Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2018)
F. Jaillet, G. Zachmann, K. Erleben, and S. Andrews (Editors)

MLS Pressure Extrapolation for the Boundary Handling in Divergence-Free SPH

Stefan Band† Christoph Gissler Andreas Peer Matthias Teschner

University of Freiburg, Germany

Figure 1: *Our boundary handling processes complex geometries with reduced artifacts and more efficiently compared to previous methods.*

Abstract

We propose a novel method to predict pressure values at boundary particles in incompressible divergence-free SPH simulations (DFSPH). Our approach employs Moving Least Squares (MLS) to predict the pressure at boundary particles. Therefore, MLS computes hyperplanes that approximate the pressure field at the interface between fluid and boundary particles. We compare this approach with two previous techniques. One previous technique mirrors the pressure from fluid to boundary particles. The other one extrapolates the pressure from fluid to boundary particles, but uses a gradient that is computed with Smoothed Particle Hydrodynamics (SPH). We motivate that gradient-based extrapolation is more accurate than mirroring. We further motivate that our proposed MLS gradient is less error prone than the SPH gradient at the boundary. In our experiments, we indicate artifacts in previous approaches. We show that these artifacts are significantly reduced with our approach resulting in simulation steps that can be twice as large compared to previous methods. We further present challenging and complex scenarios to illustrate the capabilities of the proposed boundary handling.

CCS Concepts

•***Computing methodologies*** → ***Physical simulation;*** *Massively parallel and high-performance simulations;*

1. Introduction

Iterative pressure solvers such as PCISPH [SP09], IISPH [ICS*14] or DFSPH [BK17] compute a pressure field p and apply pressure accelerations of the form $\boldsymbol{a}_i^{\mathrm{p}} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}$ to particles i. The sum considers all neighboring particles j of particle i. The variables $m, \rho, \nabla W$ denote mass, density and the gradient of the SPH kernel function W, respectively. There exist minor variations, but the solvers follow the same concept (see Section 4).

Iterative solvers typically compute pressure p_f only at fluid particles f. Pressure p_b at boundary particles b is not computed, but approximated if needed. This is, e.g., the case in the computation of the pressure acceleration $\boldsymbol{a}_f^{\mathrm{p}}$ at fluid particles f close to the boundary. Here, the respective SPH sum iterates over fluid neighbors with known pressure, but also over boundary neighbors with unknown pressure. I.e.,

$$\boldsymbol{a}_f^{\mathrm{p}} = -\sum_{f_f} m_{f_f} \left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2}\right) \nabla W_{ff_f} - \sum_{f_b} m_{f_b} \left(\frac{p_f}{\rho_f^2} + \frac{p_{f_b}}{\rho_{f_b}^2}\right) \nabla W_{ff_b} \tag{1}$$

with f_f and f_b denoting fluid and boundary neighbors of fluid par-

† bands@informatik.uni-freiburg.de

DOI: 10.2312/vriphys.20181068

ticle f, respectively. Equation (1) requires a notion of mass m_b, density ρ_b, and pressure p_b at boundary particles b. The density ρ_b is typically set to the rest density of the adjacent fluid and the mass m_b can be geometrically motivated from the boundary particle volume, see e.g. [AIA*12].

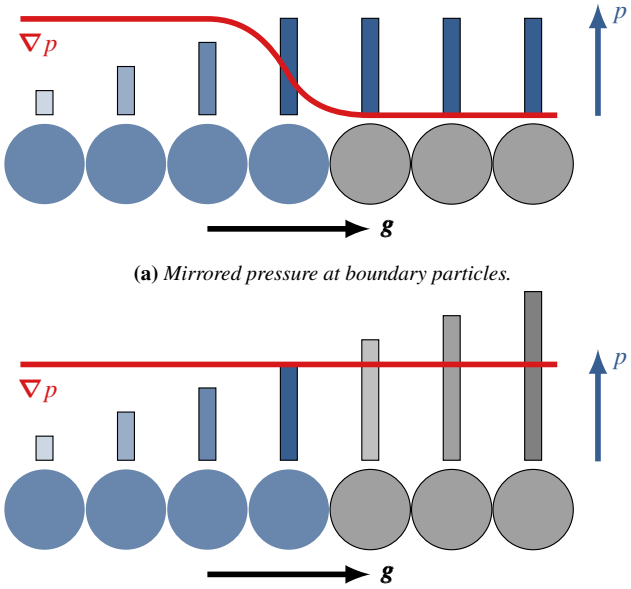
In terms of the pressure p_b, there exist different options. E.g., Akinci et al. [AIA*12] propose to mirror known pressure from fluid particles to adjacent rigid particles. Alternatively, Adami et al. [AHA12] propose to extrapolate pressure from the fluid to boundary particles using an SPH approximation of the pressure gradient.

Our contribution: We propose a novel variant to predict unknown pressure p_b at boundary particles b. In contrast to Akinci et al. [AIA*12], we compute one unique pressure value per boundary particle. We further extrapolate the pressure using the pressure gradient instead of copying pressure from the fluid to the boundary. This improves the quality of the pressure gradient of fluid particles near the boundary as illustrated in Fig. 2. In contrast to Adami et al. [AHA12], where pressure is extrapolated using SPH, we propose to use MLS [KK56, Nea04]. MLS is more accurate than SPH in case of particle deficiency which can particularly be the case near the boundary. We illustrate artifacts when using the boundary handling of Akinci et al. [AIA*12] and Adami et al. [AHA12]. We further show that these artifacts can be reduced with the proposed MLS extrapolation of pressure values. We have implemented our boundary handling in a DFSPH framework. Capabilities of the approach are illustrated for scenarios with challenging boundary setups, e.g. Fig. 1. We particularly show performance gain factors of up to two compared to [AIA*12].

Organization: The remainder of this paper is organized as follows. The following Section 2 describes existing approaches related to SPH fluid simulation and the handling of solid boundaries. In Section 3, we discuss the proposed pressure extrapolation concept whereas implementation details are described in Section 4. In Section 5, we show simulations employing our method and compare it to the boundary handling schemes of Akinci et al. [AIA*12] and Adami et al. [AHA12]. Finally, we conclude in Section 6.

