normals and accurate distance information, while non-planar boundaries are still being handled with the approach of Akinci et al. [AIA*12]. We show that our approach completely eliminates problematic oscillations in planar regions, while its computational complexity is comparable to [AIA*12]. We further show that the proposed boundary handling reduces the artificial boundary viscosity caused by the particle discretization.

For the experiments, we incorporated our MLS boundaries into the Implicit Incompressible SPH (IISPH) pressure solver [ICS*14]. Unlike state equation solvers, e.g. [MCG03, BT07], IISPH computes pressure p by solving a pressure Poisson equation (PPE) of the form $\Delta p = \boldsymbol{s}$, cf. [Cho68]. The source term $\boldsymbol{s}$ encodes a predicted density deviation at all particles that has to be corrected by pressure accelerations using the computed pressure field. The proposed boundary handling is embedded into IISPH and contributes to both sides of the PPE. While the improved distance computation contributes to the source term, the corrected normals do so in the discretization of the Laplacian.

Organization: The remainder of this paper is organized as follows. The following section describes existing approaches related to the handling of solid boundaries in SPH. The issues of [AIA*12] in planar regions are illustrated. Sections 3 and 4 discuss the proposed concept to eliminate these issues. Implementation details are described in Section 5. In Section 6, we compare our method to [AIA*12] and show how it improves the simulation results. Finally, we conclude in Section 7.

2. Related Work

SPH has been used in computer graphics first in 1995 by Stam and Fiume to simulate gaseous phenomena and fire [SF95]. Müller, Charypar and Gross extended their approach in 2003 to simulations of compressible fluids [MCG03]. From that time on, research has focused on practical formulations for incompressible fluids [SP09, HLL*12, MM13, ICS*14], multiphase simulations [MSKG05,SSP07] and highly viscous fluids [TDF*15,PT16]. We refer readers to [IOS*14] for a survey of SPH fluids. In this work, we focus on one specific aspect of SPH fluid simulations, namely the modeling of solid boundaries.

Boundary Handling in SPH: Many SPH-based fluid simulations model boundaries with particles [Mon05,ICS*14,BK16,TDNL16]. For instance, in [Mon05], boundary particles exert penalty forces on the surrounding fluid particles as soon as they are within a certain distance. Penalty forces should prevent that fluid particles penetrate the boundary, but since they also lead to large pressure variations within the fluid, small time steps are required to produce a smooth pressure field.

In order to achieve larger time steps, [BTT09] proposed the direct forcing method. Computing control forces and velocities with a predictor-corrector-scheme allows simulating one- and two-way-coupled rigid bodies. However, due to an incomplete support domain, approximating field variables with SPH is problematic at boundaries. As a result, fluid particles tend to stick to the boundary if a distance-based penalty force or the direct forcing method is used. Therefore, boundary particles should contribute to the reconstruction of field variables. For this purpose, various techniques have been developed. For example in [MFZ97, SSP07, IAGT10] boundary particles are treated like fluid particles. This means that each particle has its own density and pressure value.

Another technique to treat boundary conditions is the usage of ghost particles [CL03, YRS09, SB12]. For fluid particles that are located at a certain distance to the boundary, a ghost particle is generated, which has the same viscosity, mass, density and pressure as its associated fluid particle. But generating such ghost particles is challenging for complex boundaries. Also, the particle sampling of the boundary has a significant influence on the numerical stability and quality of the simulation. While simple objects like cuboids can be easily represented by uniformly distributed particles, for complex objects with convex or concave shapes an irregular sampling is inevitable. Furthermore, for performance reasons only the surface of these objects should be represented by particles. Akinci et al. propose to treat irregular samplings by computing volume contributions [AIA*12]. Based on the concept of the particle number density [OS03, SP08], their approach mirrors the hydrodynamic quantities of a fluid particle, i.e. density and pressure, onto its neighboring boundary particles. While adhering to the concept of SPH, their approach is efficient to compute and allows a versatile coupling of fluids and solid objects.

Alternative to particles, there exist other methods to efficiently represent boundaries, e.g. with triangle meshes [HEW15, FM15]. On the one hand, triangular meshes are advantageous for representing large planar boundaries. The reason is that, compared to particles, the number of primitives required to represent the boundary can be considerably reduced [HEW15]. Furthermore, surface normals and distances to the boundary are precisely computable [FM15]. Yet, on the other hand, handling discontinuous surface normals and non-manifold structures that cause spatial and temporal discontinuities of the fluid properties is challenging for arbitrary geometric boundaries.

Issues of Particle-based Boundary Handling: Employing the particle representation of solid boundaries described by [AIA*12], the density ρ_i of a fluid particle i at time t is computed as

$$\rho_i(t) = \underbrace{m_i \sum_f W_{if}(t)}_{\rho_{i\leftarrow\text{fluid}}} + \underbrace{\sum_b m_b W_{ib}(t)}_{\rho_{i\leftarrow\text{boundary}}}, \tag{1}$$

where m is the mass of a particle and W the SPH smoothing kernel. We use subscript f and b to distinguish between fluid and boundary neighbors, respectively. The mass m_b of boundary particle b depends on the rest density ρ_i^0 of fluid particle i (mirrored density). It is computed via the volume V_b of the boundary particle: $m_b = \rho_i^0 V_b$. The density computation can be split into two distinct parts: a part $\rho_{i\leftarrow\text{fluid}}$ that considers only how neighboring fluid particles contribute to the density and another part $\rho_{i\leftarrow\text{boundary}}$, which considers only the contribution of boundary particles. Applying the same idea to the pressure acceleration $\boldsymbol{a}_i^p$ of fluid particles (mirrored pressure) and omitting time indices results in

