Implicit Frictional Boundary Handling for SPH

Jan Bender, Tassilo Kugelstadt, Marcel Weiler, Dan Koschier

Abstract—In this paper, we present a novel method for the robust handling of static and dynamic rigid boundaries in Smoothed Particle Hydrodynamics (SPH) simulations. We build upon the ideas of the density maps approach which has been introduced recently by Koschier and Bender. They precompute the density contributions of solid boundaries and store them on a spatial grid which can be efficiently queried during runtime. This alleviates the problems of commonly used boundary particles, like bumpy surfaces and inaccurate pressure forces near boundaries. Our method is based on a similar concept but we precompute the volume contribution of the boundary geometry. This maintains all benefits of density maps but offers a variety of advantages which are demonstrated in several experiments. Firstly, in contrast to the density maps method we can compute derivatives in the standard SPH manner by differentiating the kernel function. This results in smooth pressure forces, even for lower map resolutions, such that precomputation times and memory requirements are reduced by more than two orders of magnitude compared to density maps. Furthermore, this directly fits into the SPH concept so that volume maps can be seamlessly combined with existing SPH methods. Finally, the kernel function is not baked into the map such that the same volume map can be used with different kernels. This is especially useful when we want to incorporate common surface tension or viscosity methods that use different kernels than the fluid simulation.

Index Terms—Smoothed Particle Hydrodynamics, fluid simulation, boundary handling

1 Introduction

Smoothed Particle Hydrodynamics (SPH) has become a powerful tool in computer graphics, where it is used for visual effects in movies, and in interactive applications like games or virtual training simulators. It can be used to simulate lots of different materials ranging from incompressible fluids and highly viscous liquids to granular materials and deformable solids.

Using particles as the primary material representation is appealing because it simplifies implementations. Moreover, it provides a straightforward way for two-way coupling of many different materials and for two-way coupling with rigid bodies. However, the accurate application of boundary conditions remains challenging, but is especially important when materials interact with complex static and dynamic boundaries as depicted in Fig. 1. In recent years several boundary representations based on particles, meshes or implicit representations, like signed distance fields, have been proposed. But most of them are either inaccurate or have high computational demands. For instance, sampling the boundary with particles results in bumpy surfaces which lead to inaccurate forces and introduce artificial friction or even jumping particle artifacts (cf. [1], [2]). A detailed discussion of previous methods and the associated problems can be found in Section 2.

Recently, Koschier and Bender [1] proposed an implicit boundary representation called *density maps*. They use a spatial grid to discretize the density contribution function of static and dynamic rigid obstacles which is computed using adaptive Gauss-Kronrod quadrature in a preprocessing step. To reduce the memory footprint, the density maps are only stored for a narrow band around the solid surfaces and approximated with higher-order polynomials. During runtime they can be efficiently queried and accurate forces can be computed without the aforementioned problems of particle based approaches and without significant computational overhead. Moreover, friction between particles and solids, even in complex scenarios, can be handled.

Our method is inspired by the density maps approach and is also based on an implicit boundary representation. We use the same data structures as described in [1] to store and query function values. However, instead of precomputing the density contribution of the boundary, we determine the boundary volume that overlaps with the support domain of the smoothing kernel. This has multiple advantages which are demonstrated in a series of experiments. When using the density maps approach, the derivatives of density values are determined by differentiating the shape functions of the cubic elements. However, this is prone to errors because smoothness at cell interfaces is not guaranteed [3]. Our experiments show that this is not problematic for high resolution grids but it results in visual artifacts for lower resolutions due to force discontinuities. Compared to that, we compute all derivatives by differentiating the SPH kernel which is the standard SPH formulation. This results in smooth forces even for low resolution volume maps which enables significantly faster precomputations and reduces the memory requirements drastically. Another advantage is that it directly fits into the SPH concept such that volume maps can be seamlessly integrated into existing solvers and combined easily with nearly all existing SPH methods. Finally, in contrast to the density maps approach, volume maps are independent of the smoothing kernel. Hence, a single instance of a volume map can be used in conjunction with various kernel functions. This is especially advantageous as some methods employ individual smoothing kernels for specialized purposes, e.g. surface tension or viscosity ap-

- *Jan Bender - RWTH Aachen University*
  E-mail: bender@cs.rwth-aachen.de
- *Tassilo Kugelstadt - RWTH Aachen University*
  E-mail: kugelstadt@cs.rwth-aachen.de
- *Marcel Weiler - RWTH Aachen University*
  E-mail: weiler@cs.rwth-aachen.de
- *Dan Koschier - Unaffiliated*
  E-mail: dan.koschier@gmail.com

The definitive version of record is available at http://dx.doi.org/10.1109/TVCG.2020.3004245

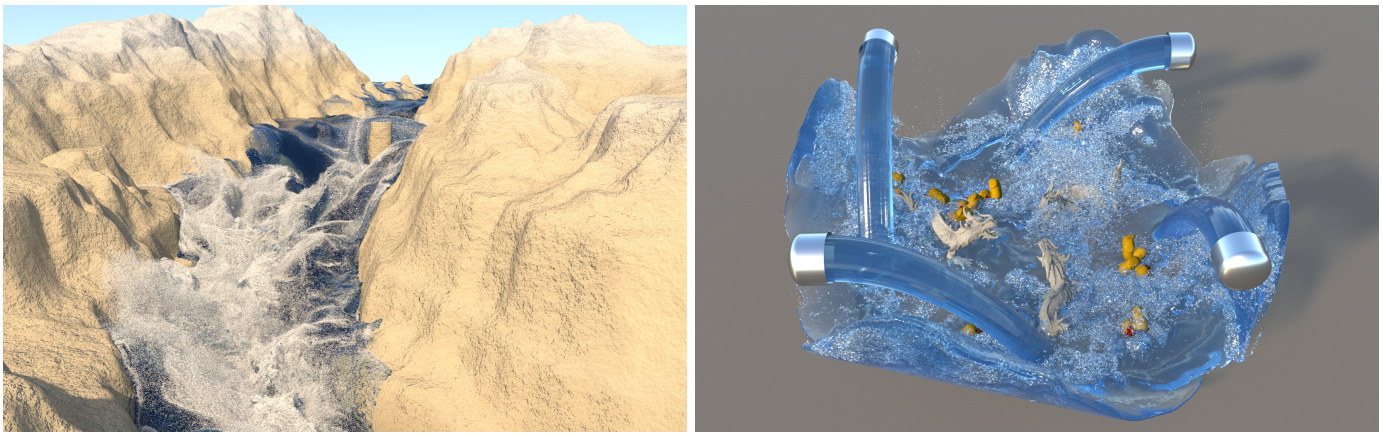


Fig. 1. Our novel implicit boundary representation based on volume maps is able to handle scenarios with complex static and dynamic boundaries. Left: 8 million turbulent fluid particles interact with a large-scale canyon boundary. Right: Four emitters generate 12 million fluid particles with high velocities that interact with static dragons and dynamic ducks.

proaches, where multiple instances of the kernel-dependent density maps would be required.

This paper is an extended version of our previous work on volume maps [4]. In addition to the original paper we introduce a method to simulate sticky boundaries and tangential friction using volume maps. Moreover, we added a section describing the volume map construction in more detail. Finally, we performed further experiments and improved the derivation of the volume maps formulation.

2 Related Work

In recent decades there has been a large variety of research on SPH methods. They have become a popular tool for simulating incompressible fluids, highly viscous liquids, and deformable solids. In this section, we briefly discuss prior works that are related to the presented method. A more general overview of recent developments in SPH can be found in the state-of-the-art report by Ihmsen et al. [5] and in the course notes of Koschier et al. [6].

SPH was introduced to the computer graphics community by Stam and Fiume [7] to simulate fire and gaseous phenomena. Later, it was also used to simulate deformable solids [8] and fluids [9]. To achieve nearly incompressible fluids, Becker and Teschner [10] used an equation of state based approach. By choosing an appropriate stiffness constant for the pressure forces they can guarantee a small maximal compression. To alleviate the small time step requirements and stability issues of explicitly integrated pressure forces, many implicit solvers have been developed. One way of enforcing incompressibility is to apply a constant density constraint as proposed by Solenthaler and Pajarola [11] who solve it with a predictive-corrective scheme. Similar approaches based on position based and projective dynamics have been proposed by Macklin and Müller [12] and Weiler et al. [13], respectively. Ihmsen et al. [14] showed that constant density can also be enforced by solving a discrete pressure Poisson equation. Another way to guarantee incompressibility is to solve a Poisson equation to make the velocity field divergence-free [15]. This has been done in a hybrid SPH and grid based approach by Raveendran et al. [16]. Bender and Koschier [17] proposed to solve for both, a divergence-free velocity field and a constant density, which increases stability and computational performance. In our experiments we use this pressure solver although the proposed boundary handling method can be combined with any of the aforementioned solvers.

Due to the particle nature of SPH it is not straightforward to enforce boundary conditions at interfaces between fluids and solid surfaces. One problem is that particles which are located close to the boundary typically suffer from neighborhood particle deficiencies leading to inaccurate density estimates and pressure forces. Another problem is that implicit pressure solvers also require pressure values inside the boundary domain which are unknown. These problems are orthogonal to each other and several methods to solve them have been proposed in recent years. Most approaches use specialized boundary representations, including particles, triangle meshes, and implicit representations such as signed distance fields.