2. Related Work

SPH is a popular choice for Lagrangian simulations in computer graphics [IOS*14]. First used by Stam and Fiume [SF95] to simulate gaseous phenomena and by Desbrun and Cani [DG96] for deformable objects, Müller et al. [MCG03] employed SPH to simulate compressible fluids. From that time on, research has focused on practical formulations of incompressible fluids with recent improvements in volume preservation [ICS*14, BK17, TDNL16], multiphase simulation [MSKG05, SP08, RLY*14, ATO17], highly viscous fluids [PICT15, TDF*15, PT17, BGFAO17, WKBB18] and deformable objects [KAG*05, SSP07, PGBT18]. Incompressibility can be enforced in various ways. Unlike non-iterative state equation solvers, e.g. [Mon94, MCG03, APKG07, BT07], iterative SPH pressure solvers, such as PCISPH [SP09], IISPH [ICS*14] and DFSPH [BK17], compute a pressure field p by solving a pressure Poisson equation (PPE) of the form $\nabla^2 p = s$, c.f. [Cho68]. Thereby, s is a source term that either encodes a predicted density deviation [SL03, SP09, ICS*14], the divergence of a velocity field [CR99, PDC*03] or a combination of both [HA07, BK17]. Computing the pressure field from a global formulation seems to improve the stability of the simulation. Rather large time steps can typically be used compared to the aforementioned non-iterative state equation solvers.



(a) *Mirrored pressure at boundary particles.*

(b) *Extrapolated pressure at boundary particles.*

Figure 2: *Mirroring pressure from fluid to boundary particles results in erroneous pressure gradients at fluid particles near the boundary. Extrapolating the pressure instead improves the gradient computation.*

This paper focuses on the optimization of the boundary handling. Therefore, we briefly discuss related works regarding the modeling of solid boundaries in the next section.

Boundary Handling in SPH As particle-based representations are very flexible and can handle arbitrarily shaped geometries, representing solid boundaries with particles is a popular choice for SPH fluid simulations, e.g. [Mon05, IAGT10, ICS*14, BK17, TDNL16]. One popular technique for handling fluid-boundary contact is to apply penalty forces between two particles as soon as they are within a certain distance, e.g. [Mon05, MST*04, MK09]. Penalty forces should prevent fluid particles from penetrating the boundary. Therefore, the magnitude of the penalty force is determined based on a penetration measure between the particles. As the results are sensitive to the stiffness parameter of the penalty force, small time steps are typically required to produce a smooth pressure field.

In order to achieve larger time steps, the direct forcing method of Becker et al. [BTT09] uses a predictor-corrector scheme to compute control forces and velocities. This method guarantees non-penetration. However, due to an incomplete support domain at the boundary, stacking of fluid particles can occur.

Another technique to treat boundaries are ghost particles [CL03, YRS09, SB12]. For fluid particles that are located at a certain dis-

tance to the boundary, a narrow layer of ghost particle is generated. Those ghost particles mirror the hydrodynamic quantities of their associated fluid particle, i.e. they have the same viscosity, mass, density and pressure. However, for complex geometries, generating such ghost particles is challenging. Furthermore, ghost particles have to be re-generated per simulation step.

Using only one layer of pre-generated boundary particles, Akinci et al. [AIA*12] treat irregular samplings by computing volume contributions, c.f. [OS03, SP08], and by mirroring the quantities of a fluid particle onto its neighboring boundary particles. While adhering to the SPH concept, this approach is efficient to compute and allows a versatile coupling of fluids and solid objects [ACAT13]. Band et al. [BGT17] proposed an extension of the boundary handling scheme of Akinci et al. [AIA*12] by employing MLS. To improve the accuracy of the density estimate and normal computation in planar regions, they locally reconstruct the surface of the true boundary by fitting boundary particles to a plane. This approach results in a smooth representation of the boundary. However, it is only applicable to planar boundaries. Furthermore, Band et al. [BGT17] do neither compute unique pressure values nor perform any pressure extrapolation at boundary particles. Instead, they use the mirroring scheme of Akinci et al. [AIA*12]. Yet, MLS techniques have been successfully applied in many research areas, e.g. [Dil99, ABCO*03, MKN*04, BRHN11].

Instead of mirroring fluid particle quantities onto the boundary, Adami et al. [AHA12] propose to use pre-generated dummy boundary particles. Thereby, fluid particles at the boundary interact with dummy particles according to the overlap of the kernel function. This has the advantage that, even for complex geometries, the boundary is well-described through-out the simulation. Furthermore, by extrapolating the pressure of boundary particles from the surrounding fluid particles, this method allows an accurate approximation of a fluid particle's pressure gradient near the boundary.

As an alternative to particles, boundaries are also representable with triangle meshes [HEW15, FM15]. Yet, as stated in [AIA*12], handling discontinuous surface normals and non-manifold structures that cause spatial and temporal discontinuities of the fluid properties is challenging for triangulated boundaries. Another alternative is an implicit representation of the boundary as proposed by Koschier and Bender [KB17]. Based on a pre-computed density map, this approach allows to efficiently evaluate the density and pressure gradient of fluid particles at the boundary.

3. Method

We first discuss the previous concepts of Akinci et al. [AIA*12] in Section 3.1 and Adami et al. [AHA12] in Section 3.2. Our approach is introduced in Section 3.3. This section focuses on the concepts. The combination of the boundary handling with DFSPH is described in Section 4.

3.1. Pressure mirroring

Pressure mirroring is motivated by its simple and efficient implementation. When a pressure acceleration is computed at a fluid particle f that has a boundary particle f_b with unknown pressure in its neighborhood, the pressure at the boundary particle is simply set to the pressure of the fluid particle, i.e. $p_{f_b} = p_f$. The computation in Eq. (1) slightly changes to

$$\boldsymbol{a}_f^{\mathrm{p}} = -\sum_{f_f} m_{f_f} \left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2} \right) \nabla W_{f f_f} - \sum_{f_b} \rho_f^0 V_{f_b} \left(\frac{p_f}{\rho_f^2} + \frac{p_f}{(\rho_f^0)^2} \right) \nabla W_{f f_b}. \tag{2}$$

Compared to Eq. (1), the mass of a boundary neighbor m_{f_b}, i.e. its contribution in the SPH sum, is computed as $m_{f_b} = \rho_f^0 V_{f_b}$ and the density of a boundary neighbor is set to $\rho_{f_b} = \rho_f^0$. Please refer to [AIA*12] for a motivation of these choices and for a discussion how to compute the volume V_{f_b}.

