Interlinked SPH Pressure Solvers for Strong Fluid-Rigid Coupling

CHRISTOPH GISSLER, University of Freiburg, Germany and FIFTY2 Technology GmbH, Germany
ANDREAS PEER, University of Freiburg, Germany
STEFAN BAND, University of Freiburg, Germany
JAN BENDER, RWTH Aachen University, Germany
MATTHIAS TESCHNER, University of Freiburg, Germany

We present a strong fluid-rigid coupling for SPH fluids and rigid bodies with particle-sampled surfaces. The approach interlinks the iterative pressure update at fluid particles with a second SPH solver that computes artificial pressure at rigid body particles. The introduced SPH rigid body solver models rigid-rigid contacts as artificial density deviations at rigid body particles. The corresponding pressure is iteratively computed by solving a global formulation which is particularly useful for large numbers of rigid-rigid contacts. Compared to previous SPH coupling methods, the proposed concept stabilizes the fluid-rigid interface handling. It significantly reduces the computation times of SPH fluid simulations by enabling larger time steps. Performance gain factors of up to 58 compared to previous methods are presented. We further show its applicability to a recent SPH solver for elastic objects. Large scenarios with up to 90M particles of various interacting materials and complex contact geometries with up to 90k rigid-rigid contacts are shown. We demonstrate the competitiveness of our proposed rigid body solver by comparing it to Bullet.


Additional Key Words and Phrases: physically-based animation, fluid simulation, Smoothed Particle Hydrodynamics, fluid-rigid coupling, rigid-rigid contacts

ACM Reference Format:

1 INTRODUCTION

Particle-sampled solids are a popular basis for the boundary handling in SPH fluid simulations (see e.g. [Ilmen et al. 2014b]). The density computation at fluid particles considers contributions from nearby boundary particles and the pressure — derived from density deviations — induces interface forces. This concept works for one-way and two-way coupling where the interface forces do not only affect the fluid velocity field, but also the velocity of rigid bodies.

When implemented in iterative pressure solvers, the computed contact forces are often only applied after finishing the pressure computation. E.g., Akinci et al. [2012] use this concept in combination with PCISPH [Solenthaler and Pajarola 2009], Ilmen et al. [2014a]...
in combination with IISPH and Bender and Koschier [2017] in combination with DFSPH. In these works, the velocities of rigid particles are kept constant during the solver process that iteratively updates the fluid pressure. We show that this concept causes substantial issues in the two-way coupling for practically relevant scenarios. E.g., instabilities can cause void regions at fluid-rigid interfaces and the simulation result is severely distorted. A simplistic solution would be a reduced time step size. This, however, lowers the performance and is also difficult to implement as the reduced step size cannot be simply derived from the CFL condition.

Contribution: We propose to stabilize the handling of SPH fluid-rigid interfaces by using a strong fluid-rigid coupling, where the velocities of rigid body particles are updated in each iteration of the fluid pressure solver. Our update considers fluid-rigid and rigid-rigid contacts that are uniformly handled with the SPH concept. While the fluid solver is responsible for the fluid-rigid contact forces, a novel system is solved to compute rigid-rigid contact forces. We therefore propose an approximate SPH formulation that works with particle-sampled rigid surfaces. If objects are in contact, an artificial SPH density deviation is determined which results in the respective pressure and contact forces that are computed using an implicit formulation.

We show that our strong coupling of fluid pressure and rigid body velocities significantly reduces instabilities, i.e. void regions at fluid-solid interfaces. We further show scenarios where a similar simulation quality can only be achieved by previous boundary handling approaches, if the time step size is reduced by one or two orders of magnitude, resulting in a performance gain factor of up to 58 for our method. Independently from the fluid-rigid coupling, we also demonstrate the competitiveness of our rigid body solver by comparing it to Bullet [Coumans 2018] and showing stability advantages and performance gains by up to a factor of 9.

Our approach can be combined with iterative SPH pressure solvers and with a boundary handling that computes fluid-rigid interface forces. Although the proposed two-way coupling is tightly interconnected with the fluid solver, it can still flexibly be used with various SPH pressure solvers. We emphasize this by deriving our concept for a generic SPH fluid solver and by presenting experiments with four different solvers: IISPH, DFSPH, a solver for highly viscous SPH fluids [Peer et al. 2015] and a solver for elastic SPH solids [Peer et al. 2018]. Our approach does not only handle large scenarios with up to 90M particles as shown in Fig. 1. It also handles complex contact geometries with up to 90k simultaneous rigid-rigid contacts.

2 RELATED WORK

The graphics community investigates the two-way coupling of fluids with solid objects in the context of various fluid solver concepts, e.g. two-way coupling for height-field fluids [Chentanez and Müller 2010; Solenthaler et al. 2011; Thürey et al. 2007], for Lattice-Boltzmann fluids [Thürey et al. 2006], for the Lagrangian vortex method [Vines et al. 2014], for model-reduced fluids [Gerszewski et al. 2015] or for MPM fluids [Yan et al. 2018].

Traditionally, a lot of interesting approaches work with Eulerian fluids. E.g., the fluid-rigid coupling proposed by Carlson et al. [2004], and also used by Kwatra et al. [2010] to simulate an articulated swimmer, treats regions occupied by rigid bodies as fluid during the pressure solve. Although solid regions are considered in the pressure solve, the rigidity constraint of these regions is only fulfilled in a second step. This is resolved by Klingner et al. [2006], where fluid and rigid constraints are simultaneously enforced. An extension to deformable solids is presented by Chentanez et al. [2006]. Lentine et al. [2011] and Tan et al. [2011] both present an improved two-way coupling for articulated swimming creatures. A strong coupling with FLIP fluids is proposed by Battet al. [2007], where rigid bodies are incorporated into the solver formulation. Rigid-rigid contacts, however, are handled sequentially in a separate step. The FLIP-based stream function solver presented by Ando et al. [2015] treats solid regions as fluid similar to Carlson et al. [2004], but presents a monolithic solver for the fluid-rigid coupling. Rigid-rigid contacts are not discussed. Guendelman et al. [2005] focus on the Eulerian coupling of deformable thin shells, where fluid and solids are solved separately. In contrast, Robinson-Mosher et al. [2008] introduce a monolithic solver based on an implicit formulation that also works for thin shells. In this context, Robinson-Mosher et al. [2009] propose an improved handling of tangential velocities at interfaces. Monolithic fluid-solid coupling and the investigation of the solvability of the respective systems are addressed, e.g., by Grétarsson et al. [2011]; Patkar et al. [2016]; Robinson-Mosher et al. [2011]. The method of Robinson-Mosher et al. [2011] is used and extended towards Chimera grids by English et al. [2013]. Lu et al. [2016] also build on the approach of Robinson-Mosher et al. [2011] to couple fluids with reduced deformable objects. A recent cut-cell method for the strong coupling of Eulerian fluids and deformable bodies is presented by Zarifi and Battet al. [2017].

