# The Reduced Immersed Method for Real-Time Fluid-Elastic Solid Interaction and Contact Simulation

CHRISTOPHER BRANDT, Delft University of Technology, The Netherlands and EPFL, Switzerland
LEONARDO SCANDOLO, ELMAR EISEMANN, and KLAUS HILDEBRANDT, Delft University of Technology, The Netherlands

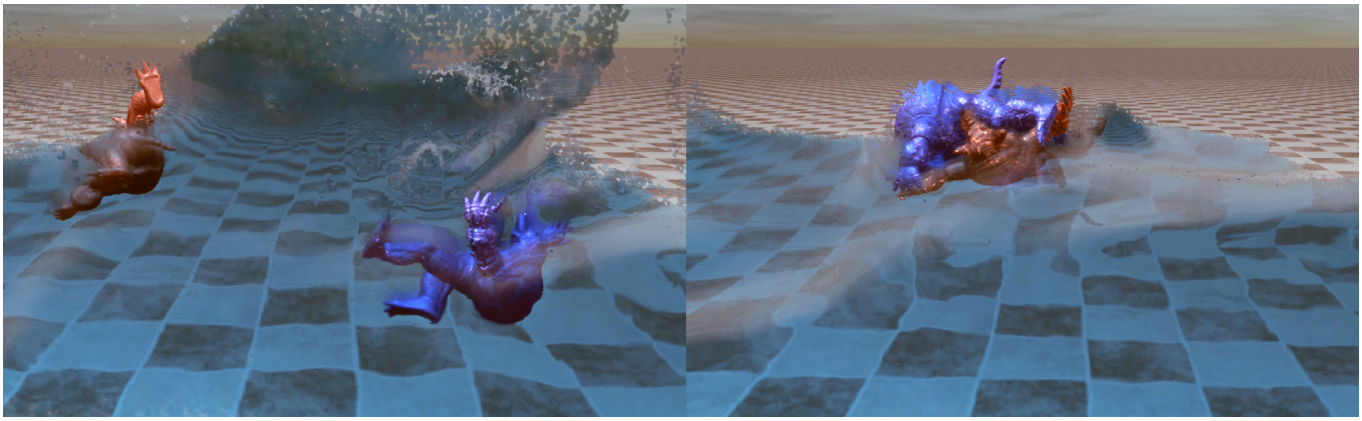


Fig. 1. Two armadillos (274k tetrahedra) in a pool of water (633k particles) simulated at 60 FPS with a time step of 1/60s. Fluid-deformable interaction and (self-)collisions are handled. The user can interact with the scene through click-and-dragging the meshes.

We introduce the Reduced Immersed Method (*RIM*) for the real-time simulation of two-way coupled incompressible fluids and elastic solids and the interaction of multiple deformables with (self-)collisions. Our framework is based on a novel discretization of the *immersed boundary equations of motion*, which model fluid and deformables as a single incompressible medium and their interaction as a unified system on a fixed domain combining Eulerian and Lagrangian terms. One advantage for real-time simulations resulting from this modeling is that two-way coupling phenomena can be faithfully simulated while avoiding costly calculations such as tracking the deforming fluid-solid interfaces and the associated fluid boundary conditions. Our discretization enables the combination of a PIC/FLIP fluid solver with a reduced-order Lagrangian elasticity solver. Crucial for the performance of RIM is the efficient transfer of information between the elasticity and the fluid solver and the synchronization of the Lagrangian and Eulerian settings. We introduce the concept of *twin subspaces* that enables an efficient reduced-order modeling of the transfer. Our experiments demonstrate that RIM handles complex meshes and highly resolved fluids for large time steps at high framerates on off-the-shelf hardware, even in the presence of high velocities and rapid user interaction. Furthermore, it extends reduced-order elasticity solvers such as Hyper-Reduced Projective Dynamics with natural collision handling.

Authors' addresses: Christopher Brandt, c.brandt@tudelft.nl, Delft University of Technology, Department of Intelligent Systems, Van Mourik Broekmanweg 6, Delft, 2628 XE, The Netherlands, EPFL, Computer Graphics and Geometry Laboratory, BC 344, Station 14, Lausanne, 1015, Switzerland; Leonardo Scandolo, l.scandolo@tudelft.nl; Elmar Eisemann, e.eisemann@tudelft.nl; Klaus Hildebrandt, k.a.hildebrandt@tudelft.nl, Delft University of Technology, Department of Intelligent Systems, Van Mourik Broekmanweg 6, Delft, 2628 XE, The Netherlands.



CCS Concepts: • **Computing methodologies** → **Real-time simulation**; **Physical simulation**.

Additional Key Words and Phrases: Real-time simulation, model reduction, subspace dynamics, solid-fluid coupling, immersed boundary method, projective dynamics, elastic solids, fluid simulation

**ACM Reference Format:**
Christopher Brandt, Leonardo Scandolo, Elmar Eisemann, and Klaus Hildebrandt. 2019. The Reduced Immersed Method for Real-Time Fluid-Elastic Solid Interaction and Contact Simulation. *ACM Trans. Graph.* 38, 6, Article 191 (November 2019), 16 pages. https://doi.org/10.1145/3355089.3356496

## 1 INTRODUCTION

The physical simulation of deformable solids and fluids in *real-time* is highly desirable for various applications of computer graphics as it enables interaction with the simulation and thereby largely enhances the users' experience in games, artistic and design applications, virtual reality, and medical training. The demands on the efficiency and robustness of such simulations pose challenging problems that require specialized computational methods. Our goal is to develop techniques targeting real-time simulations of deformable-fluid and deformable-deformable interactions involving complex geometry.

A challenge in the real-time simulation of fluid-deformable interaction is posed by the complexity of the transfer of forces between fluid and deformable. Both media have highly deforming common boundaries, on which non-linear forces are evaluated to compute the two-way interaction between them. The tracking of the interface and associated boundary conditions is prohibitive for the real-time simulation of complex deformables and highly resolved fluids, and

the transfer of forces is expensive unless the deformable and the fluid are discretized in the same manner (e.g. both as meshes or both as particle soups), such that one of them is not available in its most advantageous representation. Thus, while there are highly efficient solvers for the real-time simulation of deformables *or* fluids, it is difficult to combine them in real-time scenarios, where at most a few milliseconds can be spent on connecting the individual solvers.

As a result, recent approaches for real-time two-way interaction are limited to low resolutions or rely on simplified interaction models. In contrast, our goal is to enable the real-time simulation of complex deformables, discretized as highly resolved meshes, interacting with fluids in a physically sound manner (i.e. through the discretization of an established continuous system).

To this end, we recognize the value of *immersed methods* for the real-time simulation of fluid-deformable interaction. Our novel simulation framework, the Reduced Immersed Method (*RIM*), is based on a novel discretization of the *immersed boundary equations of motion* [Peskin 2002]. Therein, deformables and fluid are modeled as a continuous medium on a fixed Eulerian grid and additional elastic forces act on the sub-domain occupied by the deformable. This will allow us to skip the costly evaluation of the interface between the deformable and the fluid and the computation of boundary forces thereon. Our method is based on the *immersed boundary equations of motion* but the discretization and numerical schemes we use and our application focus differ substantially from the *Immersed Boundary Method* [Peskin 2002], its variants, and applications.

Our novel discretization will allow us to combine two highly efficient solvers for fluid and deformable simulation, to achieve unmatched performance for coupling simulations of massive scale (see e.g. Figure 1). Fluid and deformable are integrated in time using a *PIC/FLIP* [Zhu and Bridson 2005] variant, acting on a particle representation of the fluid *and* the deformable. The additional inner forces, acting only on the deformable, are computed using a reduced elasticity solver, in our implementation we use the *Hyper-Reduced Projective Dynamics* (*HRPD*) framework [Brandt et al. 2018]. These forces are transported to the Eulerian framework according to smooth approximations of Dirac delta functions. This will require us to transfer quantities computed on a mesh to the particle-based fluid solver, *and vice versa*, since the mesh needs to be synchronized to the updated particle state of the fluid solver after advection. We approach this challenge through the novel concept of *twin subspaces*: by constructing two subspaces, one for the particle-based and one for the mesh-based representation of the deformable, which are in direct correspondence, we are able to achieve the transport of forces and the synchronization of the solvers efficiently through the transfer of subspace coordinates and subspace projections. Since our method is based on a careful discretization of the continuous system, we are able to faithfully replicate complex two-way interaction and the associated phenomena such as buoyancy. In a direct comparison, we show that our simulations closely match those computed by an offline approach that takes more than 400 times longer to compute a time-step.

Despite the complexity of the continuous system and its discretization, the resulting framework can be described through a concise algorithm (see Algorithm 1). This is due to the fact that we embed the transfer of Lagrangian elastic forces to the Eulerian grid into PIC/FLIP's particle-to-grid transfer, in a way that aligns with the input/output structure of both solvers. This simplicity allows for full reproducibility and source code will be made available.

Fig. 2. Interactive simulations of complex elastic deformables with contact in real-time. Collisions and self-intersections are naturally handled.

Another challenging problem is the simulation of the interaction between multiple elastic defomables in real-time, since detecting and naturally handling collisions between highly resolved meshes at large time-steps requires complex computational algorithms. Through the combination of incompressibility being globally enforced on a continuos velocity field and an additional penetration prevention step, RIM also prevents (self-)collisions between incompressible elastic deformables and resolves them in a physically-based manner. In several examples, we demonstrate the capability of our method to robustly simulate real-time interaction between complex meshes subjected to gravitation and rapid user interaction.

In summary, our salient contributions are:

- We present the first immersed method for real-time fluid-deformable interaction.
- We present a novel discretization of the immersed boundary equations of motion that combines a reduced, mesh-based elasticity solver with a PIC/FLIP solver.
- The transfer of information between the two solvers and their synchronization is efficiently handled through the novel concept of twin subspaces.
- The resulting method enables the real-time simulation of fluid-deformable interaction with highly resolved meshes and fluids in unmatched performance.
- We extend the reduced-order Lagrangian elasticity method Hyper-Reduced Projective Dynamics with natural collision handling and true incompressibility.

## 2 RELATED WORK

*Real-time simulation of elastic deformables.* The simulation of elastic deformable bodies has a long history in the field of computer graphics. Our focus will be on approaches that are capable of real-time simulations, for an introduction to the general topic, we refer to the courses [Bargteil and Shinar 2018; Sifakis and Barbič 2012]. For the real-time simulation of meshes with limited resolution, *Position-based dynamics* (PBD) models of elasticity [Bender et al. 2014; Müller et al. 2014; Müller et al. 2005] have been employed successfully. *Projective Dynamics* [Bouaziz et al. 2014] enables an efficient implicit time integration by a combination of a local-gloal optimization scheme with an optimization integrator [Gast et al. 2015; Li et al. 2018] and specific choice of an elastic material model. Projective dynamics has been extended to more general materials [Liu et al. 2017; Overby et al. 2017] and schemes for accelerating the convergence of the optimization integrator have been studied [Dinev et al. 2018; Peng et al. 2018]. Gradient descent methods based on Chebyshev acceleration allow for massive parallelization [Wang 2015; Wang and Yang 2016] on the GPU. Still, the real-time simulation of deformables is limited to coarse spatial and temporal resolution.

Model reduction techniques can be used to design fast approximation algorithms for the simulation of deformables. A reduced system is constructed in an offline stage such that in the online stage the cost for integrating the reduced system is independent of the complexity mesh and its finite element space. For linear systems, a model reduction can be achieved by projecting the dynamics to a subspace of the finite element space [Hauser et al. 2003]. For non-linear systems, a second layer of reduction, a hyper-reduction, that enables the fast approximation of the non-linear forces is required. For this purpose *polynomial reduction* [Barbič and James 2005], *optimized cubature* [An et al. 2008; von Tycowicz et al. 2013; Yang et al. 2015], and a *fitting method* [Brandt et al. 2018] have been proposed.