$$\boldsymbol{a}_i^p = \underbrace{-\sum_f m_f \left(\frac{p_i}{\rho_i^2} + \frac{p_f}{\rho_f^2}\right) \nabla W_{if}}_{\boldsymbol{a}^p_{i\leftarrow\text{fluid}}} \underbrace{- \sum_b m_b \frac{p_i}{\rho_i^2} \nabla W_{ib}}_{\boldsymbol{a}^p_{i\leftarrow\text{boundary}}}, \tag{2}$$

where the terms $\boldsymbol{a}^p_{i\leftarrow\text{fluid}}$ and $\boldsymbol{a}^p_{i\leftarrow\text{boundary}}$ describe the contribution of neighboring fluid and boundary particles, respectively. To simplify the following equations, we define the vector $\boldsymbol{n}_i$ for a fluid particle i as

$$\boldsymbol{n}_i = \sum_b m_b \nabla W_{ib}\,. \tag{3}$$

Thereby, we can write the pressure acceleration caused by the boundary in a more compact form: $\boldsymbol{a}^p_{i\leftarrow\text{boundary}} = -p_i/\rho_i^2\, \boldsymbol{n}_i$. Please note that this acceleration points into the direction of $\boldsymbol{n}_i$

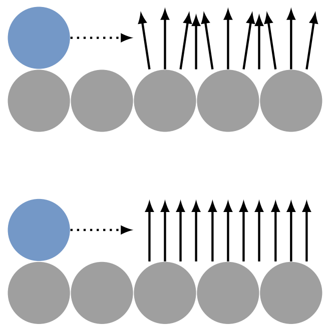


Figure 2: *Top: the boundary handling scheme of Akinci et al. [AIA*12] can lead to small oscillations of the magnitude and direction of the pressure forces exerted from the boundary. Bottom: for of a planar boundary, the pressure forces should rather point into the direction of the plane's normal and have an equal magnitude.*

and that $\boldsymbol{n}_i$ is in general not normalized. For planar boundaries, $\boldsymbol{n}_i$ should be orthogonal to the boundary.

As illustrated in Fig. 2, the boundary handling scheme of Akinci et al. [AIA*12] has the following issues: as the fluid particle moves on the planar boundary, its spatial relation to its neighboring boundary particles changes. This in turn leads to small oscillations in the distance computation to the true boundary and therefore to an erroneous evaluation of the density contribution $\rho_{i\leftarrow\text{boundary}}$. The same problem arises in the computation of the pressure acceleration term $\boldsymbol{a}^p_{i\leftarrow\text{boundary}}$. In this case, the small deviations result in oscillations of the magnitude and direction of the pressure forces, whereas the forces should always act in normal direction of the plane. We address these issues in Section 3 and propose replacements of the terms $\rho_{i\leftarrow\text{boundary}}$ and $\boldsymbol{n}_i$ by locally reconstructing the surface of the true boundary by fitting a fluid particle's boundary neighbors to a plane. Further, we will show that this allows a precise evaluation of both terms.

Point Set Surfaces: Reconstructing 3D surfaces from oriented point samples is a well-studied area of research in computer graphics and vision. For example, it allows fitting of scanned data, filling of surface holes and re-meshing of existing models, e.g. [ABCO*03, PKKG03, FCOS05]. One approach to define a surface from a set of points was introduced to computer graphics by Alexa et al. [ABCO*03]. Thereby, the surface is locally approximated with polynomials using a MLS fitting procedure. This technique is advantageous, since MLS is insensitive to noise and leads to a smooth reconstructed surface. However, it is sometimes necessary to know the intrinsic topology of the point samples and to have a parametrization of the surface before surface fitting is applicable. Since, in general, this is a non-trivial task, in this work, we restrict the fitting procedure to planar surfaces.

3. Method

In this section, we first describe how fitting a plane from the positions of a fluid particle's boundary neighbors can done very efficiently by using MLS. Furthermore, we show how the information that we gain from the fitting procedure allow an accurate computation of the density and pressure forces at the boundary.

Planar Fitting of 3D Point Samples: A plane is implicitly described by a normal vector $\boldsymbol{n} = (a,b,c)^T$ and an offset d, such that for any point $\boldsymbol{p} = (x,y,z)^T$ on the plane the following holds true:

$$ax + by + cz + d = 0. \tag{4}$$

Given a set of N point samples $\{(x_i,y_i,z_i)^T\}$, we want to determine a, b, c and d such that the resulting plane best fits the point samples. Assuming that the z-component is functionally dependent on the x- and y-components, without loss of generality we can set the coefficient $c = 1$ and minimize the squares of the deviations R of the point samples:

$$\text{minimize } R^2(a,b,d) \equiv \text{minimize } \sum_{i=1}^{N} (ax_i + by_i + z_i + d)^2. \tag{5}$$

This procedure is known as least squares [KK56]. It optimizes the squares of the residuals perpendicular to the x-, y- or z-axis, not the residuals perpendicular to the plane. But this is not an issue since all our point samples, i.e. boundary particles, should lie close to the resulting plane. The conditions for R^2 to be a minimum are

$$\frac{\partial R^2}{\partial a} = 2\sum_{i=1}^{N} (ax_i + by_i + z_i + d)\,x_i = 0 \tag{6}$$

$$\frac{\partial R^2}{\partial b} = 2\sum_{i=1}^{N} (ax_i + by_i + z_i + d)\,y_i = 0 \tag{7}$$

$$\frac{\partial R^2}{\partial d} = 2\sum_{i=1}^{N} (ax_i + by_i + z_i + d) = 0, \tag{8}$$

which in turn lead to the following equations in matrix form:

$$\begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum y_i x_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & N \end{bmatrix} \begin{bmatrix} a \\ b \\ d \end{bmatrix} = - \begin{bmatrix} \sum x_i z_i \\ \sum y_i z_i \\ \sum z_i \end{bmatrix}. \tag{9}$$

