Volume 0 (1981), Number 0 pp. 1–13 COMPUTER GRAPHICS *forum*

An Implicit SPH Formulation for Incompressible Linearly Elastic Solids

Andreas Peer Christoph Gissler Stefan Band Matthias Teschner

University of Freiburg, Germany

Abstract

We propose a novel SPH formulation for deformable solids. Key aspects of our method are implicit elastic forces and an adapted SPH formulation for the deformation gradient that—in contrast to previous work—allows a rotation extraction directly from the SPH deformation gradient. The proposed implicit concept is entirely based on linear formulations. As a linear strain tensor is used, a rotation-aware computation of the deformation gradient is required. In contrast to existing work, the respective rotation estimation is entirely realized within the SPH concept using a novel formulation with incorporated kernel gradient correction for first-order consistency.

The proposed implicit formulation and the adapted rotation estimation allow for significantly larger time steps and higher stiffness compared to explicit forms. Performance gain factors of up to one hundred are presented. Incompressibility of deformable solids is accounted for with an ISPH pressure solver. This further allows for a pressure-based boundary handling and a unified processing of deformables interacting with SPH fluids and rigids. Self-collisions are implicitly handled by the pressure solver.

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

1. Introduction

The Smoothed Particle Hydrodynamics (SPH) method [Mon92] has evolved into one of the preferred choices for simulation in computer graphics. Most commonly known for the simulation of fluids [IOS*14], its Lagrangian nature makes SPH attractive for the simulation of other materials as well, with formulations proposed for unified SPH frameworks [SSP07, LD09] and in particular, for deformables [BIT09].

Whereas in the computational physics community, SPH for deformables has been an ongoing research topic, e.g., [Mon12, CPDH12, Gan15, GSMH16], in the computer graphics community the focus for deformables was led to other concepts like the Finite Element Method (FEM) [KMBG09, JK09, DGW11, SB12b, KBT17] and Position Based Dynamics (PBD) [BMO*14]. However, the improvements made in SPH fluid simulations in the last decade render the SPH concept an interesting alternative for the simulation of deformables in computer graphics.

One of the challenging aspects in deformable modeling is the rotation handling with SPH. If a nonlinear strain measure such as the Green strain tensor is used, this is less of an issue, e.g. [Gan15]. If, however, the linear infinitesimal strain tensor is used, rotations have to be estimated and used in corotational formulations, e.g. [BIT09].

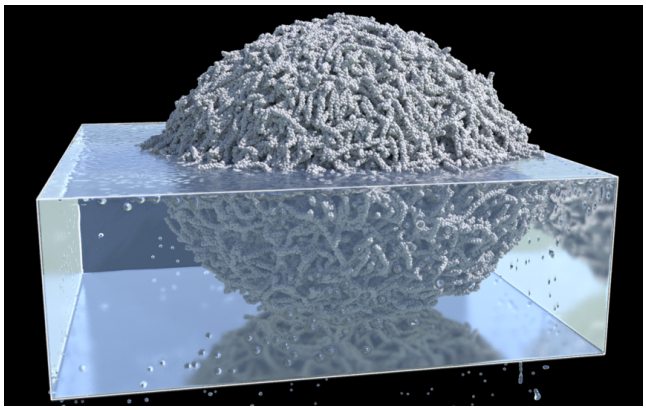


Figure 1: Particle view of NVIDIA's highly detailed deformable hairball model floating in a fluid.

As we aim at an implicit formulation, we use a corotational formulation employing the linear strain tensor, which we exploit to establish a linear relation between positions and elastic forces.

In the context of the rotation extraction, one difference of our method to [BIT09] is the SPH approximation of the deformation

gradient. As discussed in [BIT09], the SPH kernel gradient is not first-order consistent, which manifests in an erroneous representation of the rotation in the SPH deformation gradient. Therefore, we incorporate a correction of the kernel gradient for first-order consistency which allows to directly extract a meaningful rotation from the SPH deformation gradient. This is in contrast to [BIT09], where the rotation is extracted from an additional transformation matrix with the Moving Least Squares method (MLS) and also in contrast to [SSP07] where rotations are not addressed. Thus, our proposed solution is entirely based on the SPH concept.

Another difference to previous SPH methods for deformable objects is an implicit formulation that considers forces at the next timestep in the velocity update. This generally improves the performance compared to explicit formulations and we present performance gain factors of up to one hundred. While the general concept is very popular and well-established, e.g. [BW98, PICT15], the particular SPH formulation for deformables is novel. Our elastic forces are linear in the positions and we end up with a linear system. We further combine the proposed solver for elastic forces with an Incompressible SPH (ISPH) pressure solver [ICS*14] to gain the typical benefits such as incompressibility and pressure-based boundary handling. This enables versatile effects as illustrated in Fig. 1 with complex deformable objects that interact with fluids and rigids. Self-collisions are also implicitly handled by the pressure solver and phase transitions, e.g. from deformable to fluid or vice versa, can be easily simulated.

2. Related work

Particle-based formulations have a long history for the simulation of deformable bodies in computer graphics. Among the earliest and simplest models are mass-spring systems [TPF89, MP89], which have predominately been used for 1D or 2D bodies like cloths [BW98, BFA02], but also for fully 3D models [THMG04], and have gotten new attention recently [SSF13, LBOK13]. They are fast and easy to implement, but are difficult to map to the theoretical background of elasticity in physics.

For being able to use the physical formulation of elasticity, the gradient of the deformation map or alternatively, of the displacement field needs to be evaluated. This requires a way to measure the displacement, as well as a method to compute the gradient. The former is either done by using a stored reference configuration or by incrementally updating the deformation measure. The latter is done in particle-based simulations mainly using either SPH [Mon92] or MLS [LS81]. As a physically motivated approach, SPH is a popular particle-based simulation system. Mostly known for the simulation of fluids [IOS*14], with recent improvements in volume preservation [ICS*14, BK16, TL16], boundary handling [AIA*12, SB12a], multiphase simulation [SP08, RLY*14] and viscosity handling [PICT15, TDF*15, BK16, PT17], its Lagrangian nature makes it appealing for the simulation of deformable objects as well, for which it has been used in computer graphics since the beginning [DG96]. It allows to compute gradients by exploiting the gradient of the smoothing kernel. However, in the standard formulation, the SPH kernel gradient evaluation is not first-order consistent, which means that it fails to capture rotational motions. As a loophole, Müller et al. [MKN*04] introduced MLS to the computer graphics community, which computes gradients by fitting a polynomial function. Using a linear basis, it is able to correctly capture rotational motions. However, this comes with the cost of inverting a matrix, which is tricky for colinear or coplanar particle arrangements [MKB*10]. Nevertheless, MLS has been used in a variety of works ranging from plastic deformation [JWJ*14] to fracturing [PKA*05], using either a reference configuration, or updating the deformation gradient incrementally [GBB09]. Zhou et al. [ZLKW13] proposed an MLS-based implicit formulation, which is similar to ours in the sense that rotation is assumed to be constant during the simulation step in order to get a linear system to solve. Their approach is, however, purely focused on elasto-plastic solids, whereas our work is not only focused on elastic solids, but also emphasizes the interplay with other material types by employing beneficial SPH formulations.

Because of its simplicity and versatility, many works prefer SPH for the simulation of deformable objects, as it fits well into a unified framework for the simulation of a wide range of materials. Keiser et al. [KAG*05] proposed a framework for the simulation of deformable solids and fluids. An Eulerian formulation is employed, where the displacement field is evaluated in the current configuration. An initial state, but no initial connectivity is stored. They compute pressure and viscosity forces using the SPH formalism. However, elastic forces are computed as in [MKN*04], using MLS to retrieve the gradient of the displacement field. Wicke et al. [WHP*06] proposed a modified MLS-based approach that goes without initial state. Instead, they assume the particles to be aligned in a lattice, and compute a transformation for the current state that best matches the predefined lattice. Solenthaler et al. [SSP07] build on [KAG*05], but modify the approach in two ways. First, they use a Lagrangian formulation where the displacement field is evaluated in a stored reference configuration. Second, they use SPH for all computations. However, as the default SPH kernel gradient computation is not first-order consistent, they fail to correctly capture rigid-body rotations of their deformable objects. Becker et al. [BIT09] noted this and switched back to an MLS-SPH combined approach, but in contrast to [KAG*05], they use MLS just for extracting the rotational part of the deformation gradient, and further use a corotated [MDM*02] SPH formulation that incorporates the obtained rotation information. In contrast to all those works, we solely rely on SPH for capturing rotations. Furthermore, instead of using explicit force computations, we propose an implicit formulation that allows for significantly larger timesteps.

Apart from models with physical elasticity, SPH has also been combined with simpler formulations like mass-spring systems for the interaction with thin shells [LD08] or for viscoelastic flow [CBP05]. As a simple model for elasticity, Dagenais et al. [DGP12] proposed to use shape matching [MHTG05] to blend fluid and solid motion in order to mimic deformable objects.

Since computing the gradient is a non-trivial task on a randomly-arranged particle set, but comparatively easy on a regular grid, the Material Point Method (MPM) uses an underlying grid for the force computation and has been used for phase changes [SSJ*14] and dimension-reduced deformables [JGT17]. In each simulation step, the required quantities are transferred from the particles to the grid, where the forces are computed, and the resulting changes are trans-

ferred back to the Lagrangian particles. As we employ a kernel correction, we are able to reliably compute all relevant quantities on a randomly arranged particle set, and we do not need the additional grid.