A popular approach is to sample solid boundaries with particles which exert penalty forces onto nearby fluid particles to avoid penetrations [10], [18]. Instead of integrating these forces with explicit schemes, Becker et al. [19] proposed a predictor-corrector scheme. However, their method suffers from particle stacking artifacts near the boundary. Alternatively, the boundary particles can be treated as fluid particles such that they contribute to the density and pressure computations [20]. This solves the particle deficiency problem and results in smooth density distributions near solid boundaries. However, small time steps are necessary to guarantee non-penetration. Ihmsen et al. [21] combine the approaches of Becker et al. [19] and Solenthaler et al. [20] to get smooth particle distributions without requiring small time steps. Their method was further improved by Akinci et al. [22] such that two-way coupling of fluids with rigid bodies can be achieved. Moreover, they introduced normalized pseudo masses to account for non-uniform boundary particle distributions. He et al. [23] proposed to add auxiliary staggered particles to discretize a Poisson equation with appropriate boundary conditions for projecting the velocity field onto a divergence-free state. For two-way coupling of fluids and deformable solids several adaptive particle

sampling approaches have been proposed [24], [25], [26].

One problem of the particle based boundary representations is that the surfaces are typically irregular. This causes non-smooth surface normals and therefore non-penetration forces that are not completely orthogonal to the surface which leads to undesired artificial friction or even jumping particle artifacts [1]. Band et al. [2] proposed a method to alleviate this problem. They use the moving least squares (MLS) method to fit local planes to the boundary particles. This results in correct normals and eliminates most artifacts. However, it can be only applied in planar surface regions. Another disadvantage of boundary particles is that they introduce a substantial computational overhead. They need to be considered during neighborhood searches and for each fluid particle it is required to iterate over all neighboring boundary particles during force and pressure computations.

Alternatively, solid boundaries can be represented as triangle meshes. Bodin et al. [27] use unilateral constraints to prevent the fluid particles from penetrating triangulated surfaces. The two-way coupling of fluids with cloth was addressed by Huber et al. [28]. They use continuous collision detection and apply correction impulses such that the fluid particles cannot penetrate the cloth. However, neither of the approaches addresses the problem of inaccurate density estimates close to the boundary. Fujisawa and Miura [29] also use meshes as a boundary representation and adopt a solution for the density problem from the engineering community [30]. In order to correct the SPH approximations, they compute a renormalization factor which takes the overlapping volume of the SPH kernel and the boundary into account. However, they compute these factors at runtime which results in significantly higher computation times compared to the particle based approaches.

Solid boundaries can be alternatively represented with implicit functions, e.g. signed distance fields (SDFs) as proposed by Harada et al. [31], [32]. To virtually extend the fluid density into the solid they sample the density contribution of a planar surface depending on the distance of a prototype fluid particle. The resulting values are discretely sampled prior to the simulation and are then queried during runtime based on the signed distance value of the fluid particle. However, this is only correct for planar surfaces and results in incorrect density values for curved geometries. Koschier and Bender [1] proposed to precompute the density contributions of rigid boundaries and store them in a so called density map which can be efficiently queried during runtime. This results in accurate density values and avoids the artifacts of bumpy particle samplings. Since our work is inspired by the density maps approach, it will be discussed in more detail in Section 4.2.

A popular approach to solve the problem of required but unknown pressure values at the boundary is a technique called pressure mirroring (cf. [22]). While processing a fluid particle that requires to determine the pressure value at a boundary sample, pressure mirroring assumes that this value is equal to the fluid's pressure value. This is easy to implement and computationally cheap, but has the problem that the pressure values at the boundary are not unique. As shown by Band et al. [33] this reduces the stability and convergence of the pressure solver. Therefore, they suggested to include the boundary particles into the pressure projection step. In a follow-up work [34], they point out that it is even more beneficial to compute unique boundary pressures by extrapolating the values from the fluid particles using MLS. In our work we focus on removing artifacts induced by the irregularity of the typically particle-sampled surface and to enhance accuracy in the evaluation of discrete field samples and their derivatives in close proximity to the boundary. However, we would like to stress the fact that the discussed boundary pressure computation approaches are orthogonal to the proposed method.

3 Foundations

In this section we briefly introduce the governing equations to simulate incompressible fluids. Moreover, we derive the standard SPH discretization which is then extended in the next section by boundary handling.

Governing Equations: In our work we simulate fluids by using the Navier-Stokes equations for incompressible flow in Lagrangian coordinates

$$\frac{D\rho}{Dt} = 0 \quad \Leftrightarrow \quad \nabla \cdot \mathbf{v} = 0 \tag{1}$$

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{v} + \frac{\mathbf{f}}{\rho}, \tag{2}$$

where $\rho, t, \mathbf{v}, p, \nu$ and $\mathbf{f}$ denote density, time, velocity, pressure, kinematic viscosity and body forces, respectively. To solve the Navier-Stokes equations numerically a discretization is required.

Spatial Discretization: In the following we introduce the SPH formalism which is a common approach for spatial discretization. To derive the standard SPH discretization for a function $g(\mathbf{x})$, we first rewrite it using the Dirac-δ identity (cf. [6])

$$g(\mathbf{x}) = \int_\Omega g(\mathbf{x}')\ \delta(\mathbf{x} - \mathbf{x}')d\mathbf{x}', \tag{3}$$

where Ω defines the problem domain. Then, we approximate the integral by replacing δ with a kernel function W with compact support and finally approximate the resulting integral using a sum over a set of sampling points $\mathbf{x}_j$

$$g(\mathbf{x}) \approx \int_{\mathcal{N}(\mathbf{x})} g(\mathbf{x}')\ W(\|\mathbf{x} - \mathbf{x}'\|, h)d\mathbf{x}' \tag{4}$$

$$\approx \sum_j V_j g(\mathbf{x}_j) W(\|\mathbf{x} - \mathbf{x}_j\|, h), \tag{5}$$

where $\mathcal{N}(\mathbf{x})$ defines the support domain of the kernel function (see Fig. 2), h is the smoothing length of the kernel and V_j is the volume represented by a sampling point.

In an SPH simulation the fluid is discretized by particles. Each particle i represents a sampling point and its volume is computed as $\frac{m_i}{\rho_i}$. The mass m_i is determined by the size of the particle and the rest density ρ_0 of the fluid. The density ρ_i of particle i is computed using the SPH formalism

$$\rho_i = \sum_j m_j W_{ij}, \tag{6}$$

where $W_{ij} = W(\|\mathbf{x}_i - \mathbf{x}_j\|, h)$.

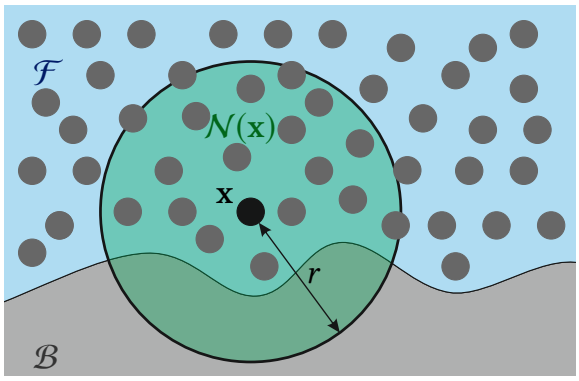


Fig. 2. Definition of the fluid domain $\mathcal{F}$ (blue), the boundary domain $\mathcal{B}$ (gray) and the support domain $\mathcal{N}(\mathbf{x})$ of the compact kernel function (green) with radius r at position $\mathbf{x}$. The fluid domain is discretized by particles.

4 Boundary Handling

Most graphics-related techniques to handle boundary conditions in the context of SPH solvers are based on the concept of virtually extending field quantities into the boundary domain, e.g. density, pressure, velocity etc. [6]. Moreover, the vast majority of proposed methods consider incompressible continua and, therefore, ensure that the incompressibility constraint (see Eq. (1)) is satisfied. By extending the density field into the boundary domain, the typically employed pressure solvers implicitly resolve boundary penetrations as the virtual density contributions lead to local compressions that violate the incompressibility constraint. Recent work has demonstrated the generality and versatility of the concept (cf. [35]). Building on this concept, we will assume that (at least) the mass density ρ is extended into the boundary such that $\rho > 0$ on $\mathcal{B}$.

Given the (extended) density field, a smoothed density sample $\rho(\mathbf{x})$ located in close proximity to the boundary domain is determined as follows. The density function is plugged into Eq. (4). We can then partition the integral into fluid and boundary portion (see Fig. 2)

$$\begin{aligned}\rho(\mathbf{x}) &\approx \int_{\mathcal{N}(\mathbf{x})} \rho(\mathbf{x}')\, W(\|\mathbf{x}-\mathbf{x}'\|, h) d\mathbf{x}' \\ &= \int_{\mathcal{N}(\mathbf{x})\cap\mathcal{F}} \rho(\mathbf{x}')\, W(\|\mathbf{x}-\mathbf{x}'\|, h) d\mathbf{x}' + \\ &\quad \int_{\mathcal{N}(\mathbf{x})\cap\mathcal{B}} \rho(\mathbf{x}')\, W(\|\mathbf{x}-\mathbf{x}'\|, h) d\mathbf{x}' \\ &= \rho_{\mathcal{F}}(\mathbf{x}) + \rho_{\mathcal{B}}(\mathbf{x}).\end{aligned} \tag{7}$$

While the fluid integral $\rho_{\mathcal{F}}$ is typically discretized into SPH particles and numerically approximated using Eq. (6), there exist various approaches to numerically approximate the boundary portion $\rho_{\mathcal{B}}$. The most popular approach is based on particle sampling. Recently, an alternative concept, referred to as density maps [1], has been proposed, where samples of $\rho_{\mathcal{B}}$ are computed using adaptive numerical quadrature to discretize the function onto a spatial grid. In the following, we first briefly describe the core concepts of particle-based approaches and density maps and then introduce our novel volume maps approach. We further discuss the differences between all approaches and review the advantages of our novel method in the next section.