While density and mass in Eq. (2) play an important role for the accurate weighting of a boundary particle in the SPH sum, the employed pressure approximation $p_{f_b} = p_f$ negatively affects the accuracy of the pressure gradient computation, i.e. the computation of the pressure acceleration. As illustrated in Fig. 2, it would be more appropriate to extrapolate the pressure from the fluid to the boundary.

Pressure mirroring does not require to iterate over boundary particles or to store pressure values at boundary particles. If a boundary pressure is required, it is simply set to the pressure of the currently processed fluid particle. While this efficiency is positive, it results in inconsistent pressure values at boundary particles. If two fluid particles f^1 and f^2 with different pressure values p_{f^1} and p_{f^2} share the same boundary particle b, the gradient computations at both fluid particles work with different pressure values. Fluid particle f^1 works with $p_b = p_{f^1}$ in Eq. (2), while the other fluid particle f^2 uses $p_b = p_{f^2}$ at the same boundary particle b.

3.2. Pressure extrapolation with SPH

Pressure extrapolation can be motivated by Pascal's law for hydrostatic pressure which states that the pressure difference at two fluid points is proportional to their height difference. For a boundary particle b and an adjacent fluid particle b_f, this can be written as $p_b = p_{b_f} + \rho_{b_f} \boldsymbol{g} \cdot \boldsymbol{x}_{bb_f}$ with gravity $\boldsymbol{g}$ and distance $\boldsymbol{x}_{bb_f} = \boldsymbol{x}_b - \boldsymbol{x}_{b_f}$. In order to handle the interaction of one boundary particle with several neighboring fluid particles, the respective contributions are weighted with the kernel function W_{bb_f}, summed up and normalized as proposed by Adami et al. [AHA12]:

$$p_b = \frac{\sum_{b_f} p_{b_f} W_{bb_f} + \boldsymbol{g} \cdot \sum_{b_f} \rho_{b_f} \boldsymbol{x}_{bb_f} W_{bb_f}}{\sum_{b_f} W_{bb_f}}. \tag{3}$$

In contrast to the pressure mirroring in [AIA*12], this approach requires an additional loop over boundary particles to compute the pressure which is also stored at boundary particles. On the other hand, boundary pressures are not inconsistent as in [AIA*12]. Instead, each boundary particle is attributed a unique pressure value. If the pressure p_b is computed for all boundary particles b, Eq. (1) can be used to compute the pressure acceleration at fluid particles.

Although the computation in Eq. (3) is normalized, it nevertheless

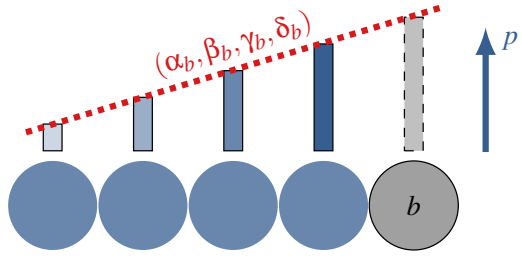


Figure 3: *MLS hyperplane fitting to compute pressure at a boundary particle b (grey). The plane parameters $\alpha_b, \beta_b, \gamma_b$ and δ_b are estimated from pressure values at adjacent fluid particles (blue).*

suffers from the typical SPH particle deficiency issue. The neighborhood of a boundary particle is only partially filled with fluid neighbors which falsifies the pressure computation.

3.3. Pressure Extrapolation with MLS

We propose to resolve the particle deficiency issue by using MLS instead of SPH for the pressure computation at boundary particles. Therefore, we fit hyperplanes that approximate the pressure field as illustrated in Fig. 3. This is conceptually different to [BGT17] where MLS is used to fit planes through boundary sample positions. In our case, MLS is performed from the perspective of a boundary particle b. It employs all pressure values p_{b_f} of fluid neighbors b_f adjacent to this boundary particle b to fit a hyperplane through the pressure field. The hyperplane parameters $\boldsymbol{c}_b = (\alpha_b, \beta_b, \gamma_b, \delta_b)^{\mathrm{T}}$ at boundary particle b are estimated by minimizing $\sum_{b_f} (\boldsymbol{b}_{b_f} \cdot \boldsymbol{c}_b - p_{b_f})^2 V_{b_f} W_{bb_f}$ with $\boldsymbol{b}_{b_f} = (1, x_{b_f}, y_{b_f}, z_{b_f})^{\mathrm{T}}$. Considering the fact that the partial derivatives of this term with respect to the hyperplane parameters should be zero at the minimum, we get the following system that has to be solved:

$$\sum_{b_f} \boldsymbol{b}_{b_f} \left(\boldsymbol{b}_{b_f} \cdot \boldsymbol{c}_b - p_{b_f} \right) V_{b_f} W_{bb_f} = \boldsymbol{0}. \tag{4}$$

In order to solve for the unknown hyperplane parameters $\boldsymbol{c}_b$, we propose to rewrite Eq. (4) as

$$\sum_{b_f} \left(\boldsymbol{b}_{b_f} \otimes \boldsymbol{b}_{b_f} \right) \boldsymbol{c}_b V_{b_f} W_{bb_f} = \sum_{b_f} \boldsymbol{b}_{b_f} p_{b_f} V_{b_f} W_{bb_f} \tag{5}$$

which corresponds to

$$\sum_{b_f} \begin{bmatrix} 1 & x_{b_f} & y_{b_f} & z_{b_f} \\ x_{b_f} & x_{b_f}^2 & x_{b_f} y_{b_f} & x_{b_f} z_{b_f} \\ y_{b_f} & x_{b_f} y_{b_f} & y_{b_f}^2 & y_{b_f} z_{b_f} \\ z_{b_f} & x_{b_f} z_{b_f} & y_{b_f} z_{b_f} & z_{b_f}^2 \end{bmatrix} \begin{bmatrix} \alpha_b \\ \beta_b \\ \gamma_b \\ \delta_b \end{bmatrix} V_{b_f} W_{bb_f} = \sum_{b_f} \begin{bmatrix} 1 \\ x_{b_f} \\ y_{b_f} \\ z_{b_f} \end{bmatrix} p_{b_f} V_{b_f} W_{bb_f}. \tag{6}$$