Another alternative for gradient computations with a particular focus on discontinuities is proposed within the peridynamics concept [Sil00]. This concept has also recently been used for the simulation of elastoplastic materials [HWW17]. Depending on the application, there is a close relation between SPH and peridynamics. If peridynamics is, e.g., applied to classical material models based on the deformation gradient, Ganzenmüller et al. [GHM15] show that the resulting forces exactly match a respective SPH discretization.

Beside particle-based formulations, a great variety of methods have been used to simulate deformable objects in computer graphics, with FEM [SB12b] being one of the most common choices. We refer the reader to [NMK*06] for a slightly outdated, but still excellent overview of the different methods. Recently, position-based dynamics [MHHR07, BMO*14] and its generalization, the projective dynamics framework [BML*14] got attention for the simulation of a great variety of materials, from fluids [MM13, WKB16] to solids [MC11] and a wide range of deformable materials, e.g., [BKCW14, CMM16, MMC16, LBK16, NOB16, TDFN16]. Its constrained-based nature makes it attractive for the unified simulation of objects with different material characteristics as well [MMCK14].

3. Method

3.1. Background and notation

Using a continuum-mechanics framework, elastic materials are characterized by an initial or reference configuration $\mathbf{x}^0$ and the current configuration $\mathbf{x}$. In this work, we will use the superscript $*^0$ to denote quantities in the initial configuration. Elastic forces act to bring an object from its current shape back to its initial shape. The deformation function ϕ maps the initial configuration to the current configuration $\mathbf{x} = \phi(\mathbf{x}^0)$. The deformation gradient $\mathbf{F} = \nabla^0 \phi = \frac{\partial \phi}{\partial \mathbf{x}^0} = \frac{\partial \mathbf{x}}{\partial \mathbf{x}^0}$ states how the deformation changes spatially. The displacement $\mathbf{u} = \mathbf{x} - \mathbf{x}^0$ is an alternative measure for it. Its gradient is given by $\nabla^0 \mathbf{u} = \mathbf{F} - \mathbf{I}$. For Cauchy elastic materials, the stress tensor from which the elastic forces are computed solely depends on $\mathbf{F}$.

We generally employ the SPH concept for the spatial discretization of all relevant equations. In SPH, a continuum volume is approximately represented by samples or particles. For the computation of a quantity A_i at position $\mathbf{x}_i$, a weighted average over neighboring particles at positions $\mathbf{x}_j$ is employed: $A_i = \sum_j V_j A_j W_{ij}$. The value V_j represents the volume of the particle at $\mathbf{x}_j$ and $W_{ij} = W(\mathbf{x}_i - \mathbf{x}_j)$ is the smoothing kernel. Similarly, the gradient can be computed, e.g., with $\nabla A_i = \sum_j V_j A_j \nabla W_{ij}$ with $\nabla W_{ij} = \nabla W\left(\mathbf{x}_i - \mathbf{x}_j\right)$ being the gradient of the smoothing kernel.

3.2. Lagrangian SPH

The distinction between initial and current configuration rises the question on which neighborhood to use for the SPH interpolation. For fluids, that typically show no elastic behavior, the well-known Eulerian SPH is used, where the neighborhood is evaluated using the current particle positions. Here, the term *Eulerian* denotes the characteristic that when looking at a single particle, other particles enter and leave its influence sphere frequently. For deformables, however, it has been shown that the Eulerian formulation leads to the tensile instability problem and that a Lagrangian formulation removes it [BGLX00, BK01]. In *Lagrangian SPH*, the initial positions are used for finding the neighborhood and therefore, the set of particles interacting with each other is kept fixed. In the following, all SPH interpolations regarding the elastic force use the initial neighborhood. To reflect that in the formulas, we use the notation $\sum_j^0$ to denote a sum over the particles in the initial neighborhood.

3.3. Overview

Algorithm 1 summarizes the general steps of the deformable update procedure. The main novelties compared to previous works lie in the computation of the deformation gradient as well as in the implicit application of the resulting force, which will be explicated in Section 3.4. In this section, each step of the update procedure will be introduced in detail.

Algorithm 1 Deformable update

Extract rotation
- compute deformation gradient $\mathbf{F}$
- extract rotational component $\mathbf{R}$ from $\mathbf{F}$

Compute elastic force using the extracted rotation
- compute corotated deformation gradient $\overset{\star}{\mathbf{F}}$ using $\mathbf{R}$
- compute strain tensor ε from $\overset{\star}{\mathbf{F}}$
- compute stress tensor $\mathbf{P}$ from ε
- compute elastic force $\mathbf{f}$ from $\mathbf{P}$

Deformation gradient

The proper computation of the deformation gradient $\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{x}^0}$ is the basis for implementing a deformable model. However, a trivial implementation in SPH as $\mathbf{F}_i = \sum_j^0 V_j^0 \mathbf{x}_{ji} \otimes \nabla W\left(\mathbf{x}_{ij}^0\right)$, with $\mathbf{x}_{ji} = \mathbf{x}_j - \mathbf{x}_i$ the distance vector, fails to capture rotational motion. As pointed out by Bonet and Lok [BL99], this is due to the fact the common SPH kernel gradient interpolation is not first-order consistent. Therefore, rotations are misinterpreted as deformations. As a consequence, the employed elastic forces are not rotationally invariant. Instead, forces are induced that rotate the object back to its initial orientation.

The deformation gradient acts as a transformation matrix that transforms an infinitesimal line element $d\mathbf{x}^0$ from an initial position to the current position, as $\frac{\partial \mathbf{x}}{\partial \mathbf{x}^0} d\mathbf{x}^0 = d\mathbf{x}$. This opens up other possibilities for computing the deformation gradient. Becker et al. [BIT09] employed shape matching – an MLS approach – to compute the deformation gradient for extracting rotation. In our work, we adopt the corotational concept of [BIT09], but propose an alternative SPH approximation for the deformation gradient that allows to extract the rotation directly from the SPH deformation gradient. MLS is avoided and the entire procedure relies on SPH only.

Rotation-aware kernel gradient

In order to correctly capture rotational motion with the SPH kernel gradient, it has to be ensured that its computation is first-order consistent. As shown by Bonet and Lok [BL99], for the kernel gradient to be first-order consistent, the condition $\sum_j^0 V_j^0 \mathbf{x}_{ji}^0 \otimes \nabla W\left(\mathbf{x}_{ij}^0\right) = \mathbf{I}$ has to be satisfied. For this sake, they propose a correction matrix

$$\mathbf{L}_i = \left(\sum_j^0 V_j^0 \nabla W\left(\mathbf{x}_{ij}^0\right) \otimes \mathbf{x}_{ji}^0\right)^{-1} \tag{1}$$

and compute the corrected kernel gradient as

$$\nabla \tilde{W}_i\left(\mathbf{x}_{ij}^0\right) = \mathbf{L}_i \nabla W\left(\mathbf{x}_{ij}^0\right). \tag{2}$$

This ensures first-order consistency, as $\sum_j^0 V_j^0 \mathbf{x}_{ji}^0 \otimes \nabla \tilde{W}_i\left(\mathbf{x}_{ij}^0\right) = \mathbf{I}$ is satisfied by construction. In Lagrangian SPH, the correction matrix is computed in the initial configuration and therefore, it has to be computed only once at the beginning of the simulation. For coplanar or collinear particle configurations, the matrix to invert in Eq. (1) is singular. In such cases, we compute its Moore–Penrose pseudoinverse, instead.

Rotation extraction

As the corrected kernel gradient is able to capture rotations, it is straightforwardly possible to use a nonlinear strain measure such as the Green strain tensor, given as $\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right)$. However, with the Green strain tensor, the elastic force is nonlinear in the position. For an implicit formulation, this is unfavorable, as it requires to solve a nonlinear system. In such a situation, one would prefer to use the infinitesimal strain tensor, given as $\varepsilon = \frac{1}{2}\left(\mathbf{F} + \mathbf{F}^T\right) - \mathbf{I}$, as with it, the force linearly depends on position, and the implicit formulation results in a more efficient linear system. Therefore, inspired by the corotational formulation of Becker et al. [BIT09], we extract rotation – which is responsible for nonlinearity – beforehand. Doing so, we are able to use a strain measure linear in position. In contrast to Becker et al., using the corrected kernel gradient, rotation can be extracted directly from the deformation gradient computed as

$$\mathbf{F}_i = \sum_j^0 V_j^0 \mathbf{x}_{ji} \otimes \nabla \tilde{W}_i\left(\mathbf{x}_{ij}^0\right). \tag{3}$$

To extract the per-particle rotation $\mathbf{R}_i$ from $\mathbf{F}_i$, we use the method of Müller et al. [MBCM16] which is fast and stable.