However, if solved directly, this formulation may lead to an ill-conditioned linear system. To avoid this, we first compute the barycenter $\bar{\boldsymbol{p}} = (\bar{x},\bar{y},\bar{z})^T = \frac{1}{N}\sum_{i=1}^{N}(x_i,y_i,z_i)^T$ of the point samples and then subtract it from each point sample. Thereby, and by substituting $\bar{x}_i = x_i - \bar{x}$, $\bar{y}_i = y_i - \bar{y}$ and $\bar{z}_i = z_i - \bar{z}$ Eq. (9) simplifies to:

$$\begin{bmatrix} \sum \bar{x}_i^2 & \sum \bar{x}_i \bar{y}_i & 0 \\ \sum \bar{y}_i \bar{x}_i & \sum \bar{y}_i^2 & 0 \\ 0 & 0 & N \end{bmatrix} \begin{bmatrix} a \\ b \\ d \end{bmatrix} = - \begin{bmatrix} \sum \bar{x}_i \bar{z}_i \\ \sum \bar{y}_i \bar{z}_i \\ 0 \end{bmatrix}. \tag{10}$$

And finally, Cramer's rule gives us $d = 0$ and

$$a = \frac{N\sum\bar{x}_i\bar{y}_i\sum\bar{y}_i\bar{z}_i - N\sum\bar{x}_i\bar{z}_i\sum\bar{y}_i^2}{N\sum\bar{x}_i^2\sum\bar{y}_i^2 - N(\sum\bar{x}_i\bar{y}_i)^2} \tag{11}$$

$$b = \frac{N\sum\bar{x}_i\bar{y}_i\sum\bar{x}_i\bar{z}_i - N\sum\bar{x}_i^2\sum\bar{y}_i\bar{z}_i}{N\sum\bar{x}_i^2\sum\bar{y}_i^2 - N(\sum\bar{x}_i\bar{y}_i)^2}. \tag{12}$$

Remember, we assumed that the z-component is functionally dependent on the x- and y-component. If this is not the case, the matrix's determinant becomes zero, which ultimately results in a division by zero. To avoid this problem, we do the above calculations

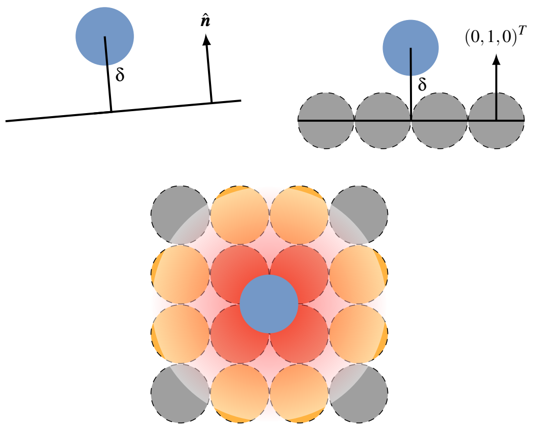


Figure 3: *Top-left: the neighboring boundary particles of the blue-colored fluid particle are fitted to a plane with normal* $\boldsymbol{n}$ *and distance* δ*. Top-right: a virtual boundary plane is created in a local coordinate system. Bottom: view of the virtual boundary from above. Boundary particles that may contribute to the SPH interpolation are colored in red and orange, whereat the red-colored particles contribute more than the orange ones.*

for all three separate assumptions, namely that each component is functionally dependent on the other two. If the point samples do span a plane, at least one of these assumptions will lead to a non-zero determinant. Therefore, we choose the most well-behaved one, i.e. the one with the largest determinant.

MLS Boundary Handling: With the procedure described above, we can now fit the neighboring boundary particles of a fluid particle to a plane. Depending on the largest determinant, this gives us a vector $\boldsymbol{n}$ and a point $\bar{\boldsymbol{p}}$ on the plane. We go ahead by normalizing $\hat{\boldsymbol{n}} = \boldsymbol{n}/||\boldsymbol{n}||$ and by defining that the fluid particle should always be located in the positive half-space of the plane. Hence, we switch the normals sign if $\hat{\boldsymbol{n}} \cdot \boldsymbol{x}_i < 0$. In a next step, to solve the problem of an inaccurate density estimation and to stabilize the magnitude of the pressure acceleration, we compute the distance $\delta = \hat{\boldsymbol{n}} \cdot (\boldsymbol{x}_i - \bar{\boldsymbol{p}})$ of the fluid particle to the fitted plane, as illustrated in the top-left image of Fig. 3. With this distance it would be possible to analytically compute the boundary's contribution to the density, e.g. as done in [FM15]. But since we want the computational overhead to be as low as possible, and even more important, to avoid sharp contrasts of the density and pressure acceleration near regions, where the fitting procedure might fail, we proceed by numerically computing the boundary's contribution with SPH as follows.