Our paper focuses on Lagrangian fluid formulations. Müller et al. [2004] show the interaction of fluids and deformable objects. Fluid and rigid objects are solved sequentially by Clavet et al. [2005]. In the rigid update step, the fluid particles are considered as rigid and their penetration into rigid bodies is resolved. Oger et al. [2006] integrate pressure values of the fluid particles over the surface of the rigid to compute the coupling forces. Keiser et al. [2006] treat rigid body particles as fluid to compute interacting forces similar to the forces used by Carlson et al. [2004]. As these approaches base on explicit formulations, the simulation time step is rather limited. A predictor-corrector scheme for two-way fluid-rigid coupling is proposed by Becker et al. [2009]. Although the coupling bases on a global formulation to solve for valid relative velocities at interfaces, non-penetration is only enforced in a subsequent correction of particle positions. An alternative two-way coupling with pressure-based SPH boundary forces is introduced by Akinci et al. [2012]. The force formulation is particularly useful for incomplete neighborhoods and non-uniform boundary samplings, i.e. one-layer-boundaries with particles of varying size can be handled. Rigid-rigid contacts and predicted velocities of rigid objects are not updated during the fluid solve. Akinci et al. [2013b] extend the concept of Akinci et al. [2012] to elastic solids. Fujisawa and Miura [2015] propose to adapt the density computation of fluid particles such that no boundary particles are needed. However, their approach does not support two-way coupling. Similarly, Koschier and Bender [2017] propose density maps instead of boundary particles to compute the influence...
of a rigid body onto the fluid. Here, two-way coupling is realized by mirroring the interface forces from the fluid to a rigid similar to the method of Akinci et al. [2012]. Takahashi and Lin [2016] and Takahashi et al. [2017] classify particles depending on the boundary condition to improve the convergence of the fluid solver. The computed interface forces are similar to the ones of Akinci et al. [2012]. Oh et al. [2009] propose an impulse-based fluid-rigid coupling which also can be used to simulate solid-solid interactions with particles. The approach, however, is local and multiple simultaneous contacts are resolved subsequently ignoring previous results. Two-way coupling solutions with particle-sampled rigid objects are also used in frameworks for multiple materials, e.g. [Keiser et al. 2005; Macklin et al. 2014; Solenthaler et al. 2007]. Recently, Akbay et al. [2018] proposed an improved two-way coupling which is applicable to both Eulerian and Lagrangian fluids.

In contrast to the discussed coupling methods, our concept interconnects two SPH pressure solvers to simultaneously resolve fluid-rigid and rigid-rigid contacts. Rigid-rigid contacts are modeled as deviations of an artificial SPH density at rigid body particles that represent the surface of a rigid body. A global formulation is solved to handle rigid-rigid contacts. The two-way coupling is tightly connected to the fluid solver, but can still be applied to various SPH pressure solvers.

3 METHOD

We first motivate the addressed two-way coupling issue in existing iterative SPH fluid solvers in Subsection 3.1. Subsection 3.2 introduces the concept to resolve this issue. In Subsection 3.3, we detail the fluid-rigid force formulations we use. The proposed rigid body solver used for our two-way coupling is introduced in Subsection 3.4. Here, we first derive the implicit formulation that we solve. We then describe the SPH implementation of the respective solver. Finally, in Subsection 3.5, we explain the combination of our rigid body solver with a generic iterative SPH pressure solver.

3.1 Generic iterative pressure solver with the boundary handling of Akinci et al. [2012]

Iterative SPH pressure solvers such as PCISPH [Solenthaler and Pajarola 2009], LPSPH [He et al. 2012], IISPH [Ihmnsen et al. 2014a] and DFSPH [Bender and Koschier 2017] compute a pressure field and velocity changes from pressure gradients to preserve the rest density of the fluid at all particles (see Algorithm 1). A predicted fluid velocity \( v^*_f \) is initialized by applying all non-pressure induced velocity changes and then refined in each solver iteration \( I \). Velocities at rigid particles \( v^*_r \) are predicted accordingly, but not refined during the solver iterations according to the concept of Akinci et al. [2012].

The iterative refinement of fluid velocities \( v^*_f \) depends on a pressure-induced force \( F^p_f \) which in turn depends on the iteratively refined pressure field \( p^l_f \). Finally, the refinement of the pressure field \( p^l_f \) depends on predicted fluid velocities \( v^*_f \) that are updated in each iteration and on predicted velocities of rigid particles \( v^*_r \) that are constant during the iterations.

Keeping rigid body velocities constant during the iterations is a simplification that introduces errors in the pressure computation, i.e. in the computation of the final velocities of the fluid and the rigid objects. This is due to the fact that each iteration computes forces at the fluid-rigid interface which are applied to fluid particles, but are neglected at rigid particles. The pressure-induced fluid-rigid interface forces at rigid particles \( F^p_r \) are only applied after the iterations by using the final pressure field. Rigid-rigid contacts and the resulting forces \( F^rr \) are also computed and applied once after the solver iterations. This means that the predicted velocities at rigid particles \( v^*_r \) are erroneous during the solver iterations. As these velocities influence the pressure computation, the pressure field is negatively affected which finally introduces errors to the velocity field of the fluid.

1. Initialize \( I = 0 \), \( p^l_f, v^*_f, v^*_r \)
2. while Density deviation too large do
3. Compute fluid pressure forces \( F^p_f \)
4. Compute predicted fluid velocity \( v^*_f = v^*_l + (F^p_f) \)
5. Compute pressure \( p^l_f = (v^*_f - v^*_l) \)
6. \( I = I + 1 \)
7. Update fluid
8. Compute fluid-rigid interface forces \( F^p_r \)
9. Compute rigid-rigid contact forces \( F^rr \)
10. Update rigid bodies

Algorithm 1. Generic iterative pressure solver with the boundary handling of Akinci et al. [2012]. Predicted velocities of fluid particles \( v^*_f \) are updated in each iteration \( I \), while rigid particle velocities \( v^*_r \) are kept constant. Please note that actual solvers might update more quantities or in a different order which does not affect the concept.

3.2 Iterative pressure solver with interleaved fluid-rigid velocity update

We propose an improved two-way coupling where forces at the fluid-rigid interface are not only applied to fluid particles, but also to rigid particles in each iteration. I.e., instead of working with a predicted velocity \( v^*_r \) at rigid particles that is constant during the iterations, we propose to update the velocity \( v^*_r \) in each iteration \( I \). In contrast to Algorithm 1, interface forces \( F^p_r \) and forces due to rigid-rigid contact \( F^rr \) refine velocities \( v^*_r \) of rigid particles per iteration as illustrated in Algorithm 2. This refinement now constitutes the actual effect of the interface forces not only onto the fluid velocities, but also onto the velocities of the rigid bodies. The erroneous assumption of constant velocities of rigid particles as in Algorithm 1 is not applied. As the improved predicted velocities \( v^*_r \) influence the pressure refinement \( p^l_f \), they also affect the fluid velocities \( v^*_f \).