This 4×4 system can be rewritten such that α_b can be directly computed and the parameters β_b, γ_b, δ_b can be computed by solving a 3×3 system. We therefore perform a basis transform: all considered positions $\boldsymbol{x}_{b_f}$ and also $\boldsymbol{x}_b$ are translated to positions $\bar{\boldsymbol{x}}_{b_f}$ and $\bar{\boldsymbol{x}}_b$ by

$$\boldsymbol{d}_b = \frac{\sum_{b_f} \boldsymbol{x}_{b_f} V_{b_f} W_{bb_f}}{\sum_{b_f} V_{b_f} W_{bb_f}} \tag{7}$$

i.e.

$$\bar{\boldsymbol{x}}_{b_f} = \left(\bar{x}_{b_f}, \bar{y}_{b_f}, \bar{z}_{b_f} \right)^{\mathrm{T}} = \boldsymbol{x}_{b_f} - \boldsymbol{d}_b \tag{8}$$

$$\bar{\boldsymbol{x}}_b = \left(\bar{x}_b, \bar{y}_b, \bar{z}_b \right)^{\mathrm{T}} = \boldsymbol{x}_b - \boldsymbol{d}_b. \tag{9}$$

This basis transform, i.e. the translation of all incorporated particle positions by the same vector $\boldsymbol{d}_b$, does not affect the parameters of the hyperplane. Thus, instead of solving Eq. (6), we now consider the following system with the same solution:

$$\sum_{b_f} \begin{bmatrix} 1 & \bar{x}_{b_f} & \bar{y}_{b_f} & \bar{z}_{b_f} \\ \bar{x}_{b_f} & \bar{x}_{b_f}^2 & \bar{x}_{b_f} \bar{y}_{b_f} & \bar{x}_{b_f} \bar{z}_{b_f} \\ \bar{y}_{b_f} & \bar{x}_{b_f} \bar{y}_{b_f} & \bar{y}_{b_f}^2 & \bar{y}_{b_f} \bar{z}_{b_f} \\ \bar{z}_{b_f} & \bar{x}_{b_f} \bar{z}_{b_f} & \bar{y}_{b_f} \bar{z}_{b_f} & \bar{z}_{b_f}^2 \end{bmatrix} \begin{bmatrix} \alpha_b \\ \beta_b \\ \gamma_b \\ \delta_b \end{bmatrix} V_{b_f} W_{bb_f} = \sum_{b_f} \begin{bmatrix} 1 \\ \bar{x}_{b_f} \\ \bar{y}_{b_f} \\ \bar{z}_{b_f} \end{bmatrix} p_{b_f} V_{b_f} W_{bb_f}. \tag{10}$$

Please note that W_{bb_f} in Eq. (10) depends on the distance between $\boldsymbol{x}_b$ and $\boldsymbol{x}_{b_f}$. Now, the elements $\bar{x}_{b_f}$, $\bar{y}_{b_f}$ and $\bar{z}_{b_f}$ can be replaced by zero based on the following observation:

$$\begin{aligned} \sum_{b_f} \bar{\boldsymbol{x}}_{b_f} V_{b_f} W_{bb_f} &= \sum_{b_f} \left(\boldsymbol{x}_{b_f} - \boldsymbol{d}_b \right) V_{b_f} W_{bb_f} \\ &= \sum_{b_f} \boldsymbol{x}_{b_f} V_{b_f} W_{bb_f} - \boldsymbol{d}_b \sum_{b_f} V_{b_f} W_{bb_f} \\ &= \boldsymbol{d}_b \sum_{b_f} V_{b_f} W_{bb_f} - \boldsymbol{d}_b \sum_{b_f} V_{b_f} W_{bb_f} \\ &= \boldsymbol{0}. \end{aligned} \tag{11}$$

Therefore, $\sum_{b_f} \bar{x}_{b_f} \beta_b V_{b_f} W_{bb_f} = \beta_b \sum_{b_f} \bar{x}_{b_f} V_{b_f} W_{bb_f}$ in Eq. (10) can be written as $\sum_{b_f} 0 \cdot \beta_b V_{b_f} W_{bb_f}$. In the same way, the coefficients $\bar{y}_{b_f}$ and $\bar{z}_{b_f}$ in the matrix in Eq. (10) can be replaced by zeros, resulting in the following form:

$$\sum_{b_f} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \bar{x}_{b_f}^2 & \bar{x}_{b_f} \bar{y}_{b_f} & \bar{x}_{b_f} \bar{z}_{b_f} \\ 0 & \bar{x}_{b_f} \bar{y}_{b_f} & \bar{y}_{b_f}^2 & \bar{y}_{b_f} \bar{z}_{b_f} \\ 0 & \bar{x}_{b_f} \bar{z}_{b_f} & \bar{y}_{b_f} \bar{z}_{b_f} & \bar{z}_{b_f}^2 \end{bmatrix} \begin{bmatrix} \alpha_b \\ \beta_b \\ \gamma_b \\ \delta_b \end{bmatrix} V_{b_f} W_{bb_f} = \sum_{b_f} \begin{bmatrix} 1 \\ \bar{x}_{b_f} \\ \bar{y}_{b_f} \\ \bar{z}_{b_f} \end{bmatrix} p_{b_f} V_{b_f} W_{bb_f}. \tag{12}$$

Now, we can solve for α_b:

$$\alpha_b = \frac{\sum_{b_f} p_{b_f} V_{b_f} W_{bb_f}}{\sum_{b_f} V_{b_f} W_{bb_f}}. \tag{13}$$

The other hyperplane parameters β_b, γ_b and δ_b are obtained by solving

$$\begin{bmatrix} \beta_b \\ \gamma_b \\ \delta_b \end{bmatrix} = \left(\sum_{b_f} \begin{bmatrix} \bar{x}_{b_f}^2 & \bar{x}_{b_f}\bar{y}_{b_f} & \bar{x}_{b_f}\bar{z}_{b_f} \\ \bar{x}_{b_f}\bar{y}_{b_f} & \bar{y}_{b_f}^2 & \bar{y}_{b_f}\bar{z}_{b_f} \\ \bar{x}_{b_f}\bar{z}_{b_f} & \bar{y}_{b_f}\bar{z}_{b_f} & \bar{z}_{b_f}^2 \end{bmatrix} V_{b_f} W_{bb_f} \right)^{-1} \sum_{b_f} \begin{bmatrix} 1 \\ \bar{x}_{b_f} \\ \bar{y}_{b_f} \\ \bar{z}_{b_f} \end{bmatrix} p_{b_f} V_{b_f} W_{bb_f} . \tag{14}$$