First, we locally transform the coordinate system such that the fluid particle is located at the origin. Further, the coordinate system is rotated such that the normal $\hat{\boldsymbol{n}}$ is axis-aligned with, e.g., the y-axis. In this local coordinate system, we sample a virtual plane at the distance δ below the fluid particle with virtual boundary particles, each having the same mass. Next, we evaluate the SPH kernel and the magnitude of its gradient for each virtual boundary particle. This can be done very efficiently by employing the symmetry and anti-symmetry properties of the kernel, i.e. an explicit representation of the virtual plane is not required. For example, in the setting of Fig. 3, where the SPH smoothing length is chosen as $2h$, the fluid particle with diameter h can at most interact with the orange- and red-colored boundary particles. Since each of the red-colored boundary particles has the same distance $d_{\text{red}} = ||(0.5h, \delta, 0.5h)^T||$ to the fluid particle, and all orange-colored particles have the same distance $d_{\text{orange}} = ||(1.5h, \delta, 0.5h)^T||$ to the fluid particle, the cost for computing the virtual boundary's contributions is reduced severely. Overall, an accurate computation of the density contribution $\rho_{i\leftarrow\text{boundary}}$ simplifies to just two evaluations of the kernel:

$$\rho_{i\leftarrow\text{boundary}} = 4\,W(d_{\text{red}}) + 8\,W(d_{\text{orange}})\,. \qquad (13)$$

To get an improved magnitude of the vector $\boldsymbol{n}_i$, the magnitude of the kernel gradient has to be evaluated two times. The final value for $\boldsymbol{n}_i$, including the direction, is then obtained by a multiplication with the normal $\hat{\boldsymbol{n}}$ of the fitted plane:

$$\boldsymbol{n}_i = (4\,\nabla W(d_{\text{red}}) + 8\,\nabla W(d_{\text{orange}}))\,\hat{\boldsymbol{n}}\,. \qquad (14)$$

4. Discussion

Applicability of MLS boundaries: In the previous section, we derived new formulations for computing the density and pressure acceleration of fluid particles at the boundary. These formulations can be used as a replacement for the boundary contributions in Eqs. (1) and (2). However, naively fitting each fluid particle's boundary neighbors to a plane might fail or even lead to wrong simulation results in the following cases: first, if a fluid particle does not have at least three boundary neighbors, a plane is not uniquely definable. Second, while fitting a plane from boundary particles might be possible, the real boundary and the fitted plane might significantly differ, e.g. at corners or edges of a rectangular box. Besides, if the boundary particles lie exactly in a plane, but the plane has holes, we also do not want to apply the fitting procedure. This is because our virtual boundary is always sampled completely. And therefore, we would lose the information of the holes. For these reasons, our MLS boundaries are not applicable to the whole simulation domain.

A solution to these problems is detecting the above cases and then use the boundary handling of Akinci et al. [AIA*12]. However, as a consequence, we handle the transition between the two boundary handling schemes carefully - for instance, discontinuities in the density field have to be avoided.

Comparison to mesh-based approaches: If the boundary is given by a triangle mesh, a typical approach is to sample the boundary with particles and pre-compute surface normals for each boundary particle using the normals of the triangles. To compute a normal at an arbitrary position, the normals of the nearby boundary particles are interpolated. However, while this technique produces smooth surface normals in planar regions, it requires the storage of one normal per boundary particle. In contrast to this, our approach does not have an additional storage requirement, since the boundary normals are computed on the fly. Another conceptual difference is that whether or not our MLS boundaries approach is used is decided by the fluid particles and not by the boundary itself, i.e. fluid particles can choose the best model of the boundary's interface at runtime and account for errors in the density and normal estimations.

Algorithm 1 Simulation update for IISPH with MLS boundaries

```
procedure PREDICT DENSITY
    for each fluid particle $i$ do
        compute density $\rho_i(t)$ (1)
    for each fluid particle $i$ do
        predict velocity $\boldsymbol{v}_i^\star = \boldsymbol{v}_i(t) + \Delta t\, \boldsymbol{a}_i^{\text{non-pressure}}(t)$
    for each fluid particle $i$ do
        if FIT PLANE($i$) then                                ▷ MLS boundaries
            compute improved normal $\boldsymbol{n}_i$ (14)
            compute improved $\rho_{i\leftarrow\text{boundary}}$ (13)
            recompute density $\rho_i(t)$ (1)
        else                                       ▷ Akinci's boundary handling
            compute normal $\boldsymbol{n}_i = \sum_b m_b \nabla W_{ib}$
        predict density $\rho_i^\star = \rho_i(t) - \Delta t\, \rho_i^0 \nabla \cdot \boldsymbol{v}_i^\star$
        compute diagonal element $a_{ii}$
        initialize pressure $p_i = 0$
procedure COMPUTE PRESSURE                                    ▷ solve PPE
    while not converged do
        for each fluid particle $i$ do
            compute pressure acceleration $\boldsymbol{a}_i^p$ (2)
        for each fluid particle $i$ do
            compute divergence $\nabla \cdot \boldsymbol{a}_i^p$
            compute source term $s_i = \frac{\rho_i^0 - \rho_i^\star}{\Delta t^2}$
            update pressure $p_i = \max\left(0, p_i + \frac{\omega}{a_{ii}}\left(s_i - \nabla \cdot \boldsymbol{a}_i^p\right)\right)$
procedure INTEGRATION
    for each fluid particle $i$ do
        $\boldsymbol{v}_i(t+\Delta t) = \boldsymbol{v}_i^\star + \Delta t\, \boldsymbol{a}_i^p$
        $\boldsymbol{x}_i(t+\Delta t) = \boldsymbol{x}_i(t) + \Delta t\, \boldsymbol{v}_i(t+\Delta t)$
```

Neighborhood size: Another approach to reduce the erroneous oscillations at the boundary is to increase the smoothing length of the SPH kernel. However, even if boundaries are represented by only one layer of particles, the number of boundary neighbors per fluid particle grows quadratically with the smoothing length. Therefore, this approach also dramatically increases the computation cost. In contrast to this, our approach achieves very good results in planar regions while only using a small neighborhood size.