Corotated deformation gradient

Similar to Becker et al. [BIT09], we use the extracted rotation for computing a corotated deformation gradient that only contains shear and expansion parts. Because of objectivity of Cauchy elastic materials, the computed stress tensor, and therefore the elastic force, is invariant to rotation. This allows us to compute all quantities in an arbitrary rotational frame, as long as we make sure the resulting forces are rotated back to the actual frame. Using the per-particle rotation extracted above, Becker et al. propose to compute the corotated deformation gradient by means of the displacement field as $\mathbf{F} = \mathbf{I} + \nabla^0 \mathbf{u} = \mathbf{I} + \nabla^0\left(\mathbf{x} - \mathbf{x}^0\right)$ and by further rotating the current positions back to the initial positions which translates to SPH as $\mathbf{F}_i = \mathbf{I} + \sum_j^0 V_j^0 \left(\mathbf{R}_i^{-1}\mathbf{x}_{ji} - \mathbf{x}_{ji}^0\right) \otimes \nabla W\left(\mathbf{x}_{ij}^0\right)$. We modify this computation in the following way. Instead of rotating the current configuration back to the initial configuration, we rotate the initial configuration to the current configuration. Doing so, we have to rotate all initial quantities, which means that the corrected kernel has to be evaluated in the current frame as well. For that, we introduce the rotated kernel gradient as

$$\nabla \overset{\star}{W}_i\left(\mathbf{x}_{ij}^0\right) = \mathbf{R}_i \mathbf{L}_i \nabla W\left(\mathbf{x}_{ij}^0\right) \tag{4}$$

which rotates the corrected kernel gradient to the frame of the current configuration. With these modifications, the deformation gradient is finally computed as

$$\overset{\star}{\mathbf{F}}_i = \mathbf{I} + \sum_j^0 V_j^0 \left(\mathbf{x}_{ji} - \mathbf{R}_i \mathbf{x}_{ji}^0\right) \otimes \nabla \overset{\star}{W}_i\left(\mathbf{x}_{ij}^0\right). \tag{5}$$

Note that in this computation of the deformation gradient, only displacements in the initial configuration are prefactored by the rotation matrix. This will turn advantageous when introducing the implicit formulation, as those costly matrix-vector multiplications have to be performed only once during the set-up of the system to solve, reducing the cost of each solver iteration.

Linear elasticity

As a constitutive model, we use the linear elasticity material model. Its linearity in position results in a linear system for our implicit formulation, which greatly reduces the computational effort for solving the system.

Strain The strain tensor used in the linear elasticity model is the infinitesimal strain tensor

$$\varepsilon_i = \frac{1}{2}\left(\overset{\star}{\mathbf{F}}_i + \overset{\star}{\mathbf{F}}_i^T\right) - \mathbf{I} \tag{6}$$

In contrast to the Green strain tensor $\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right)$, position does not appear quadratically, but only linear.

Stress In the linear elasticity model, stress depends linearly on strain. The first Piola-Kirchhoff stress tensor $\mathbf{P}$ is given as

$$\mathbf{P}_i = 2\mu\varepsilon_i + \lambda \mathrm{tr}(\varepsilon_i)\mathbf{I} \tag{7}$$

with the Lamé parameters λ and μ. Alternatively, the stress tensor can be decomposed into a volumetric and a deviatoric part,

$$\begin{aligned}\mathbf{P}_i =& 2G\left(\varepsilon_i - \frac{1}{3}\mathrm{tr}(\varepsilon_i)\mathbf{I}\right) + K\mathrm{tr}(\varepsilon_i)\mathbf{I} \\ =& 2G\varepsilon_i + \left(K - \frac{2}{3}G\right)\mathrm{tr}(\varepsilon_i)\mathbf{I}\end{aligned} \tag{8}$$

with shear modulus $G = \mu$ and bulk modulus $K = \lambda + \frac{2\mu}{3}$. We parameterize the simulations in our experiments with those moduli.

Elastic force

We compute the elastic force from the divergence of the stress tensor, which in the simplest case can be computed as $\mathbf{f}_i = \sum_j^0 V_i^0 V_j^0 \left(\mathbf{P}_i + \mathbf{P}_j\right) \nabla W\left(\mathbf{x}_{ij}^0\right)$. Note that in contrast to [BIT09], we do not rotate the force, as we compute the deformation gradient in the current frame already. However, as the rotated and corrected

kernel depends on the particle of evaluation, to get a symmetric force, each stress tensor has to be multiplied by the corresponding kernel gradient separately [Gan15], resulting in

$$\mathbf{f}_i = \sum_j^0 V_i^0 V_j^0 \left(\mathbf{P}_i \nabla \overset{\star}{W}_i \left(\mathbf{x}_{ij}^0 \right) - \mathbf{P}_j \nabla \overset{\star}{W}_j \left(\mathbf{x}_{ji}^0 \right) \right). \tag{9}$$

3.4. Implicit formulation

The previous section has explained all equations that are used in our computation of the elastic force. Depending on the choice of arguments, it translates to either an explicit or an implicit formulation. When approximately assuming the extracted rotation to remain constant during the simulation step, the elastic force in Equation (9) can be regarded as a function $\mathbf{f}(\mathbf{d}, \mathbf{d}^0)$ that linearly depends on two position vectors $\mathbf{d}$ and $\mathbf{d}^0$. The function is given as

$$\mathbf{f}_i(\mathbf{d}, \mathbf{d}^0) = \sum_j^0 V_i^0 V_j^0 \Big(\mathbf{P}_i(\mathbf{d}, \mathbf{d}^0) \nabla \overset{\star}{W}_i \left(\mathbf{x}_{ij}^0 \right) - \mathbf{P}_j(\mathbf{d}, \mathbf{d}^0) \nabla \overset{\star}{W}_j \left(\mathbf{x}_{ji}^0 \right) \Big) \tag{10}$$

with stress, strain and the deformation gradient expressed similarly as

$$\mathbf{P}_i(\mathbf{d}, \mathbf{d}^0) = 2G\varepsilon_i(\mathbf{d}, \mathbf{d}^0) + \left(K - \frac{2}{3} G \right) \mathrm{tr}\left(\varepsilon_i(\mathbf{d}, \mathbf{d}^0) \right) \mathbf{I} \tag{11}$$

$$\varepsilon_i(\mathbf{d}, \mathbf{d}^0) = \frac{1}{2} \left(\overset{\star}{\mathbf{F}}_i(\mathbf{d}, \mathbf{d}^0) + \overset{\star}{\mathbf{F}}_i(\mathbf{d}, \mathbf{d}^0)^T \right) - \mathbf{I} \tag{12}$$

$$\overset{\star}{\mathbf{F}}_i(\mathbf{d}, \mathbf{d}^0) = \mathbf{I} + \sum_j^0 V_j^0 \left(\mathbf{d}_{ji} - \mathbf{R}_i \mathbf{d}_{ji}^0 \right) \otimes \nabla \overset{\star}{W}_i \left(\mathbf{x}_{ij}^0 \right). \tag{13}$$

For the explicit formulation, the two parameters are given by the position at the current timestep and the initial position, i.e., $\mathbf{d} = \mathbf{x}^t$ and $\mathbf{d}^0 = \mathbf{x}^0$, respectively. The explicit velocity update can then be written as

$$\mathbf{v}_{exp}^{t+\Delta t} = \mathbf{v}^t + \Delta t \frac{\mathbf{f}(\mathbf{x}^t, \mathbf{x}^0)}{m} \tag{14}$$

with m denoting the particle mass. For an implicit formulation, the position at the next timestep is used instead, such that $\mathbf{d} = \mathbf{x}^{t+\Delta t}$, and the velocity update changes to

$$\mathbf{v}_{imp}^{t+\Delta t} = \mathbf{v}^t + \Delta t \frac{\mathbf{f}(\mathbf{x}^{t+\Delta t}, \mathbf{x}^0)}{m}. \tag{15}$$

Using a backward difference for the position at the next timestep, $\mathbf{x}^{t+\Delta t} = \mathbf{x}^t + \Delta t \mathbf{v}^{t+\Delta t}$, the implicit velocity update is given as

$$\mathbf{v}^{t+\Delta t} = \mathbf{v}^t + \Delta t \frac{\mathbf{f}(\mathbf{x}^t + \Delta t \mathbf{v}^{t+\Delta t}, \mathbf{x}^0)}{m} \tag{16}$$

where the velocity subscript has been omitted for readability. As we assume the rotation to remain constant during the update, the elastic force linearly depends on the position, and the computation of $\mathbf{f}$ can be split into $\mathbf{f}(\mathbf{x}^t + \Delta t \mathbf{v}^{t+\Delta t}, \mathbf{x}^0) = \mathbf{f}(\mathbf{x}^t, \mathbf{x}^0) + \mathbf{f}(\Delta t \mathbf{v}^{t+\Delta t}, \mathbf{0})$ to get

$$\mathbf{v}^{t+\Delta t} - \Delta t \frac{\mathbf{f}(\Delta t \mathbf{v}^{t+\Delta t}, \mathbf{0})}{m} = \mathbf{v}^t + \Delta t \frac{\mathbf{f}(\mathbf{x}^t, \mathbf{x}^0)}{m} \tag{17}$$

where $\mathbf{0}$ denotes the zero vector. This forms a linear system of the form $\mathbf{A}\mathbf{v}^{t+\Delta t} = \mathbf{b}$ for the unknown velocities at the next timestep.

Discussion In our implicit formulation, we use the simplifying assumption that the rotations remain constant during each simulation step. This allows to extract the rotations beforehand and therefore, to derive corotated forces that are linear in positions. This largely reduces the computational effort for solving the resulting system. A similar assumption is taken in [ZLKW13]. With fixed rotations, the simulation outcome will slightly differ compared to a nonlinearized formulation, as potential rotation changes are not considered. In practice, however these changes are rather small and the simulations look plausible.