4.1 Particle-Based Approaches

The core idea of particle-based approaches is to discretize the boundary geometry with particles [18], [21], [22]. These boundary particles typically have the same size as the fluid particles and serve as additional sampling points. In this way the boundary portion $\rho_{\mathcal{B}}(\mathbf{x})$ can be discretized analogously to the fluid portion using a sum (see Eq. (6))

$$\rho_i = \rho_{\mathcal{F}}(\mathbf{x}_i) + \rho_{\mathcal{B}}(\mathbf{x}_i) \approx \sum_j m_j W_{ij} + \sum_k \tilde{m}_k W_{ik}, \tag{8}$$

where j and k denote the indices of particle i's fluid and boundary particle neighbors, respectively. Note that the boundary masses $\tilde{m}_k$ are independent of the boundary object's material properties and solely fulfill the purpose of extending the fluid's mass density field into the boundary. The values $\tilde{m}_k$ are therefore often called *pseudo masses*.

One of the most popular approaches for particle-based boundary handling was proposed by Akinci et al. [22]. In this approach the pseudo masses are chosen such that $\rho_{\mathcal{B}} \approx \rho_0$ inside the boundary domain. They compute the pseudo mass of a boundary particle k as $\tilde{m}_k = \frac{\rho_0}{\sum_l W_{kl}}$, where l denotes the indices of the neighboring boundary particles of k, to account for non-uniform boundary particle distributions.

For a more detailed discussion of particle-based approaches we would like to refer the reader to the work of Koschier et al. [6].

4.2 Density Maps

Instead of sampling the boundary with particles one could also solve the integral over the boundary part in Eq. (7) by numerical integration methods such as Gauss quadrature while setting the density in the boundary to ρ_0. However, the problem with this approach is twofold. First, the numerical integration is expensive and significantly decreases the performance of the simulation. Second, we have to integrate a discontinuous, piecewise constant function as the density field is zero outside of the boundary and takes on ρ_0 inside the boundary. This leads to staircase artifacts in the resulting function due to the limited accuracy of fixed pattern quadrature schemes (see Fig. 3).

The density maps concept [1] solves both problems. The boundary density function $\rho_{\mathcal{B}}$ is discretized and precomputed on a spatial grid to solve the performance problem. The discontinuities are removed by extending the boundary density to the outside of the boundary to get a smooth function. Since our novel boundary handling approach is inspired by the density maps method, we will discuss this concept here in more detail.

In contrast to particle-based approaches, the density maps concept is based on an implicit representation of the boundary. Building on a signed distance function $\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}$ whose zero iso-surface describes the boundary geometry, the authors extend the density field into the boundary using the extension function $\gamma : \mathbb{R} \rightarrow \mathbb{R}^+$. They further convolve the density extension function with the kernel in order to ensure compliance with the SPH formalism (cf. Eq. (7)), i.e.

$$\rho_{\mathcal{B}}(\mathbf{x}) = \int_{\mathcal{N}(\mathbf{x})} \gamma(\Phi(\mathbf{x}'))\, W(\|\mathbf{x}-\mathbf{x}'\|, h) d\mathbf{x}', \tag{9}$$

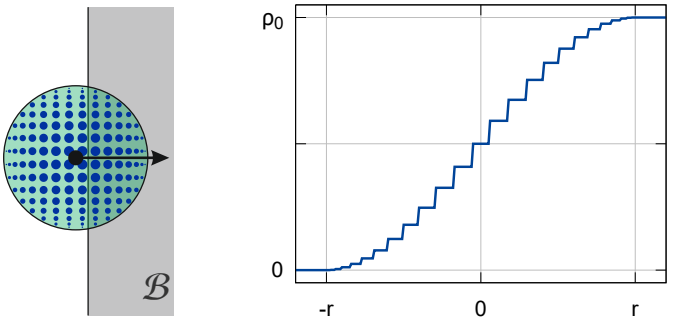


Fig. 3. Left: As the black fluid particle with support radius r enters the boundary, the boundary part of the density $\rho_{\mathcal{B}}$ increases. Right: Computing $\rho_{\mathcal{B}}$ by numerical integration leads to a discontinuity each time a sampling point (blue) enters the boundary.

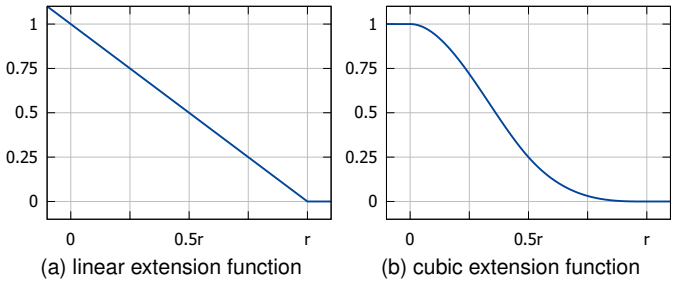


Fig. 4. (a) The linear extension function γ (Eq. (11)) used for density maps is not smooth at $x = r$. (b) In contrast our cubic function γ^* (Eq. (13)) is continuously differentiable. Note that the plot in (a) shows the normalized function γ/ρ_0.

where the signed distance function is defined as

$$\Phi(\mathbf{x}) = \begin{cases} -\inf_{\tilde{\mathbf{x}} \in \partial\mathcal{B}} \|\mathbf{x} - \tilde{\mathbf{x}}\| & \text{if } \mathbf{x} \in \mathcal{B} \\ \inf_{\tilde{\mathbf{x}} \in \partial\mathcal{B}} \|\mathbf{x} - \tilde{\mathbf{x}}\| & \text{otherwise.} \end{cases} \tag{10}$$

The extension function is modeled as a linear polynomial that takes the rest density value on the boundary surface

$$\gamma(x) = \begin{cases} \rho_0 \left(1 - \frac{x}{r}\right) & \text{if } x < r \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

This means we have a linear function increasing from 0 to ρ_0 within the support radius r. In this way the extension function has no discontinuity within the support radius. In order to evaluate the boundary density function $\rho_{\mathcal{B}}$ at a certain sample position $\mathbf{x}$, the integral is numerically approximated using adaptive Gauss-Kronrod quadrature. As this procedure is computationally expensive for arbitrarily complex boundary shapes and therefore not bearable at runtime, $\rho_{\mathcal{B}}$ is discretized on a regular grid with cubic shape functions of Serendipity type. The discretized function can then be efficiently accessed at runtime such that boundary density values and gradients can be computed in $\mathcal{O}(1)$.

4.3 Volume Maps

Our novel volume maps approach is inspired by the density maps concept and is also based on an implicit boundary representation. Moreover, we also define an extension function to avoid discontinuities and use a spatial grid to improve performance. However, instead of computing the density part of the boundary directly, we determine the intersection volume of the sphere around a fluid particle that is defined by the support radius r and the boundary (see Fig. 2). This offers several advantages as shown in the following.

The most important advantage is that the boundary volume can be directly used in the general SPH formulation (see Eq. (5)) which enables a consistent computation of the density gradient. Volume maps only determine the intersection volume between kernel support domain and boundary and use the SPH formulation to compute the density gradient. In contrast, density maps determine the gradient by differentiating the cubic shape functions of their spatial grid discretization. The latter way to compute the gradient has a significant drawback. It is not guaranteed that a density map is smooth over the interfaces of neighboring grid cells [3]. This leads to 'kinks' in the discretized representation of the density function when using a coarse resolution. At the position of such a kink the described computation of the density gradient leads to pressure forces that are not consistently oriented and therefore to noticeable visual artifacts (see Section 6). This can be improved by using a high-resolution density map which is computationally expensive to generate and has significantly higher memory requirements. In our method we use the map only to determine the boundary volume. The gradient is computed with the classical SPH formulation. Therefore, our method yields smooth pressure forces that do not cause visual artifacts even for low resolution maps which is shown in Section 6.

In order to generate a volume map, we determine the boundary volume as

$$V_{\mathcal{B}}(\mathbf{x}) = \int_{\mathcal{N}(\mathbf{x})} \gamma^*(\Phi(\mathbf{x}'))d\mathbf{x}', \tag{12}$$

where Φ is the signed distance function (see Eq. (10)). Note that this formulation is similar to the integral for the boundary density in Eq. (9). However, the computation of the volume does not require a convolution with the kernel function. Therefore, in contrast to density maps, volume maps are independent of the kernel and a single map can be used in conjunction with various kernel functions. Moreover, in our formulation we use a cubic extension function instead of a linear one (cf. Eq. (11))

$$\gamma^*(x) = \begin{cases} \frac{C(x)}{C(0)} & \text{if } 0 < x < r \\ 1 & \text{if } x \leq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

where C denotes the cubic spline kernel [36]. Note that this kernel is only used to define the cubic function γ^* while an arbitrary kernel can be used in the SPH formulation. Using a cubic extension function has the advantage that in contrast to a linear one, we get a smooth transition at distance $x = r$ (see Fig. 4). Moreover, the fact that our cubic extension function is continuously differentiable significantly improves the Gauss quadrature approximation of the integral in Eq. (12) in comparison to using the non-smooth linear function [37]. This means we can reach the same approximation accuracy with a lower number of sample points which reduces the precomputation time. Finally, the integral in Eq. (12) is numerically solved using Gauss-Legendre quadrature and the

volume function is spatially discretized on a grid structure. For further details see Section 5.