5. Implementation Details

In this section, we explain the details of our implementation. Therefore, we show how our novel MLS boundary handling scheme is incorporated into a SPH framework, exemplified at the IISPH pressure solver described by Ihmsen et al. [ICS*14]. Although we embedded our approach only into IISPH, we suppose that it is usable in any SPH fluid solver that models boundaries with [AIA*12].

Algorithm 1 summarizes the simulation update. Since our proposed boundary handling scheme should only be applied to fluid particles near planar boundaries, our implementation of the pressure solver is compatible to [ICS*14]. The only thing that changed is the Predict Density procedure. In this procedure, we precompute and store the coefficients required in the Jacobi update of the pressure solver, e.g. the diagonal element a_{ii}. Additionally, we now try to locally fit a plane from the boundary neighbors of each fluid particle. If we succeed, then we overwrite the coefficients with the improved values, i.e. with Eqs. (13) and (14). Otherwise, we keep them and therefore get the same behavior as in [ICS*14]. Like in [ICS*14], we only need to store seven additional scalar values per particle. But in contrast to [ICS*14], we have a slightly increased cost of at most three iterations over a fluid particle's boundary neighbors: one to compute their barycenter, one to compute their plane's normal and another one to check, whether all the following criteria are satisfied - since, as discussed in Section 4, we have to take care of where to use MLS boundaries. Overall, we apply our new MLS scheme only for fluid particles if:

- The fluid particle has at least three boundary particle neighbors to fit a plane.
- The largest determinant in the plane fitting procedure is not zero.
- The largest projected distance of a neighboring boundary particle to the fitted plane is smaller than a specified threshold, e.g. $0.05h$. This criterion describes how well the boundary particles really match the shape of a plane.
- We project the fluid particle's position onto the fitted plane. If the minimal distance of a neighboring boundary particle to this projected position is larger than an user-defined threshold, e.g. h, we do not apply our method, because the plane has a hole beneath the fluid particle.
- For a smooth transition between the two boundary handling schemes, neither the deviation of the density nor the pressure force's magnitude should be larger than 10 % compared to [AIA*12].

In our implementation, we use compact hashing [IABT11] to find particle neighbors. Furthermore, we parallelize all computations with Intel Threading Building Blocks [Phe08]. For the SPH interpolation, we use the cubic spline kernel [Mon05]. Viscosity is modeled as proposed by [MFZ97]. If not stated otherwise, we chose $\mu = 0.002$ for all experiments. Surface tension and adhesion effects are mimicked as explained in [AAT13].

6. Results

In this section, we compare our novel MLS-based boundary handling method to [AIA*12]. We used different particle diameters h for the simulations, yet the SPH smoothing length was always chosen as $2h$. The rest density of the simulated fluids was $1000\,\text{kg/m}^3$ while the largest permissible degree of compression was kept at 0.1 %. We computed all simulations on a 12-core @2.6 GHz Intel Xeon E5-2690 with 32 GB of RAM.

Erroneous Speed and Density: First, we compare our new approach to [AIA*12] in a simple setting, where 225 fluid particles are resting on a solid plane (see Fig. 1). We simulated this scenario with a fixed time step $\Delta t = 0.001\,\text{s}$ and with a particle diameter $h = 0.025\,\text{m}$. The plane is inclined at an angle $\alpha = 0.6^\circ$, is uniformly sampled and has free-slip boundary conditions. Therefore, we would expect that the fluid particles will slowly begin to slide down the plane without changing their initial pattern. While this is not the case for [AIA*12], with our approach the particles behave as expected. Furthermore, with the boundary handling of Akinci et

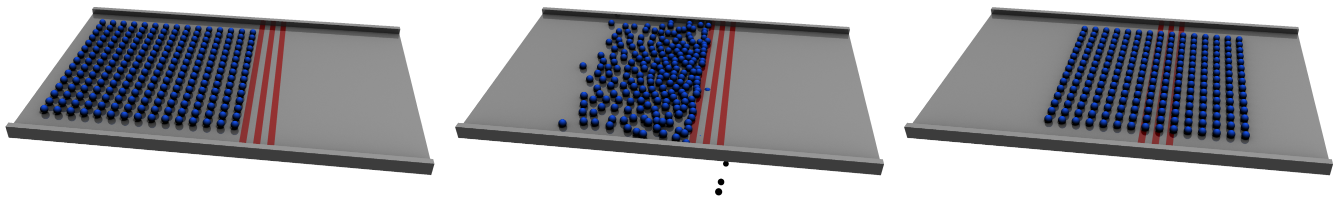


Figure 4: *The simulation setting used for the comparison with the boundary handling scheme of Akinci et al. [AIA*12]. Left: initial setting. The red areas show oversampled regions. Middle: as the blue-colored fluid particles move closer to these regions, their computed density is erroneous. As a result, fluid particles get stuck and can not move past these regions. Even a few particles leak through the boundary. Right: our proposed MLS-based boundary handling scheme does not suffer from these issues.*