The system in Eq. (17) shares some similarities with the system for implicit viscous forces in [TDF*15], as both viscous and elastic forces are computed from the divergence of a stress tensor. In fact, the system in [TDF*15] can be regarded a special case of our system with no rotation handling and kernel correction employed. In contrast to [TDF*15], we do not pursue to extract the coefficients of the system matrix, but we propose a more efficient matrix-free implementation, instead.

4. Implementation

4.1. Initial configuration

In the course of the deformables update, no initial positions, but initial distance vectors are processed. Therefore, following [SSP07] and [BIT09], we do not store the absolute initial position $\mathbf{x}_i$ of a particle, but the distance vectors to its neighbors $\mathbf{x}_{ji}$, instead. Whenever new particles are inserted, a neighborhood search is performed and the initial distance vectors for all particle-neighbor pairs are stored. Furthermore, the kernel correction matrix is precomputed using Eq. (1) and stored for each particle.

4.2. Solving the system

For our deformable formulation, the linear system in Eq. (17) needs to be solved. We propose a matrix-free implementation to efficiently achieve this using the conjugate gradient method. This is accomplished in three steps. First, we precompute and store the rotated kernel gradient for each particle-neighbor pair by evaluating Eq. (4). Then, we compute the right-hand side of Eq. (17). Finally, we solve the system iteratively.

Both the computation of the left-hand side and the right-hand side require the evaluation of the elastic force function in Eq. (10). It can be evaluated efficiently by iterating twice over all particles and their neighbors, as outlined in Algorithm 2. In the first iteration, we loop once over all neighbors to compute $\overset{\star}{\mathbf{F}}$ using Eq. (13), and we hereupon compute ε and $\mathbf{P}$ using Eq. (12) and Eq. (11), respectively. Finally, we store $\mathbf{P}$. As it is symmetric, only six independent components need to be stored. Then, in the second iteration, we loop once again over all neighbors to compute the divergence of $\mathbf{P}$ using Eq. (10).

Right-hand side The right-hand side of Eq. (17) just computes the velocity update due to an explicit elastic force as in Eq. (14), which can be computed by iterating twice over all particles and neighbors as described above. Moreover, when using the corrected kernel gradient, the computation of the corotated deformation gradient as in

Algorithm 2 Computation of $\mathbf{f}(\mathbf{d}, \mathbf{d}^0)$

for all particle i **do**
$\overset{\star}{\mathbf{F}}_{temp} = \mathbf{I} + \sum_j^0 V_j^0 \left(\mathbf{d}_{ji} - \mathbf{R}_i \mathbf{d}_{ji}^0\right) \otimes \nabla \overset{\star}{W}_i\left(\mathbf{x}_{ij}^0\right)$
$\varepsilon_{temp} = \frac{1}{2}\left(\overset{\star}{\mathbf{F}}_{temp} + (\overset{\star}{\mathbf{F}}_{temp})^T\right) - \mathbf{I}$
$\mathbf{P}_i = 2G\varepsilon_{temp} + \left(K - \frac{2}{3}G\right)\mathrm{tr}(\varepsilon_{temp})\mathbf{I}$
for all particle i **do**
$\mathbf{f}_i = \sum_j^0 V_i^0 V_j^0 \left(\mathbf{P}_i \nabla \overset{\star}{W}_i\left(\mathbf{x}_{ij}^0\right) - \mathbf{P}_j \nabla \overset{\star}{W}_j\left(\mathbf{x}_{ji}^0\right)\right)$

Eq. (13) simplifies to

$$\overset{\star}{\mathbf{F}}_i(\mathbf{x}, \mathbf{x}^0) = \sum_j^0 V_j^0 \mathbf{x}_{ji} \otimes \nabla \overset{\star}{W}_i\left(\mathbf{x}_{ij}^0\right), \tag{18}$$

as $\sum_j^0 V_j^0 \mathbf{R}_i \mathbf{x}_{ji}^0 \otimes \nabla \overset{\star}{W}_i\left(\mathbf{x}_{ij}^0\right) = \mathbf{I}$. Strain, stress and force are then computed as usual using the respective Eqs. (10) to (12).

Solver iteration In each iteration, the matrix-vector product $\mathbf{Ap}$, with $\mathbf{p}$ the basis vector in case of the conjugate gradient method, has to be computed. We do not build the matrix $\mathbf{A}$ explicitly, but rather compute the product, which is given by the left-hand side of Eq. (17) for each particle as

$$(\mathbf{Ap})_i = \mathbf{p}_i - \frac{\Delta t}{m}\mathbf{f}_i(\Delta t \mathbf{p}, \mathbf{0}). \tag{19}$$

The complexity lies in computing the elastic force $\mathbf{f}$. However, as shown above, this can be efficiently done by iterating twice over all particles and neighbors. The first iteration in the procedure described above, which consists in the computation of the deformation gradient and the strain and stress tensors, can further be slightly optimized. We note that when computing the corotated deformation gradient using Eq. (13) as

$$\overset{\star}{\mathbf{F}}_i(\Delta t \mathbf{p}, \mathbf{0}) = \mathbf{I} + \Delta t \sum_j^0 V_j^0 \mathbf{p}_{ji} \otimes \nabla \overset{\star}{W}_i\left(\mathbf{x}_{ij}^0\right), \tag{20}$$

and strain using Eq. (12) as

$$\varepsilon_i(\Delta t \mathbf{p}, \mathbf{0}) = \frac{1}{2}\left(\overset{\star}{\mathbf{F}}_i(\Delta t \mathbf{p}, \mathbf{0}) + \left(\overset{\star}{\mathbf{F}}_i(\Delta t \mathbf{p}, \mathbf{0})\right)^T\right) - \mathbf{I} \tag{21}$$

the addition of $\mathbf{I}$ in the deformation gradient is canceled by its subtraction in the strain tensor. Therefore, following most computer graphics papers, in lieu of the deformation gradient, we compute the gradient of the displacement field as $\nabla^0 \mathbf{u} = \mathbf{F} - \mathbf{I}$, resulting in

$$\overset{\star}{\nabla^0} \mathbf{u}_i(\Delta t \mathbf{p}, \mathbf{0}) = \Delta t \sum_j^0 V_j^0 \mathbf{p}_{ji} \otimes \nabla \overset{\star}{W}_i\left(\mathbf{x}_{ij}^0\right). \tag{22}$$

Strain is then computed as

$$\varepsilon_i(\Delta t \mathbf{p}, \mathbf{0}) = \frac{1}{2}\left(\overset{\star}{\nabla^0} \mathbf{u}_i(\Delta t \mathbf{p}, \mathbf{0}) + \left(\overset{\star}{\nabla^0} \mathbf{u}_i(\Delta t \mathbf{p}, \mathbf{0})\right)^T\right), \tag{23}$$

and we can continue with the calculation of stress as usual.

4.3. Conjugate gradient method

In our implementation, we use the conjugate gradient method for solving the linear system, which requires a symmetric system matrix in order to converge. From Eqs. (10), (13) and (17) it follows that this requires the same initial volume and the same mass for all particles. Therefore, in addition to assigning all particles the same mass $m_0 = h^3 \rho_0$ with h denoting the particle spacing and ρ_0 the rest density, we initially distribute the particles homogeneously to the given volume. Then, all particles have the same mass m_0 and roughly the same volume $V_0^0 = \frac{m_0}{\rho_0}$. With these assumptions, the system matrix is symmetric. As stopping criterion, we use the average absolute per-particle error of the system. In our experiments, this error is set to 1×10^{-3}. Furthermore, to improve convergence, we perform a "warm start" by initializing the solver with the temporary intermediate velocity $\mathbf{v}^t + \Delta t_c \mathbf{a}^*$, as introduced in the next section.

4.4. Embedding in the fluid solver

Our elastic solver is integrated into an IISPH framework as outlined in Alg. 3. The IISPH pressure solver accounts for collision handling and incompressibility. As correct collision handling is crucial for the simulation, the pressure solver is executed as the last step of the fluid update. However, the accelerations induced by elastic forces may introduce high velocities that violate the CFL condition and thus undermine collision handling. Therefore, our proposed update procedure is as follows. First, we estimate a candidate timestep Δt_c from the current velocities. We compute intermediate accelerations $\mathbf{a}^*$ induced by gravity, viscosity and friction. We use the candidate timestep to integrate to a temporary intermediate velocity $\mathbf{v}^t + \Delta t_c \mathbf{a}^*$, for which the elastic system in Eq. (17) is solved. The system to solve is

$$\mathbf{v}_{elast} - \Delta t_c \frac{\mathbf{f}(\Delta t_c \mathbf{v}_{elast}, \mathbf{0})}{m} = \mathbf{v}^t + \Delta t_c \left(\mathbf{a}^* + \frac{\mathbf{f}(\mathbf{x}^t, \mathbf{x}^0)}{m}\right) \tag{24}$$

where compared to Eq. (17) the current velocity has been replaced by the intermediate velocity, such that the right-hand side incorporates the velocity change induced by explicit forces. With the solution velocities $\mathbf{v}_{elast}$, the timestep is reestimated in order to ensure that the CFL condition is still satisfied. Pressure projection as well as velocity and position integration is then done using the reestimated timestep.