To compute the density of a particle we use the SPH formulation in Eq. (5). Here a quantity at a position $\mathbf{x}$ is determined by a sum over a set of sampling points in the neighborhood of $\mathbf{x}$. Each sampling point represents a volume and its quantity is weighted by a kernel function depending on the distance between the sampling point and $\mathbf{x}$. To determine the boundary portion of the density $\rho_{\mathcal{B}}$ using our volume maps approach, we add a single sampling point $\mathbf{x}^*$ on the boundary which represents the boundary volume $V_{\mathcal{B}}$. Moreover, we extend the density field to the boundary and assume that the density in the boundary volume $V_{\mathcal{B}}$ is ρ_0. Substituting the sampling point and the density in the integral for the boundary density from Eq. (7) allows us to solve the integral:

$$\begin{aligned}\rho_{\mathcal{B}}(\mathbf{x}) \approx& \int_{\mathcal{N}(\mathbf{x})\cap\mathcal{B}} \rho_0\, W(\|\mathbf{x}-\mathbf{x}^*\|,h)d\mathbf{x}' \\ =&\rho_0\, W(\|\mathbf{x}-\mathbf{x}^*\|,h)\int_{\mathcal{N}(\mathbf{x})\cap\mathcal{B}} d\mathbf{x}' \\ =&\rho_0\, W(\|\mathbf{x}-\mathbf{x}^*\|,h)V_{\mathcal{B}}(\mathbf{x}).\end{aligned} \tag{14}$$

In our work we define $\mathbf{x}^*$ as the closest point to $\mathbf{x}$ on the boundary. This point can be easily determined by the signed distance function Φ that we also use to generate the volume map.

The volume map is generated in a precomputation step before the simulation. To compute the density contribution of the boundary during runtime, in each simulation step we have to determine the closest point $\mathbf{x}^*$ on the surface of the boundary and the boundary volume $V_{\mathcal{B}}$ for each fluid particle $\mathbf{x}$ that is close to boundary. Note that both quantities can be obtained by simple lookups in the map which require constant time. The resulting density contribution can be easily used in any pressure solver. The required density gradient can be determined analogously by taking the gradient of the kernel. Finally, the symmetric pressure force [36] for a particle i is computed as

$$\begin{aligned}\mathbf{F}_i^p =& -m_i\sum_j m_j\left(\frac{p_i}{\rho_i^2}+\frac{p_j}{\rho_j^2}\right)\nabla W(\|\mathbf{x}-\mathbf{x}_j\|,h) \\ &- m_i V_{\mathcal{B}}\rho_0\left(\frac{p_i}{\rho_i^2}+\frac{\tilde{p}}{\tilde{\rho}^2}\right)\nabla W(\|\mathbf{x}-\mathbf{x}^*\|,h) \\ =&\mathbf{F}^p_{i\leftarrow\mathcal{F}}+\mathbf{F}^p_{i\leftarrow\mathcal{B}},\end{aligned} \tag{15}$$

where $\mathbf{F}^p_{i\leftarrow\mathcal{F}}$ is the part of the pressure force that is exerted by the neighboring fluid particles and $\mathbf{F}^p_{i\leftarrow\mathcal{B}}$ is the part that is exerted by the boundary. $\tilde{p}$ and $\tilde{\rho}$ are the pressure and the density at the boundary, respectively. In our work we used pressure mirroring [22] and therefore set $\tilde{p}=p_i$ and $\tilde{\rho}=\rho_0$. However, we plan to compute the boundary pressure value by extrapolation as proposed by Band et al. [34] in the future. Their pressure extrapolation can be combined easily with our approach by setting $\tilde{p}$ to the extrapolated value.

In case of dynamic boundaries we apply the negative force $\mathbf{F}^p_{\mathcal{B}\leftarrow i}=-\mathbf{F}^p_{i\leftarrow\mathcal{B}}$ at the position $\mathbf{x}^*$ to the boundary object in order to realize two-way coupling. This two-way coupling is shown in the simulations in Fig. 1 and 12.

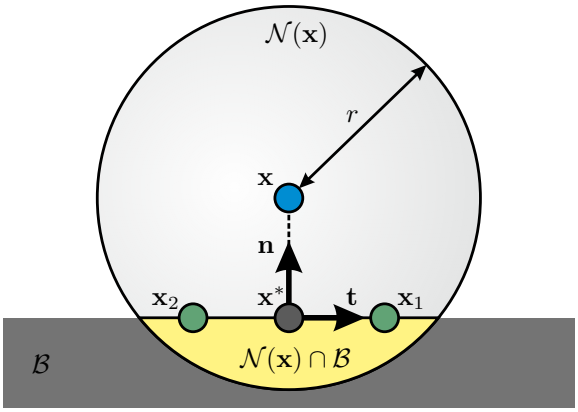


Fig. 5. In a simulation with volume maps, the computation of boundary viscosity requires the definition of additional points $\mathbf{x}_1$ and $\mathbf{x}_2$ (green) on the boundary surface. When a particle $\mathbf{x}$ (blue) is closer to the boundary than the support radius r, the volume maps approach determines the closest point on surface $\mathbf{x}^*$ (gray), the surface normal $\mathbf{n}$ and the volume of the intersection $\mathcal{N}(\mathbf{x})\cap\mathcal{B}$ (yellow) of the support domain $\mathcal{N}$ and the boundary domain $\mathcal{B}$. By computing a tangent vector $\mathbf{t}$ which is perpendicular to $\mathbf{n}$ two additional surface points are generated.

4.4 Boundary Viscosity

In the following we describe how volume maps can be combined with a method for the simulation of boundary viscosity in order to simulate sticky boundaries and tangential friction.

Sticky Boundaries: In our work we use the method of Weiler et al. [38] to simulate highly viscous materials. Weiler et al. introduce an implicit method to compute the viscosity force $\mathbf{f}_{\text{visco}}=\mu\nabla^2\mathbf{v}$ which is defined by the Navier-Stokes equations. The Laplacian of the velocity field is approximated by combining an SPH derivative with a finite difference derivative [36]:

$$\nabla^2\mathbf{v}_i = 2(d+2)\sum_j V_j\frac{\mathbf{v}_{ij}\cdot\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2+0.01h^2}\nabla W_{ij}, \tag{16}$$

where $\mathbf{x}_{ij}=\mathbf{x}_i-\mathbf{x}_j$, $\mathbf{v}_{ij}=\mathbf{v}_i-\mathbf{v}_j$ and d is the number of spatial dimensions. The sum in Eq. (16) considers all neighboring fluid particles j. Highly viscous materials like honey tend to stick to the boundary. To simulate this effect Weiler et al. propose to consider also all neighboring boundary particles in the sum of Eq. (16).

However, when using volume maps this method cannot directly be applied. The only point we have on the boundary surface is the closest point $\mathbf{x}^*$ (cf. Section 4.3). The vector from this point to $\mathbf{x}_i$ points exactly in the direction of the surface normal (see Fig. 5). Hence, if we solely use this boundary point, the term $\mathbf{v}_{ij}\cdot\mathbf{x}_{ij}$ in Eq. (16) would only represent the velocity difference in normal direction while the velocity in tangential direction is neglected. To solve this problem, we temporarily add additional surface points to evaluate the Laplacian of the velocity field.

To generate additional points on the boundary surface we first determine two orthogonal vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ in the tangential plane perpendicular to the normal vector $\mathbf{n}$. Then we define four points in this plane by $\mathbf{x}_{1/2}=\mathbf{x}^*\pm d\mathbf{t}_1$ and $\mathbf{x}_{3/4}=\mathbf{x}^*\pm d\mathbf{t}_2$. In our work we use a distance of $d=0.5r$ to the point $\mathbf{x}^*$. When reducing the distance factor, the points get closer to $\mathbf{x}^*$ so that the tangential components

of the velocity have less influence. For larger factors the points are far away from $\mathbf{x}$ so that they only lie within the support radius when the fluid particle is very close to the surface. In several experiments the value $d = 0.5r$ gave us the best results. Fig. 5 shows the two-dimensional case with one tangent vector $\mathbf{t}$ and two additional points $\mathbf{x}_1$ and $\mathbf{x}_2$.

In our simulation we evaluate Eq. (16) only for the four additional surface points. Since the four points represent the volume $V_{\mathcal{B}}$ of the intersection region $\mathcal{N}(\mathbf{x}) \cap \mathcal{B}$, we set $V_j = 0.25\, V_{\mathcal{B}}$. Finally, we need the velocity of each point which is computed as

$$\mathbf{v}_j = \mathbf{v}_{\text{rb}} + \boldsymbol{\omega}_{\text{rb}} \times (\mathbf{x}_j - \mathbf{x}_{\text{rb}}), \tag{17}$$

where $\mathbf{x}_{\text{rb}}$, $\mathbf{v}_{\text{rb}}$ and $\boldsymbol{\omega}_{\text{rb}}$ are the center of mass, the velocity and the angular velocity of the corresponding rigid body, respectively.

Tangential Friction: When simulating deformable solids, the bodies typically do not stick to the boundaries. However, tangential friction occurs between the solids and the boundaries which can be simulated in a similar way.