3.3 Fluid-rigid interface forces

The coupling between fluid and rigid objects requires the computation of the fluid-rigid interface force \( F^p_r \) as detailed in the previous section. Multiple different options to compute such an interface force have been proposed in previous work, e.g. by Adami et al. [2012]; Akinci et al. [2012]; Band et al. [2018a,b, 2017]; Koschier and Bender [2017]; Monaghan [1994]; Schechter and Bridson [2012]. Our strong coupling formulation does not rely on a specific fluid-rigid interface computation as long as a force can be computed per rigid
1. Initialize $l = 0$. $p_{f}^{0}, v_{f}^{0}, v_{r}^{0}$
2. while Density deviation too large do
3. Compute fluid pressure forces $p_{f}^{l+1}(p_{f}^{l})$
4. Compute predicted fluid velocity $v_{f}^{l+1}(p_{f}^{l})$
5. Compute fluid-rigid interface forces $F_{lr}^{l+1}(p_{f}^{l})$
6. Compute rigid-rigid interface forces $F_{rr}^{l+1}(p_{f}^{l})$
7. Compute predicted rigid velocity $v_{r}^{l+1}(p_{f}^{l}, F_{fr}^{l+1})$
8. Compute pressure $p_{r}^{l+1}(v_{f}^{l+1}, v_{r}^{l+1})$
9. $l += 1$
10. Update fluid
11. Update rigid bodies

Algorithm 2. Generic iterative pressure solver with the proposed interleaved fluid-rigid velocity update. In contrast to Algorithm 1, predicted velocities of fluid and rigid particles — $v_{f}^{l}$ and $v_{r}^{l}$ — are updated in each iteration $l$. As the updated predicted velocities of fluid and rigid particles influence the pressure refinement $p_{f}^{l}$, the interleaved velocity update results in an improved pressure field which in turn results in an improved fluid velocity field and in an improved rigid body movement compared to Algorithm 1.

Both interface forces determine a pressure value for each rigid particle in contact with at least one fluid particle. This pressure value is considered when computing a pressure gradient for the fluid particle using SPH. For two-way coupling, the pairwise force acting from a rigid particle onto a fluid particle is mirrored and assumed to act on the respective rigid. The approaches by Akinci et al. [2012] and Band et al. [2018b] differ in how they determine the pressure value of the rigid particle.

Akinci et al. [2012] determine the pressure value at a rigid particle by assuming that it has the same pressure value as the respective fluid particle. In contrast, Band et al. [2018b] determine the pressure value at a rigid particle by doing an extrapolation from the fluid pressure values onto the boundary. They employ MLS for this pressure extrapolation to improve the robustness to sampling disorder compared to an SPH interpolation.

### 3.4 Rigid body solver

Each iteration of the proposed solver requires the computation of fluid-rigid interface forces $F_{fr}^{l}$ and forces due to rigid-rigid contact $F_{rr}^{l}$. As discussed in the previous section, we employ the fluid-rigid interface forces proposed by Akinci et al. [2012] and Band et al. [2018b].

Motivated by a unified computation of all forces with SPH, we propose to compute the rigid-rigid contact forces $F_{rr}^{l}$ based on artificial density deviations at rigid particles. We therefore assume an artificial rest density, e.g. $\rho_{r}^{0} = 1$. If there is a rigid-rigid contact, we calculate a density $\rho_{r} > \rho_{r}^{0}$. We then determine contact forces $F_{rr}^{l}$ such that $\rho_{r} = \rho_{r}^{0}$ for all rigid particles after applying the forces $F_{rr}^{l}$ to the rigid bodies and updating their state.

This is realized by computing an artificial pressure field $p_{r}$ and deriving artificial pressure forces from the pressure gradient at all rigid particles. These forces cause velocity changes whose divergence cancels the density errors. In the following, we first derive the formulation to compute $p_{r}$ in Subsection 3.4.1. Then, in Subsection 3.4.2, we show how to implement the presented solver with SPH.

#### 3.4.1 System

The proposed system can be derived from the continuity equation $\frac{\partial \rho}{\partial t} = -\rho \nabla \cdot v$ with $\rho$ being the density and $v$ being the velocity of a rigid particle $r$. Discretizing time with a backward difference and introducing the constraint that the density at time $t + \Delta t$ with $\Delta t$ being the simulation time step should be equal to the desired rest density, i.e. $\rho_{r}^{0} = \rho_{r}^{0}$, we get

$$\frac{\rho_{r}^{0} - \rho_{r}}{\Delta t} = -\rho_{r} \nabla \cdot v_{r}^{\text{next}}$$

(1)

with $v_{r}^{\text{next}}$ being the desired velocity of a rigid particle at time $t + \Delta t$ to obtain the desired density $\rho_{r}^{0}$. Note, that the velocity divergence at particle $r$ is zero with respect to other particles of the same rigid body, but may be non-zero with respect to particles of other rigid bodies. The velocity $v_{r}^{\text{next}}$ can be written as

$$v_{r}^{\text{next}} = v_{r}^{\text{next}} + \omega_{R}^{\text{next}} \times r_{r}^{\text{next}}$$

(2)

with $v_{r}^{\text{next}}$ and $\omega_{R}^{\text{next}}$ being the linear and angular velocities of the respective rigid body $R$ at time $t + \Delta t$, respectively (see e.g. [Bender et al. 2014]). $r_{r}^{\text{next}}$ denotes the vector from the center of mass of $R$ to the position of the particle $r$ at time $t + \Delta t$. Using Euler integration and Newton’s second law, we write the linear velocity $v_{r}^{\text{next}}$ as

$$v_{r}^{\text{next}} = v_{r} + \Delta t \left( F_{R} + \sum_{k} F_{k}^{rr} \right)$$

(3)

with $M_{R}$ being the mass of rigid body $R$. The force $F_{R}$ comprises all momentum-changing sources except the unknown rigid-rigid contact forces $F_{rr}^{l}$. This includes, e.g., gravitational force and the fluid-rigid interface forces. The sum considers all particles $k$ of the rigid body $R$. The angular velocity $\omega_{R}^{\text{next}}$ in Eq. (2) can be written as

$$\omega_{R}^{\text{next}} = \omega_{R} + \Delta t I_{R}^{-1} \left( \tau_{R} + (I_{R} \omega_{R}) \times \omega_{R} + \sum_{k} r_{k} \times F_{k}^{rr} \right)$$

(4)