Algorithm 3 Deformable SPH fluid solver

estimate candidate timestep Δt_c from $\mathbf{v}^t$
compute gravity, viscosity and friction to get acceleration $\mathbf{a}^*$
extract rotations using $\mathbf{x}^t$
solve $\mathbf{v}_{elast} - \Delta t_c \frac{\mathbf{f}(\Delta t_c \mathbf{v}_{elast}, \mathbf{0})}{m} = \mathbf{v}^t + \Delta t_c \left(\mathbf{a}^* + \frac{\mathbf{f}(\mathbf{x}^t, \mathbf{x}^0)}{m}\right)$
update acceleration: $\mathbf{a}^{**} \leftarrow \frac{\mathbf{v}_{elast} - \mathbf{v}^t}{\Delta t_c}$
estimate timestep Δt from $\mathbf{v}_{elast}$
compute press. and press. force using $\mathbf{v}^t$, $\mathbf{a}^{**}$ and Δt to get $\mathbf{a}^{***}$
update velocity: $\mathbf{v}^{t+\Delta t} \leftarrow \mathbf{v}^t + \Delta t \mathbf{a}^{***}$
update position: $\mathbf{x}^{t+\Delta t} \leftarrow \mathbf{x}^t + \Delta t \mathbf{v}^{t+\Delta t}$

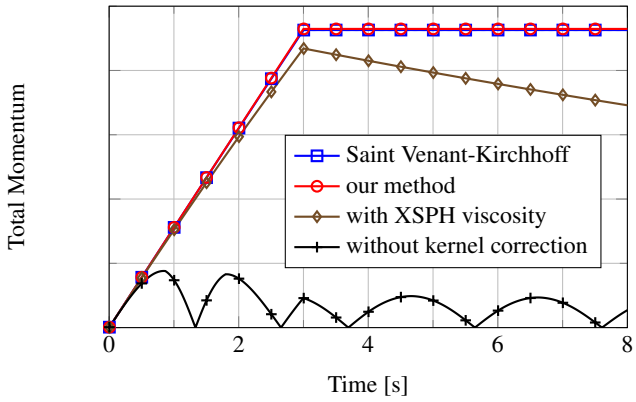


Figure 2: Momentum of a cube during and after 3 s within a rotational force field. Our rotation handling is of similar accuracy compared to the nonlinear Saint Venant-Kirchhoff model. Employing XSPH viscosity damps rotation. Not employing the kernel correction suppresses rotation.

5. Results

This section illustrates the properties of our proposed SPH-based deformables solver. Our solver is integrated into an IISPH framework [ICS*14]. As kernel, the quintic formulation of Wendland (Wendland C^2) is used [Wen95] with a support radius of twice the particle spacing. From a theoretical perspective, this kernel avoids the pairing instability [DA12]. In practice, however, we did not experience differences to the commonly used Cubic spline kernel which works as well in our setting. Boundary handling and two-way coupling is done using particles [AIA*12]. For multiphase simulations, the number density approach is employed [SP08] with an additional surface tension force based on [AAT13] for the water phase. We further use a drag force for all phases [GBP*17] and employ XSPH viscosity [Mon89, SB12a] in the material space. A viscosity force as in [MFZ97] is used to model friction at the rigid interface. Particles for deformable objects are generated with Poisson-disk sampling using a dart-throwing approach based on [CJW*09]. Our implementation is fully parallelized [IABT11], and the experiments were run on a 16-core Intel Xeon CPU with 3.10 GHz. The scenes were rendered using Houdini's physically based Mantra renderer [Sid16] at 50 frames per second.

5.1. Rotation handling

We compare the rotation handling of our implicit formulation to a Saint Venant–Kirchhoff model which uses the nonlinear, rotation invariant Green strain tensor. We further compare to our formulation without kernel correction and to our formulation with an additional small XSPH viscosity force. In the experiment, a deformable cube is accelerated for 3 s in a rotational force field. The plot in Fig. 2 shows the resulting total momentum. Our method shows the same perfect rotation conservation as the Saint Venant–Kirchhoff model. However, when further employing an XSPH viscosity force, as it is typically done in practice, rotation is damped. Without kernel correction, the cube is pulled back to its original orientation and rotation is suppressed.

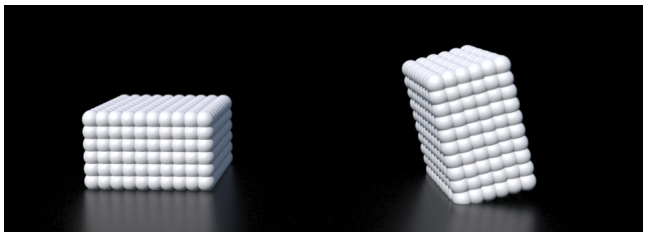


Figure 3: Two cuboids falling onto a plane. For the cuboid on the right, kernel correction is disabled, which prevents it from rotating.

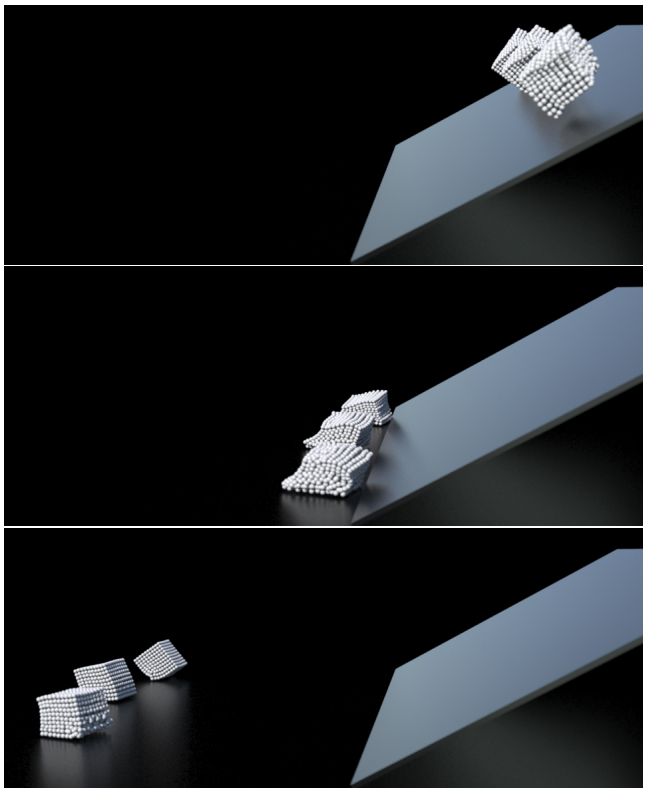
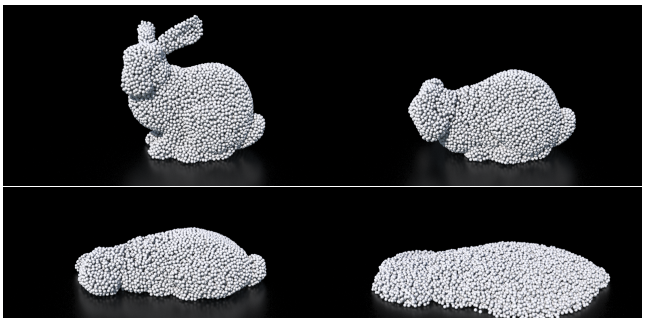


Figure 4: Comparison to Saint Venant-Kirchhoff. The nearest cuboid is simulated with the Saint Venant-Kirchhoff model. The cuboid in the middle is simulated with our method using the same timestep. The farthest cuboid is simulated with our method using a timestep that is five times larger.

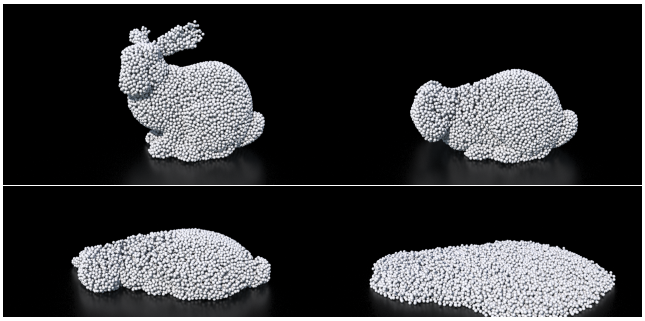
This is further illustrated with the scene in Fig. 3, where two initially slightly rotated cuboids fall onto a plane. Whereas the cuboid with kernel correction is able to rotate freely and to adjust its orientation to match the plane orientation, the cuboid without kernel correction is pulled back to its initial orientation and is hindered from adjusting to the plane orientation.

5.2. Comparison to Saint Venant-Kirchhoff

In the scene shown in Fig. 4, we compare our method to the nonlinear Saint Venant–Kirchhoff model in a more complex scenario. When both simulations run with the same timestep, our method



(a) Our implicit formulation



(b) Becker et al.

Figure 5: Comparison of our implicit formulation to the method of Becker et al. [BIT09]. The overall simulation behavior is comparable. The method of Becker et al. is unstable for high shear and bulk moduli. In this scenario, the performance gain factor of our method compared to Becker et al. is more than one hundred.