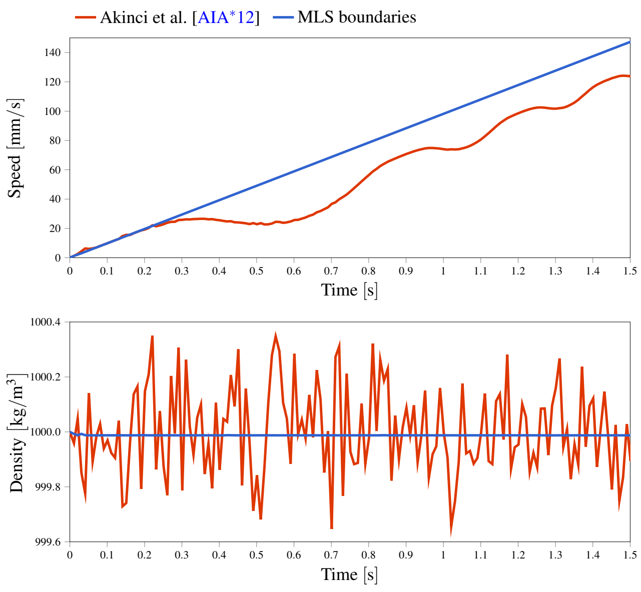
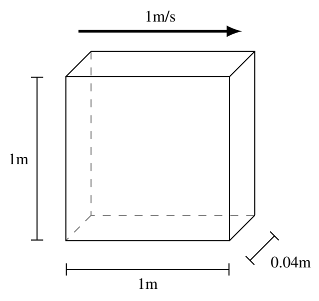


Figure 5: *Speed (top) and density (bottom) over time of a fluid particle that slides down a inclined plane with free-slip boundary conditions. With our approach the particle's speed matches exactly the expected value $v(t) = t\,g\,sin(\alpha)$.*

al. [AIA*12] the speed and density of a single tracked particle differ much from the expected values (see Fig. 5). The particle even slows down a few times instead of constantly increasing its speed.

Leakage: A second scenario for comparing our approach with [AIA*12] is illustrated in Fig. 4. As in the first scenario, we initialized the fluid above an inclined plane and simulated it with a fixed time step $\Delta t = 0.001\,\mathrm{s}$ and particle diameter $h = 0.025\,\mathrm{m}$. Additionally, the three red-colored areas show regions, where the boundary plane is oversampled by a factor of five, i.e. these regions are represented by five times more particles than the rest of the plane. Interestingly enough this degree of oversampling is not unusual for scenarios with complex geometric boundaries. Again, as the fluid particles start to move, with [AIA*12] they immediately break up their initial sampling pattern. Due to erroneous computations of the density and pressure accelerations near the oversampled regions, even 61 particles leak through the plane. In contrast to this, the fluid particles neither break up their initial sampling pattern nor move through the boundary with our new approach - they can even pass the oversampled regions without experiencing any distortions.

Artificial Boundary Viscosity: Next, we compare our new approach to [AIA*12] in a 3D lid-driven cavity scenario. The lid-driven cavity is commonly used in computational fluid dynamics, e.g. [LVFK14], for evaluating the quality of the simulation. Thereby, as illustrated in Fig. 6, a viscous fluid is placed inside a rectangular box. All walls of this box, except the top, have no-slip boundary conditions. At the top fluid particles move with a constant speed of $1\,\mathrm{m/s}$, driving the fluid under the effect of viscosity. We



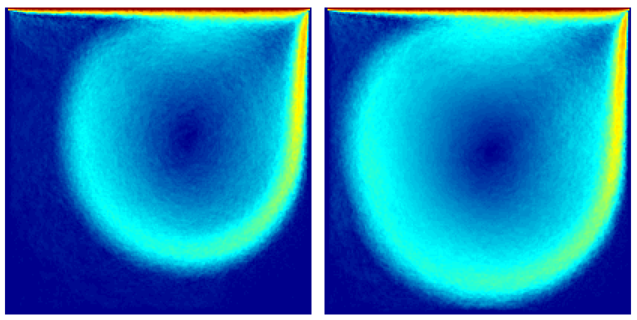


Figure 6: *Top: initial setting for the 3D lid-driven cavity scenario. Velocities are color-coded, where red indicates a high and blue a low speed. Bottom: compared to [AIA*12] (left) our method (right) reduces the artificial viscosity caused by the particle-based boundary handling.*