Finally, the pressure p_b at boundary particle b is computed as

$$p_b = (1, \bar{x}_b, \bar{y}_b, \bar{z}_b)^{\mathrm{T}} \cdot \boldsymbol{c}_b . \tag{15}$$

If the matrix in Eq. (14) is not invertible, we set β_b, γ_b and δ_b to zero resulting in

$$p_b = (1, \bar{x}_b, \bar{y}_b, \bar{z}_b)^{\mathrm{T}} \cdot (\alpha_b, 0, 0, 0)^{\mathrm{T}} = \alpha_b = \frac{\sum_{b_f} p_{b_f} V_{b_f} W_{bb_f}}{\sum_{b_f} V_{b_f} W_{bb_f}} . \tag{16}$$

This pressure corresponds to the weighted average of the pressure values of fluid particles b_f adjacent to boundary particle b. In our experiments, however, we only experienced issues with a singular matrix in Eq. (14) for boundary particles with a single fluid neighbor and for boundary particles whose fluid neighbors lie exactly on a line or plane. In these cases, employing the proposed basis transform sets many coefficients of the matrix to zero.

Similar to the SPH pressure extrapolation, our approach performs a loop over boundary particles to compute and store pressure at boundary particles. Each boundary particle has a unique pressure value. In contrast to the SPH extrapolation, our MLS approach does not suffer from the particle deficiency issue.

4. Implementation

We have combined our proposed boundary handling with a slightly modified DFSPH solver [BK17] that is outlined in Algorithm 1. We follow the idea of combining two solvers, one for the velocity divergence and one for the density invariance. As a minor notation change to DFSPH, our two solvers compute pressure p instead of stiffness parameter κ. DFSPH implicitly introduces κ with the relation $\nabla p_f = \sum_{f_j} m_{f_j} \kappa_{f_j} \nabla W_{ff_j}$ with f_j denoting a fluid or a boundary neighbor of f. From the SPH formulation $\nabla p_f = \sum_{f_j} \frac{m_{f_j}}{\rho_{f_j}} p_{f_j} \nabla W_{ff_j}$, it follows that $p_{f_j} = \kappa_{f_j} \rho_{f_j}$. So, we use the DFSPH solver, but multiply all κ values with the density ρ to get pressure values.

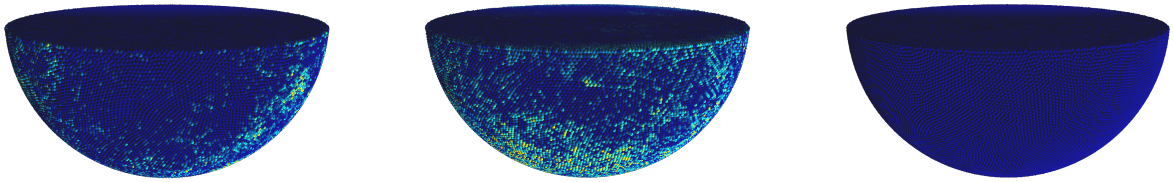
The functions CorrectDivergenceError and CorrectDensityError in Algorithm 1 compute pressure at fluid particles, but they are also responsible for the pressure computation at boundary particles. In case of pressure mirroring, the pressure is not explicitly computed, but just considered in the computation of $\boldsymbol{a}^*$ and $\boldsymbol{a}^{**}$ by using Eq. (2) instead of Eq. (1). The SPH extrapolation and our proposed MLS extrapolation loop over the boundary particles to compute and store pressure. Then, Eq. (1) is used to compute the pressure accelerations $\boldsymbol{a}^*$ and $\boldsymbol{a}^{**}$. Finally, we update the pressure values p_f of fluid particles f in a Jacobi step. In our experiments, we set the relaxation coefficient $\omega = 0.5$.

Algorithm 1 DFSPH with MLS pressure extrapolation

procedure PERFORM SIMULATION
- **for each** fluid particle f **do**
  - find neighborhoods $N_f(t)$
- **for each** fluid particle f **do**
  - compute density $\rho_f(t)$
  - compute factor $\alpha_f \leftarrow \frac{\left|\left|\sum_j m_j \nabla W_{fj}\right|\right|^2 + \sum_{f_f} m_f m_{f_f} \left|\left|\nabla W_{ff_f}\right|\right|^2}{-\rho_f(t)^2}$
- $\boldsymbol{a}^* \leftarrow$ CORRECT DIVERGENCE ERROR ▷ divergence-free
- **for each** fluid particle f **do**
  - predict velocity $\boldsymbol{v}_f^* \leftarrow \boldsymbol{v}_f(t) + \Delta t\, \boldsymbol{a}_f^*$
- **for each** fluid particle f **do**
  - predict velocity $\boldsymbol{v}_f^{**} \leftarrow \boldsymbol{v}_f^* + \Delta t\, \boldsymbol{a}_f^{\text{non-pressure}}$
- $\boldsymbol{a}^{**} \leftarrow$ CORRECT DENSITY ERROR ▷ density invariant
- **for each** fluid particle f **do**
  - update velocity $\boldsymbol{v}_f(t+\Delta t) \leftarrow \boldsymbol{v}_f^{**} + \Delta t\, \boldsymbol{a}_f^{**}$
  - update position $\boldsymbol{x}_f(t+\Delta t) \leftarrow \boldsymbol{x}_f(t) + \Delta t\, \boldsymbol{v}_f(t+\Delta t)$

procedure CORRECT DIVERGENCE ERROR
- **for each** fluid particle f **do**
  - compute source term $s_f \leftarrow -\frac{1}{\Delta t} \nabla \cdot \boldsymbol{v}_f(t)$
  - initialize pressure $p_f \leftarrow 0$
- **while** not converged **do**
  - **for each** boundary particle b **do**
    - compute pressure p_b using MLS ▷ Eq. (15)
  - **for each** fluid particle f **do**
    - compute pressure acceleration $\boldsymbol{a}_f^*$ ▷ Eq. (1)
  - **for each** fluid particle f **do**
    - set pressure $p_f \leftarrow p_f + \frac{\omega}{\alpha_f} \left(s_f - \nabla \cdot \boldsymbol{a}_f^* \right)$
- **return** $\boldsymbol{a}^*$

procedure CORRECT DENSITY ERROR
- **for each** fluid particle f **do**
  - compute source term $s_f \leftarrow \frac{1}{\Delta t^2} \frac{\rho_f(t) - \rho_f^0}{\rho_f(t)} - \frac{1}{\Delta t} \nabla \cdot \boldsymbol{v}_f^{**}$
  - initialize pressure $p_f \leftarrow 0$
- **while** not converged **do**
  - **for each** boundary particle b **do**
    - compute pressure p_b using MLS ▷ Eq. (15)
  - **for each** fluid particle f **do**
    - compute pressure acceleration $\boldsymbol{a}_f^{**}$ ▷ Eq. (1)
  - **for each** fluid particle f **do**
    - set pressure $p_f \leftarrow \max\left(0, p_f + \frac{\omega}{\alpha_f} \left(s_f - \nabla \cdot \boldsymbol{a}_f^{**} \right)\right)$
- **return** $\boldsymbol{a}^{**}$