*Reduced solid-solid interaction.* The detection of contact of reduced deformables can be accelerated using a specialized *bounding volume hierarchy* [James and Pai 2004]. For fast detection of self-collision, culling methods based on *collision certificates* [Barbič and James 2010] and *bounded-normal trees* [Schvartzman et al. 2009] have been proposed. These works focus only on the detection of collision. Accelerated models for contact between two reduced deformables [Barbič and James 2008], reduced deformables and static objects [Harmon and Zorin 2013], and self-contact of articulated deformables [Teng et al. 2014] have been proposed. Our approach addresses solid-solid contact and self-contact scenarios in one framework that is based on two principles. Firstly, the combination of global incompressibility and elastic forces is a natural physical model for contact. This type of contact model was used in [Peer et al. 2018; Teng et al. 2016]. Secondly, the evolution of a shape in continuous, single-valued velocity fields precludes interpenetration [Sulsky et al. 1995]. Single-valued divergence-free velocity fields were used for a collision-free shape editing approach [von Funck et al. 2006]. We will not discuss offline contact simulation approaches here, but note that the approaches in [Fan et al. 2013; Jiang et al. 2017; Levin et al. 2011] relate to the way in which we handle contact on a Eulerian grid.

*Real-time simulation of fluids.* When restricting to coarse grid resolutions and low particle counts, many fluid simulation techniques can be used for real-time simulation. For example, GPU implementations of the PIC/FLIP algorithm [Zhu and Bridson 2005] are capable of real-time rates on modern hardware. Likewise, Smoothed Particle Hydrodynamics (SPH) [Monaghan 1992] methods can be elevated to interactive rates, as demonstrated in [Müller et al. 2003]. More recent approaches employ hybrid grids and specialized multi-grid algorithms [Chentanez and Müller 2011] to achieve real-time rates on large scale liquids or adapt the FLIP method to a sparse hierarchy of grids to allow for the simulation of tens of millions of particles [Wu et al. 2018]. Our method employs a modified GPU implementation of the PIC/FLIP algorithm.

*Real-time fluid-deformable interaction.* While efficient methods for the real-time simulation of either deformables *or* fluids have been introduced, the real-time simulation of the interaction between deformables and fluids still poses a great challenge. As such, efficient simulation frameworks that enable the computation of a time step in complex fluid-deformable coupling simulations in real-time are limited. [Yang et al. 2012] combine incompressible SPH and an explicit FEM solver for elasticity through a CUDA implementation and handle the computation and distribution of coupling forces through proxy particles. The explicit elasticity computations, the complex computation of coupling forces at common interfaces and the need to enforce non-penetrations as an additional subsequent step, heavily limit the step-size, such that simulations run at about one twentieth of real-time speed and at low frame-rates for high resolutions. To improve the performance of the computation of coupling forces, other real-time methods represent both fluid and deformable through particles. PBD models for elasticity can be combined with a PBD model of fluids [Macklin and Müller 2013] in a unified frame work [Macklin et al. 2014]. While this approach is fast, drawbacks, such as difficulties in handling soft constraints and a coupling of material stiffness to the time step and iteration count, are reported [Weiler et al. 2016]. A related elasticity model is shape matching [Müller et al. 2005], which has been coupled to incompressible SPH [Shao et al. 2015] and a PIC/FLIP solver in [Gao et al. 2018a]. We discuss our method in relation to [Macklin et al. 2014] and [Gao et al. 2018a] in Section 6.1

*Offline fluid-deformable interaction.* Traditional approaches for the simulation of fluids interacting with solids are not focused on real-time performance and computation timings for one time step range from seconds to hours. Typically, a Lagrangian elastic solid and a fluid simulation on an Eulerian grid are combined.

*Weak*, or *partitioned*, coupling methods [Degroote 2013; Génevaux et al. 2003; Guendelman et al. 2005; Hou et al. 2012], solve the fluid and solid simulation separately and use interface conditions at the common boundary to pass information from one to the other. Their advantage is the ability to use two separate specialized discretizations and solvers for fluid and elasticity computations. Since the boundary conditions cannot be known in advance, partitioned approaches require multiple iterations between the fluid and elasticity solvers to compute a single time-step. [Akbay et al. 2018] supplement arbitrary partitioned methods with a reduced-order monolithic

system and an impulse based interaction term to significantly reduce the number of iterations between solvers before convergence.

*Monolithic* schemes, such as [Chentanez et al. 2006; Robinson-Mosher et al. 2011, 2008], integrate the solid and fluid dynamics to a single system that includes the interface conditions. Challenges arise in ensuring that the resulting system is positive-definite and symmetric.

Both approaches (weak and monolithic) need to represent the boundary of the deforming solid on the Eulerian grid. Accuracy of interfaces can be increased by adaptive refinement [Losasso et al. 2004] or by inpainting boundary-conforming tetrahedral mesh [Chentanez et al. 2007]. In cut-cell methods [Batty et al. 2007; Zarifi and Batty 2017] the grid cells are clipped against the solid's geometry and interface conditions are enforced at the clipped cells. We evaluate our method by comparing simulations using our method to simulations using [Zarifi and Batty 2017] in Section 6.2. A method for monolithic solid-fluid interaction using reduced deformable objects has been introduced in [Lu et al. 2016]. Even when efficient solvers are combined, the handling of the boundary and the computation of coupling forces enable interactive rates only for very coarse resolutions. Additionally, time steps are usually very small to prevent stability issues related to the fluid simulation, the elasticity simulation or solid-solid and fluid-solid penetration.

For the interaction between rigid solids and fluid, Carlson, Mucha and Turk [2004] propose solving for a unified velocity field, on which they enforce incompressibility and rigidity of the solids via projections, which in turn approximate the interaction behavior.

Mesh-based purely Lagrangian methods, such as [Clausen et al. 2013; Misztal et al. 2014], profit from a unified representation of fluid and solids and sharply resolve the interfaces. However, the mesh constantly needs to be restructured, which limits real-time applicability. *Smoothed Particle Hydrodynamics* (SPH) [Monaghan 1992] can be used for fluid simulations [Becker and Teschner 2007; Ihmsen et al. 2014; Müller et al. 2003; Solenthaler and Pajarola 2009], which results in a meshless Lagrangian method.

The SPH model can be used for both fluid *and* deformables, as proposed in [Peer et al. 2018]. This requires an SPH formulation for deformable solids. Since the SPH kernel gradient is not first-order consistent, rotation extraction requires a correction of the kernel. Real-time simulation in this framework is limited to coarse resolutions and the deformables are represented as particles.

Recently, the Material Point Method [Sulsky et al. 1995], which is a meshless Lagrangian/Eulerian hybrid method has found attention in various computer graphics applications, including fluid-solid coupling [Hu et al. 2018; Jiang et al. 2015; Tampubolon et al. 2017]. As in our method, efficient Lagrangian to Eulerian transfers are essential to MPM approaches, and a GPU optimization has been explored in [Gao et al. 2018b].

*Immersed methods.* An alternative Lagrangian-Eulerian approach to solid-fluid interaction are immersed methods, originating from the immersed boundary method [Peskin 2002], which was used, for example, to study flow patterns around heart valves. The basic idea of the immersed boundary method is to simulate fluid and incompressible deformables on a common, fixed Eulerian grid, through a unified system of equations of motion that contains both Eulerian and Lagrangian terms. The deformable's elasticity is modeled by an additional force term, which is computed in a Lagrangian solid simulation, and a transfer of these forces to the Eulerian grid using Dirac delta functions. Several variants of the immersed boundary method have been proposed, such as the immersed finite elements method [Zhang et al. 2004; Zhang and Gay 2007], the immersogeometric method [Kamensky et al. 2015], or approaches aiming at better volume preservation [Bao et al. 2017]. A benefit of immersed methods is that no additional interaction forces need to be imposed on the time-varying fluid-solid interface, as they are implicitly handled through the underlying unified equations of motion.

We recognize the benefits of immersed methods for the real-time simulation of deformable-fluid coupling simulations in computer graphics and propose a novel discretization of the immersed boundary equations of motion from [Peskin 2002]. In contrast to other immersed methods, we make use of an additional particle representation of the deformable and use the PIC/FLIP interpolation formulas to transfer elastic forces and velocities between the Lagrangian and Eulerian settings. Moreover, we use a *reduced-order* Lagrangian simulation and introduce the twin subspaces, which enable information transfer of elastic forces into the Eulerian grid, as well as the synchronization of the Lagrangian and Eulerian settings after advection at low computational cost. To the best of our knowledge this is the first immersed method making use of a reduced elasticity solver.

## 3 THE REDUCED IMMERSED METHOD

In the following we will describe our simulation framework *RIM*. In Section 3.1 we introduce the continuous system underlying our simulation framework. A novel discretization of this system is proposed in Section 3.2, which combines a mesh-based, reduced, Lagrangian elasticity solver (Section 3.2.1) with a variant of the PIC/FLIP method (Section 3.2.2). The joint simulation and the algorithm for *RIM* are described in Section 3.2.3. In our method, fluid-deformable coupling is realized through the concept of twin subspaces, described in Section 3.2.4, which enables the highly efficient transfer of forces between the two solvers. The method is stabilized through an additional layer of intersection prevention, detailed in Section 3.2.5. The simulation of multiple deformables in contact, i.e. the handling of collisions and self-collisions, is discussed in Section 4.

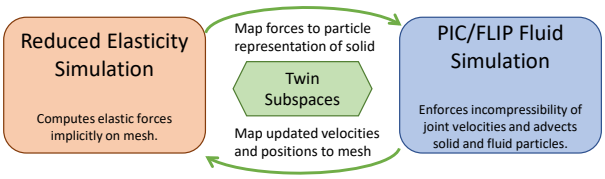



Fig. 3. Overview of our algorithmic pipeline.

In Figure 3 we provide a simplified overview for the final algorithm, where a reduced elasticity solver and a PIC/FLIP fluid solver are connected through the twin subspaces to solve the immersed

boundary equations of motion. The final algorithm is listed in Algorithm 1. The key algorithmic aspects for the performance of this method are:

- Elastic forces are evaluated implicitly, using a mesh, with displacements restricted to a linear subspace.
- The elastic forces are transported to solid particles (as velocity updates), using a second subspace on the solid particles, that is in correspondence to the former.
- Incompressibility of the joint (fluid and deformable) velocity field is enforced on the grid by solving a pressure projection. This is done using a limited number of Jacobi iterations to find a good trade-off between accuracy and performance (see Appendix C).
- Updated positions and velocities of the deformable are transported back to the mesh, again making use of the twin subspaces.
- Velocities are corrected to prevent interpenetration between solids and fluid, which can be done efficiently using updated approximated particle normals, acquired by using the specific structure of the *Hyper-Reduced Projective Dynamics* [Brandt et al. 2018] subspace.

This fairly simple algorithm will be shown to approximate our continuous model presented in the next sections.

### 3.1 Continuous model

Our framework is based on the immersed boundary equations of motion [Peskin 2002] for fluid-solid interaction. The equations model the union of the fluid and the elastic solids as a single incompressible medium on a fixed domain. Elastic forces, acting only on the deformables, are transported from a Lagrangian to the Eulerian setting via Dirac delta distributions. The fluid and deformables interact on a bounded domain $\Omega \subset \mathbb{R}^3$. We assume that the deformables occupy a subset $\Omega_S \subset \Omega$ at time 0, which is their initial, undeformed configuration. The initial densities of fluid and solid are given by a scalar function $\rho_0$ on $\Omega$. The internal forces acting on the deformables are given by $F_{\text{int}}(y,t)$ for $y \in \Omega_S$, where $y$ parameterizes $\Omega_S$. The motion of fluid and deformables will be described by a Eulerian velocity field $v$, which is governed by

$$\rho(x,t)\left(\frac{\partial v(x,t)}{\partial t} + \nabla v(x,t)\cdot v(x,t)\right) + \nabla p(x,t) = \mathbf{f}_{\text{int}}(x,t) + \mathbf{f}_{\text{ext}}(x,t) \tag{1}$$

$$\nabla \cdot v(x,t) = 0 \tag{2}$$

$$\rho(x,t) = \int_\Omega \rho_0(y)\ \delta\left(x - \varphi(y,t)\right)\ \mathrm{d}y \tag{3}$$

$$\mathbf{f}_{\text{int}}(x,t) = \int_{\Omega_S} F_{\text{int}}(y,t)\ \delta\left(x - \varphi(y,t)\right)\ \mathrm{d}y \tag{4}$$

$$\frac{\partial \varphi(x,t)}{\partial t} = v\left(\varphi(x,t),t\right) \tag{5}$$

$$\text{For } x \in \partial\Omega\ :\ v(x,t)\cdot n(x) = 0 \tag{6}$$

Here, $p$ is the unknown pressure enforcing incompressibility and $\mathbf{f}_{\text{ext}}$ are external, Eulerian body forces. For the boundary conditions (6), $n(x)$ is the outer normal to $\Omega$ at $x \in \partial\Omega$. Initial velocities are given as $v(\cdot,0) = v_0$. Densities and elastic forces are treated as Lagrangian terms and transported to the Eulerian system by convolution integrals over Dirac delta distributions, (3) and (4), whose centers move with the flow map $\varphi$. Specifics about the elastic forces used in our simulations are delayed until Section 3.2.1, where they are stated in the discretized setting.