For each solid particle i which is closer to the boundary than its support radius we first determine additional surface points $\mathbf{x}_j$ as described above. Since we are only interested in the relative tangential motion of the bodies, we then compute the relative tangential velocities of a solid particle i and its corresponding additional boundary surface points

$$\mathbf{v}_{ij}^t = \mathbf{v}_{ij} - (\mathbf{v}_{ij} \cdot \mathbf{n}_i)\mathbf{n}_i, \tag{18}$$

where $\mathbf{n}_i$ is the surface normal at the closest point to $\mathbf{x}_i$ and $\mathbf{v}_j$ is determined by Eq. (17). Finally, we compute the Laplacian of the velocity field using the relative tangential velocity:

$$\nabla^2 \mathbf{v}_i = 2(d+2) \sum_j \frac{m_j}{\rho_j} \frac{\mathbf{v}_{ij}^t \cdot \mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2} \nabla W_{ij}. \tag{19}$$

5 Volume Map Discretization

As previously explained, every evaluation of the boundary volume function $V_{\mathcal{B}}$ requires one to numerically solve an integral over the cubically extended volume contribution as defined in Eq. (12). Since our goal is to support arbitrarily complex boundary surface shapes, carrying out a sufficiently accurate approximation of $V_{\mathcal{B}}$ is computationally involved. Fortunately, under the assumption that the boundary geometry solely undergoes rigid transformations, we can discretize the function prior to the simulation and efficiently evaluate boundary volume samples using the discrete field at runtime.

Generally, we are not restricted to a specific type of discretization. However, given that we expect that $V_{\mathcal{B}}$ is sufficiently smooth and given that the cubic extension introduced in Eq. (13) is smooth and non-linear, it is sensible to employ a piecewise polynomial discretization in order to capture the function with sufficient accuracy. Therefore, we suggest to use cubic polynomials on a regular hexahedral grid. An obvious choice for the cubic elements would be to use isoparametric tensor-product polynomials leading to 64 node elements which in turn require 64 evaluations of $V_{\mathcal{B}}$ for construction. In order to minimize both the number of function evaluations and storage requirements, we decided to use cubic elements of Serendipity type instead, reducing the number of element nodes to 32 which is in line with the method used by Koschier and Bender [1]. The Serendipity elements still guarantee C^0 continuity over element boundaries. The loss in accuracy compared to tensor-product elements is furthermore small, as the Serendipity element solely dismisses basis functions of mixed higher-order. However, those mixed higher-order terms add typically little to the overall approximation quality. To further reduce the memory requirements, grid cells sufficiently far away from and sufficiently deep inside the boundary object can be discarded resulting in a sparse grid which only stores cells in a narrow band around the surface. Finally, the volume map construction is carried out in three steps:

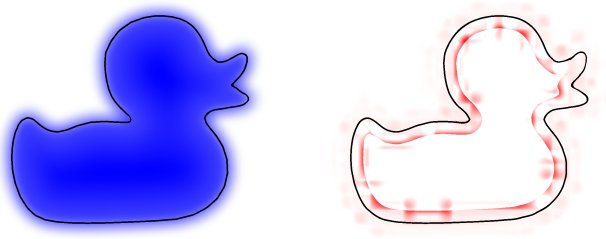
1) Signed distance field discretization
2) Volume map discretization
3) Enforcing grid sparsity

In the first step, a discretization domain is determined by computing the axis-aligned bounding box of the boundary surface dilated by (at least) the SPH kernel support radius. Presuming a user-specified grid resolution, the signed distance to the surface is then sampled at every node of the cubic Serendipity elements in the grid, e.g. using the method of Bærentzen and Aanæs [39]. This preliminary discretization allows us to efficiently compute signed distances in the next steps at arbitrary points within the discretized domain. Additionally, we can reuse the discretized signed distance field for collision detection which is required when simulating interacting rigid bodies.

In the second step, we sample the boundary volume function $V_{\mathcal{B}}$ at every element node. As previously discussed, this requires to numerically solve the integral in Eq. (12). For all of our results we have used a Gauss-Legendre quadrature rule of order 30 which was enough to yield sufficiently accurate results for all of our boundary geometries. Generally, using more elaborate quadrature techniques such as adaptive quadrature might be beneficial if even more complex boundary shapes are employed or if higher accuracy is required. Fig. 6a shows the volume map of a duck model which was generated with the method described above. The map has a resolution of $20 \times 20 \times 20$.

In the third and final step, we discard grid cells which are located sufficiently far away from the boundary or deep inside the boundary object in order to reduce the memory requirements. The volume integral in Eq. (12) using the cubic extension function defined in Eq. (13) is zero for all $\mathbf{x}$ with a distance of $\Phi(\mathbf{x}) > 2r$. Moreover, we assume that a particle never penetrates the boundary more than the same distance. Therefore, we discard the grid cells, where all points in the cell fulfill $|\Phi(\mathbf{x})| > 2r$. For the dragon models in Fig. 1 and 12 we used a map resolution of $60 \times 40 \times 40$. Storing the signed distance field and the volume map as dense grid requires 35.7 MB while the sparse grid only requires 6.4 MB which reduces the memory consumption by a factor of 5.5.

The discretization described above enables us to approximate the boundary volume $V_{\mathcal{B}}$ efficiently. While the volume maps approach provides a quite accurate solution at the grid nodes, it interpolates the solution in the interior of a grid cell by cubic shape functions. To measure the



(a) Volume map (b) Error of a volume map

Fig. 6. (a) A 2D slice of the duck volume map used in the simulations in Fig. 1 and 12. The size of the duck is $1.3\,m \times 1.3\,m \times 1.5\,m$. The volume is color-coded from white to blue. We used a coarse map resolution of $20 \times 20 \times 20$. The object surface is represented by the black contour. (b) For each pixel in this plot the volume is computed once using our volume maps approach and once by solving the corresponding volume integral accurately. The figure shows the deviation of our volume maps approach from the accurate solution. White means no error and red is the maximum volume error which was $0.000012\,m^3$ in this comparison.

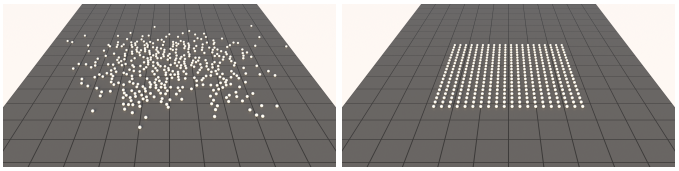
accuracy of this approximation we compared the volume maps solution to an accurate solution of the volume integral. The accurate solution was computed by solving Eq. (12) at every point (not only at the grid nodes) using a Gauss-Legendre quadrature rule of order 100. The difference of both solutions is shown in Fig. 6b, where the error is color-coded from white (no error) to red. The maximum volume error in this experiment was $0.000012\,m^3$ which occurs in the dark red regions. The small error value shows that our volume maps approach gives a good approximation of the boundary volume even for low map resolutions.

Finally, we performed an experiment where a particle with radius 0.01 m was moved in small steps towards a planar boundary until it touches the boundary. In each step we compared the exact boundary intersection of its support domain and the boundary volume determined by our approach. The average volume error was $0.000016\,m^3$ for a low-resolution map ($10 \times 10 \times 10$). This small error is caused by the map discretization, which we use to significantly improve the performance, and the extension function, which is required to avoid discontinuities.

6 Results and Discussion

In this section, we present results and compare our volume maps concept to the particle-based method of Akinci et al. [22] and the density maps approach [1]. For the comparison we implemented all methods in the open-source SPH library SPlisHSPlasH [40]. All simulations were performed using the implicit pressure solver DFSPH with an adaptive time-stepping scheme based on the CFL condition [17]. In fluid simulations we further employed a micropolar model to simulate vorticity [41]. Dynamic rigid bodies are simulated using a position-based method [42], [43]. The performance was measured on two Intel Xeon E5-2697 processors with 2.3GHz.

Comparisons: To compare our method to the particle-based approach, we performed several experiments. In the first simulation we dropped a regular grid of fluid particles on an inclined plane (see Fig. 7). The space between the particles is larger than the support radius so that the particles do not influence each other. The particle-based sampling of the boundary causes a chaotic movement of the particles (see Fig. 7a) while the simulation with our volume maps approach shows the expected smooth motion (see Fig. 7b).



(a) Akinci et al. (b) Volume Maps

Fig. 7. A regular grid of fluid particles is dropped on an inclined plane. (a) The particle-based sampling causes a chaotic motion. (b) Our approach shows the expected smooth motion.

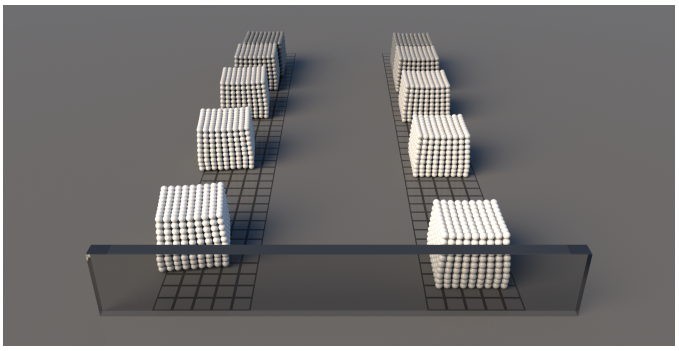


Fig. 8. Sequence of five steps in two simulations of a sliding cube on a frictionless inclined plane comparing the method of Akinci et al. (left) and our method (right). While our method shows the expected behavior (right), the cube simulated using the particle-based approach of Akinci et al. (left) slightly drifts to the left, starts to rotate and gets slower due to artificial friction.