The torque $\tau_{R}$ contains all sources except the unknown rigid-rigid contact forces. Again, the fluid-rigid interface forces are included in $\tau_{R}$. The matrix $I_{R}$ is the inertia tensor of $R$ and the sum considers all particles $k$ of rigid body $R$. Using Eqs. (2) to (4), we write Eq. (1) as

$$\frac{\rho_{r}^{0} - \rho_{r}}{\Delta t} = -\rho_{r} \nabla \cdot \left( v_{r} + \Delta t \frac{1}{M_{R}} F_{R} \right) - \rho_{r} \nabla \cdot \left( \Delta t \frac{1}{M_{R}} \sum_{k} F_{k}^{rr} \right)$$

$$- \rho_{r} \nabla \cdot \left( \omega_{R} + \frac{\Gamma_{r}}{\Delta t} \left( \tau_{R} + \Delta t (I_{R} \omega_{R}) \times \omega_{R} \right) \right) \times r_{r}^{\text{next}}$$

$$- \rho_{r} \nabla \cdot \left( \Delta t \frac{1}{I_{R}} \sum_{k} r_{k} \times F_{k}^{rr} \right) \times r_{r}^{\text{next}}.$$

(5)

Compared to Eq. (1), unknown velocities are replaced by unknown rigid-rigid contact forces in Eq. (5). We introduce the approximation $r_{r}^{\text{next}} = r_{r}$ and the term $s_{r}$ with $s_{r} = \frac{\rho_{r}^{0} - \rho_{r}}{\Delta t} + \rho_{r} \nabla \cdot v_{r}^{\text{next}}$ with $v_{r}^{\text{next}} = v_{r} + \Delta t \frac{1}{M_{R}} F_{R} + (\omega_{R} + \frac{\Gamma_{r}}{\Delta t} (\tau_{R} + \Delta t (I_{R} \omega_{R}) \times \omega_{R})) \times r_{r}$. Accordingly,
with unknown artificial pressure. We have

\[ s_r = -\rho_r \nabla \cdot \left( \Delta t \frac{1}{M_r} \sum_k F^v_k + \left( \Delta t I_R^{-1} \sum_k r_k \times F^s_k \right) \times \mathbf{r}_r \right) \quad (6) \]

which can be written as

\[ s_r = -\rho_r \nabla \cdot \left( \Delta t \sum_k \mathbf{K}_r \mathbf{F}^s_k \right) \quad (7) \]

with \( \mathbf{K}_r = \frac{1}{M} \mathbf{I} - \mathbf{r}_s \mathbf{K}_r^{-1} \mathbf{r}_s \) where \( \mathbf{I} \) is the identity matrix and \( \mathbf{r}_s \) is the skew-symmetric cross-product matrix of vector \( \mathbf{r}_r \). The matrix \( \mathbf{K} \) is referred to as collision matrix and has been proposed by Mirtich [1996]. Equation (7) applies to particle \( r \) of rigid body \( R \) and the sum considers all particles \( k \) of the same rigid body \( R \). For two particles \( r \) and \( k \), the term \( \Delta t \mathbf{K}_r \mathbf{F}^s_k \) gives the velocity change at particle \( r \) due to the contact force \( \mathbf{F}^s_k \) at particle \( k \). Equation (7) shares the same positive characteristics as other implicit formulations, e.g. [Xu et al. 1996; Gingold and Monaghan 1977; Lucy 1977; Monaghan 2012; Müller et al. 2003; Stam and Fiume 1995]. \( W_{ij} \) is a kernel function, where we use the cubic spline kernel as, e.g., described by Monaghan [2005]. The sum considers all particles \( j \) within a given distance to the position of sample \( i \). In rest state, fluid particles are sampled at distance \( h \) and the support of our kernel function is \( 2h \). We also use SPH to compute spatial derivatives. In particular, we use \( \nabla A_i = \rho_i \sum_j m_j \left( \frac{A_i}{\rho_j^2} + \frac{A_i}{\rho_j^2} \right) \nabla W_{ij} \) to approximate the gradient and \( \nabla \cdot A_i = \frac{1}{\rho_i} \sum_j m_j (A_i - A_j) \nabla W_{ij} \) to approximate the divergence. These approximations are commonly used. They are variants of the generalized SPH derivative operators as, e.g., discussed by Price [2012].

Artificial rest volume: Following, e.g. Solenthaler and Pajarola [2008] or Akinci et al. [2012], the rest volume \( V_0^r \) of a rigid particle is computed as \( V_0^r = \frac{1}{\gamma} \sum V_w^k \). The sum considers rigid particles \( k \) of the same object within the kernel support at particle \( r \). We use \( \gamma = 0.7 \). We motivate our choice of this value and demonstrate the result of changing \( \gamma \) in detail in Appendix A.

Artificial rest density: The artificial rest density \( \rho_r^0 \) of a rigid particle is set to one. As we parameterize the rigid body solver with a relative density error at rigid particles, this choice is irrelevant for the simulation.

Artificial density: The actual density \( \rho_r \) of a rigid particle is computed as \( \rho_r = \sum \rho_i^0 V_i^0 W_{ik} \). Here, the sum considers all particles \( k \) from all rigid bodies within the kernel support at \( r \). If other objects are in contact at the location of particle \( r \), we get a density deviation, i.e. \( \rho_r > \rho_r^0 \).

Artificial volume: The actual volume \( V_r \) in case of \( \rho_r > \rho_r^0 \) is computed as \( V_r = \frac{\rho_r^0}{\rho_r} V_r^0 \).

Computation of \( s_r \): The computation of \( s_r \) requires the computation of \( F_R \) and \( \mathbf{r}_R \) which include all forces and torques except the rigid-rigid contact forces. As our proposed rigid body solver is integrated into an iterative fluid solver, \( s_r \) is updated in each iteration of the rigid solver due to the fact that the fluid-rigid interface forces are updated in each iteration of the fluid solver. All other forces contributing to \( F_R \) and \( \mathbf{r}_R \) are assumed to be constant and we compute them once in the initialization step of the rigid body solver.

The term \( s_r \) further requires the divergence \( \nabla \cdot v^* \) which is computed as:

\[ \nabla \cdot v^* = \frac{1}{\rho_r} \sum_k V_k \rho_k (v_k^* - v^*_r) \cdot \nabla W_{rk}. \quad (9) \]

The sum considers rigid neighbors \( k \) of \( r \) with particles \( k \) belonging to rigid bodies \( K, r \) to a rigid body \( R \) and \( R \notin K \). This is due to the fact that the divergence term encodes density changes due to the relative movement of rigid bodies \( K \) in contact with \( R \). For neighboring particles in \( R \), the divergence term should be zero.