This system of equations unifies the dynamics of incompressible elastic deformables immersed in an incompressible fluid. It is equivalent to other common formulations for fluid-solid interaction, but formulates the jump in the fluid stress across the common interfaces implicitly, instead of posing explicit boundary conditions. In fact, the difficulty of tracking the common interface and solving for unknown boundary conditions is replaced by the transfer between the Lagrangian and Eulerian setting. The integrals (3) and (4) need to be evaluated to keep track of densities and elastic forces within the Eulerian setting, which requires the stable computation of the flowmap (5). Also, to compute $F_{\text{int}}$, a Lagrangian representation of the deformable which is synchronized to the Eulerian state has to be available. We propose a novel discretization of this system which makes use of modern and efficient solvers ([Brandt et al. 2018; Zhu and Bridson 2005]) and introduces a reduced-order method to compute the transfers and synchronization between the Lagrangian and Eulerian settings.

The immersed boundary equations of motion pose two limitations. Firstly, the system implicitly imposes no-slip boundary conditions between different deformables or fluid and deformables, as the velocity field $v$ will always be continuous. In practice, the effect of this restriction is reduced by our specific discretization and we observe relative tangential motion for both fluid-deformable and deformable-deformable interaction. Secondly, all deformables are assumed to be incompressible, which enables the omission of explicit boundary conditions to model fluid-deformable interaction.

### 3.2 Discretization

We approach our discretization by employing the common approach of operator splitting to approximate the equations of motion (1) - (6). We solve for velocities $v$ and pressure $p$ by splitting the equations of motion into an *elasticity step* (which will include the external body forces acting on the deformable),

$$\frac{\partial v(x,t)}{\partial t} = \frac{1}{\rho(x,t)}\left(\mathbf{f}_{\text{int}}(x,t) + \mathbf{f}_{\text{ext}}(x,t)\right) \text{ for } x \in \varphi(\Omega_S,t), \tag{7}$$

a step to apply external forces on the fluid,

$$\frac{\partial v(x,t)}{\partial t} = \frac{1}{\rho(x,t)}\mathbf{f}_{\text{ext}}(x,t) \text{ for } x \in \varphi(\Omega\backslash\Omega_S,t), \tag{8}$$

a *projection step* to enforce incompressibility on both deformables and fluid,

$$\frac{\partial v(x,t)}{\partial t} = -\frac{1}{\rho(x,t)}\nabla p(x,t) \tag{9}$$

$$\nabla \cdot v(x,t) = 0 \tag{10}$$

and an *advection step*

$$\frac{\partial v(x,t)}{\partial t} + \nabla v(x,t)\cdot v(x,t) = 0. \tag{11}$$

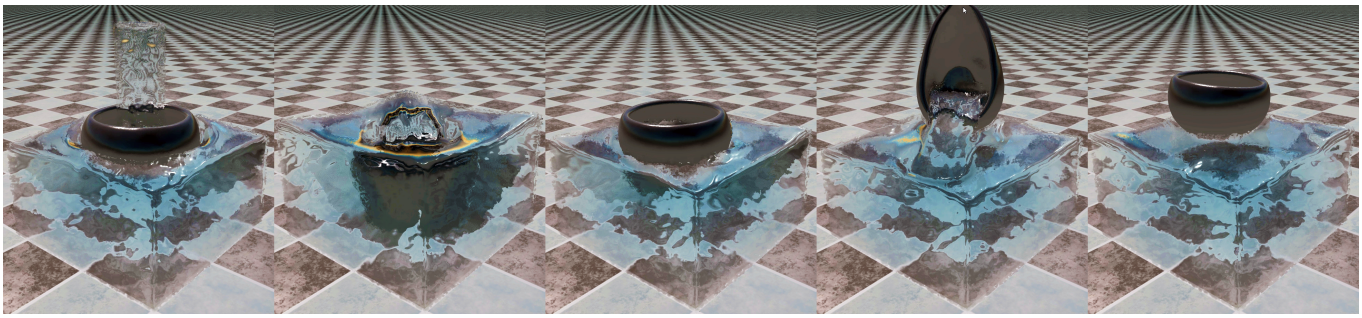


Fig. 4. A stream of water is poured into an elastic cup in a pool. Through user interaction, the liquid is then poured from the cup into the pool.

This splitting allows us to use the most efficient representation for each step individually, and thus we choose mesh and particle representations of the deformable to discretize (7), a regular grid to discretize (9) and particles to discretize (8) and (11). Note that the deformable is present in all three discretizations.

The steps of applying external forces on the fluid (8), projection (9), and advection (11) are approximated by the PIC/FLIP variant described in Section 3.2.2. After these steps, positions and velocities of the mesh representation of the deformable are updated according to the changes of particle and grid representations.

In the *elasticity step*, we first compute Lagrangian elastic forces $F_{\text{int}}$ using the mesh-based, implicit approach, which is described in Section 3.2.1. We then compute intermediate velocities according to the elastic and external forces on the Lagrangian mesh. These velocities are transferred to the particles representing the deformable in the PIC/FLIP algorithm, where they are used as input to compute projection and advection. This discretizes the update of Eulerian velocities, (7), including the transfer of forces. Hence, the evaluation of the integrals (4) is delayed, and embedded into the PIC/FLIP algorithm's particle-to-grid transfer, such that it comes at no additional computational cost. As an additional advantage, the velocity update is performed on the particle/mesh representations, where the boundary between deformable and fluid is sharply resolved, which is not the case on the Eulerian grid.

The update of the Lagrangian mesh, as well as the transfer of the updated velocities from the mesh to the particle representation are efficiently realized through the concept of twin subspaces, which will be described in Sections 3.2.3 and 3.2.4.

*3.2.1 Lagrangian elasticity computations.* In this step, our task is to update the velocities of the deformable, disregarding incompressibility and fluid interaction. To discretize this step, the undeformed solid occupying the domain $\Omega_S$ is represented by a tetrahedral mesh with vertex positions $X_0^m \in \mathbb{R}^{3n}$. Each vertex receives an associated mass equal to the integrated rest density $\rho_0$ over an area given by the Voronoi cell of that vertex. The associated masses are collected in a diagonal matrix $M$. The motion of the deformable is described by time-dependent vertex positions $X^m(t)$. The elastic energy associated to deformed vertex positions $X^m$ is discretized as a volume weighted sum $E(X^m) = \sum \text{Vol}_j W_{\text{el}}^j(X^m)$, where the energy density $W_{\text{el}}^j$ is constant at each tetrahedron. In Projective Dynamics [Bouaziz et al. 2014], the $W_{\text{el}}^j$ that are used to model an elastic, volumetric deformable, are given by

$$W_{\text{el}}^j(X^m) = \lambda_j \left\| S_j X - p_j(X^m) \right\|^2$$

where $\lambda_j > 0$ is the stiffness of the material, $S_j$ is the matrix which computes the deformation gradient of the $j$th tetrahedron from the vector $X$ of vertex positions, and the *constraint projections* $p_j$ compute the rotational parts of the deformation gradients. The discrete Lagrangian internal forces are then given by $F_{\text{int}}(X^m) = -\partial E(X^m)/\partial X^m = -\sum_j \nabla W_{\text{el}}^j(X^m)$.

With this, the spatially-discrete version of (7) can be rewritten using Lagrangian coordinates as

$$M\ddot{X}^m = F(X^m) + F_{\text{ext}} \tag{12}$$

To compute the updated velocities, we will compute a discrete time step of (12) and infer updated velocities as $(X^m(t+h) - X^m(t))/h$. To compute this time step at reduced cost, we will use the reduced-order, local-global optimization-based time stepping scheme proposed in Hyper-Reduced Projective Dynamics [Brandt et al. 2018]. This approach makes use of reduced coordinates $X^m \approx \mathbf{U}^m \tilde{X}$, for a linear, $d$-dimensional subspace $\mathbf{U}^m$, and approximates the nonlinear part of the dynamics from only a small fraction of evaluated constraint projections $p_j$. We will not describe Projective Dynamics or the model reduction approach in more detail here, as we use it as a black-box that takes as input subspace coordinates of the current vertex positions $\tilde{X}(t)$, subspace coordinates of the per-vertex velocities $\tilde{V}(t)$ and outputs $\tilde{V}(t+h)$. Note, that the input positions $\tilde{X}(t)$ and velocities $\tilde{V}(t)$ have to be such that they reflect the state of the Eulerian system on which projection and advection are computed.

Any method for the simulation of elastic deformables, that is constrained to a linear subspace, can be used in place of the Hyper-Reduced Projective Dynamics framework to compute the intermediate velocities of the elasticity step. However, we benefit from the stability that is provided by the optimization-based implicit time stepping in combination with the regularization through the subspaces.

*3.2.2 Projection and advection.* To compute projection and advection, fluid, deformable and their associated quantities (velocity, density and pressure) are discretized as particles on a staggered MAC-grid. In particular, in addition to its mesh representation, the

deformable will be represented through particles, which are initialized as the intersection of a regular grid with the volumetric mesh. Equations (8)-(11) are discretized and solved using the PIC/FLIP algorithm from [Zhu and Bridson 2005].

The algorithm was modified to take into account the immersed incompressible deformable. Specifically, the particles representing the deformable are flagged as *solid* and their velocities are modified according to the elastic forces computed in the *elasticity step*. This transfer is detailed in the following section. Furthermore, we need to account for varying densities of fluid and deformable, which requires an adapted pressure projection.

Various small additional modifications were made to increase performance and to prevent volume loss for large velocities and time steps. The modiciations to the original PIC/FLIP algorithm are detailed in Appendix A and the full modified algorithm is listed there, as Algorithm 2.

*3.2.3 Joint simulation.* To finalize the *RIM* algorithm, we need to detail the transfers of the Lagrangian quantities and the update of the Lagrangian setting according to the Eulerian velocity field. As described above, this amounts to the following steps in our discretization:

- The velocities acquired from the mesh-based elasticity step are transferred to the solid particles, where they replace the velocities attached to the particles from the last time step.
- After projection and advection, the positions and velocities of the solid particles need to be transferred to the mesh as input for the elasticity solve. This last step is critical, since both solvers need to be synchronized in order for the simulation to remain stable over time.

We make use of the restriction of the elasticity solver to a linear subspace $\mathbf{U}^m$ by realizing the operations above via a *twin subspace*, that relates quantities of the deformables in their mesh representation to quantities for the particle representation. That is, we will create a second subspace $\mathbf{U}^p$, such that, for subspace coordinates $\tilde{X}$ encoding a deformation of the solid, the mesh $X^m = \mathbf{U}^m\tilde{X}$ and the solid particles $X^p = \mathbf{U}^p\tilde{X}$ are in correspondence. Moreover, the same subspace can be used to transfer per-vertex velocities $V^m = \mathbf{U}^m\tilde{V}$ to per particle velocities $V^p = \mathbf{U}^p\tilde{V}$.