(a) Fixed shear modulus and varying bulk modulus

(b) Varying shear modulus and fixed bulk modulus

(c) Matching shear and bulk modulus

Figure 6: Illustration of the influence of bulk and shear modulus.

shows comparable behavior. We also note that our method shows smaller local oscillations compared to Saint Venant-Kirchhoff. When the timestep of our method is increased, the behavior slightly diverges. This is not only due to our linear approximation in time, but also due to the pressure solver and the viscous force, whose behavior is timestep dependent.

5.3. Comparison to Becker et al. [BIT09]

The experiment shown in Fig. 5 consists of a bunny whose shear and bulk modulus are first decreased gradually and then increased again. We did the same experiment replacing our implicit formluation by the method of Becker et al. [BIT09]. The overall behavior of the bunny is comparable for both methods. Nevertheless, although a timestep smaller by a factor of two hundred was chosen, the simulation using the method of Becker et al. is unstable for high shear and bulk moduli. These instabilities could only be removed by increasing viscosity or by reducing the timestep further. Therefore, with the method of Becker et al. we were not able to get a stable simulation while preserving small-scale details. Furthermore, with the given parameters, the method of Becker et al. was 108 times slower compared to our implicit formulation.

5.4. Parameter evaluation

The scenario in Fig. 6 illustrates the influence of shear and bulk modulus. It consists of three parts, and in each part three armadillos with different parameters fall onto a plane. In the first experiment, the shear modulus is equal for all armadillos, while the bulk modulus varies. In the second experiment, the bulk modulus is equal and the shear modulus varies. In the third experiment, shear and bulk modulus are set to the same magnitude for each armadillo, but vary amongst the armadillos. Table 1 summarizes the set-up and the results. The images and timings indicate that matching the bulk modulus to the shear modulus gives convincing results at a fair cost. Therefore, in the following experiments, we will typically set both moduli to match.

5.5. Phase transition

Figure 7 shows a scenario where a bunny dissolves into liquid with shear and bulk modulus set to zero. After that, it is reassembled as shear and bulk modulus are increased. Our implicit formulation is able to cope with large particle displacements that result from the dissolution. This is further illustrated with the scenario in Figure 8, where four armardillos in a box dissolve into fluid and are hereupon reassembled sequentially. Whereas the iteration count in Table 1 reveals that for most other scenarios the pressure solver is

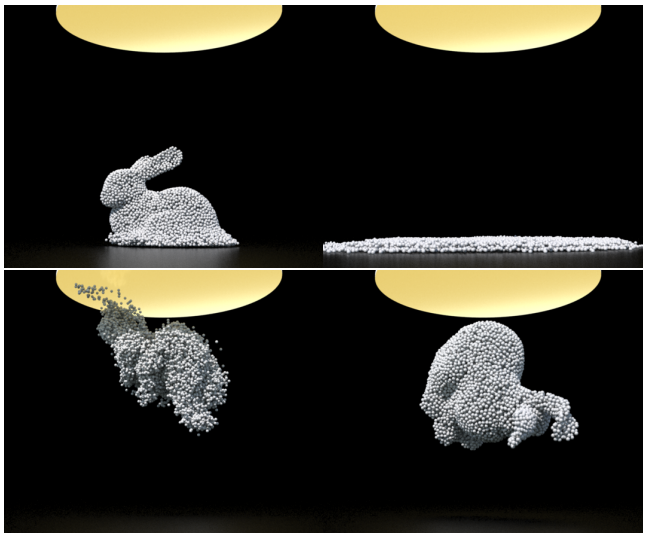


Figure 7: A bunny dissolves into a liquid and is reassembled thereafter.

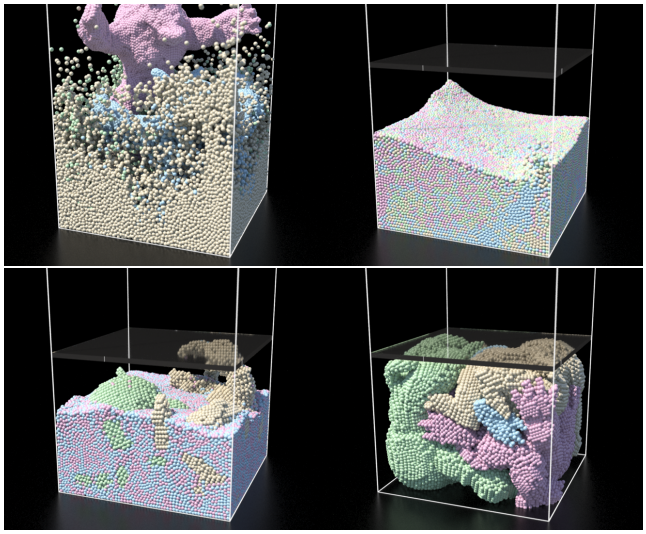


Figure 8: Four armadillos in a box dissolve into liquid and are then reassembled sequentially.

not challenged, it needs an increased number of iterations in the present scenario, as it is responsible for separating the armadillos. This is further aggravated by the fact that the armadillos cannot fully unfold to their rest shape due to the limited space available in the closed box.

5.6. Self-collision

With the scenario in Figure 9, the capability of detecting and resolving self-collisions is illustrated with a cloth-like object falling onto a solid sphere. The object consists of one layer of particles only. To get a stable simulation, an additional zero-energy mode suppression force [Gan15] is applied. As for cloth-like materials,

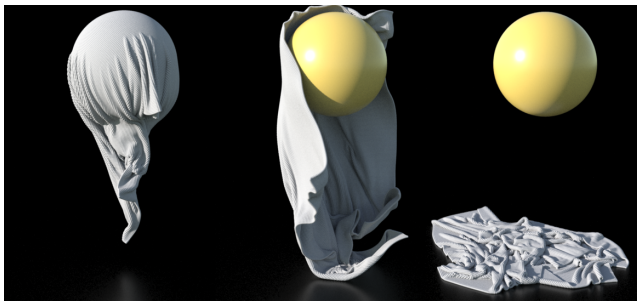


Figure 9: A piece of cloth slides over a sphere and falls to the ground. This employs various self-collision, which are handled by the pressure solver.

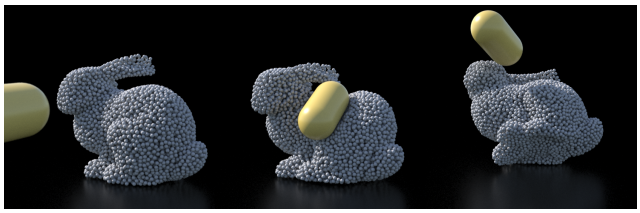


Figure 10: A deformable bunny is shot by a rigid capsule. The interaction is accomplished by the pressure solver.

bending is desired whereas stretching is not, the bulk modulus is set to a higher value than the shear modulus in this scene as stated in Table 1.

5.7. Two-way coupling

The scenario in Figure 10 illustrates two-way coupling of our deformables formulation with solid objects of different densities. A deformable bunny with a density of $1000\,\mathrm{kg/m^3}$ is hit by a fast-moving capsule-shaped object. The scenario is simulated three times with the density of the capsule set to $200\,\mathrm{kg/m^3}$, $2000\,\mathrm{kg/m^3}$ and $20\,000\,\mathrm{kg/m^3}$, respectively. As shown in Table 1, the performance is comparable for all three simulations. Interestingly, in contrast to multiphase simulations, the pressure solver is not challenged in this scenario. This is probably due to the short time span the capsule and the bunny are in direct contact. To prevent the capsule from penetrating the bunny, the bulk modulus is increased compared to the shear modulus in this scenario.

5.8. Multiphase

Figure 11 shows a scenario where deformable objects with different shear and bulk moduli interact. Furthermore, they interact with a water-like phase, where both shear and bulk modulus are set to zero. The water phase has a density of $1000\,\mathrm{kg/m^3}$. The white-colored phase has the moduli set to $G = K = 3 \times 10^4\,\mathrm{Pa}$ and a density of $333\,\mathrm{kg/m^3}$, whereas for the yellow-colored phase, $G = K = 1 \times 10^6\,\mathrm{Pa}$, and the density is $666\,\mathrm{kg/m^3}$. Apart from showing the interaction between the phases, the scenario also illustrates the rotation handling of our approach. In contrast to the

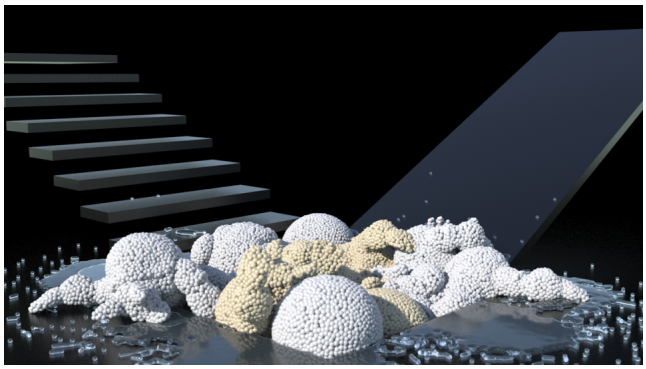


Figure 11: Multiple deformable armadillos, bunnies and spheres slide, roll and fall into a pool of water. The interaction between the different objects, as well as the interaction with the water, is accomplished by the pressure solver.