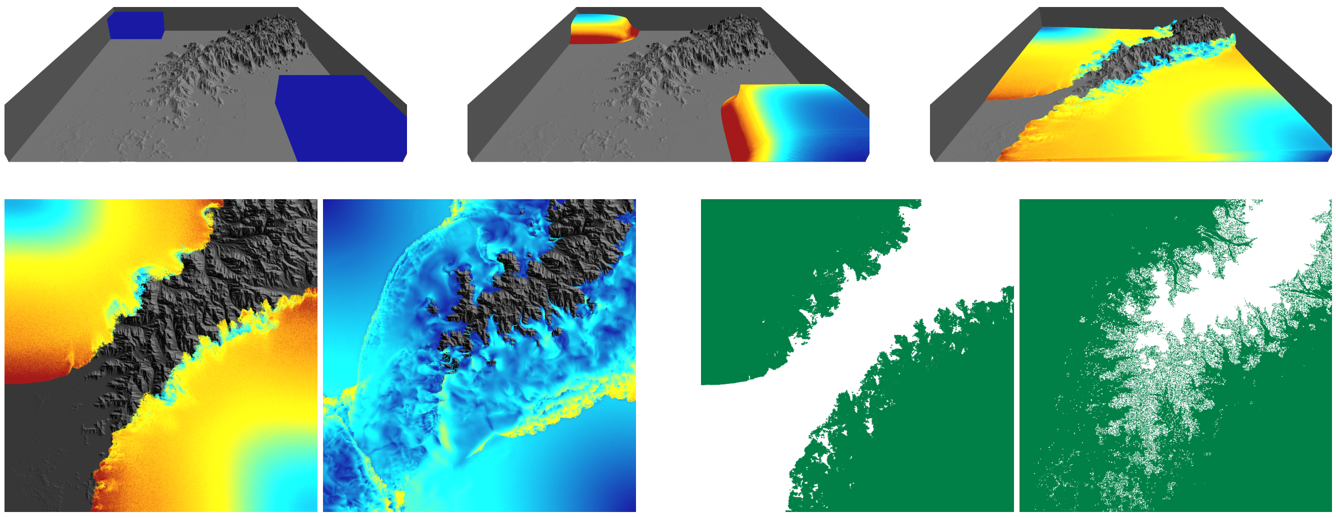


Figure 7: *Top: two breaking dams with a complex geometric boundary at different times and color-coded velocities, where blue corresponds to a small, red to a high velocity. Bottom: the scene from bird's eye view. Our MLS boundary handling scheme is applied in the green-colored planar regions in the two images on the right.*

used a rectangular box with dimensions $1\,\text{m} \times 1\,\text{m} \times 0.04\,\text{m}$. The particle diameter was $h = 0.005\,\text{m}$, thus making a total of 320 k particles. We fixed the time step at $\Delta t = 0.002\,\text{s}$. Furthermore, we kept the pressure solver's number of iterations fixed at two to measure the performance overhead of the solver's initialization procedure. In order to compare our approach to [AIA*12], we placed a two-dimensional sensor plane at the box's center and measured the fluid's speed. As shown in the two bottom images of Fig. 6, our approach reduces the artificial viscosity effects of the boundary, i.e. the fluids speed is less reduced and therefore the fluid can reach the bottom wall of the box. This effect can also be seen by taking a look at the average kinetic energy, i.e. velocity, of a fluid particle. With our approach, a fluid particle has an average speed of $161.87\,\text{mm/s}$ in all simulation steps, while with [AIA*12] the average speed is $116.65\,\text{mm/s}$, i.e. 28 % smaller. Furthermore, our new boundary handling scheme does not add much performance overhead to the simulation. In this scenario, many fluid particles can see the boundary, since the box is thin. Moreover, the boundary is planar and hence our new method is applicable nearly to all fluid particles at the boundary. We fall back to [AIA*12] only at the corners and edges of the rectangular box. Overall, in this scenario the cost per simulation step is increased with our new approach only by 2.5 %.

Large-scale Scenario: Finally, to show that our approach can also handle more complex scenarios, we simulated a fluid that is represented by 52.3 million particles (see Fig. 7). Furthermore, to improve the visual quality of the simulation, we added surface tension and adhesion forces as proposed by [AAT13] with $\gamma = 0.1$ and $\beta = 0.01$. Each particle has a diameter $h = 0.0045\,\text{m}$. We used an adaptive time step [IAGT10], which was on average $\Delta t = 0.005\,\text{s}$. Overall, the average computation time was 17.87 s per simulation step and the pressure solver required an average of 6.65 iterations. Since the scene has many planar regions, our MLS boundaries are applicable to a large number of fluid particles near the boundary. This is indicated in the two bottom-right images of Fig. 7. Also important to mention is that we get smooth transitions between regions, where our new approach is applicable and the non-planar regions, where we use [AIA*12].

7. Conclusion and Future Work

We presented a novel approach to improve the handling of planar boundaries in SPH fluid simulations. For that purpose, we first showed that, due to erroneous oscillations in the density and pressure force calculations, the boundary handling scheme of Akinci et al. [AIA*12] suffers from perceivable issues near planar regions. To address these issues, we combined a particle representation of boundaries with a MLS technique. Thereby, we first try to locally fit a fluid particle's boundary neighbors to a plane. In a next step, we compute the boundary's contribution by evaluating the distance to virtual boundary particles on the fitted plane instead of evaluating the density and pressure forces for the real boundary particles. While adding only a small amount of performance overhead, our approach completely eliminates the problematic oscillations. And furthermore, it reduces the artificial boundary viscosity caused by the particle discretization scheme.

As future work, we plan to extent our method to locally reconstruct more complex surfaces from the boundary particles, other than planes. Using a more exact representation of the boundary would have several benefits: first, we may avoid the critical transition between two boundary handling schemes. And second, missing contributions of overlapping boundaries could be taken into account more accurately. This would stabilize the density and pressure force calculations even further. Moreover, we plan to extend our approach such that it also stabilizes two-way interactions between fluids and solid objects.

Acknowledgments

We would like to thank the reviewers for their helpful comments. This project is supported by the German Research Foundation (DFG) under contract number TE 632-1/2.

References

[AAT13] AKINCI N., AKINCI G., TESCHNER M.: Versatile Surface Tension and Adhesion for SPH Fluids. *ACM Transactions on Graphics (TOG) 32*, 6 (Nov. 2013), 182:1–182:8. 5, 7

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., T. SILVA C.: Computing and Rendering Point Set Surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (Jan. 2003), 3–15. 1, 3

[ACAT13] AKINCI N., CORNELIS J., AKINCI G., TESCHNER M.: Coupling Elastic Solids with SPH Fluids. *Computer Animation and Virtual Worlds 24*, 3-4 (2013), 195–203. 1

[AIA*12] AKINCI N., IHMSEN M., AKINCI G., SOLENTHALER B., TESCHNER M.: Versatile Rigid-fluid Coupling for Incompressible SPH. *ACM Transactions on Graphics (TOG) 31*, 4 (July 2012), 62:1–62:8. 1, 2, 3, 4, 5, 6, 7