(a) *Pressure mirroring* **(b)** *Pressure extrapolation with SPH* **(c)** *MLS pressure extrapolation*

Figure 4: *Comparison of the different boundary handling schemes. Velocities are color-coded with blue corresponding to minimal and red corresponding to maximal velocity. In contrast to pressure mirroring 4a and pressure extrapolation with SPH 4b, our approach 4c does not show an incorrect movement of the fluid particles.*

5. Results

In this section, we compare our proposed MLS pressure extrapolation to the pressure mirroring of Akinci et al. [AIA*12] and the pressure extrapolation with SPH of Adami et al. [AHA12]. We show that our approach can handle challenging scenarios, such as complex and fast-moving boundary geometries and high water depths. We use different particle spacings and time steps for the simulations. For the SPH interpolation, we use the cubic spline kernel [Mon05] with a support of two times the particle spacing. The rest density of the fluids is $1000\,\mathrm{kg\,m^{-3}}$ while the largest permissible compression error is 0.1 %. In our implementation, we employ compact hashing [IABT11] for the neighbor search. We apply a drag force to the fluid as described in [GBP*17] and model surface tension as proposed in [AAT13]. Viscosity is modeled as proposed by [MFZ97]. To reduce the loss in turbulent details, we use a micropolar material model [BKKW17]. All computations are fully parallelized with Intel Threading Building Blocks [Phe08]. We use [FIF18] to reconstruct the fluid surface. The ray-traced images are rendered with [Sid18]. All presented scenarios have been computed on a 12-core 2.6 GHz Intel Xeon E5-2690 with 32 GB of RAM.

5.1. Rotating Sphere

First, we compare our new approach to [AIA*12] and [AHA12] in a setting where fluid is placed inside a rigid sphere with free-slip boundary conditions as illustrated in Fig. 4. The boundary sphere has a radius of 3 m and is rotating slowly at 7 revolutions per minute. We use a particle spacing of 5 cm. The scenario consists of 417 k fluid and 44.5 k boundary particles and is simulated for ten seconds with a fixed time step of 1 ms. The number of density invariant iterations of our DFSPH solver was fixed to ten while the divergence-free iterations count was set to zero. This resulted in a density error of approximately 0.037 %.

In this experiment, the boundary particles should not influence the fluid velocities and the fluid should rest inside the sphere. However, as shown in Fig. 4, this is not the case for [AIA*12] and [AHA12]. Both boundary handling schemes introduce an artificial viscosity, which causes an incorrect movement of the fluid particles. In contrast to this, our MLS pressure extrapolation does not suffer from artificial viscosity at the boundary. The computation time for the pressure field is very similar for all approaches (our: 64.29 ms, [AIA*12]: 63.12 ms, [AHA12]: 63.44 ms).

5.2. Breaking Dam

In order to compare the solver iteration counts of our approach with [AIA*12] and [AHA12], we simulate a breaking dam scenario inside a cylindric-shaped domain of size $3\,\mathrm{m} \times 3\,\mathrm{m} \times 0.5\,\mathrm{m}$ with a particle spacing of 8 mm. Thus, making a total of 2.95 million fluid and 182.2 k boundary particles. The scenario is illustrated in Fig. 5. Furthermore, we use different fixed time step sizes. Table 1 summarizes the iteration measurements and computation times for a simulation over ten seconds.

In our experiment, our MLS pressure extrapolation approach always requires the minimum number of iterations per simulation step. For larger time step sizes, our approach outperforms [AHA12] due to more accurate pressure gradients at the boundary. Computing unique pressure values for boundary particles is not expensive. For [AHA12], we measured an average computation time for the boundary pressures of 1.12 ms per iteration. For our approach, as we have to iterate twice over fluid neighbors of the boundary particles, the computation time slightly increased to 1.87 ms.

5.3. Vase

As shown in [BGI*18], computing unique pressure values for boundary particles can be beneficial for scenarios with a high water depth. In order to show that our approach can also handle such challenging scenarios, we simulate a vase of height 10 m that is filled with water over a duration of eighteen seconds. The scene is illustrated in Fig. 6. The particle spacing is 2 cm and the adaptive time step [IAGT10] is 0.47 ms on average. The scene consists of up to 13.33 million fluid and 826 k boundary particles. The total computation time per simulation step is 3.70 s on average with MLS pressure extrapolation, 3.72 s for pressure mirroring and 3.84 s for SPH pressure extrapolation. The reduced computation time of our approach is the result of a reduced solver iteration count.