Computation of \( \rho_r \cdot \nabla \cdot (\Delta t \sum_k V_k \mathbf{K}_r \mathbf{F}^v_k \nabla \rho_k) \) of Eq. (8): A first loop over all rigid particles computes \( \nabla \cdot \mathbf{F}_{r} = \rho_r \sum_k V_k \rho_k \left( \frac{\rho_r}{\rho_k^2} + \frac{\rho_r}{\rho_k^2} \right) \nabla W_{rk} \). The same loop accumulates \( -\Delta t \frac{1}{M_R} V_r \nabla \rho_r \) and \( -\Delta t \sum_k I_{rr}^k V_{rk} \times \mathbf{r}_r \) per rigid body and stores the result of the two sums as \( \omega^m_{rk} \) and \( \omega^g_{rk} \) at each rigid body, respectively. A second loop over all rigid particles computes the velocity \( \mathbf{v}^*_r = \mathbf{v}^*_r + \omega^m_{rk} \times \mathbf{r}_r \), i.e. \( \mathbf{v}^*_r = s_r \).
with $\Delta t \sum_k V_k \mathbf{K}_{rk} \nabla p_k$. This is the velocity change at particle $r$ due to the applied contact forces $\mathbf{F}^c_k$ derived from $\nabla p_k$ at all particles $k$ of the same rigid body. The second loop also computes the divergence $\rho_r \nabla \cdot (\Delta t \sum_k V_k \mathbf{K}_{rk} \nabla p_k) = -\rho_r \nabla \cdot \mathbf{v}^p_r = \sum_k V_k \rho_k (v^p_k - \mathbf{v}^p_r) \cdot \nabla W_{rk}$ with $k$ being neighboring rigid particles of $r$.

**Solver step:** We use a relaxed Jacobi solver, i.e. we update pressure from iteration $l$ to iteration $l+1$ using

$$p^{l+1}_r = p^l_r + \frac{\beta^{RI}}{b_r} (s_r - \rho_r \nabla \cdot (\Delta t \sum_k V_k \mathbf{K}_{rk} \nabla p_k))$$

(10)

with $\beta^{RI}$ being the relaxation coefficient and $b_r$ being the diagonal element of the linear system. Both variables are discussed below. Additionally, we clamp negative values of $p^{l+1}_r$ to 0 to prevent attraction forces due to the SPH pressure gradient.

**Computation of $\beta^{RI}$:** As basis for the relaxation coefficient $\beta^{RI}$, we use a value of 0.5 as proposed by Ihmsen et al. [2014a]. We additionally adapt its value to improve the convergence of our employed relaxed Jacobi solver for solving the rigid-rigid contacts. For this, we follow the idea of Tonge et al. [2012] and divide the relaxation factor by the overall number of contacts of the rigid body to which particle $r$ belongs: $\beta^{RI} = \frac{0.5}{\text{num contacts}}$.

**Computation of $b_r$:** We computeEq. (8) for all rigid body particles in the scene and thus get a system of equations that can be written as $\mathbf{s} = \mathbf{B} \mathbf{p}$ with $\mathbf{s}$ and $\mathbf{p}$ denoting vectors that contain $s_r$ and $p_r$ of all rigid particles $r$, respectively. The relaxed Jacobi step in Eq. (10) requires the diagonal elements of matrix $\mathbf{B}$ with $b_r$ denoting the diagonal element in line $r$. Therefore, we accumulate all coefficients of $p_r$ in Eq. (8) to compute $b_r$.

**Friction:** We employ an explicit per-particle friction force based on the Coulomb model (see e.g. [Bender et al. 2014]) to handle friction between rigid bodies. First, we compute a normal at each rigid particle $r$ in contact as $n_r = x_r - \frac{\sum_k x_k W_{rk}}{\sum_k W_{rk}}$. We furthermore compute the averaged relative predicted velocity due to other external forces at particle $r$ and map this to a normal velocity and tangential velocity $\mathbf{v}^{rel,1}_r$ using $n_r$. We estimate a contact impulse based on the normal velocity which together with the tangential velocity direction allows us to compute a friction force using the Coulomb model. To ensure that the friction force only decreases the relative velocity of the whole rigid body, we clamp its value to a maximum of $\frac{1}{\Delta t \text{num contacts}} \frac{1}{\sum_k \mathbf{K}_{rk}} \mathbf{v}^{rel,1}_r$ per particle where $t_r$ is the computed tangent at $r$.

### 3.5 Combining fluid and rigid body solver

The rigid body solver computes the rigid-rigid contact forces and the predicted rigid velocities in lines 6 and 7 of the proposed concept in Algorithm 2. Algorithm 3 shows the same concept, extended by the embedding of the introduced rigid body solver.

The fluid and rigid body solvers are closely coupled. The fluid solver provides fluid-rigid interface forces $\mathbf{F}^f_r$ to the rigid body solver. In particular, $\mathbf{F}^f_r$ contributes to $\mathbf{F}_r$ and $\mathbf{r}_r$ used in $\mathbf{v}^c_r$ and finally in $s_r$ (see Eq. (9) and line 6 in Algorithm 3). In turn, the rigid body solver computes updated predicted rigid velocities $\mathbf{v}^c_r$ that are used by the fluid solver. The fluid solver computes predicted density deviations from the divergence of predicted particle velocities, e.g. IIISP, or from predicted positions using predicted velocities, e.g. PCISPH. Thus, $\mathbf{v}^c_r$ influences the pressure computation at fluid particles.

The order of the rigid body computations in lines 6, 7, 8 and 10 in Algorithm 3 is determined by the fact that these computations constitute a solver iteration for Eq. (8). The term $s_r$ in line 6 is the left-hand side of this equation, while $\mathbf{F}^f_r$ in line 7 and $\mathbf{v}^c_r$ in line 8 are estimated during the computation of the right-hand side of Eq. (8). Note that $\mathbf{v}^c_r = \mathbf{v}^p_r + \mathbf{v}^f_r$ and $\mathbf{v}^{rel,1}_r = \Delta t \sum_k V_k \mathbf{K}_{rk} \nabla p_k$. After the computations in lines 6, 7 and 8, the solver step can be performed in line 10 using Eq. (10).

The proposed concept interlinks two pressure solvers. As each update of the fluid solver affects the rigid solver and each rigid solver step influences the next fluid solver step, it is natural to couple them in a one-to-one manner as realized in Algorithm 3.

When solving both the fluid and the rigid-rigid collision in a coupled system as shown in Algorithm 3, a unified stopping criterion must be used. Our used stopping criterion is fulfilled when the stopping criterion of our employed fluid solver is fulfilled and additionally the average density deviation of all rigid particles in contact with other rigid objects is smaller than 0.1%.

### 3.6 Discussion

We propose to couple our novel pressure-based rigid body solver with an existing iterative fluid pressure solver in order to achieve strong coupling. By coupling them as shown in Algorithm 3, we no longer have a partitioned solver for fluid dynamics and rigid body dynamics in contrast to the current state-of-the-art approach by Akinci et al. [2012]. In fact, our approach describes a monolithic, unified system that solves fluid and rigid dynamics simultaneously.
There is an equation for each fluid and rigid particle in the scene with an unknown pressure value for each respective particle.