In the second experiment we simulated the motion of a deformable cube on a frictionless inclined plane (see Fig. 8). The elastic behavior is simulated using the implicit SPH formulation for deformable solids of Peer et al. [44]. The results show that the non-smooth surface of the particle-based sampling leads to an undesired drift, artificial friction and a rotational motion. Since the volume maps are better suited to represent smooth surfaces, we did not experience any artifacts when using our novel method.

Finally, we simulated another deformable cube that rotates without friction for 100 seconds on a plane (see Fig. 9). Again the particle-based sampling caused an undesired drift and artificial damping. The cubes started with an angular velocity of 5.1 rad/s. At the end of the simulation the cube which was simulated with the boundary handling of Akinci et al. lost almost 90% of its angular velocity and the cube's center shifted substantially. Using our novel method the loss of angular velocity was less than 1% and no drift occurred.

To compare our method to the density maps approach of Koschier et al., we computed the pressure force for a single fluid particle which is moved along the normal direction to a planar boundary. The plots in Fig. 10 show the normal component of the resulting force (the other components are zero) for both methods. For the density map construction we solved the integral in Eq. (9) for each grid point which is close to the boundary using a high-resolution Gauss-Legendre quadrature with 17.5k sampling points. Since the

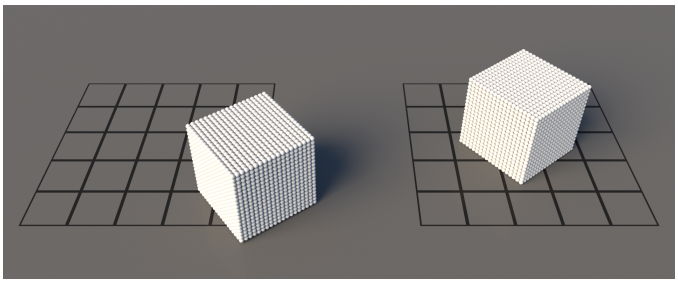


Fig. 9. Simulation of a deformable cube that rotates on a plane without friction. Left: In the simulation with the method of Akinci et al. the cube looses about 90% of its angular velocity and it drifts to the right. Right: Using our method no undesired drift is noticeable and the angular velocity is almost maintained.

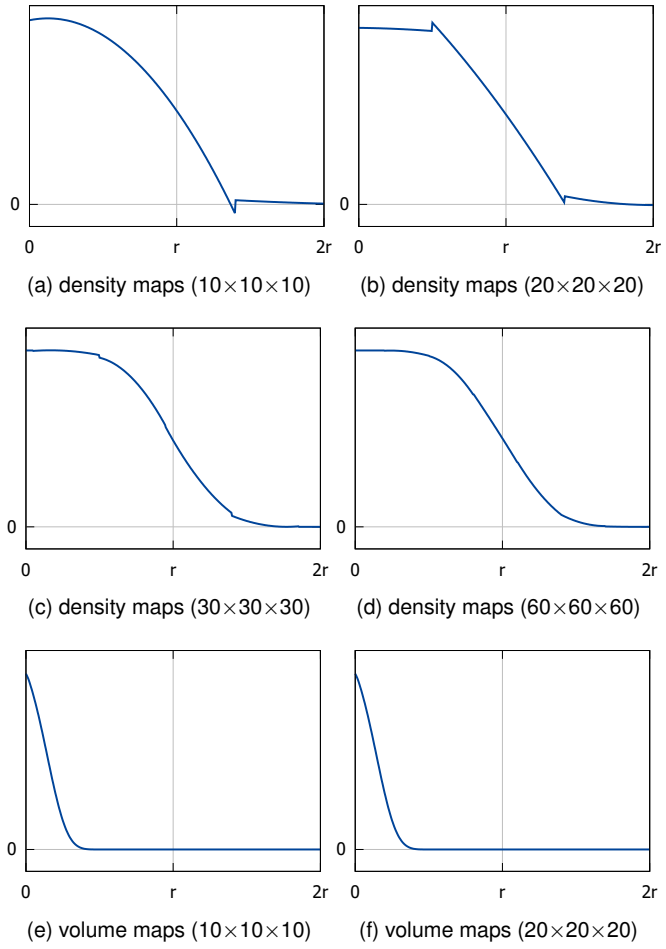


Fig. 10. Pressure force. A single fluid particle is moved in normal direction towards a boundary plane. The plots show the pressure force in normal direction (y-axis) for different map resolutions in relation to the distance to the boundary (x-axis). When using density maps, the pressure force increases slowly and is not smooth for low resolution maps (a,b,c,d). The force increases faster when using volume maps and is smooth even for low resolutions (e,f).

extension function of density maps is not continuously differentiable, this high resolution of sampling points is required to get a sufficient approximation when using Gauss quadrature [37]. Fig. 10 shows that low resolution density maps lead to discontinuities in the pressure force which in turn cause visual artifacts in a simulation as demonstrated in Fig. 11. The volume map construction was also performed by solving an integral (see Eq. (12)) using Gauss-Legendre quadrature. However, due to our smooth cubic extension function 4k sampling points were already sufficient. As shown in Fig. 10 the resulting force was continuous even for low resolution maps and no visual artifacts occurred during the dam break simulation in Fig. 11.

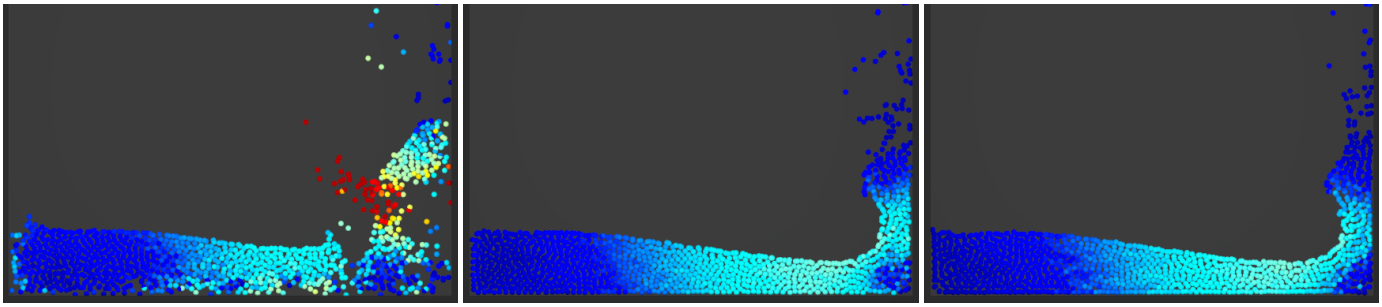
We believe that the kinks in the density maps pressure forces (see Fig. 10) are caused by small discontinuities between neighboring cells of the spatial grid (cf. [3]). Since the density maps approach uses the shape functions of the spatial discretization to compute the density gradient, these discontinuities have a large influence on the resulting pressure force and cause noticeable artifacts (see Fig. 11). In contrast our volume maps method only determines the volume using a map while the gradient is computed using the standard SPH formulation. Therefore, the influence on the pressure force is only small and neither noticeable in the graph (see Fig. 10) nor in the simulation (see Fig. 11).

Finally, we compared the construction times and the memory requirements of the volume and density maps. We compared the 3D maps of the unit cube that were used to generate the graphs in Fig. 10. At the same resolution the requirements are comparable. However, the density map required a resolution of $60 \times 60 \times 60$ while a volume map only needed a resolution of $10 \times 10 \times 10$ to obtain a smooth result. We observed the same resolution requirements in our 2D dam break simulation. The corresponding density map required 78.2 MB while the volume map only needed 392 KB due to its lower resolution. Hence, the memory consumption of the volume map was 200 times lower. The density map construction took 175 s while the generation of the volume map only needed 0.3 s. This yields a speedup factor of about 580. Naturally the speedup and memory requirements depend on the geometry that is discretized. Considering a complex geometry with small features, the discrete map naturally requires a higher resolution. However, the density map still relies on a far higher resolution to avoid the stability issues. At this point we would like to mention that the current implementation uses a regular grid to store the map. In the future, we plan to improve our implementation by using an adaptive grid which will further reduce the memory requirements.

Complex Boundaries: We performed two simulations to show that our novel method is able to handle scenarios with complex static and dynamic boundaries. First, we simulated a breaking dam in a canyon (see Fig. 1, left). To capture the detailed canyon surface we used a map resolution of $128 \times 64 \times 768$ in this scenario. We want to emphasize that this high resolution map is not required to guarantee a stable simulation. A realistic fluid-air interaction is simulated using the method of Gissler et al. [45]. The simulation shows that our method is able to handle 8 million turbulent fluid particles that interact with a large-scale complex boundary.

In the second simulation 12M fluid particles, which were generated by four emitters, interact with three static dragons and ten dynamic ducks (see Fig. 1, right). This demonstrates that our method robustly handles coupling between a turbulent fluid and dynamic bodies.

Combination with Existing SPH Methods: Our novel boundary handling method can be easily combined with



(a) Density Maps 10×10 (b) Density Maps 60×60 (c) Volume Maps 10×10

Fig. 11. Comparison of the density and volume maps using a 2D dam break simulation. (a) Low resolution density maps cause visual artifacts. (b) We can alleviate this problem by using a high-resolution density map. (c) Volume maps avoid these artifacts by a consistent gradient computation and do not suffer from visual artifacts even when using a low-resolution map.