5.4. Teacup

In order to demonstrate the applicability of our approach to two-way coupled dynamic objects, we integrated the Bullet physics

	average per time step					
	iterations			computation time		
Δt	Mirroring	SPH extrapolation	MLS extrapolation	Mirroring	SPH extrapolation	MLS extrapolation
0.25 ms	4.0	4.0	4.0	116 ms	120 ms	123 ms
0.50 ms	5.7	5.7	5.6	186 ms	192 ms	192 ms
1.00 ms	12.0	12.2	11.8	448 ms	467 ms	462 ms
1.50 ms	14.9	16.0	14.9	559 ms	618 ms	587 ms

Table 1: *Comparison of the different boundary handling schemes using different time steps for the Breaking Dam scenario.*

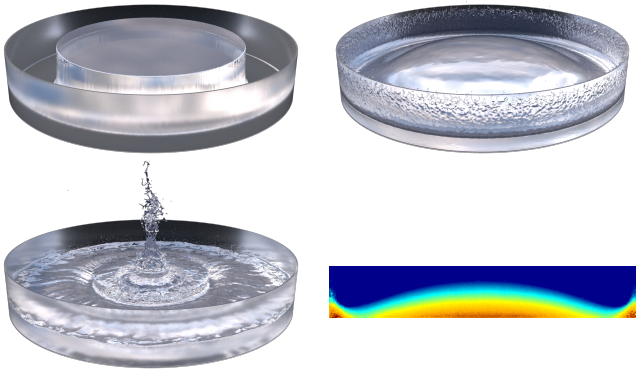


Figure 5: *Cylindrical breaking dam with* 2.95 *million fluid particles simulated with our MLS pressure extrapolation approach. The smooth color-coded pressure field on the bottom-right corresponds to the top-right frame.*

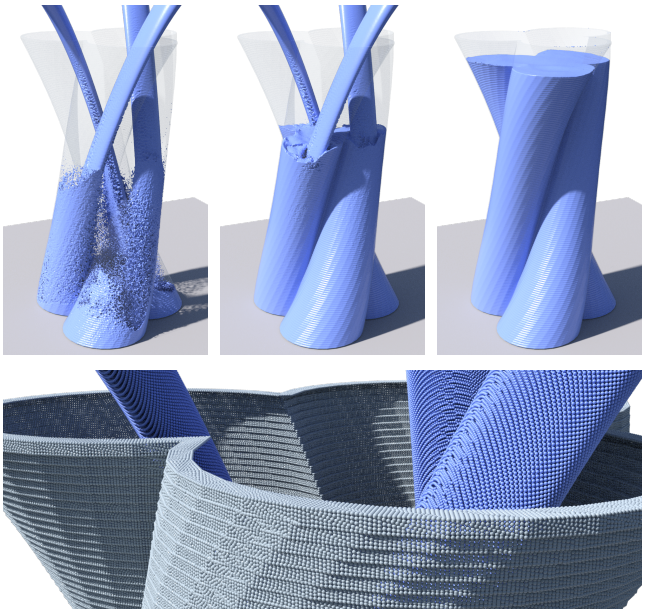


Figure 6: *Vase scenario with up to* 13.33 *million fluid particles. The bottom image shows a closeup, visualizing the complex boundary geometry and the particles.*

library [Cou18] in our simulation framework. Fig. 1 shows a teacup that contains a two-way coupled rigid rubber duck that has a rest density of 500 $\mathrm{kg\,m^{-3}}$. As fluid is filled into the cup, the rubber duck begins to rise. We have simulated the scene for ten seconds. It consists of up to 3.57 million fluid particles and 1.25 million boundary particles. The particle spacing is 3 mm. Our DFSPH solver requires a total of 13.22 iterations on average per simulation step with the adaptive time step being 0.28 ms on average. The total average computation time per simulation step is 1.112 s whereof computing the boundary pressures takes 34.33 ms.

5.5. Washing Machine

Our proposed method is particularly appropriate for fast-moving and complex boundaries. This is indicated in Fig. 7 where we simulate a washing machine that contains seven two-way coupled rigid spheres with different radii. The washing drum is animated and contains holes, i.e. the fluid drains. The particle spacing is 2 cm and the scene consists of up to 2.04 million fluid particles and 1.18 million boundary particles. We use an adaptive time step with an average of 0.5 ms. Overall, with our approach the average computation time is 409 ms per simulation step whereof computing the boundary pressures takes 17.6 ms. The average iteration count is 4.42. Due to instabilities at the fast-moving boundary, pressure mirroring requires a time step that is half as large compared to our MLS extrapolation. This results in a speed-up of factor 1.8 compared to Akinci et al. [AIA*12]. This speed-up factor is particularly remarkable considering the fact that [AIA*12] typically works for time steps that correspond to rather large CFL numbers.

6. Conclusion and Future Work

MLS pressure extrapolation at boundaries reduces artifacts at fluid-solid interfaces which can improve the performance of the pressure computation in iterative solvers. We have shown that pressure mirroring and SPH extrapolation suffer from artificial viscosity which is not the case for the proposed MLS extrapolation. We have also shown that the reduced velocity artifacts in our boundary handling can positively influence the pressure computation time for challenging scenarios. As one of the next steps, we plan to investigate properties of the MLS gradient estimation for other purposes, e.g. the computation of the pressure acceleration at fluid particles.

Figure 7: *Animated washing machine with two-way coupled rigid spheres.*

Acknowledgments

The vase model is courtesy of Eckerput at https://www.cgtrader.com and is licensed under Royalty Free License. The cup model is courtesy of nerosoft at https://www.cgtrader.com and is licensed under Royalty Free License. The saucer model is courtesy of trapdormi at https://www.cgtrader.com and is licensed under Royalty Free License. The model of the washing machine is courtesy of vikinger at https://www.cgtrader.com and is licensed under Royalty Free License. The rubber duck is courtesy of willie at www.thingiverse.com and is licensed under Creative Commons - Public Domain Dedication license.

References

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

[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 (2003), 3–15. 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. 3

[AHA12] ADAMI S., HU X. Y., ADAMS N. A.: A generalized wall boundary condition for smoothed particle hydrodynamics. *Journal of Computational Physics 231*, 21 (2012), 7057 – 7075. 2, 3, 6

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

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively Sampled Particle Fluids. *ACM Transactions on Graphics 26*, 3 (2007). 2

[ATO17] ALDUÁN I., TENA A., OTADUY M. A.: DYVERSO: A versatile multi-phase position-based fluids solution for VFX. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 32–44. 2

[BGFAO17] BARREIRO H., GARCÍA-FERNÁNDEZ I., ALDUÁN I., OTADUY M. A.: Conformation constraints for efficient viscoelastic fluid simulation. *ACM Transactions on Graphics 36*, 6 (2017), 221. 2

[BGI*18] BAND S., GISSLER C., IHMSEN M., CORNELIS J., PEER A., TESCHNER M.: Pressure Boundaries for Implicit Incompressible SPH. *ACM Transactions on Graphics 37*, 2 (2018), 14:1–14:11. 6

[BGT17] BAND S., GISSLER C., TESCHNER M.: Moving Least Squares Boundaries for SPH Fluids. In *Virtual Reality Interactions and Physical Simulations* (2017), Eurographics Association. 3, 4