With this construction at hand, after we performed the elasticity step, we can simply transfer the velocities to the solid particles in the fluid solver, by passing the small, $d$-dimensional vector $\tilde{V}$ and computing the sparse matrix vector product $\mathbf{U}^p\tilde{V}$ (the subspaces used in HRPD are sparse). Then, we use Algorithm 2 to acquire updated solid particle positions $X_S^p$ and velocities $V_S^p$.

Thereafter, we need to transfer these quantities to the mesh representation for the next elasticity step. This transfer is also realized through the use of the twin subspaces, namely by projecting the solid particle positions to the subspace

$$\mathbf{U}^{p\,T}\mathbf{U}^p\tilde{X}^* = \mathbf{U}^{p\,T}X_S^p. \tag{13}$$

and then inferring the mesh velocities as $\tilde{V}^* = \left(\tilde{X}^* - \tilde{X}\right)/h$. This amounts to solving a small, prefactorized linear system in every time step. Finally, to synchronize the two solvers, the solid particle positions and velocities need to be updated to account for this projection (i.e. $X_S^p = \mathbf{U}^p\tilde{X}^*$ and $V_S^p = \mathbf{U}^p\tilde{V}^*$). The proposed transfers and synchronization mechanisms are fast, even for highly resolved meshes, since all operations have a complexity that is independent of the number of mesh vertices. In Appendix B we comment on the performance of this approach in comparison to a potential unreduced approach.

We are now able to summarize our algorithm for the simulation of fluid-deformable interaction, which is listed in Algorithm 1. The construction of the subspaces $\mathbf{U}^m$ and $\mathbf{U}^p$ will be described below. We start the simulation with initial subspace coordinates $\tilde{X}(0) = \tilde{X}_0$ of the deformable (and thus mesh node positions $X^m(0) = \mathbf{U}^m\tilde{X}_0$ and solid particle positions $X_S^p(0) = \mathbf{U}^p\tilde{X}_0$), initial velocities of the deformable $\tilde{V}(0) = \tilde{V}_0$ (yielding $V^m(0) = \mathbf{U}^m\tilde{V}$ and $V_S^p(0) = \mathbf{U}^p\tilde{V}$), initial fluid particle positions $X_F^p(0)$ and initial fluid particle velocities $V_F^p(0)$. Then, starting from $t = 0$, we perform the steps listed in Algorithm 1.

**Algorithm 1** The *RIM* algorithm

1: From $\tilde{X}(t)$ and $\tilde{V}(t)$ compute intermediate mesh velocities $\tilde{V}(t+h)$ using the HRPD algorithm.
2: Transfer these subspace velocities to intermediate solid particle velocities $V_S^{p\,*} = \mathbf{U}^p\tilde{V}(t+h)$.
3: Perform Algorithm 2 using $X^p(t)$, $V^p(t)$ and $V_S^{p\,*}$ as input for particle positions, velocities and intermediate particle velocities. This yields updated particle positions $X^{p\,*}$ and particle velocities $V^{p\,*}$.
4: The output positions and velocities for *fluid* particles are used in the next time step: $X_F^p(t+h) = X_F^{p\,*}$ and $V_F^p(t+h) = V_F^{p\,*}$.
5: The updated *solid* particle positions $X_S^{p\,*}$ are projected to updated subspace positions of the deformable $\tilde{X}(t+h)$ by solving (13).
6: Compute the updated subspace velocities of the deformable as $\tilde{V}(t+h) = (\tilde{X}(t+h) - \tilde{X}(t))/h$.
7: To synchronize the two representations, update the solid particle positions and velocities as $X_S^p(t+h) = \mathbf{U}^p\tilde{X}(t+h)$ and $V_S^p(t+h) = \mathbf{U}^p\tilde{V}(t+h)$.
8: Update solid particle normals as described in Section 3.2.5.
9: $t \leftarrow t+h$

*3.2.4 Twin subspace construction.* It remains to construct the subspaces $\mathbf{U}^p$ and $\mathbf{U}^m$ that are used to link mesh and particle representations of the deformable object. We start by initializing the solid particles representing the undeformed configuration by intersecting the undeformed tetrahedral mesh with a regular grid, whose cell widths are equal to the initial particle spacing chosen in the PIC/FLIP solver. For each particle, we store its barycentric coordinates within the tetrahedron it resides in, along with that tetrahedron's index.

The subspace for mesh deformations and velocities $\mathbf{U}^m$ is constructed as described in the *Hyper-Reduced Projective Dynamics* framework. That is, a set of sample vertices is chosen and skinning weights are computed from radial basis functions with limited support centered at these samples, using distances to these samples.

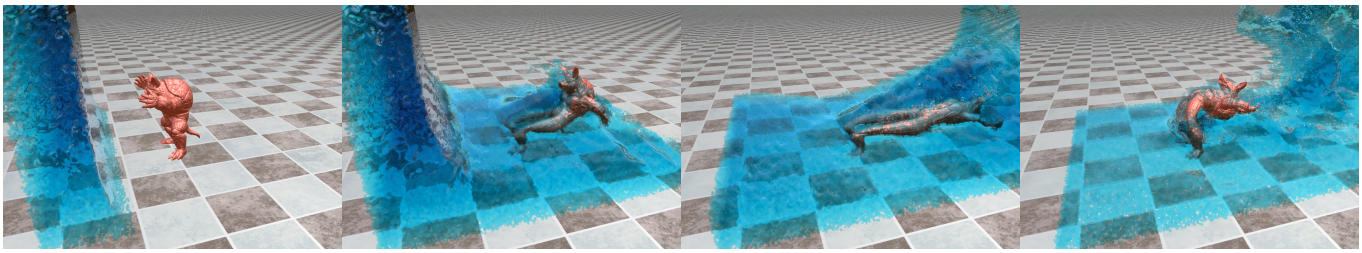


Fig. 5. An armadillo with clamped feet is subjected to a dam break.

The subspace $\mathbf{U}^m$ is then designed to contain all skinning deformations supported by the computed weights, such that the subspace coordinates correspond to affine transformations prescribed at each vertex sample.

The subspace for the solid particles is then constructed as $\mathbf{U}^p = \mathbf{B}\mathbf{U}^m$, where $\mathbf{B}$ is the matrix that puts each particle into the corresponding mesh tetrahedron at the stored barycentric coordinates.

*3.2.5 Preventing penetrations.* Our continuous model naturally prevents intersections between fluid and deformables, due to incompressibility of a global velocity field in combination with elastic forces. In practice, however, we encounter inaccuracies due to the large step-sizes and high velocities encountered in interactive real-time simulations, in combination with a limited grid resolution and particle spacing. These inaccuracies can lead to fluids penetrating the deformables. To prevent such occurrences, we add an additional velocity correction step to the fluid solver, where the velocity $V_f^p$ of a fluid particle $f$ that is closer than $h|V_f^p|$ to a solid boundary particle $s$ is required to satisfy

$$V_f^p \cdot n_s \geq V_s^p \cdot n_s. \tag{14}$$

This requires up-to-date particle normals for the outer particles representing the deformable. However, computing such normals at low particle resolutions and under strong deformations is inaccurate and for high resolutions it is prohibitively costly. Instead, we make use of the twin subspace construction. We initialize the normals of the boundary particles of the deformable once and update them efficiently by making use of the specific structure of our subspaces.

The initial solid particle normals are acquired from weighted averages of the closest mesh vertices. Then, we update particle normals by making use of the subspace coordinates corresponding to the deformed body, received from the elasticity step. For the specific subspaces proposed in Section 3.2.4, subspace coordinates correspond to a set of $k$ affine transformations. This means, we can approximate the particle normals by applying the weighted combinations of the linear parts of these transformations to the initial normals. For meshes, similar update mechanisms were proposed in [James and Pai 2002] and [Kavan et al. 2007]. Since the skinning weights in our subspace construction are sparse, this update mechanism is fast. This allows us to rapidly infer outer normals on the deformed particle representation, without having to compute the updated normals on the mesh. The penetration constraints are enforced just before advection (Line 11 in Algorithm 2) as a post-processing of particle velocities. We check, for each pair of nearby particles, whether the velocities satisfy the non-penetration constraints (14), and, if not, project the velocities to the closest configuration that satisfies them.

Using these normals to enforce (14) prevented fluid-deformable penetrations in all our examples. An example sequence with and without using the penetration prevention constraints can be seen in the appendix, in Figure 12. There, an extreme case is shown where fluid particles with high velocities fall onto a thin layer, using a large time-step.

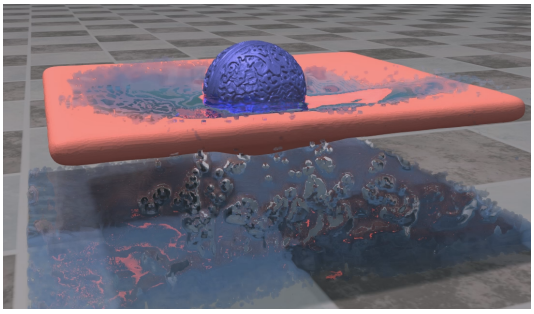
## 4 REAL-TIME COLLISION HANDLING

Our simulation framework offers a natural way to prevent and handle collisions between elastic deformables in a physically-based manner. In particular, we enrich the Projective Dynamics framework with natural collision handling for real-time simulations and highly resolved meshes. Contact is handled on the particle level, such that the level of detail at which intersections are prevented can be defined independently of the mesh resolution. This allows us to perform reduced-order simulations of complex meshes in contact, where common (self-)collision approaches would be prohibitive in real time settings.

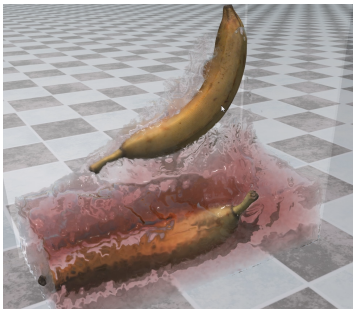
Our collision handling approach is fundamentally different from approaches, where existing intersections are detected, and either distance constraints or artificial repulsion forces are added to remove the intersections and handle the contact in a natural fashion. Instead, we anticipate and prevent collisions in three layers:

- All deformables are advected on a global velocity field. For small enough time steps, this prevents collisions, as individual trajectories in time- and space-continuous vector fields do not cross.
- Enforcing incompressibility jointly on a Eulerian grid does not permit intersections, since this would lead to locally increased densities. Additionally, the elastic energy acting on the deformables will prevent them from 'mixing', i.e. exchanging small amounts of particles, which can be observed for fluids.
- For large time-steps and velocities, both mechanisms above can fail due to numerical inaccuracies. Hence, in addition to enforcing the penetration prevention condition (14) for fluid particle velocities, we additionally enforce the following inequalities for all pairs $s_1, s_2$ of solid particles which are

(a) A simulation of four complex elastic deformables in contact.



(b) Interaction between an elastic mat, an elastic ball and water.



(c) Textured bananas in a pool.

Fig. 6. Various examples for complex deformable-deformable and deformable-fluid interactions.

facing each other, i.e.

$$n_{s_1} \cdot n_{s_2} \leq 0 \quad \text{and}$$
$$(X^p_{s_1} - X^p_{s_2}) \cdot n_{s_1} \geq 0$$

have to satisfy

$$V^p_{s_1} \cdot n_{s_2} \geq V^p_{s_2} \cdot n_{s_2}. \tag{15}$$

As for the fluid-solid non-penetration constraints in the previous section, they are enforced as a post-processing step before advection (Line 12 of Algorithm 2).

The capability of our method to prevent collisions in real-time simulations or in simulations with smaller time-steps is evaluated and discussed in Section 5.2.

## 5 EXPERIMENTS

The **supplementary video** accompanying this publication shows recordings (screen captured during simulation in real time) of all experiments discussed in this paper. The experiments were conducted on an AMD Ryzen Threadripper 1950X CPU with a GeForce RTX 2080 Ti. Hyper-Reduced Projective Dynamics was implemented to run on the CPU, using 8 threads in parallel, while the PIC/FLIP algorithm was implemented to run on the GPU using NVIDIA CUDA.