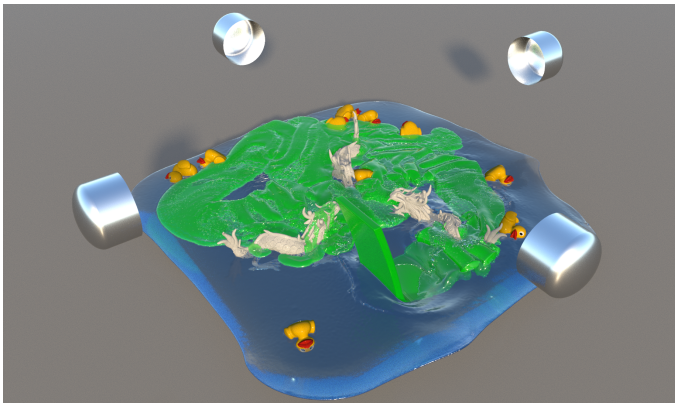


Fig. 12. Multiphase simulation where 5 million fluid particles interact with 1.3 million particles of a highly viscous material and multiple dynamic rigid bodies. The simulation demonstrates that our volume maps boundary handling can be easily combined with an implicit viscosity solver and an SPH multiphase method.

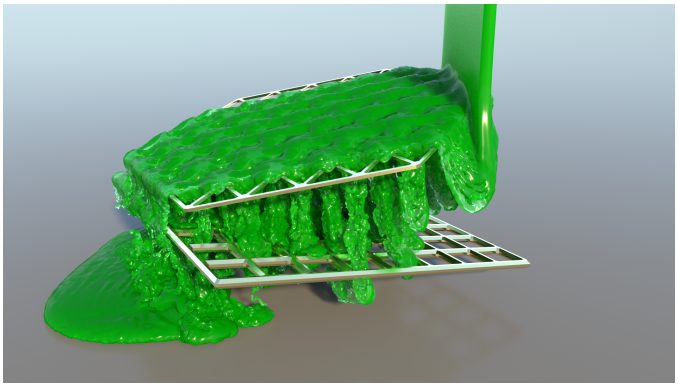


Fig. 13. A viscous fluid is poured over two grids and sticks to them. This example demonstrates that our extension enables the simulation of viscous friction between fluids and solids. It also shows that even fine structures like the grid can be reliably captured using volume maps.

other existing SPH methods. To demonstrate this we created a scenario consisting of several emitters, static dragons and dynamic ducks (see Fig. 12). This time we combined our method with the multiphase approach of Solenthaler and Pajarola [46] and the implicit viscosity solver of Weiler et al. [38] to simulate the interaction of water, highly viscous material and dynamic and static boundaries. In this experiment the highly viscous material sticks to the boundary. This effect was simulated using the extension introduced in Section 4.4. We used 5 million particles for the water phase and 1.3 million particles for the highly viscous material. The simulation shows that our novel boundary handling method can be seamlessly combined in other existing SPH methods.

Boundary Viscosity: In Section 4.4 we introduced extensions of the volume maps approach to simulate sticky boundaries and tangential friction. We performed three experiments to test both extensions in complex simulations.

In simulations with highly viscous materials, such as honey, that typically stick to the boundary, we use the first extension to simulate the sticking effect. Fig. 12 shows a highly viscous material that interacts with dynamic rigid bodies and a static boundary. In this simulation we set the fluid viscosity coefficient to $\mu = 1000\,\frac{kg}{m \cdot s}$ and the boundary viscosity coefficient to $\mu_{\mathcal{B}} = 10\,\frac{kg}{m \cdot s}$. The accompanying video shows that the viscous material tends to stick to the boundary which is the effect of our proposed extension.

In a second experiment we demonstrate the sticking effect without further interaction forces (see Fig. 13). A viscous fluid was poured over two grids and adhered to the boundary. Even though the grids were relatively thin, their interaction with the fluid was accurately simulated with our method. A map resolution of $100 \times 10 \times 100$ was sufficient to capture their boundary volumes. The fluid viscosity coefficient was set to $\mu = 500\,\frac{kg}{m \cdot s}$ while the boundary viscosity coefficient was $\mu_{\mathcal{B}} = 10000\,\frac{kg}{m \cdot s}$ to get a highly sticky behavior.

In the last boundary viscosity experiment we tested the tangential friction method introduced in Section 4.4. Six deformable cubes were simulated with different friction parameters (see Fig. 14). Again we used the method of Peer et al. [44] to simulate the elastic behavior. At the beginning all boxes had the same initial velocity of $5\,\frac{m}{s}$ and tangential friction was turned off. After two seconds of simulation we added tangential friction by setting the friction coefficients of the six cubes to values ranging from $\mu_{\mathcal{B}} = 0.0125\,\frac{kg}{m \cdot s}$ to $\mu_{\mathcal{B}} = 0.4\,\frac{kg}{m \cdot s}$ (cf. Fig. 14).

Performance: In our last experiment we simulated a double dam break consisting of 3 million fluid particles (see Fig. 15). This scenario was simulated once using the boundary han-

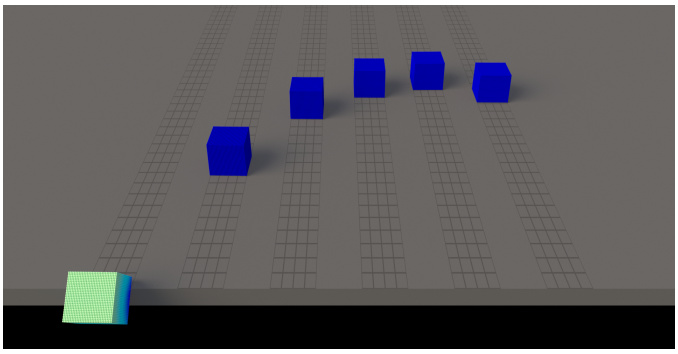


Fig. 14. Six deformable cubes with different friction coefficients slide on a plane. The coefficients $\mu_{\mathcal{B}}$ were set to (from left to right): $0.0125\ \frac{kg}{m \cdot s}$, $0.025\ \frac{kg}{m \cdot s}$, $0.05\ \frac{kg}{m \cdot s}$, $0.1\ \frac{kg}{m \cdot s}$, $0.2\ \frac{kg}{m \cdot s}$, and $0.4\ \frac{kg}{m \cdot s}$. Note that our tangential friction model only affects the tangential velocity.

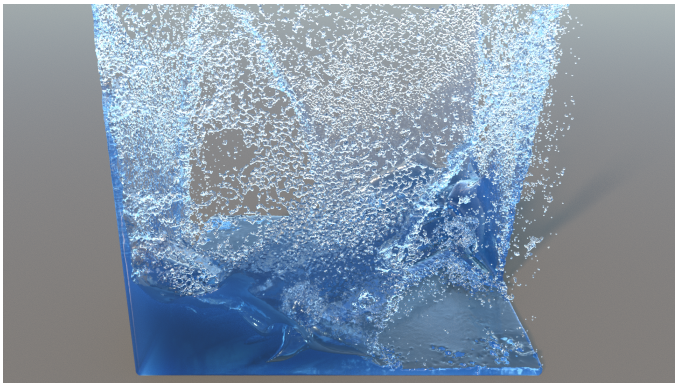


Fig. 15. Double dam break simulation with 3 million water particles.

dling of Akinci et al., once using density maps and once using our novel volume maps formulation to compare the performance. The average times per simulation step for the neighborhood search and the pressure and non-pressure forces are given in Table 1. The results show that the neighborhood search is faster when using density or volume maps. The reason for this is that no boundary particles need to be considered. The computation of the pressure and non-pressure forces using a map is also slightly faster than for the particle-based method since instead of a sum over all boundary neighbors only a lookup in a map is required. The comparison shows that our novel method has a performance that is comparable to the other methods and is even slightly faster than the particle-based approach.

7 Conclusion

In this paper we introduced a novel implicit boundary representation called volume maps. The most popular SPH boundary handling methods are based on a particle representation of the surface. However, the boundary particles lead to bumpy surfaces and inaccurate pressure forces. Density maps provide an implicit representation of the surfaces which avoids these problems. However, high-resolution maps are required since the density gradient computation suffers from discontinuities in the map. In contrast to the density maps approach our method only determines the overlapping volume of the boundary and the support domain of the smoothing kernel. The gradient computation

TABLE 1
Average times per step required for the neighborhood search, the pressure solve and the computation of the non-pressure forces in a double dam break simulation (see Fig. 15).

	Akinci et al.	Density Maps	Volume Maps
neigh. search	411 ms	379 ms	369 ms
pressure	1090 ms	1054 ms	1078 ms
non-pressure	357 ms	339 ms	342 ms
total	1858 ms	1715 ms	1789 ms

uses the classical SPH formulation which provides a consistent pressure force even for low resolution maps and does not suffer from map discontinuities. Therefore, in comparison to density maps we need about two orders of magnitude less precomputation time and memory. Since our method directly fits into the classical SPH formulation, it can be easily combined with any existing SPH method. Finally, in contrast to density maps our representation is independent of a specific kernel function and a single map can be combined with methods that use different kernels.

Our method has the same limitations as the density maps approach. A volume map cannot be precomputed for a deformable solid and it is computationally expensive to compute the map at runtime. Moreover, the cubic shape functions that we use for our map smooth out sharp features in the boundary model. However, this problem can be alleviated by using an adaptive discretization for the map which is a goal for our future work. Another future goal is to combine our volume maps with pressure extrapolation to further improve the boundary handling.

References

[1] D. Koschier and J. Bender, "Density maps for improved sph boundary handling," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 2017, pp. 1–10.