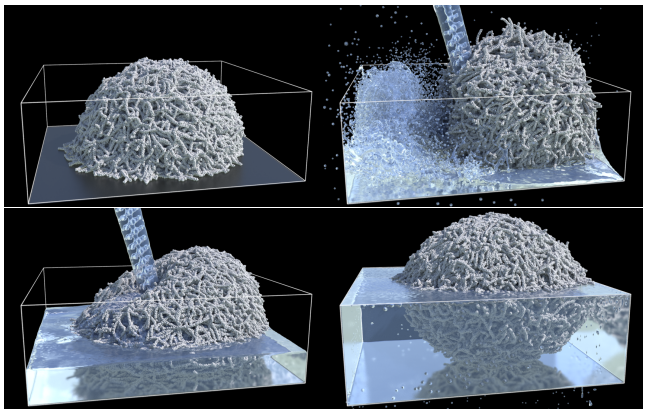


Figure 12: A deformable hairball is flooded with water. The hairball is sampled from a complex geometry that includes channels and cavities. The scenarios consists of up to 5.6M particles.

previous scenarios with one phase, the pressure solver is more challenged, as it has to separate the phases.

5.9. Large scale

The scenario in Fig. 12 consists of a deformable hairball [McG11] with a density of 300 kg/m^3 that is sampled with 1M particles. It interacts with up to 4.6M fluid particles with a density of 1000 kg/m^3. As revealed by Table 1, simulating one frame of this scenario took less than two minutes. Furthermore, the scenario shows the ability to efficiently represent deformable objects with complex geometry.

6. Conclusion

We have presented a novel SPH formulation for elastic solids. In contrast to previous work, kernel-gradient correction is employed to enable the rotation extraction from an SPH deformation gradient. Comparisons with the nonlinear Saint Venant-Kirchhoff model indicate the accuracy of the proposed rotation handling. We have further introduced an implicit formulation for the velocity update that significantly improves the performance compared to [BIT09], a previous SPH formulation for elastic solids. The embedding of our approach into an ISPH fluid solver [ICS*14] elegantly accounts for boundary handling, self-collisions, phase transitions and multi-phase fluids.

Although we have achieved an efficiency gain factor of up to one hundred compared to [BIT09], there are still limiting factors for the timestep that could be addressed. In particular, our implementation employs rather large explicit friction forces at the rigid interface that limit the overall performance. Another limiting factor is the maximum density ratio between different phases. Here, we only work with small ratios of one order of magnitude not to negatively affect the performance.

References

[AAT13] AKINCI N., AKINCI G., TESCHNER M.: Versatile surface tension and adhesion for SPH fluids. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 182:1–182:8. 7

[AIA*12] AKINCI N., IHMSEN M., AKINCI G., SOLENTHALER B., TESCHNER M.: Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans. Graph. 31*, 4 (July 2012), 62:1–62:8. 2, 7

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 594–603. 2

[BGKLPX00] BELYTSCHKO T., GUO Y., KAM LIU W., PING XIAO S.: A unified stability analysis of meshless particle methods. *International Journal for Numerical Methods in Engineering 48*, 9 (2000), 1359–1400. 3

[BIT09] BECKER M., IHMSEN M., TESCHNER M.: Corotated SPH for deformable solids. In *Proceedings of the Fifth Eurographics Conference on Natural Phenomena* (Aire-la-Ville, Switzerland, 2009), NPH'09, Eurographics Association, pp. 27–34. 1, 2, 3, 4, 5, 8, 10

[BK01] BONET J., KULASEGARAM S.: Remarks on tension instability of Eulerian and Lagrangian corrected smooth particle hydrodynamics (CSPH) methods. *International Journal for Numerical Methods in Engineering 52*, 11 (2001), 1203–1220. 3

[BK17] BENDER J., KOSCHIER D.: Divergence-free SPH for incompressible and viscous fluids. *IEEE Transactions on Visualization and Computer Graphics 23*, 3 (Mar. 2017), 1193–1206. 2

[BKCW14] BENDER J., KOSCHIER D., CHARRIER P., WEBER D.: Position-based simulation of continuous materials. *Computers & Graphics 44* (2014), 1–10. 3

[BL99] BONET J., LOK T.-S.: Variational and momentum preservation aspects of smooth particle hydrodynamic formulations. *Computer Methods in Applied Mechanics and Engineering 180*, 1 (1999), 97–115. 3, 4

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. 33*, 4 (July 2014), 154:1–154:11. 3

[BMO*14] BENDER J., MÜLLER M., OTADUY M. A., TESCHNER M., MACKLIN M.: A survey on position-based simulation methods in computer graphics. *Computer Graphics Forum 33*, 6 (Sept. 2014), 228–251. 1, 3

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 43–54. 2

scene		figure	particles	spacing [m]	G [Pa]	K [Pa]	average Δt [ms]	average iterations def.	average iterations press.	average time/frame [s]
comparison	StVK	Fig. 4	1k	0.05	$[1\times10^4]$		1.0	-	1	0.033
	our method						1.0	1	1	0.041
	our method						5.0	3	3.6	0.014
comparison	Becker et al.	Fig. 5	15k	0.04	$[5\times10^2-5\times10^5]$		0.01	–	1	40.1
	our method						2.0	18	1.4	0.37
parameter evaluation		Fig. 6	129k	0.05	2×10^5	2×10^3	1.0	3	1	8.3
						2×10^5		7		
						2×10^7		80		
					2×10^3		1.0	5	1	6.4
					2×10^5	2×10^5		7		
					2×10^7			59		
					2×10^3	2×10^3	1.0	1	1	8.4
					2×10^5	2×10^5		7		
					2×10^7	2×10^7		89		
phase trans.	bunny	Fig. 7	15k	0.04	$[0-5\times10^4]$		2.0	6	1.2	0.22
	armadillos	Fig. 8	123k	0.03	$[0-2\times10^5]$		1.4	10	17	4.3
self-collision		Fig. 9	58k	0.0125	5×10^2	5×10^3	1.0	3	1	0.87
coupling	2×10^2 kg/m^3	Fig. 10	11k	0.05	2×10^5	2×10^6	1.0	13.6	1.0	0.57
	2×10^3 kg/m^3							13.5	1.0	0.58
	2×10^4 kg/m^3							14.1	1.1	0.61
multiphase		Fig. 11	135k + 278k	0.05	$[3\times10^4-1\times10^6]$		2.0	34	6	3.9
large scale		Fig. 12	1M + 4.6M	0.025	$[2\times10^4]$		1.0	43	8	112

Table 1: Scene parameters and performance data of the simulated scenarios. Two readings in the "particles" column indicate the number of deformable and water particles, respectively. Square brackets in the "G" and "K" column indicate matching shear and bulk moduli. The timings given in the last column represent the total time for computing one frame, i.e. the time needed to advance the simulation by 20 ms.

[CBP05] CLAVET S., BEAUDOIN P., POULIN P.: Particle-based viscoelastic fluid simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 219–228. 2

[CJW*09] CLINE D., JESCHKE S., WHITE K., RAZDAN A., WONKA P.: Dart throwing on surfaces. In *Proceedings of the Twentieth Eurographics Conference on Rendering* (Aire-la-Ville, Switzerland, 2009), EGSR'09, Eurographics Association, pp. 1217–1226. 7

[CMM16] CHENTANEZ N., MÜLLER M., MACKLIN M.: Real-time simulation of large elasto-plastic deformation with shape matching. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, 2016), SCA '16, Eurographics Association, pp. 159–167. 3

[CPDH12] CLEARY P. W., PRAKASH M., DAS R., HA J.: Modelling of metal forging using SPH. *Applied Mathematical Modelling 36*, 8 (2012), 3836–3855. 1

[DA12] DEHNEN W., ALY H.: Improving convergence in smoothed particle hydrodynamics simulations without pairing instability. *Monthly Notices of the Royal Astronomical Society 425*, 2 (2012), 1068–1082. 7

[DG96] DESBRUN M., GASCUEL M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '96* (New York, NY, USA, 1996), Springer, pp. 61–76. 2

[DGP12] DAGENAIS F., GAGNON J., PAQUETTE E.: A prediction-correction approach for stable SPH fluid simulation from liquid to rigid. *Proceedings of the Computer Graphics International 2012* (jun 2012). 2

[DGW11] DICK C., GEORGII J., WESTERMANN R.: A hexahedral multigrid approach for simulating cuts in deformable objects. *IEEE Transactions on Visualization and Computer Graphics 17*, 11 (Nov. 2011), 1663–1675. 1

[Gan15] GANZENMÜLLER G. C.: An hourglass control algorithm for Lagrangian smooth particle hydrodynamics. *Computer Methods in Applied Mechanics and Engineering 286* (2015), 87–106. 1, 5, 9

[GBB09] GERSZEWSKI D., BHATTACHARYA H., BARGTEIL A. W.: A point-based method for animating elastoplastic solids. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 133–138. 2

[GBP*17] GISSLER C., BAND S., PEER A., IHMSEN M., TESCHNER M.: Generalized drag force for particle-based simulations. *Computers & Graphics 69* (2017), 1–11. 7

[GHM15] GANZENMÜLLER G. C., HIERMAIER S., MAY M.: On the similarity of meshless discretizations of peridynamics and smooth-particle hydrodynamics. *Computers & Structures 150* (2015), 71–78. 3

[GSMH16] GANZENMÜLLER G. C., SAUER M., MAY M., HIERMAIER

S.: Hourglass control for smooth particle hydrodynamics removes tensile and rank-deficiency instabilities. *The European Physical Journal Special Topics 225*, 2 (2016), 385–395. 1