The data for our experiments can be found in Table 1. The first FPS measurement considers the time it takes to do a full time step and write all positions of mesh vertices and fluid particles into buffers on the GPU that allows an external environment to render the scene. The second FPS measurement is taken directly from the rendering environment that we used to create our animated scenes and thus includes the time to render the scene. Fluid rendering consists of a real-time post-process pass which uses screen-space reflections and refractions. The values for time step, diameters, densities and stiffness of the deformables are chosen to resemble real scenarios. The deformable diameter is measured as the longest side of a bounding box of one of the deformables in the scene. Fluid density (1000 kg/m$^3$) and gravitational acceleration (9.81 m/s$^2$) are the same throughout all simulations and are thus not listed in the table. For the Hyper-Reduced Projective Dynamics simulation, we use a constant number of ten local-global iterations and evaluate between 1000 and 2000 constraint projections to compute the approximated forces. For the PIC/FLIP simulation we use 3% PIC to 97% FLIP for fluid particles and 50% PIC to 50% FLIP for solid particles in all experiments. We solve for the unknown pseudo-pressure using a fixed number of Jacobi iterations. While allowing for a fast and highly memory efficient GPU implementation, the Jacobi method is known for slow convergence. Since in our model the solid-fluid interaction relies on the incompressibility of the joint velocity field, this requires further analysis, which is conducted in Appendix C.

We want to highlight that all simulations discussed in the following are simulated and rendered in 60 FPS with a time step of 1/60s, such that real-time interaction can be observed. The only exception to this is the simulation depicted in Figure 6a, which is simulated at 30 FPS with a time step of 1/256s and the comparison to an offline method shown in Figure 9.

We further break down the timings for one time step of our simulation for the example of an Armadillo mesh in a pool (Figure 8, Table 1 leftmost column) in Figure 7. There, we list the timings for HRPD (further split up into local and global iterations, as well as time taken to update relevant vertex positions), the time to perform Algorithm 2, the timings for the transfer between mesh and particle representations (including the evaluation of normals for solid boundary particles) and the timings for the rest of the simulation (handling interaction, writing particle and vertex positions into buffers, handling the program flow). As in all our experiments, the elasticity step takes up the largest part of our computation time.

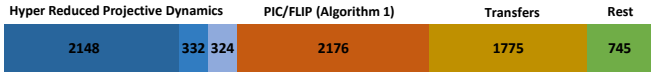



Fig. 7. Breakdown of timings (in microseconds) for one time step in the simulation from Figure 8.

There is a precomputation phase, in which we initialize the reduced dynamics as described in [Brandt et al. 2018], sample the particles for solids and fluids, generate the twin subspace for the deformable particles and transfer normals from the mesh to the boundary solid particles. These times range from 12 seconds, for the buoyancy example shown in Figure 8, to 96 seconds, for the simulation of four complex meshes shown in Figure 6a.

Table 1. Data for the experiments shown in the figures and the supplementary video. See Section 5 for further details.

| Name | Buoyancy (Fig. 8) | Armadillos in pool (Fig. 1) | Elephant and Lion (Fig. 2) | Cup in Pool (Fig. 4) | Plate and Ball (Fig. 6b) | Four Meshes (Fig. 6a) |
|---|---|---|---|---|---|---|
| # mesh tetrahedra | 137330 | 274660 | 250154 | 368709 | 713012 | 371561 |
| # solid particles | 4826 | 4253 | 53082 | 9811 | 84499 | 241983 |
| # fluid particles | 189540 | 633750 | 0 | 274076 | 89090 | 0 |
| grid dimensions | 27 × 99 × 26 | 61 × 81 × 65 | 87 × 92 × 63 | 33 × 147 × 33 | 59 × 152 × 48 | 59 × 152 × 48 |
| Subspace Dimension | 1188 | 1188 | 1188 | 1188 | 1188 | 2400 |
| diameter of deformable (m) | 2.86 | 2.86 | 1.85 (Elephant) | 2.02 | 2.11 (Ball) | 2.86 (Armadillo) |
| time step (s) | 1/60 | 1/60 | 1/60 & 1/30 | 1/60 | 1/60 | 1/256 |
| init. particle spacing (m) | 0.073 | 0.096 | 0.037 | 0.051 | 0.025 | 0.032 |
| solid density (kg/m$^3$) | 540 / 1000 / 1850 | 660 | 1000 | 430 | 2330 | 1000 |
| Stiffness parameter | 150000 | 500000 | 100000 & 20000 | 250000 | 300.000 | 150000 |
| **fps** | **126** | **98** | **94** | **129** | **80** | **31** |
| fps incl. rendering | 77 | 60 | 72 | 60 | 55 | 25 |

### 5.1 Fluid-deformable interaction

In Figure 1, we show a snapshot of a simulation of two armadillo meshes (137k tetrahedra and 4253 solid particles each) in a pool of water (633750 fluid particles). As in all our simulations, the user can interact by clicking and dragging the meshes to introduce temporary position constraints into the elasticity step. The scene shows that two-way coupling between fluid and deformable is faithfully handled and that the simulation remains stable, even though large velocities (which violate the CFL conditions) are present.

Figure 8 shows that buoyancy is correctly handled by our method. Armadillos of different densities are dropped into a pool of water and sink to the bottom, come to rest within the pool or rise to the top, depending on the density ratios. Note that drag forces are implicitly handled by our model and do not have to be computed and transferred to solid particles or the mesh in any way.

More complex fluid-deformable interactions are shown in Figures 9, left, 4 and 5. In these simulations, we show that we successfully prevent liquid from penetrating moderately thin deformable surfaces. The simulation in Figure 9, left, shows liquid deforming an elastic mat, whose inner forces cause an opposing motion that, in turn, causes the fluid to splash upwards. Figure 4 shows a snapshot of a simulation where an elastic, low density cup in a pool is filled with water, which causes the cup to deform and sink. Through user interaction the cup is then lifted such that the liquid is poured into the pool. The full cup settles at a different height than the empty cup, which shows another complex layer of interaction.

In Figure 5, we show an elastic body with low density being subjected to a flood of water, which heavily deforms the solid. The supplementary video includes two variants of this sequence with varying material stiffness for the solid, which lead to clearly distinct interaction behavior. Our simulation makes use of the mesh-based representation for elasticity computations and rendering, where it is advantageous over particle or grid based representations. To highlight another advantage of the mesh representation, a simulation with textured objects, interacting with each other and with fluid, is shown in Figure 6c.

### 5.2 Deformable contact simulation

Figure 2 shows interactive simulations of complex elastic deformables in contact, where a user pushes and drags elastic bodies against each other. Figure 6b shows a simulation that exhibits rich interactions between two deformables (the elastic mat and an elastic ball) and fluid. Data concerning these simulations is included in Table 1. The example demonstrates the capabilities of our method to handle collisions and self-intersections in real-time. The deformables interact naturally and most collisions and self-intersections are prevented. Like any collision handling approach for elastic deformables, the large velocities encountered in interactive simulations, in combination with the large time steps, can cause intersections if the collision primitives (in our case particles) are advected further than their confidence radius in a single time-step. For applications that require guarantees for the absence of intersections, one can either reduce the time step depending on the largest velocities or conduct an additional position correction step (which can either make use of the mesh, particle or grid data available in our method). Figure 6a shows a snapshot of a simulation with four complex meshes in contact and a time step of 1/256s, which is free of collisions.

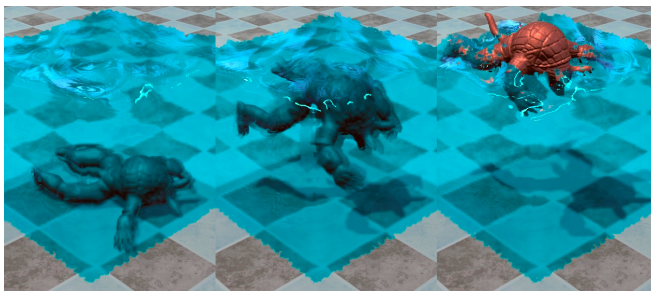


Fig. 8. Armadillos with different densities are being dropped into a pool of water, the snapshot is taken after the armadillos settle. The armadillo's density is 540, 1000 and 1850 kg/m$^3$, respectively.

## 6 COMPARISONS

### 6.1 Other real-time approaches

Approaches for real-time fluid-deformable coupling are listed in Section 2. We would like to put our method in relation with the FLIP and shape matching approach [Gao et al. 2018a] and the unified PBD approach [Macklin et al. 2014] in more detail, as these two methods report the highest performance in terms of computation times, resolutions and time-step sizes. Both methods are purely particle-based and use shape-matching to model the deformable objects. While shape-matching approaches are fast, they are also simple models of deformable objects. As a consequence, "only small scale elastic deformations are supported" ([Macklin et al. 2014]), and, for highly-resolved deformables, the stiffness of the objects is difficult to control (see e.g. [Weiler et al. 2016]). In contrast, our approach uses a non-linear mesh-based elasticity solver and large deformations can be observed in all our examples, see Figure 5 for an example involving particularly large deformations. Moreover, having mesh representations of the deformables readily available is beneficial for rendering the scenes (especially in real-time approaches), see e.g. Figure 6c for an example with textured meshes.

In the unified particle-based approach [Macklin et al. 2014], fluids are simulated using density constraints (akin to [Macklin and Müller 2013]) and coupling is achieved by including the solid particles to the density estimation, which is used for enforcing the density constraints. The convergence of this approach is tied to the number of particles, such that, for example, buoyancy is reported to be affected by the sizes of the solids, instead of solely their density.

[Gao et al. 2018a] achieve coupling by computing a FLIP step for all particles and then applying a shape-matching step that modifies the positions of the solid particles. Since this will lead to penetrations between fluid and solid, subsequent correction steps are conducted. This coupling approach does not derive from a continuous system and "cannot handle elastic objects interacting with fluid well" ([Gao et al. 2018a]). In contrast, our system derives from an established continuous system, which we discretize using two specialized solvers and force/velocity transfers from mesh to grid.

### 6.2 Comparison to an offline method

We can verify the ability of RIM to accurately simulate complex interaction scenarios by comparing it to offline methods, which spend a considerable amount of time on computing each time-step with the goal of being as accurate as possible. To this end, we compare *RIM* to a recent approach presented in [Zarifi and Batty 2017]. As an experimental setup, we pour fluid onto an elastic, low density mat. The setup is chosen such that strong interactions in both directions (i.e. fluid strongly deforming the solid and the solid applying forces to the fluid with visible effect) can be observed and difficulties in preventing penetrations of fluid through the thin mat arise. To keep the computation time of the offline method low, we use a low resolution to represent the elastic mat (3025 vertices). The fluid is represented by roughly 59k particles, and the simulation grid consists of $100^3$ cells. The time step is set to to 1/240 and gravitational acceleration to $3m/s^2$, in order to keep the offline method from taking more than two sub-iterations and to prevent fluid from penetrating the mat. Since the elastic material model used in [Zarifi and Batty 2017] is different from ours, we chose the material stiffness parameter of our method such that a visually comparable behavior is obtained. In Figure 9, we show frames of the simulations produced by both methods side-by-side, the full comparison is shown in our supplementary video. Our method, which requires 17ms computation time per frame, is able to faithfully simulate the two-way coupling phenomena observed in the offline method, which requires 7480ms per frame on average.[1] Some differences in the simulations are expected, as the method of [Zarifi and Batty 2017] resolves the fluid-solid interface sharply, uses a different material model, uses free-slip boundary conditions and simulates compressible deformables.