Chentanez et al. [2006] and Klingner et al. [2006] initially proposed to solve fluid and solids simultaneously to achieve a strong coupling. In contrast to their approach which uses an Eulerian fluid discretization and a Lagrangian solid discretization, we use a unified SPH-based discretization for fluids and rigid objects. For Position Based Dynamics methods, there exist different approaches to handle rigid-rigid and two-way coupling constraints. Macklin et al. [2014] first consider the rigid particles belonging to the same rigid object as being unconnected and then later constrain the movement of these particles to a rigid body movement using shape matching. Instead, we directly consider the rigid body movement in our system of equations. This is similar to the approach of Deul et al. [2014], who integrate the rigid body motion into the rigid-rigid constraints.

4 RESULTS

We have combined our strong two-way coupling with two fluid solvers, IISPH [Ihmsen et al. 2014a] and DFSPH [Bender and Koschier 2017], with a solver for highly viscous fluids [Peer et al. 2015] and also with a solver for elastic solids [Peer et al. 2018]. Vorticity is modeled with the micropolar formulation of Bender et al. [2017]. Viscosity follows the idea of Morris and et al. [1997]. Fluid-air interaction in free-surface scenarios is modeled with the drag force proposed by Gissler et al. [2017]. Surface tension is based on the method of Akinci et al. [2013a]. Multi-phase simulations employ the number-density concept of Solenthaler and Pajarola [2008]. We sample the surfaces of triangle meshes with boundary particles with an algorithm that is similar to the one proposed by Bell et al. [2005].

The experiments are organized as follows. Subsection 4.1 illustrates the improved stability and discusses the performance gain of the proposed fluid-rigid coupling compared to the state-of-the-art approach by Akinci et al. [2012]. In Subsection 4.2, we demonstrate the utility of our approach in complex scenarios with large particle numbers, large numbers of interacting rigid bodies, large numbers of rigid-rigid contacts, and with multiple fluid phases. In Subsection 4.3, we compare our rigid body solver with Bullet. Subsection 4.4 emphasizes the advantage of having a unified SPH-based rigid body solver by showing phase transitions and by combining it with highly viscous fluids and elastic solids.

Table 1 summarizes particle numbers for all scenarios. Table 2 shows particle sizes, time steps, iteration counts and computation time per frame. The experiments have been computed on a 16-core 3.1 GHz Intel Xeon workstation. We use a framerate of 50 frames/s for all scenes apart from the AR scene where we use 60 frames/s to match framerate of the video recording.

4.1 Comparison to the method of Akinci et al. [2012]

4.1.1 Rising sphere. A 2D simulation of a sphere with a density of 100 kg m\(^{-3}\) rising in an IISPH fluid with a density of 1000 kg m\(^{-3}\) is used to illustrate stability improvements and performance gain of our concept compared to the method of Akinci et al. [2012]. Figure 2a illustrates that their approach is unstable when using the same time step as for our concept in Fig. 2d which results in a stable simulation. Reducing the time step for the method of Akinci et al. [2012] in

<table>
<thead>
<tr>
<th>Scene</th>
<th>particles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fluid</td>
</tr>
<tr>
<td>Rising sphere</td>
<td>Fig. 2</td>
</tr>
<tr>
<td>Propeller pump</td>
<td>Fig. 3</td>
</tr>
<tr>
<td>Orion splashdown</td>
<td>Fig. 4</td>
</tr>
<tr>
<td>Moored buoys</td>
<td>Fig. 5</td>
</tr>
<tr>
<td>Armadillo drain</td>
<td>Fig. 6</td>
</tr>
<tr>
<td>AR scene</td>
<td>Fig. 7</td>
</tr>
<tr>
<td>Water gate</td>
<td>Fig. 1</td>
</tr>
<tr>
<td>Nut and bolt</td>
<td>Fig. 9</td>
</tr>
<tr>
<td>Duck production</td>
<td>Fig. 10</td>
</tr>
<tr>
<td>Armadillo pool</td>
<td>Fig. 11</td>
</tr>
<tr>
<td>Valley</td>
<td>Fig. 12</td>
</tr>
</tbody>
</table>

Table 1. The maximum number of particles and maximum simultaneous rigid-rigid contacts per scene. For the Armadillo pool and Valley scene, the number of fluid particles also includes the particles of the viscous fluid and deformable objects.

<table>
<thead>
<tr>
<th>Scene</th>
<th>part. size [mm]</th>
<th>time step [ms]</th>
<th>iterations</th>
<th>time per frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rising sphere</td>
<td>Akinci et al. [2012]</td>
<td>50</td>
<td>0.02</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>our concept</td>
<td>50</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Propeller pump</td>
<td>Akinci et al. [2012]</td>
<td>50</td>
<td>0.3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>our concept</td>
<td>50</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>Orion splashdown</td>
<td>Akinci et al. [2012]</td>
<td>200</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>our concept</td>
<td>200</td>
<td>20</td>
<td>19</td>
</tr>
<tr>
<td>Moored buoys</td>
<td>20</td>
<td>0.1</td>
<td>25</td>
<td>18 min</td>
</tr>
<tr>
<td>Armadillo drain</td>
<td>15</td>
<td>0.2</td>
<td>2</td>
<td>58 s</td>
</tr>
<tr>
<td>AR scene</td>
<td>our concept</td>
<td>10</td>
<td>0.25</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>our concept</td>
<td>10</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Water gate</td>
<td>8</td>
<td>0.3</td>
<td>5</td>
<td>15 min</td>
</tr>
<tr>
<td>Nut and bolt</td>
<td>10</td>
<td>0.1</td>
<td>2</td>
<td>19.9 min</td>
</tr>
<tr>
<td>Duck production</td>
<td>5</td>
<td>0.1</td>
<td>1</td>
<td>13.2 s</td>
</tr>
<tr>
<td>Armadillo pool</td>
<td>30</td>
<td>1</td>
<td>6</td>
<td>9 s</td>
</tr>
<tr>
<td>Valley</td>
<td>50</td>
<td>1</td>
<td>8</td>
<td>6 min</td>
</tr>
</tbody>
</table>

Table 2. Particle sizes, maximum time steps, average solver iterations and computation time per frame for all scenes. Comparative numbers for the method of Akinci et al. [2012] with similar simulation results are given for the first three scenes. For the Armadillo pool and Valley scene, pressure solver iterations are given, viscosity and elasticity solver iterations are not included.

Figs. 2b and 2c reduces artifacts and improves the stability, but is more expensive to compute. To get comparable simulation results to our approach, the time step has to be reduced by two orders of
magnitude for the formulation of Akinci et al. [2012], resulting in a performance gain factor of 58 for our technique.