[2] S. Band, C. Gissler, and M. Teschner, "Moving least squares boundaries for sph fluids," in *Virtual Reality Interactions and Physical Simulations*. Eurographics Association, 2017, pp. 21–28.

[3] D. Koschier, C. Deul, M. Brand, and J. Bender, "An hp-adaptive discretization algorithm for signed distance field generation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 10, pp. 2208–2221, 2017.

[4] J. Bender, T. Kugelstadt, M. Weiler, and D. Koschier, "Volume maps: An implicit boundary representation for sph," in *Motion, Interaction and Games*. ACM, 2019.

[5] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner, "SPH Fluids in Computer Graphics," in *Eurographics (State of the Art Reports)*, 2014, pp. 21–42.

[6] D. Koschier, J. Bender, B. Solenthaler, and M. Teschner, "Smoothed particle hydrodynamics for physically-based simulation of fluids and solids," in *EUROGRAPHICS 2019 Tutorials*, 2019.

[7] J. Stam and E. Fiume, "Depicting fire and other gaseous phenomena using diffusion processes," in *Proc. Computer Graphics and Interactive Techniques*, 1995.

[8] M. Desbrun and M.-P. Gascuel, "Smoothed Particles: A new paradigm for animating highly deformable bodies," in *Eurographics Workshop on Computer Animation and Simulation*, 1996, pp. 61–76.

[9] M. Müller, D. Charypar, and M. Gross, "Particle-Based Fluid Simulation for Interactive Applications," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003.

[10] M. Becker and M. Teschner, "Weakly compressible SPH for free surface flows," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2007, pp. 1–8.

[11] B. Solenthaler and R. Pajarola, "Predictive-corrective incompressible SPH," *ACM Transactions on Graphics*, vol. 28, no. 3, 2009.

[12] M. Macklin and M. Müller, "Position Based Fluids," *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 1–5, 2013.

[13] M. Weiler, D. Koschier, and J. Bender, "Projective Fluids," in *ACM Motion in Games*. New York, NY, USA: ACM, 2016, pp. 79–84.

[14] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath, and M. Teschner, "Implicit incompressible SPH," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 3, 2014.

[15] J. Cornelis, J. Bender, C. Gissler, M. Ihmsen, and M. Teschner, "An optimized source term formulation for incompressible sph," *The Visual Computer*, vol. 35, no. 4, pp. 579–590, 2019.

[16] K. Raveendran, C. Wojtan, and G. Turk, "Hybrid smoothed particle hydrodynamics," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2011, pp. 33–42.

[17] J. Bender and D. Koschier, "Divergence-Free SPH for Incompressible and Viscous Fluids," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 3, pp. 1193–1206, 2017.

[18] J. Monaghan, "Simulating Free Surface Flows with SPH," *Journal of Computational Physics*, vol. 110, no. 2, pp. 399–406, 1994.

[19] M. Becker, H. Tessendorf, and M. Teschner, "Direct Forcing for Lagrangian Rigid-Fluid Coupling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 3, pp. 493–503, 2009.

[20] B. Solenthaler, J. Schläfli, and R. Pajarola, "A unified particle model for fluid-solid interactions," *Computer Anim. and Virtual Worlds*, vol. 18, no. 1, pp. 69–82, 2007.

[21] M. Ihmsen, N. Akinci, M. Gissler, and M. Teschner, "Boundary handling and adaptive time-stepping for PCISPH," in *Virtual Reality Interactions and Physical Simulations*, 2010, pp. 79–88.

[22] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, and M. Teschner, "Versatile rigid-fluid coupling for incompressible SPH," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 62:1–62:8, 2012.

[23] X. He, N. Liu, S. Li, H. Wang, and G. Wang, "Local Poisson SPH for Viscous Incompressible Fluids," *Computer Graphics Forum*, vol. 31, pp. 1948–1958, 2012.

[24] M. Müller, S. Schirm, M. Teschner, B. Heidelberger, and M. Gross, "Interaction of fluids with deformable solids," *Computer Anim. and Virtual Worlds*, vol. 15, no. 34, pp. 159–171, 2004.

[25] L. Yang, S. Li, A. Hao, and H. Qin, "Realtime Two-Way Coupling of Meshless Fluids and Nonlinear FEM," *Computer Graphics Forum*, vol. 31, no. 7, pp. 2037–2046, 2012.

[26] N. Akinci, J. Cornelis, G. Akinci, and M. Teschner, "Coupling elastic solids with smoothed particle hydrodynamics fluids," *Computer Anim. and Virtual Worlds*, vol. 24, no. 3-4, pp. 195–203, 2013.

[27] K. Bodin, C. Lacoursière, and M. Servin, "Constraint fluids," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, 2012.

[28] M. Huber, B. Eberhardt, and D. Weiskopf, "Boundary Handling at Cloth-Fluid Contact," *Comp. Graphics Forum*, vol. 34, no. 1, 2015.

[29] M. Fujisawa and K. Miura, "An Efficient Boundary Handling with a Modified Density Calculation for SPH," *Computer Graphics Forum*, vol. 34, no. 7, pp. 155–162, 2015.

[30] S. Kulasegaram, J. Bonet, R. W. Lewis, and M. Profit, "A variational formulation based contact algorithm for rigid boundaries in two-dimensional SPH applications," *Computational Mechanics*, vol. 33, no. 4, pp. 316–325, 2004.

[31] T. Harada, S. Koshizuka, and Y. Kawaguchi, "Smoothed Particle Hydrodynamics on GPUs," in *Comp. Graphics International*, 2007.

[32] ——, "Smoothed particle hydrodynamics in complex shapes," in *Spring Conf. on Computer Graph.*, 2007, pp. 191–197.

[33] S. Band, C. Gissler, M. Ihmsen, J. Cornelis, A. Peer, and M. Teschner, "Pressure boundaries for implicit incompressible sph," *ACM Transactions on Graphics*, vol. 37, no. 2, 2018.

[34] S. Band, C. Gissler, A. Peer, and M. Teschner, "MLS pressure boundaries for divergence-free and viscous SPH fluids," *Computers & Graphics*, vol. 76, pp. 37–46, 2018.

[35] C. Gissler, A. Peer, S. Band, J. Bender, and M. Teschner, "Interlinked sph pressure solvers for strong fluid-rigid coupling," *ACM Transactions on Graphics*, vol. 38, no. 1, pp. 5:1–5:13, 2019.

[36] J. J. Monaghan, "Smoothed Particle Hydrodynamics," *Reports on Progress in Physics*, vol. 68, no. 8, pp. 1703–1759, 2005.

[37] B. Müller, F. Kummer, M. Oberlack, and Y. Wang, "Simple multidimensional integration of discontinuous functions with application to level set methods," *International Journal for Numerical Methods in Engineering*, vol. 92, no. 7, pp. 637–651, 2012.

[38] M. Weiler, D. Koschier, M. Brand, and J. Bender, "A physically consistent implicit viscosity solver for sph fluids," *Computer Graphics Forum*, vol. 37, no. 2, 2018.

[39] J. Bærentzen and H. Aanæs, "Signed distance computation using the angle weighted pseudonormal," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 3, pp. 243–253, 2005.

[40] J. Bender, "SPlisHSPlasH Library," https://github.com/InteractiveComputerGraphics/SPlisHSPlasH, 2020.

[41] J. Bender, D. Koschier, T. Kugelstadt, and M. Weiler, "Turbulent micropolar sph fluids with foam," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 6, pp. 2284–2295, 2018.

[42] C. Deul, P. Charrier, and J. Bender, "Position-based rigid-body dynamics," *Computer Anim. and Virtual Worlds*, vol. 27, no. 2, 2014.

[43] J. Bender, M. Müller, and M. Macklin, "Position-based simulation methods in computer graphics," in *EUROGRAPHICS 2017 Tutorials*. Eurographics Association, 2017.

[44] A. Peer, C. Gissler, S. Band, and M. Teschner, "An implicit sph formulation for incompressible linearly elastic solids," *Computer Graphics Forum*, vol. 37, no. 6, pp. 135–148, 2017.

[45] C. Gissler, S. Band, A. Peer, M. Ihmsen, and M. Teschner, "Generalized drag force for particle-based simulations," *Computers & Graphics*, vol. 69, pp. 1–11, 2017.

[46] B. Solenthaler and R. Pajarola, "Density Contrast SPH Interfaces," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2008, pp. 211–218.

Jan Bender is professor of computer science and leader of the Computer Animation Group at RWTH Aachen University. He received his diploma, PhD and habilitation in computer science from the University of Karlsruhe. His research interests include interactive simulation methods, multibody systems, deformable solids, fluid simulation, collision handling, cutting, fracture, GPGPU and real-time visualization.

Tassilo Kugelstadt is a PhD student at RWTH Aachen University. He received his BSc degree in physics from JGU Mainz in 2013 and his MSc degree in Computer Science in the Natural Sciences from JGU Mainz in 2015. His research interests include physically-based simulation of deformable solids, elastic rods and fluids.

Marcel Weiler is a PhD student at RWTH Aachen University. He received his MSc degree in Computer Science from Technische Universität Darmstadt in 2015. His research interest include the physically-based simulation of fluids and deformable solids, rendering and GPU computing.

Dan Koschier has been a research associate in the Smart Geometry Processing group at University College London until 2019 and has since moved on to work in the medical technology industry. He received his PhD in Computer Science from RWTH Aachen University in 2018. His research interests include physically-based simulation of deformable solids, cutting, fracture, and fluids and machine learning aided physical simulations.