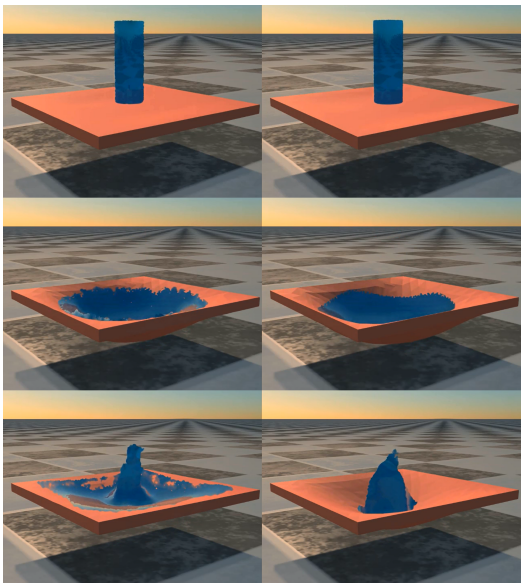


Fig. 9. Corresponding frames of simulations using *RIM* (left, 17ms per frame) and a recent offline method [Zarifi and Batty 2017] (right, 7480ms per frame).

To demonstrate the scalability of our method, in our supplementary video, we show a similar simulation using a larger amount of fluid, a more finely resolved elastic mat, higher gravitational acceleration and a larger time-step, while allowing for user interaction. Moreover, our frameworks simulates deformable-deformable contact without any computational overhead, see for example Figure 6b.

## 7 LIMITATIONS AND FUTURE WORK

Maintaining real-time rates, robustness, and stability for deformable-fluid scenarios is a challenging problem. Hence, our method has several restrictions.

The equations of motion underlying *RIM* pose two limitations. The first is that the elastic materials must be incompressible. Since this requirement is closely tied to the way our framework realizes fluid-deformable interaction, extending the framework to include compressible material poses a challenge. The second limitation is that the interface is restricted to no-slip boundary conditions. In our

[1] For the method from [Zarifi and Batty 2017], we used a CPU implementation provided by the authors, which performs all simple loops in parallel. At the heart of the algorithm is a linear solve of a time-varying system which has to be assembled in each iteration.

experiments, sticky surfaces can be observed (see Figure 9, left and 6b). However, simulations also show that fluid can flow tangentially across deformable surfaces. For example, fluid runs across the the armadillo's skin as it surfaces from the pool in Figure 8, right, leaving it completely dry. Similarly, deformables in contact can be observed sliding across each other (see Figure 2). This is caused by factors such as the PIC/FLIP update, which involves not only the grid velocities but also their previous velocities and temporal changes of the grid velocities. It is an interesting task to extend *RIM* to include other types of boundary conditions.

Our current implementation uses a basic implementation of the PIC/FLIP solver with several modifications to accommodate for large time-steps and low computation times. In order to achieve more accurate and rich dynamics, it would be beneficial to extend the fluid solver by handling free surfaces correctly, resolving the discontinuities in the densities and pressure more accurately and use a more involved volume preservation mechanism for large time steps. While these features have been introduced in several variants, preliminary experiments have shown that work is required to adapt them to real-time scenarios.

It would be interesting to explore the use of dimensional reduction techniques for the fluid solver in our framework, e.g., for an accelerated pressure projection [Ando et al. 2015] or by restricting velocities to a subspace [Cui et al. 2018; Liu et al. 2015].

To allow fluids and deformables to freely move in space, unrestricted by simulation boundaries, sparse and adaptable grids as proposed in [Wu et al. 2018] would be beneficial. Note that in simulations where only deformables are present, we can move the grid to always encompass the current positions of the deformable.

## ACKNOWLEDGMENTS

We would like to thank Niels de Hoon for helpful suggestions and feedback, Omar Zarifi and Christopher Batty for sharing their implementation of the cut-cell method, and the anonymous reviewers for their constructive comments and suggestions.

## REFERENCES

Muzaffer Akbay, Nicholas Nobles, Victor Zordan, and Tamar Shinar. 2018. An extended partitioned method for conservative solid-fluid coupling. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 86.

Steven S. An, Theodore Kim, and Doug L. James. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph.* 27, 5 (2008), 165:1–165:10.

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2015. A Dimension-reduced Pressure Solver for Liquid Simulations. *Comput. Graph. Forum* 34, 2 (2015), 473–480.

Yuanxun Bao, Aleksandar Donev, Boyce E. Griffith, David M. McQueen, and Charles S. Peskin. 2017. An Immersed Boundary method with divergence-free velocity interpolation and force spreading. *J. Comput. Phys.* 347 (2017), 183 – 206.

Jernej Barbič and Doug L. James. 2005. Real-Time Subspace Integration for St. Venant–Kirchhoff Deformable Models. *ACM Trans. Graph.* 24, 3 (2005), 982–990.

Jernej Barbič and Doug L. James. 2008. Six-DoF Haptic Rendering of Contact Between Geometrically Complex Reduced Deformable Models. *IEEE Trans. Haptics* 1, 1 (2008), 39–52.

Jernej Barbič and Doug L. James. 2010. Subspace Self-collision Culling. *ACM Trans. Graph.* 29, 4 (2010), 81:1–81:9.

Adam Bargteil and Tamar Shinar. 2018. An Introduction to Physics-based Animation. In *ACM SIGGRAPH 2018 Courses (SIGGRAPH '18)*. ACM, 6:1–6:57.

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A Fast Variational Framework for Accurate Solid-fluid Coupling. *ACM Trans. Graph.* 26, 3 (2007).

Markus Becker and Matthias Teschner. 2007. Weakly compressible SPH for free surface flows. In *Symposium on Computer Animation*. 209–217.

Jan Bender, Dan Koschier, Patrick Charrier, and Daniel Weber. 2014. Position-based simulation of continuous materials. *Computers & Graphics* 44 (2014), 1–10.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 33, 4 (2014), 154:1–154:11.

Landon Boyd and Robert Bridson. 2012. MultiFLIP for energetic two-phase fluid simulation. *ACM Transactions on Graphics (TOG)* 31, 2 (2012), 16.

Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-Reduced Projective Dynamics. *ACM Transactions on Graphics* 37, 4 (2018), 80:1–80:13.

Mark Carlson, Peter J. Mucha, and Greg Turk. 2004. Rigid Fluid: Animating the Interplay Between Rigid Bodies and Fluid. *ACM Trans. Graph.* 23, 3 (2004), 377–384.

Nuttapong Chentanez, Bryan E. Feldman, François Labelle, James F. O'Brien, and Jonathan R. Shewchuk. 2007. Liquid Simulation on Lattice-Based Tetrahedral Meshes. In *Symposium on Computer Animation*. 219–228.

Nuttapong Chentanez, Tolga G. Goktekin, Bryan E. Feldman, and James F. O'Brien. 2006. Simultaneous Coupling of Fluids and Deformable Bodies. In *ASymposium on Computer Animation*. 83–89.

Nuttapong Chentanez and Matthias Müller. 2011. Real-time Eulerian Water Simulation Using a Restricted Tall Cell Grid. *ACM Trans. Graph.* 30, 4 (2011), 82:1–82:10.

Pascal Clausen, Martin Wicke, Jonathan R. Shewchuk, and James F. O'Brien. 2013. Simulating Liquids and Solid-liquid Interactions with Lagrangian Meshes. *ACM Trans. Graph.* 32, 2 (2013), 17:1–17:15.

Qiaodong Cui, Pradeep Sen, and Theodore Kim. 2018. Scalable Laplacian Eigenfluids. *ACM Trans. Graph.* 37, 4 (2018), 87:1–87:12.

Joris Degroote. 2013. Partitioned simulation of fluid-structure interaction. *Archives of computational methods in engineering* 20, 3 (2013), 185–238.

Dimitar Dinev, Tiantian Liu, Jing Li, Bernhard Thomaszewski, and Ladislav Kavan. 2018. FEPR: fast energy projection for real-time simulation of deformable objects. *ACM Trans. Graph.* 37, 4 (2018), 79:1–79:12.

Ye Fan, Joshua Litven, David I. W. Levin, and Dinesh K. Pai. 2013. Eulerian-on-Lagrangian Simulation. *ACM Trans. Graph.* 32, 3 (2013), 22:1–22:9.

Ming Gao, Xinlei Wang, Kui Wu, Andre Pradhana, Eftychios Sifakis, Cem Yuksel, and Chenfanfu Jiang. 2018b. GPU Optimization of Material Point Methods. *ACM Trans. Graph.* 37, 6, Article 254 (Dec. 2018), 12 pages. https://doi.org/10.1145/3272127.3275044

Yang Gao, Shuai Li, Hong Qin, Yinghao Xu, and Aimin Hao. 2018a. An efficient FLIP and shape matching coupled method for fluid–solid and two-phase fluid simulations. *The Visual Computer* (2018).

Theodore F. Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang, and Joseph M. Teran. 2015. Optimization Integrator for Large Time Steps. *IEEE Trans. Vis. Comput. Graph.* 21, 10 (2015), 1103–1115.

Olivier Génevaux, Arash Habibi, and Jean-Michel Dischler. 2003. Simulating Fluid-Solid Interaction. In *Proceedings of the Graphics Interface*. 31–38.

Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. 2005. Coupling Water and Smoke to Thin Deformable and Rigid Shells. *ACM Trans. Graph.* 24, 3 (2005), 973–981.

David Harmon and Denis Zorin. 2013. Subspace integration with local deformations. *ACM Trans. Graph.* 32, 4 (2013), 107:1–107:10.

Kris K. Hauser, Chen Shen, and James F. O'Brien. 2003. Interactive Deformation Using Modal Analysis with Constraints. In *Graphics Interface*. 247–256.

Gene Hou, Jin Wang, and Anita Layton. 2012. Numerical methods for fluid-structure interaction - a review. *Communications in Computational Physics* 12, 2 (2012), 337–377.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 150.

Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. 2014. Implicit Incompressible SPH. *IEEE Trans. Vis. Comput. Graph.* 20, 3 (2014), 426–435.

Doug L. James and Dinesh K. Pai. 2002. DyRT: Dynamic Response Textures for Real Time Deformation Simulation with Graphics Hardware. *ACM Trans. Graph.* 21, 3 (2002), 582–585.

Doug L. James and Dinesh K. Pai. 2004. BD-tree: Output-sensitive Collision Detection for Reduced Deformable Models. *ACM Trans. Graph.* 23, 3 (2004), 393–398.

Chenfanfu Jiang, Theodore Gast, and Joseph Teran. 2017. Anisotropic Elastoplasticity for Cloth, Knit and Hair Frictional Contact. *ACM Trans. Graph.* 36, 4, Article 152 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073623

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 51.

David Kamensky, Ming-Chen Hsu, Dominik Schillinger, John A. Evans, Ankush Aggarwal, Yuri Bazilevs, Michael S. Sacks, and Thomas J.R. Hughes. 2015. An immersogeometric variational framework for fluid-structure interaction: Application to bioprosthetic heart valves. *Computer Methods in Applied Mechanics and Engineering* 284 (2015), 1005–1053.