4.1.2 Propeller pump. Figure 3 shows a scenario where a fast rotating propeller pumps fluid from one side of a tank to the other one. The propeller is two-way coupled, accelerated by a force and reaches a maximum speed of 160 rpm. The DFSFP fluid has a density of 1000 kg m$^{-3}$, the rubber ducks have a density of 500 kg m$^{-3}$ and the propeller has a density of 1500 kg m$^{-3}$. The scene is simulated with a time step of 3 ms using our strong coupling method. Using the weak coupling of Akinci et al. [2012] requires a reduced time step of 0.3 ms to obtain a stable result. Accordingly, using our approach results in a performance gain factor of 4.6.

4.1.3 Orion splashdown. The simulated splashdown of an Orion spacecraft capsule with the method of Akinci et al. [2012] and with our approach is compared in Fig. 4. The density of the IISPH fluid is 1000 kg m$^{-3}$ and the density of the capsule is 1200 kg m$^{-3}$. When using the same time step of 20 ms for both approaches, our method is stable, while the weak coupling of Akinci et al. [2012] leads to an artificial void region below the capsule. This void region disturbs the interface handling and results in an unstable simulation. A stable result using their weak coupling can only be obtained with a reduced time step of 2 ms which results in a performance gain factor of 3.4 for our method.

4.2 Rigid body solver

4.2.1 Moored buoys. The scene in Fig. 5 illustrates our two-way coupling for multiphase fluids and also the rigid-rigid contact handling. Three buoys swim in two IISPH fluids with densities of 750 kg m$^{-3}$ and 1500 kg m$^{-3}$. Since the three buoys have different densities (orange: 100 kg m$^{-3}$, green: 1100 kg m$^{-3}$, turquoise: 2000 kg m$^{-3}$), they swim at different levels. The surfaces of the buoys and of the chain elements are particle-sampled. All rigid-rigid contacts of the respective chain elements and the buoys are handled by our proposed rigid body solver.

4.2.2 Armadillo drain. Figure 6 illustrates that our SPH-based rigid body solver can simulate a large number of interacting, geometrically complex rigid bodies. Additionally, the 1500 Armadillos in
this scene are two-way coupled with a DFSPH fluid. The particle numbers for the fluid and the Armadillos and also the number of simultaneous contact points are given in Table 1.

4.2.3 AR scene. In the scene shown in Fig. 7, we use the fluid-rigid interface force proposed by Band et al. [2018b] in contrast to the interface force by Akinci et al. [2012] which we use in other scenes. This demonstrates that we can use our strong coupling approach with different fluid-rigid interface forces. Furthermore, we simulated the scene with different time steps as shown in Table 2 to demonstrate that it is generally beneficial for the overall computation time to use larger time steps.

4.2.4 Water gate. Figure 1 illustrates that our method can handle large-scale scenarios with millions of fluid and rigid body particles and also with tens of thousands of simultaneous rigid-rigid contacts. A water wheel is connected to a water gate with a chain and multiple gears. The chain and the gears are fully simulated with our rigid body solver. The connections between the respective chain elements are purely solved based on rigid-rigid contacts. All dynamic rigid bodies are two-way coupled with an IISPH fluid. Figure 8 shows a close-up of the chain and its particle representation.

4.2.5 Nut and bolt. Our rigid body solver stably simulates complex, interacting, concave geometry with many simultaneous contacts which is demonstrated in the scene shown in Fig. 9. We simulated a bolt inside a nut acting as a valve for water. The bolt is fully modeled as rigid body with six degrees of freedom. Using a screwdriver, the
bolt is slowly loosened until it falls out. In this scene, we employed the fluid-rigid interface force proposed by Band et al. [2018b].

4.3 Comparison to Bullet [Coumans 2018]

To evaluate our rigid body solver independently of the proposed strong fluid-rigid coupling, we compare it to Bullet [Coumans 2018] which is a popular state-of-the-art rigid body engine that implements a PGS LCP solver [Bender et al. 2014].

4.3.1 Armadillo drain. We adapted the Armadillo drain scene shown in Subsection 4.2.2 by removing the fluid to compare our rigid body solver with Bullet [Coumans 2018]. The resulting timings from the comparison are shown in Table 3. Bullet is faster than our solver when discretizing the Armadillo with a single convex hull. However, this simplifies the problem drastically since the Armadillo mesh is highly concave. When using 36 convex hulls, Bullet’s overall computation time is slower compared to our solver where we discretized the Armadillo mesh with 274 boundary particles.

4.3.2 Nut and bolt. We use the nut and bolt from the scene shown in Fig. 9 for a comparison with interacting geometry with a lot of simultaneous contact points. We only simulated the bolt being affected by gravity and resting inside the nut. This is a similar setup to the scene used by Xu et al. [2014] to compare their proposed rigid body solver to Bullet. As proposed by them, we also employ a convex decomposition of the meshes when simulating them with Bullet.

Table 4 shows the result of the comparison. Using our proposed rigid body solver, we were able to stably simulate the 4 s of physical time in 1 min and 21 s. For the same overall computation time, Bullet does not produce a stable simulation as shown in the accompanying video. For Bullet to stably simulate this scene, we needed to reduce the time step to 0.03 ms which resulted in a computation time which was larger by a factor of 9.5 compared to using our solver.

In addition to the computation time disadvantage of Bullet, we needed to create the convex decomposition of the nut and bolt meshes. We were not able to get an automatic convex decomposition of the meshes which was accurate enough for the use with Bullet. Instead, we needed to manually separate the meshes into convex parts which was time consuming. In contrast, with our rigid body solver, only the surface of the rigid body needs to be sampled with particles which can be robustly done, e.g., with the algorithm proposed by Bell et al. [2005].

4.3.3 Discussion. We have shown two examples in which our proposed rigid body solver is faster and more stable than Bullet which shows that our proposed rigid body solver is competitive. However, obviously, one could construct scenes where our solver has disadvantages compared to Bullet. For example, our solver may use a lot of particles to discretize a simple box depending on the particle resolution. Accordingly, Bullet could be faster in scenes where only simple rigid body objects like boxes are involved for which Bullet uses specialized collision algorithms. In contrast, our solver is especially well suited for many interacting, complex and concave rigid bodies.

4.4 Unified SPH solver

4.4.1 Duck production. The duck production scenario in Fig. 10 shows a setup where fluids solidify to rigid bodies and later become fluid again. This shows an advantage of having a unified solver based on SPH for fluid dynamics and rigid body dynamics. It allows to switch the existing fluid particles to be simulated as a rigid body and back to be simulated as fluid without instabilities due to the conversion. Furthermore, since everything is simulated based on particles, it is not necessary to have a triangle mesh or to be able to create a collision shape which would be needed when simulating this scene with for example Bullet [Coumans 2018].