[BK17] BENDER J., KOSCHIER D.: Divergence-Free SPH for Incompressible and Viscous Fluids. *IEEE Transactions on Visualization and Computer Graphics 23*, 3 (2017), 1193–1206. 1, 2, 5

[BKKW17] BENDER J., KOSCHIER D., KUGELSTADT T., WEILER M.: A Micropolar Material Model for Turbulent SPH Fluids. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2017), ACM, pp. 4:1–4:8. 6

[BRHN11] BILOTTA G., RUSSO G., HÉRAULT A., NEGRO C. D.: Moving least-squares corrections for smoothed particle hydrodynamics. *Annals of Geophysics 54*, 5 (2011). 3

[BT07] BECKER M., TESCHNER M.: Weakly Compressible SPH for Free Surface Flows. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), Eurographics Association, pp. 209–217. 2

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

[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 (2003), 448–475. 2

[Cou18] COUMANS, ERWIN: The bullet physics library. `www.bulletphysics.org`, 2018. 7

[CR99] CUMMINS S. J., RUDMAN M.: An SPH Projection Method. *Journal of Computational Physics 152*, 2 (1999), 584–607. 2

[DG96] DESBRUN M., GASCUEL M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation*. Springer, 1996, pp. 61–76. 2

[Dil99] DILTS G. A.: Moving-least-squares-particle hydrodynamics I: Consistency and stability. *International Journal for Numerical Methods in Engineering 44*, 8 (1999), 1115–1155. 3

[FIF18] FIFTY2 TECHNOLOGY: PreonLab. `www.fifty2.eu`, 2018. 6

[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. 3

[GBP*17] GISSLER C., BAND S., PEER A., IHMSEN M., TESCHNER M.: Generalized drag force for particle-based simulations. *Computers & Graphics 69* (2017). 6

[HA07] HU X. Y., ADAMS N. A.: An incompressible multi-phase sph method. *Journal of Computational Physics 227*, 1 (2007), 264–278. 2

[HEW15] HUBER M., EBERHARDT B., WEISKOPF D.: Boundary Handling at Cloth-Fluid Contact. *Computer Graphics Forum 34*, 1 (2015), 14–25. 3

[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. 6

[IAGT10] IHMSEN M., AKINCI N., GISSLER M., TESCHNER M.: Boundary Handling and Adaptive Time-stepping for PCISPH. In *Virtual Reality Interactions and Physical Simulations* (2010). 2, 6

[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

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH Fluids in Computer Graphics. In *Eurographics (State of the Art Reports)* (2014). 2

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

[KB17] KOSCHIER D., BENDER J.: Density Maps for Improved SPH Boundary Handling. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2017), ACM, pp. 1:1–1:10. 3

[KK56] KENNEY J., KEEPING E.: *Mathematics of Statistics Part I.* Van Nostrand, 1956. 2

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

[MFZ97] MORRIS J. P., FOX P. J., ZHU Y.: Modeling Low Reynolds Number Incompressible Flows Using SPH. *Journal of Computational Physics 136*, 1 (1997), 214–226. 6

[MK09] MONAGHAN J., KAJTAR J.: SPH particle boundary forces for arbitrary boundaries. *Computer Physics Communications 180*, 10 (2009), 1811 – 1820. 2

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M. H., ALEXA M.: Point Based Animation of Elastic, Plastic and Melting Objects. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), Eurographics Association, pp. 141–151. 3

[Mon94] MONAGHAN J. J.: Simulating Free Surface Flows with SPH. *Journal of Computational Physics 110*, 2 (1994), 399–406. 2

[Mon05] MONAGHAN J. J.: Smoothed Particle Hydrodynamics. *Reports on Progress in Physics 68*, 8 (2005), 1703. 2, 6

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

[MST*04] MÜLLER M., SCHIRM S., TESCHNER M., HEIDELBERGER B., GROSS M.: Interaction of Fluids with Deformable Solids: Research Articles. *Computer Animation and Virtual Worlds 15*, 3-4 (2004), 159–171. 2

[Nea04] NEALEN A.: An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation. 2

[OS03] OTT F., SCHNETTER E.: A modified SPH approach for fluids with large density differences. In *ArXiv Physics e-prints* (2003), p. 3112. 3

[PDC*03] PURCELL T. J., DONNER C., CAMMARANO M., JENSEN H. W., HANRAHAN P.: Photon Mapping on Programmable Graphics Hardware. In *ACM SIGGRAPH/Eurographics Conference on Graphics Hardware* (2003), Eurographics Association, pp. 41–50. 2

[PGBT18] PEER A., GISSLER C., BAND S., TESCHNER M.: An Implicit SPH Formulation for Incompressible Linearly Elastic Solids. *Computer Graphics Forum* (2018). 2

[Phe08] PHEATT C.: Intel® Threading Building Blocks. *Journal of Computing Sciences in Colleges 23*, 4 (2008), 298–298. 6

[PICT15] PEER A., IHMSEN M., CORNELIS J., TESCHNER M.: An implicit viscosity formulation for SPH fluids. *ACM Transactions on Graphics 34*, 4 (2015), 114. 2

[PT17] PEER A., TESCHNER M.: Prescribed Velocity Gradients for Highly Viscous SPH Fluids with Vorticity Diffusion. *IEEE Transactions on Visualization and Computer Graphics 23*, 12 (2017), 2656–2662. 2

[RLY*14] REN B., LI C., YAN X., LIN M. C., BONET J., HU S.-M.: Multiple-fluid sph simulation using a mixture model. *ACM Transactions on Graphics 33*, 5 (2014), 171:1–171:11. 2

[SB12] SCHECHTER H., BRIDSON R.: Ghost SPH for Animating Water. *ACM Transactions on Graphics 31*, 4 (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

[Sid18] SIDE EFFECTS SOFTWARE: Houdini. www.sidefx.com, 2018. 6

[SL03] SHAO S., LO E. Y.: Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Advances in Water Resources 26*, 7 (2003), 787 – 800. 2

[SP08] SOLENTHALER B., PAJAROLA R.: Density Contrast SPH Interfaces. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), pp. 211–218. 2, 3

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

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

[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

[WKBB18] WEILER M., KOSCHIER D., BRAND M., BENDER J.: A Physically Consistent Implicit Viscosity Solver for SPH Fluids. *Computer Graphics Forum* (2018). 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