Nahyup Kang, Jinho Park, Junyong Noh, and Sung Yong Shin. 2010. A hybrid approach to multiple fluid simulation using volume fractions. In *Computer Graphics Forum*,

Vol. 29. 685–694.
Ladislav Kavan, Rachel McDonnell, Simon Dobbyn, Jiří Žára, and Carol O'Sullivan. 2007. Skinning arbitrary deformations. In *Symposium on Interactive 3D Graphics and Games*. ACM, 53–60.
David I. W. Levin, Joshua Litven, Garrett L. Jones, Shinjiro Sueda, and Dinesh K. Pai. 2011. Eulerian Solid Simulation with Contact. *ACM Trans. Graph.* 30, 4 (2011), 36:1–36:10.
Jing Li, Tiantian Liu, and Ladislav Kavan. 2018. Laplacian Damping for Projective Dynamics. In *VRIPHYS 2018*. 29–36.
Bei-Bei Liu, Gemma Mason, Julian Hodgson, Yiying Tong, and Mathieu Desbrun. 2015. Model-reduced variational fluid simulation. *ACM Trans. Graph.* 34, 6 (2015), 244:1–244:12.
Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-Newton Methods for Real-Time Simulation of Hyperelastic Materials. *ACM Trans. Graph.* 36, 3 (2017), 23:1–23:16.
Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating Water and Smoke with an Octree Data Structure. *ACM Trans. Graph.* 23, 3 (2004), 457–462.
Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. 2006. Multiple interacting liquids. In *ACM Tran. Graph.*, Vol. 25. ACM, 812–819.
Wenlong Lu, Ning Jin, and Ronald Fedkiw. 2016. Two-way Coupling of Fluids to Reduced Deformable Bodies. In *Symposium on Computer Animation*. 67–76.
Miles Macklin and Matthias Müller. 2013. Position Based Fluids. *ACM Trans. Graph.* 32, 4 (2013), 104:1–104:12.
Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2014. Unified Particle Physics for Real-time Applications. *ACM Trans. Graph.* 33, 4 (2014), 153:1–153:12.
Marek Krzysztof Misztal, Kenny Erleben, Adam W. Bargteil, Jens Fursund, Brian Bunch Christensen, Jakob Andreas Bærentzen, and Robert Bridson. 2014. Multiphase Flow of Immiscible Fluids on Unstructured Moving Meshes. *IEEE Trans. Vis. Comput. Graph.* 20, 1 (2014), 4–16.
Joe J. Monaghan. 1992. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics* 30, 1 (1992), 543–574.
Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based fluid simulation for interactive applications. In *Ssymposium on Computer Animation*. 154–159.
Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. 2014. Strain Based Dynamics. In *Symposium on Computer Animation*. 149–157.
Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus H. Gross. 2005. Meshless deformations based on shape matching. *ACM Trans. Graph.* 24, 3 (2005), 471–478.
NVIDIA 2007. *CUDA: Compute Unified Device Architecture Programming Guide*. NVIDIA.
Matthew Overby, George E. Brown, Jie Li, and Rahul Narain. 2017. ADMM ⊇ Projective Dynamics: Fast Simulation of Hyperelastic Models with Dynamic Constraints. *IEEE Trans. Vis. and Comp. Graph.* 23, 10 (2017), 2222–2234.
Andreas Peer, Christoph Gissler, Stefan Band, and Matthias Teschner. 2018. An Implicit SPH Formulation for Incompressible Linearly Elastic Solids. *Computer Graphics Forum* 37, 6 (2018), 135–148.
Yue Peng, Bailin Deng, Juyong Zhang, Fanyu Geng, Wenjie Qin, and Ligang Liu. 2018. Anderson acceleration for geometry optimization and physics simulation. *ACM Trans. Graph.* 37, 4 (2018), 42:1–42:14.
Charles S. Peskin. 2002. The immersed boundary method. *Acta Numerica* 11 (2002), 479–517.
Avi Robinson-Mosher, Craig Schroeder, and Ronald Fedkiw. 2011. A Symmetric Positive Definite Formulation for Monolithic Fluid Structure Interaction. *J. Comput. Phys.* 230, 4 (2011), 1547–1566.
Avi Robinson-Mosher, Tamar Shinar, Jon Gretarsson, Jonathan Su, and Ronald Fedkiw. 2008. Two-way Coupling of Fluids to Rigid and Deformable Solids and Shells. *ACM Trans. Graph.* 27, 3 (2008), 46:1–46:9.
Sara C. Schvartzman, Jorge Gascón, and Miguel A. Otaduy. 2009. Bounded Normal Trees for Reduced Deformations of Triangulated Surfaces. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 75–82.
X Shao, Z Zhou, Nadia Magnenat-Thalmann, and W Wu. 2015. Stable and fast fluid–solid coupling for incompressible SPH. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 191–204.
Eftychios Sifakis and Jernej Barbič. 2012. FEM simulation of 3D deformable solids: A practitioner's guide to theory, discretization and model reduction. In *SIGGRAPH Courses*. 20:1–20:50.
Barbara Solenthaler and Renato Pajarola. 2009. Predictive-corrective incompressible SPH. In *ACM Trans. Graph.*, Vol. 28. ACM, 40.
Deborah Sulsky, Shi-Jian Zhou, and Howard L. Schreyer. 1995. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications* 87, 1 (1995), 236–252. Particle Simulation Methods.
Andre Pradhana Tampubolon, Theodore Gast, Gergely Klár, Chuyuan Fu, Joseph Teran, Chenfanfu Jiang, and Ken Museth. 2017. Multi-species simulation of porous sand and water mixtures. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 105.
Yun Teng, David I. W. Levin, and Theodore Kim. 2016. Eulerian Solid-fluid Coupling. *ACM Trans. Graph.* 35, 6 (2016), 200:1–200:8.
Yun Teng, Miguel A. Otaduy, and Theodore Kim. 2014. Simulating Articulated Subspace Self-contact. *ACM Trans. Graph.* 33, 4 (2014), 106:1–106:9.
Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. 2006. Vector field based shape deformations. *ACM Trans. Graph.* 25, 3 (2006), 1118–1125.
Christoph von Tycowicz, Christian Schulz, Hans-Peter Seidel, and Klaus Hildebrandt. 2013. An Efficient Construction of Reduced Deformable Objects. *ACM Trans. Graph.* 32, 6 (2013), 213:1–213:10.
Huamin Wang. 2015. A Chebyshev Semi-iterative Approach for Accelerating Projective and Position-based Dynamics. *ACM Trans. Graph.* 34, 6 (2015), 246:1–246:9.
Huamin Wang and Yin Yang. 2016. Descent Methods for Elastic Body Simulation on the GPU. *ACM Trans. Graph.* 35, 6 (2016), 212:1–212:10.
Marcel Weiler, Dan Koschier, and Jan Bender. 2016. Projective Fluids. In *Proc. ACM Motion in Games*. 79–84.
Kui Wu, Nghia Truong, Cem Yuksel, and Rama Hoetzlein. 2018. Fast Fluid Simulations with Sparse Volumes on the GPU. *Computer Graphics Forum* 37, 2 (2018), 157–167.
Lipeng Yang, Shuai Li, Aimin Hao, and Hong Qin. 2012. Realtime Two-Way Coupling of Meshless Fluids and Nonlinear FEM. *Computer Graphics Forum* 31, 7 (2012), 2037–2046.
Yin Yang, Dingzeyu Li, Weiwei Xu, Yuan Tian, and Changxi Zheng. 2015. Expediting Precomputation for Reduced Deformable Simulation. *ACM Trans. Graph.* 34, 6 (2015), 243:1–243:13.
Omar Zarifi and Christopher Batty. 2017. A Positive-definite Cut-cell Method for Strong Two-way Coupling Between Fluids and Deformable Bodies. In *Symposium on Computer Animation*. 7:1–7:11.
Lucy Zhang, Axel Gerstenberger, Xiaodong Wang, and Wing Kam Liu. 2004. Immersed finite element method. *Computer Methods in Applied Mechanics and Engineering* 193, 21 (2004), 2051–2067. Flow Simulation and Modeling.
Lucy T. Zhang and Mickaël Gay. 2007. Immersed finite element method for fluid-structure interactions. *Journal of Fluids and Structures* 23, 6 (2007), 839–857.
Yongning Zhu and Robert Bridson. 2005. Animating Sand As a Fluid. *ACM Trans. Graph.* 24, 3 (2005), 965–972.

## A MODIFIED PIC/FLIP ALGORITHM

We modified the PIC/FLIP algorithm [Zhu and Bridson 2005] to accomodate for the additional elastic forces and the large-time steps and performance requirements posed by interactive simulations. Our PIC/FLIP variant is listed in Algorithm 2.

The following changes were made to account for the immersed deformable object and the additional elastic forces acting on it:

- Particles are equipped with a flag that discerns particles representing the deformable (*solid particles*) and particles representing the fluid. Each particle carries its respective initial density, such that the evaluation of $\rho(x, t)$ for (8) and (9) based on the integral in (3) is approximated by interpolating these quantities on the grid; see Line 5.
- To take into account the elastic forces acting on the deformables, we will transfer the updated velocities from the elasticity step (which accounts for external forces as well) to the solid particles in Line 3 (this transfer detailed in Sections 3.2.3 and 3.2.4). Fluid particle velocities still need to be updated according to external forces (Line 2). Then, the updated particle velocities are interpolated to the grid to obtain $v^c_{\text{new}}$. While we listed this as three different steps in the algorithm, both $v^c_{\text{old}}$ and $v^c_{\text{new}}$ can be computed within a single loop over all nearby particles, preventing re-evaluation of the trilinear weights.
- We need to account for different densities in the projection step in Lines 8 and 9. Let $v^c$ be the velocity field that was interpolated on the grid from solid and fluid particles. In each grid cell $c$, we interpolate the density values carried by nearby particles (using the trilinear weights) to get an

estimated density value $\rho^c$ within this cell[2]. The time-discrete, grid-discrete version of equations (9) and (10) then read

$$\tilde{v}^c = v^c - h\frac{1}{\rho^c}(\nabla p)^c \tag{16}$$

$$(\nabla \cdot \tilde{v})^c = 0 \tag{17}$$

where $p$ is an unknown pseudo-pressure field on the grid. Applying the (discrete) divergence operator to both sides of (16) and using (17), results in the discrete, density weighted Poisson equation

$$h\left(\nabla \cdot \frac{1}{\rho^c}\nabla p\right)^c = (\nabla \cdot v)^c, \tag{18}$$

which we solve using a fixed-number of Jacobi iterations (200 in all experiments, see Appendix C). Such variable coefficient Poisson equations often appear in multiple fluid simulation approaches (such as [Boyd and Bridson 2012; Kang et al. 2010; Losasso et al. 2006]), where great care is taken to resolve the discontinuities in the density (and the resulting pressure) in their discretization. We resolve the issue of discontinuities by interpolating the values on the grid, that is, assigning a mix between fluid and deformable densities in cells where both are present. This leads to inaccuracies in the projection step. Note, however, that we are not dealing with the complex phenomena associated to fluid-fluid interactions (bubbles, mixing, foam), that lead to topological changes at fine scales. In our experiments (Section 5), we show that we can simulate a broad range of fluid-deformable interaction phenomena despite this inaccuracy.

Furthermore, in order to account for large time steps and to decrease the computation time we introduce the following modifications to the PIC/FLIP algorithm:

- In Line 7 we modify the computation of divergence, where, if the sum of trilinear weights from nearby particles in a cell exceeds that value for a cell filled with regularly sampled particles, we subtract a fraction of the difference from the divergence. This fraction is given by a parameter $\alpha$, that is chosen in the range of 0.1 to 1 in our examples. This simple modification allows us to recover from temporary volume loss in a natural manner, but does not modify computations at all, in case no volume loss occurs. The sums of trilinear weights are already available from the velocity interpolation step.
- In interactive simulations, we do not adapt the step-size $h$ when velocities exceed the threshold given by the CFL conditions, as this is prohibitive in a real time setting, where a stable ratio between framerate and time step is required. For non-interactive simulations the step-size can be adapted when velocities from the elasticity step exceed a threshold.
- We do not extend the velocity field to empty grid-cells, which allows us to skip the computation of the distance field to the water surface. This causes the fluid to decelerate more when entering empty grid cells and may also cause unnatural splashing due to not accounting for surface tension. This trade-off is made to allow for real-time simulations with highly resolved fluids.
- We will not sample air particles in our PIC/FLIP simulation, such that grid-cells without fluid or deformable particles are skipped in divergence computations and the projection step. This is a common simplification for fluid simulations in computer graphics.

[2]Note that the interpolation of density values on the grid can be performed at the same time as velocities are interpolated to increase performance.