4.4.2 Armadillo pool. To illustrate the flexibility of our concept, various highly viscous [Peer et al. 2015], elastic [Peer et al. 2018] and rigid Armadillos (simulated with our rigid body solver) are two-way coupled with an ISPH fluid. The fluid-viscous and fluid-elastic coupling of the highly viscous and elastic Armadillos is realized using the methods of Peer et al. [2015] and Peer et al. [2018], respectively. Our approach is used for the coupling of the rigid Armadillos to the fluid phase and to the viscous and elastic Armadillos. An image of the scene is shown in Fig. 11.
Interlinked SPH Pressure Solvers for Strong Fluid-Rigid Coupling

5 LIMITATIONS

The accuracy of the rigid-rigid contact handling is related to the size of a rigid body particle which in turn is related to the size of a fluid particle. Thus, the accuracy of the contact handling is coupled to the accuracy of the fluid simulation. Our approach can handle rigid body particle sizes that vary to some extent, but we assume that they are generally similar to the size of a fluid particle. Using fluid and rigid particles of largely different sizes as, e.g., in Winchenbach et al. [2017], is definitely an interesting opportunity for future research as it would decouple the accuracy of the fluid and the rigid body solver. This, however, might induce adjustments in the fluid solver, e.g., adaptive kernel supports, which is currently not required in our coupling approach. Another consequence of using particles for the rigid-rigid contact handling is that their size and velocity influence the maximum allowed time step due to the CFL condition. However, this time step restriction generally applies for particle-based boundary handling when the fluid is in contact with a moving rigid body. Furthermore, in our scenes, the CFL condition of the fluid solver normally dominated the time step.

As already illustrated in Subsection 4.1, the performance gain factor varies and depends on the scenario. Our approach is especially useful in scenes with large velocities or large accelerations at particles. It is also advantageous in scenes with large density ratios. In the rising sphere scenario in Fig. 2, e.g., we use a moderate density ratio of 1:10. If we would use a larger ratio, we could present larger speedups. On the other hand, the performance gain is reduced in scenarios with smaller density ratios and also in calm scenes with slowly moving particles.

6 CONCLUSION

We have introduced a strong two-way coupling method for SPH fluids and rigid bodies with particle-sampled surfaces.

On one hand, the technique shares characteristics of a unified approach by solving pressure at fluid and rigid body particles based on SPH density deviations and also by deriving fluid-rigid and rigid-rigid contact forces from pressure using the SPH methodology. Further, the proposed rigid body solver is closely interconnected with the fluid solver. Each fluid solver iteration influences the rigid body solver and each iteration of the rigid body solver affects the fluid solver.

On the other hand, although we solve a monolithic system, we show that it is a flexible and useful extension for iterative SPH pressure solvers in general. This is emphasized by the description of the coupling concept, where we specify the interface between
fluid and rigid body solver, but keep the fluid solver in a general format. The flexibility is further shown in the experiments, where we combine our coupling with DFSPH, ISPH, highly viscous fluids and elastic solids. Large scenarios with up to 90M particles have been presented.

Our two-way coupling stabilizes the fluid-rigid interface handling. This is illustrated in comparisons with the approach of Akinci et al. [2012]. This stabilization enables larger simulation time steps which corresponds to significant performance improvements of up to a factor of 58. The introduced SPH-based rigid-rigid contact handling does not only promote the two-way coupling, but is also able to handle complex contact geometries between rigid objects with tens of thousands of simultaneous contacts.

In the future, we plan to investigate the applicability of our concept to alternative Lagrangian fluid formulations, i.e. position-based fluids [Macklin and Müller 2013] due to the versatility of the concept and power particles [de Goes et al. 2015] due to the improved accuracy compared to SPH interpolation. Both approaches can handle particle sampled boundaries, but the interface to our two-way coupling has to be investigated. In our current setting, the fluid solver provides fluid-rigid interface forces to the rigid bodies. Further, the fluid solver has to be able to process velocity changes at rigid body particles during the iterations. Additionally, it could be interesting to combine the proposed strong coupling with alternative particle-based rigid body solvers (e.g., solvers based on DEM as proposed by Bell et al. [2005] or Harada [2007]).

ACKNOWLEDGMENTS

The rubber duck by willie is licensed under CC0 1.0. The rubber duck mold by rocketboy is licensed under CC BY-NC 3.0. The nut and bolt models by queezz88 are licensed under CC BY 3.0. The propeller model by Parkinbo is licensed under CC BY-NC 3.0. The Armadillo model is courtesy of the Stanford University Computer Graphics Laboratory. The Orion capsule is courtesy of NASA. The creature model is courtesy of Eric Mootz. The tree model is from www.cadnav.com. The rock models by Andrew Hansen are licensed under CC0 1.0. We thank FIFTY2 Technology’s team members for their support.

A CHOICE OF $\gamma$

We use a value of 0.7 for $\gamma$ for the computation of the artificial rest volume of a rigid particle in Subsection 3.4.2. This follows from the following observation. If a volume is uniformly sampled with particles at distance $h$, we get $\frac{1}{V_{r,k}} = \frac{1}{r^3}$ for a particle $r$ with complete neighborhood and neighbors $k$, i.e. its volume $h^3$ is correctly computed from the inverse of the number density. If we remove particles from the neighborhood of $r$, its volume would be overestimated. These missing contributions can be compensated by $\gamma$. If 3D particles are uniformly sampled in a 2D plane which we assume to be the case in our rigid body representation, the desired volume would still be $h^3$, but we get $\frac{1}{V_{r,k}} \approx \frac{h^3}{0.7}$. Computing $0.7 \frac{1}{V_{r,k}} = 10$ instead, i.e. $\gamma = 0.7$, results in the desired volume $h^3$. The effect of using different values for $\gamma$ is demonstrated in Fig. 13.

REFERENCES


Impulse-based Dynamic Simulation of Rigid Body Systems
Brian Vincent Mirtich. 1996.

Wenlong Lu, Ning Jin, and Ronald P. Fedkiw. 2016. Two-way Coupling of Fluids
Dan Koschier and Jan Bender. 2017. Density Maps for Improved SPH Boundary Han-
Richard Keiser, Bart Adams, Dominique Gasser, Paolo Bazzi, Philip Dutre, and Markus
Christoph Gissler, Stefan Band, Andreas Peer, Markus Ihmsen, and Matthias Teschner.
R. Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. 2013. Chimera Grids for Water
Crispin Deul, Patrick Charrier, and Jan Bender. 2014. Position-Based Rigid Body
Mathieu Desbrun, Marie-Paule Cani, et al
Fernando de Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun.

ACM Trans. Graph., Vol. 38, No. 1, Article 5. Publication date: January 2019.