[BK16] BENDER J., KOSCHIER D.: Divergence-Free SPH for Incompressible and Viscous Fluids. *IEEE Transactions on Visualization and Computer Graphics* (2016). 2

[BT07] BECKER M., TESCHNER M.: Weakly Compressible SPH for Free Surface Flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), Eurographics Association, pp. 209–217. 1

[BTT09] BECKER M., TESSENDORF H., TESCHNER M.: Direct Forcing for Lagrangian Rigid-Fluid Coupling. *IEEE Transactions on Visualization and Computer Graphics 15*, 3 (May 2009), 493–503. 2

[Cho68] CHORIN A. J.: Numerical Solution of the Navier-Stokes Equations. *Mathematics of Computation 22*, 104 (1968), 745–762. 2

[CL03] COLAGROSSI A., LANDRINI M.: Numerical Simulation of Interfacial Flows by Smoothed Particle Hydrodynamics. *Journal of Computational Physics 191*, 2 (Nov. 2003), 448–475. 2

[FCOS05] FLEISHMAN S., COHEN-OR D., SILVA C. T.: Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics (TOG) 24*, 3 (July 2005), 544–552. 3

[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. 2, 4

[HEW15] HUBER M., EBERHARDT B., WEISKOPF D.: Boundary Handling at Cloth-Fluid Contact. *Computer Graphics Forum 34*, 1 (Feb. 2015), 14–25. 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. 2

[IABT11] IHMSEN M., AKINCI N., BECKER M., TESCHNER M.: A Parallel SPH Implementation on Multi-Core CPUs. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 99–112. 5

[IAGT10] IHMSEN M., AKINCI N., GISSLER M., TESCHNER M.: Boundary Handling and Adaptive Time-stepping for PCISPH. In *Workshop in Virtual Reality Interactions and Physical Simulation VRIPHYS* (2010). 1, 2, 7

[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, 2, 5

[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). 2

[KK56] KENNEY J., KEEPING E.: *Mathematics of Statistics Part I*. Van Nostrand, 1956. 3

[LVFK14] LEROY A., VIOLEAU D., FERRAND M., KASSIOTIS C.: Unified semi-analytical wall boundary conditions applied to 2-D incompressible SPH. *Journal of Computational Physics 261* (2014), 106–129. 6

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based Fluid Simulation for Interactive Applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 154–159. 1, 2

[MFZ97] MORRIS J. P., FOX P. J., ZHU Y.: Modeling Low Reynolds Number Incompressible Flows Using SPH. *Journal of Computational Physics 136*, 1 (Sept. 1997), 214–226. 2, 5

[MM13] MACKLIN M., MÜLLER M.: Position Based Fluids. *ACM Transactions on Graphics (TOG) 32*, 4 (July 2013), 104:1–104:12. 2

[Mon92] MONAGHAN J. J.: Smoothed Particle Hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 1 (1992), 543–574. 1

[Mon05] MONAGHAN J. J.: Smoothed Particle Hydrodynamics. *Reports on Progress in Physics 68*, 8 (2005), 1703. 2, 5

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based Fluid-fluid Interaction. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 237–244. 2

[OS03] OTT F., SCHNETTER E.: A modified SPH approach for fluids with large density differences. In *ArXiv Physics e-prints* (2003), p. 3112. 2

[Phe08] PHEATT C.: Intel® Threading Building Blocks. *Journal of Computing Sciences in Colleges 23*, 4 (Apr. 2008), 298–298. 5

[PKKG03] PAULY M., KEISER R., KOBBELT L. P., GROSS M.: Shape Modeling with Point-sampled Geometry. In *ACM SIGGRAPH 2003 Papers* (2003), pp. 641–650. 3

[PT16] PEER A., TESCHNER M.: Prescribed Velocity Gradients for Highly Viscous SPH Fluids with Vorticity Diffusion. *IEEE Transactions on Visualization and Computer Graphics* (2016). 2

[SB12] SCHECHTER H., BRIDSON R.: Ghost SPH for Animating Water. *ACM Transactions on Graphics (TOG) 31*, 4 (July 2012), 61:1–61:8. 2

[SF95] STAM J., FIUME E.: Depicting Fire and Other Gaseous Phenomena Using Diffusion Processes. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (1995), pp. 129–136. 2

[SP08] SOLENTHALER B., PAJAROLA R.: Density Contrast SPH Interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), pp. 211–218. 2

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective Incompressible SPH. *ACM Transactions on Graphics (TOG) 28*, 3 (July 2009), 40:1–40:6. 2

[SSP07] SOLENTHALER B., SCHLÄFLI J., PAJAROLA R.: A Unified Particle Model for Fluid&Ndash;Solid Interactions. *Computer Animation and Virtual Worlds 18*, 1 (Feb. 2007), 69–82. 2

[TDF*15] TAKAHASHI T., DOBASHI Y., FUJISHIRO I., NISHITA T., LIN M. C.: Implicit Formulation for SPH-based Viscous Fluids. *Computer Graphics Forum 34*, 2 (2015), 493–502. 2

[TDNL16] TAKAHASHI T., DOBASHI Y., NISHITA T., LIN M. C.: An Efficient Hybrid Incompressible SPH Solver with Interface Handling for Boundary Conditions. *Computer Graphics Forum* (2016). 2

[YRS09] YILDIZ M., ROOK R., SULEMAN A.: SPH with the multiple boundary tangent method. *International Journal for Numerical Methods in Engineering 77*, 10 (2009), 1416–1438. 2