**Algorithm 2** The modified PIC/FLIP algorithm used for *RIM*

**Input:** Step-size $h$, particle positions $X^p$, current particle velocities $V^p$, intermediate solid particle velocities $V_S^{p*}$, grid-based external forces $f_{\text{ext}}$.
**Output:** Updated particle positions $X^{p*}$ and velocities $V^{p*}$

1: Interpolate particle velocities $V^p$ on grid nodes $c$, using trilinear weights: $v^c_{\text{old}}$
2: Update *fluid* particle velocities from external forces: $V_F^p \leftarrow V_F^p + h * f_{\text{ext}}$
3: Replace the velocities on *solid* particles by the intermediate solid particle velocities: $V_S^p \leftarrow V_S^{p*}$
4: Interpolate the updated particle velocities $V^p$ on grid nodes $c$, using trilinear weights: $v^c_{\text{new}}$
5: Interpolate particle densities on grid nodes $c_i$, using trilinear weights: $\rho^c$
6: Compute divergence of $v^c_{\text{new}}$ on grid: $d^{c_i}$
7: Subtract divergence from cells where the sum of trilinear weights is above rest-value: $d^c \leftarrow d^c - \max(0, \alpha(w^c - w_{\text{rest}}))$
8: Solve Poisson equation using Jacobi iterations to get pseudo-pressure on grid: $p^c$
9: Make velocity field divergence free: $v^c_{\text{new}} \leftarrow v^c_{\text{new}} - h\frac{1}{\rho^c}\nabla p^c$
10: Interpolate grid velocities on particles, using both $v_{\text{new}}$ and $v_{\text{old}}$ in PIC/FLIP manner: $V^{p*}$
11: Enforce fluid-deformable penetration prevention constraints (14) from Section 3.2.5
12: Enforce deformable-deformable non-intersection constraints (15) from Section 4
13: Advect each particle using a Runge–Kutta second order step to get updated particle positions $X^{p*}$

We implemented Algorithm 2 on a GPU using CUDA [NVIDIA 2007]. Most steps of this algorithm can be trivially parallelized (over the grid or over the particles). To efficiently interpolate particle quantities on the grid, a data structure that assigns particles to the grid cell they reside in is required. To this end, we first create a list of key-value pairs that contain, for each particle, its own ID and the ID of the grid cell it resides in. We then sort this list by the grid cell IDs (using a GPU-based parallel sorting algorithm). From the sorted list, in parallel over all particles, we create a data structure that enables efficient iterations over particles that reside in a specific cell.

## B BENEFITS OF THE TWIN SUBSPACES

The twin subspace construction enables the rapid transfer between and synchronization of the two solvers employed in our framework. In Figure 7 we report the time spent on transferring velocities from

mesh to particles, transferring positions and velocities from particles to mesh and updating the normals of the boundary particles in a typical simulation scenario (Figure 8), which amounts to 1775 microseconds. The timings for transfer and synchronization between the solvers depends only on the number of solid particles and the subspace dimension, but is independent of the mesh resolution.

Without twin subspaces there are various problems preventing an efficient simulation scheme. In the following, we list potential alternatives for each of the transfers:

- We use the twin subspaces for transferring velocities from the elasticity solver to the solid particles. Without the twin subspace concept, we would need to evaluate the per-vertex velocities (from the subspace coordinates) and compute the particle velocities via barycentric interpolation.
- We use the twin subspace to efficiently update normal vectors at the solid boundary in the particle representation. Without the twin subspace mechanism, we would need to compute updated normals for the mesh boundary. Hence all vertex (or triangle normals) would have to be computed, which requires full vertex positions evaluated from subspace coordinates. Then, we would need to transfer these to the particles as weighted sums and normalize them. Alternatively, particle normals could be computed from the particle positions directly, but this is inaccurate at the particle resolutions used in our examples.
- We use the twin subspaces for transferring updated positions and velocities from the solid particles back to the elasticity solver. It is unclear how this could be done without the twin subspaces since there are significantly less particles than vertices in our examples, and there are vertices outside of the convex hull of the particle representation. A transfer via interpolating per-vertex velocities from the grid, a subsequent subspace projection and explicit, Lagrangian advection would be possible, but would not bring the mesh and particle representations into correspondence, since their respective advections would yield significantly different results. These errors would accumulate without additional synchronization of vertex and particle positions, which could be achieved via barycentric interpolation. Note that this would cause solid particles to change their positions, while fluid particles remain fixed, and would lead to volume loss and fluid-deformable penetration. [3]

In summary, a potential unreduced scheme for our proposed discretization of the immersed boundary equations of motion is possible, but increases computation time in a mesh-resolution dependent manner and introduces problems due to difficulties in the synchronization of the representations.

## C APPROXIMATED PRESSURE PROJECTION

To find a decent trade-off between accuracy and performance, we approximate the pressure projection solve (see equation (18)) by a fixed number of Jacobi iterations, namely 200 in all our experiments. Since our solid-fluid interaction model relies on the incompressibility of the joint velocity field, residual divergence might not only cause volume loss, but also inaccurate interaction behavior. However, we found that divergence is sufficiently reduced that the visual quality of the simulation does not suffer from the approximation. Note that long-term volume loss is prevented due to the modified divergence used on the right hand side of the pressure projection (cells with too many particles get assigned a decreased divergence, see Line 7 of Algorithm 2). Furthermore, interaction artifacts, such as fluid and solid intersecting, can be remedied by the particle penetration prevention step described in Section 3.2.5. For a visual comparison of our method using limited Jacobi iterations to a fully monolithic method that solves the pressure projection exactly, see Figure 9.

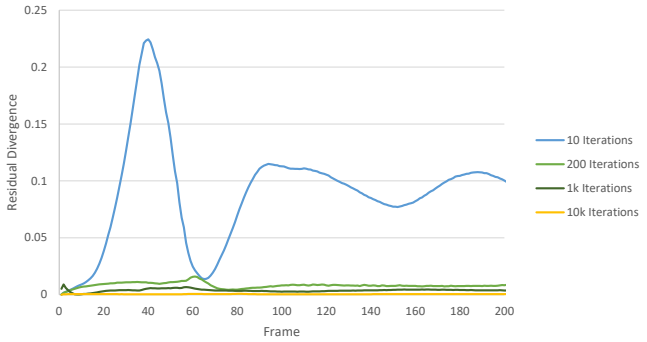



Fig. 10. Average absolute residual divergence per cell at each frame of the animation shown in Figure 11.

In Figure 10 we plot the average absolute residual divergence per cell of the velocity field used for advection in a fluid-deformable interaction scene for each frame and for varying Jacobi iteration counts. While residual divergence can be completely removed after about 10k iterations, this results in a low FPS count (23 FPS in the shown scene, versus 126 FPS when using 200 iterations). On the other hand, the simulations using 200 iterations and 10000 iterations (snapshots in Figure 11) are visually close, such that we decide to limit the number of Jacobi iterations to enable real-time performance. Moreover, in our experiments, a high iteration count yields more numerical dissipation. Lastly, note that our method is not limited to 200 Jacobi iterations, and most examples in this paper will still maintain real-time rates for much higher iteration counts.

[3] When using twin-subspaces, a synchronization also takes place (Algorithm 1, Line 7), where solid particles are moved such that they form a subspace configuration. For the subspace sizes used in our examples, we found that the positional changes are negligible compared to integration errors resulting from advection.

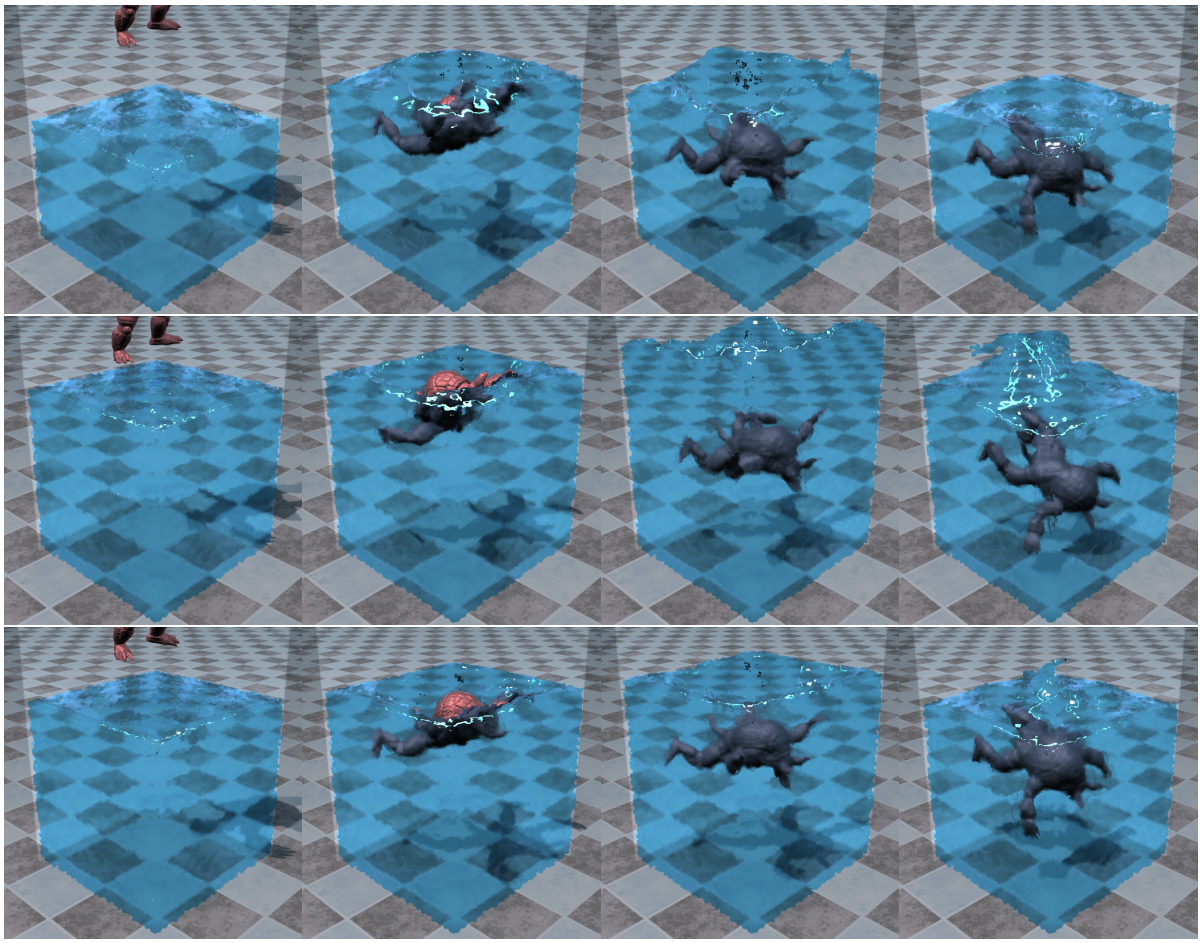


Fig. 11. The same simulation of an armadillo dropping in a pool with 10 (top row), 200 (middle row) and 10000 (bottom row) Jacobi iterations. While volume loss is clearly visible for a low number of iterations, the results for 200 iterations (which we use in all experiments) and 10000 iterations (which yields almost zero residual) are visually close, the lower iteration count even leading to less numerical dissipation. Frames 30, 70, 100 and 200 are shown, corresponding to the residual divergence plots in Figure 10.

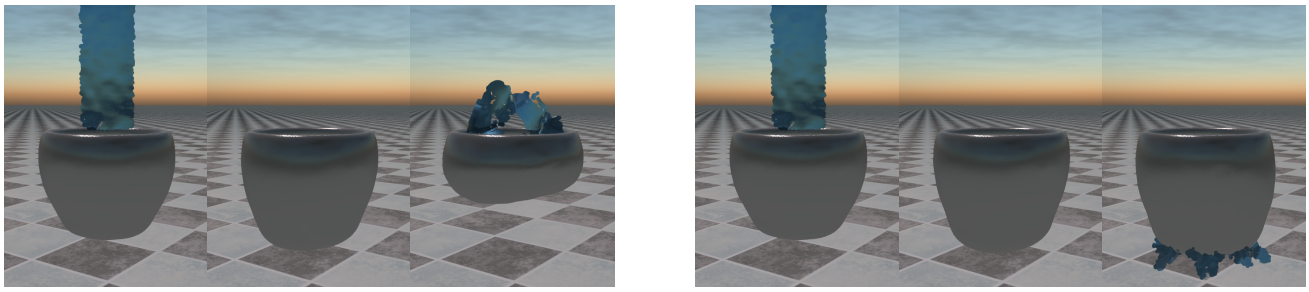


Fig. 12. The same simulation of a cup, which is kept fixed at the opening and filled with water, on the left with using the penetration constraints from Section 3.2.5, and on the right without, where, due to high velocities and a large time-step (1/60s), fluid particles are able to move through and into the thin bottom of the cup.