[HWW17] HE X., WANG H., WU E.: Projective peridynamics for modeling versatile elastoplastic materials. *IEEE Transactions on Visualization and Computer Graphics* (2017). 3

[IABT11] IHMSEN M., AKINCI N., BECKER M., TESCHNER M.: A parallel SPH implementation on multi-core cpus. *Computer Graphics Forum 30*, 1 (2011), 99–112. 7

[ICS*14] IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (Mar. 2014), 426–435. 2, 7, 10

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH fluids in computer graphics. In *Eurographics 2014 - State of the Art Reports* (2014), The Eurographics Association, pp. 21–42. 1, 2

[JGT17] JIANG C., GAST T., TERAN J.: Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Trans. Graph. 36*, 4 (July 2017), 152:1–152:14. 2

[JK09] JEŘÁBKOVÁ L., KUHLEN T.: Stable cutting of deformable objects in virtual environments using XFEM. *IEEE Comput. Graph. Appl. 29*, 2 (Mar. 2009), 61–71. 1

[JWJ*14] JONES B., WARD S., JALLEPALLI A., PERENIA J., BARGTEIL A. W.: Deformation embedding for point-based elastoplastic simulation. *ACM Trans. Graph. 33*, 2 (Apr. 2014), 21:1–21:9. 2

[KAG*05] KEISER R., ADAMS B., GASSER D., BAZZI P., DUTRÉ P., GROSS M.: A unified Lagrangian approach to solid-fluid animation. In *Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics* (Aire-la-Ville, Switzerland, 2005), SPBG'05, Eurographics Association, pp. 125–133. 2

[KBT17] KOSCHIER D., BENDER J., THUEREY N.: Robust eXtended finite elements for complex cutting of deformables. *ACM Trans. Graph. 36*, 4 (July 2017), 55:1–55:13. 1

[KMBG08] KAUFMANN P., MARTIN S., BOTSCH M., GROSS M.: Flexible simulation of deformable models using discontinuous galerkin fem. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, 2008), SCA '08, Eurographics Association, pp. 105–115. 1

[LBK16] LIU T., BOUAZIZ S., KAVAN L.: Towards real-time simulation of hyperelastic materials. *arXiv preprint arXiv:1604.07378* (2016). 3

[LBOK13] LIU T., BARGTEIL A. W., O'BRIEN J. F., KAVAN L.: Fast simulation of mass-spring systems. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 214:1–214:7. 2

[LD08] LENAERTS T., DUTRÉ P.: *Unified SPH model for fluid-shell simulations*. Report CW 530, Departement of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium, November 2008. 2

[LD09] LENAERTS T., DUTRÉ P.: *An architecture for unified SPH simulations*. Report CW 559, Departement Computerwetenschappen, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium, 2009. 1

[LS81] LANCASTER P., SALKAUSKAS K.: Surfaces generated by moving least squares methods. *Mathematics of Computation 37*, 155 (1981), 141–158. 2

[MBCM16] MÜLLER M., BENDER J., CHENTANEZ N., MACKLIN M.: A robust method to extract the rotational part of deformations. In *Proceedings of the 9th International Conference on Motion in Games* (New York, NY, USA, 2016), MIG '16, ACM, pp. 55–60. 4

[MC11] MÜLLER M., CHENTANEZ N.: Solid simulation with oriented particles. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 92:1–92:10. 3

[McG11] MCGUIRE M.: Computer graphics archive, August 2011. URL: http://graphics.cs.williams.edu/data. 10

[MDM*02] MÜLLER M., DORSEY J., MCMILLAN L., JAGNOW R., CUTLER B.: Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2002), SCA '02, ACM, pp. 49–54. 2

[MFZ97] MORRIS J. P., FOX P. J., ZHU Y.: Modeling low reynolds number incompressible flows using SPH. *Journal of Computational Physics 136*, 1 (Sept. 1997), 214–226. 7

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2 (Apr. 2007), 109–118. 3

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 471–478. 2

[MKB*10] MARTIN S., KAUFMANN P., BOTSCH M., GRINSPUN E., GROSS M.: Unified simulation of elastic rods, shells, and solids. In *ACM SIGGRAPH 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 39:1–39:10. 2

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, 2004), SCA '04, Eurographics Association, pp. 141–151. 2

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Trans. Graph. 32*, 4 (July 2013), 104:1–104:12. 3

[MMC16] MACKLIN M., MÜLLER M., CHENTANEZ N.: XPBD: Position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (New York, NY, USA, 2016), MIG '16, ACM, pp. 49–54. 3

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (July 2014), 153:1–153:12. 3

[Mon89] MONAGHAN J. J.: On the problem of penetration in particle methods. *Journal of Computational Physics 82*, 1 (May 1989), 1–15. 7

[Mon92] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics 30*, 1 (1992), 543–574. 1, 2

[Mon12] MONAGHAN J.: Smoothed particle hydrodynamics and its diverse applications. *Annual Review of Fluid Mechanics 44* (2012), 323–346. 1

[MP89] MILLER G., PEARCE A.: Globular dynamics: A connected particle system for animating viscous fluids. *Computers & Graphics 13*, 3 (1989), 305–309. 2

[NMK*06] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4 (2006), 809–836. 3

[NOB16] NARAIN R., OVERBY M., BROWN G. E.: ADMM $\supseteq$ projective dynamics: Fast simulation of general constitutive models. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, 2016), SCA '16, Eurographics Association, pp. 21–28. 3

[PICT15] PEER A., IHMSEN M., CORNELIS J., TESCHNER M.: An implicit viscosity formulation for SPH fluids. *ACM Trans. Graph. 34*, 4 (July 2015), 114:1–114:10. 2

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRÉ P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 957–964. 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 (Dec 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 Trans. Graph. 33*, 5 (Sept. 2014), 171:1–171:11. 2

[SB12a] SCHECHTER H., BRIDSON R.: Ghost SPH for animating water. *ACM Trans. Graph. 31*, 4 (July 2012), 61:1–61:8. 2, 7

[SB12b] SIFAKIS E., BARBIC J.: FEM simulation of 3d deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses* (New York, NY, USA, 2012), SIGGRAPH '12, ACM, pp. 20:1–20:50. 1, 3

[Sid16] SIDE EFFECTS SOFTWARE: Houdini, 2016. URL: http://www.sidefx.com. 7

[Sil00] SILLING S. A.: Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids 48*, 1 (2000), 175–209. 3

[SP08] SOLENTHALER B., PAJAROLA R.: Density contrast SPH interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, 2008), SCA '08, Eurographics Association, pp. 211–218. 2, 7

[SSF13] SU J., SHETH R., FEDKIW R.: Energy conservation for the simulation of deformable bodies. *IEEE Transactions on Visualization and Computer Graphics 19*, 2 (2013), 189–200. 2

[SSJ*14] STOMAKHIN A., SCHROEDER C., JIANG C., CHAI L., TERAN J., SELLE A.: Augmented mpm for phase-change and varied materials. *ACM Trans. Graph. 33*, 4 (July 2014), 138:1–138:11. 2

[SSP07] SOLENTHALER B., SCHLÄFLI J., PAJAROLA R.: A unified particle model for fluid–solid interactions. *Computer Animation and Virtual Worlds 18*, 1 (Feb. 2007), 69–82. 1, 2, 5

[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 (May 2015), 493–502. 2, 5

[TDFN16] TAKAHASHI T., DOBASHI Y., FUJISHIRO I., NISHITA T.: Volume preserving viscoelastic fluids with large deformations using position-based velocity corrections. *The Visual Computer 32*, 1 (2016), 57–66. 3

[THMG04] TESCHNER M., HEIDELBERGER B., MULLER M., GROSS M.: A versatile and robust model for geometrically complex deformable solids. In *Proceedings of the Computer Graphics International* (Washington, DC, USA, 2004), CGI '04, IEEE Computer Society, pp. 312–319. 2

[TL16] TAKAHASHI T., LIN M. C.: A multilevel SPH solver with unified solid boundary handling. In *Proceedings of the 24th Pacific Conference on Computer Graphics and Applications* (Goslar, Germany, 2016), PG '16, Eurographics Association, pp. 517–526. 2

[TPF91] TERZOPOULOS D., PLATT J., FLEISCHER K.: Heating and melting deformable models. *The Journal of Visualization and Computer Animation 2*, 2 (1991), 68–73. 2

[Wen95] WENDLAND H.: Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in computational Mathematics 4*, 1 (1995), 389–396. 7

[WHP*06] WICKE M., HATT P., PAULY M., MÜLLER M., GROSS M.: Versatile Virtual Materials Using Implicit Connectivity. In *Symposium on Point-Based Graphics* (2006), Botsch M., Chen B., Pauly M., Zwicker M., (Eds.), The Eurographics Association. 2

[WKB16] WEILER M., KOSCHIER D., BENDER J.: Projective fluids. In *Proceedings of the 9th International Conference on Motion in Games* (New York, NY, USA, 2016), MIG '16, ACM, pp. 79–84. 3

[ZLKW13] ZHOU Y., LUN Z., KALOGERAKIS E., WANG R.: Implicit integration for particle-based simulation of elasto-plastic solids. *Computer Graphics Forum 32*, 7 (2013), 215–223. 2, 5