

Implicit Incompressible SPH

Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner

Abstract—We propose a novel formulation of the projection method for Smoothed Particle Hydrodynamics (SPH). We combine a symmetric SPH pressure force and an SPH discretization of the continuity equation to obtain a discretized form of the pressure Poisson equation (PPE). In contrast to previous projection schemes, our system does consider the actual computation of the pressure force. This incorporation improves the convergence rate of the solver. Furthermore, we propose to compute the density deviation based on velocities instead of positions as this formulation improves the robustness of the time-integration scheme. We show that our novel formulation outperforms previous projection schemes and state-of-the-art SPH methods. Large time steps and small density deviations of down to 0.01 percent can be handled in typical scenarios. The practical relevance of the approach is illustrated by scenarios with up to 40 million SPH particles.

Index Terms—Physically based simulation, fluid dynamics, smoothed particle hydrodynamics, incompressibility, implicit integration

1 INTRODUCTION

THE SPH method [1] has proven to be effective at generating compelling water effects in computer graphics. Enforcing incompressibility in SPH is important for the visual quality of a simulated scenario and this paper contributes to that topic.

Standard SPH (SSPH) uses an equation of state (EOS) to compute pressure that results in forces penalizing the current compression [1]. Pressure is computed from locally evaluated compression weighted with a user-defined stiffness value. The straightforward computation of pressure values makes SSPH well suited for efficient simulations of compressible fluids, for example, [2], [3]. For weakly compressible fluids (WCSPH), for example, [4], [5], a rather stiff EOS is used that imposes a severe time step restriction limiting the overall performance.

The performance of SSPH has been significantly improved by EOS-based predictor-corrector schemes, for example, PCISPH [6] and local Poisson SPH [7]. Pressure forces are modeled as constraint forces that resolve compression induced by nonpressure forces. The respective pressure values are computed by iteratively predicting and correcting the particle positions based on an EOS. The approaches still rely on local information, but in contrast to SSPH and WCSPH, the EOS does not contain a user-defined stiffness parameter. PCISPH and local Poisson SPH handle time steps that are up to two orders of magnitude larger compared to WCSPH, while the overall speedup with respect to WCSPH can be up to 55 or 23, respectively [6], [7].

As an alternative to EOS approaches with locally computed penalty forces, projection schemes, also referred to as splitting [8], can be used to compute the pressure field in SPH. First, intermediate velocities are predicted without considering the pressure forces. Then, a PPE is solved to compute pressure such that the resulting pressure forces correct the intermediate velocities to a divergence-free state. This is a standard technique in grid-based approaches, for example, [8], [9], [10], but a detailed discussion of grid-based variants is beyond the scope of this paper. In Lagrangian approaches, projection schemes can be distinguished with respect to the source term in the Poisson formulation. Here, either the divergence of the intermediate velocity field, for example, [11], [12], the compression due to the intermediate velocity field, for example, [13], [14], [15], or a combination of both, for example, [16], [17], are used. As incompressibility is an important ingredient for realistic fluid animations, the compression formulation seems to be preferable in SPH. As discussed for SPH in [11], and for the *moving particle semi-implicit* method (MPS) in [12], only using the divergence term tends to result in perceivable compression.

The numerically challenging approximation of the Laplacian with SPH is a major issue in SPH projection schemes. Directly discretizing the Laplace operator with second-order kernel derivatives is known to be sensitive to the sampling. This is discussed in, for example, [18], [11], where approximations are proposed. To avoid the SPH discretization of the Laplacian, some authors propose to compute the pressure field on a background grid at a different, typically lower resolution, for example, [19], [17], [20], [21]. After transferring the pressure values from the grid to the particles, particle pressure values can be refined, for example, using an EOS as in [21].

SPH projection schemes, also referred to as incompressible SPH (ISPH) methods, are currently considered impractical in the context of computer graphics. As stated in, for example, [11], [22], the performance of ISPH does not scale well with the problem domain and is particularly an issue for large-scale scenarios. Our work addresses this point. We propose a discretization of the PPE that significantly improves the convergence of the solver and

• M. Ihmsen, J. Cornelis, and M. Teschner are with the Department of Computer Science, Faculty of Engineering, Albert-Ludwigs-University Freiburg, Freiburg im Breisgau 79110, Germany.
E-mail: ihmisen@informatik.uni-freiburg.de.

• B. Solenthaler is with the Computer Graphics Laboratory, Department of Computer Science, ETH Zuerich, Zuerich 8092, Switzerland.

• C. Horvath is with the Pixar Animation Studios, 1215 45th Street, Emeryville, CA 94608.

Manuscript received 12 Apr. 2013; revised 11 July 2013; accepted 13 July 2013; published online 25 July 2013.

Recommended for acceptance by S. Hu.

For information on obtaining reprints of this article, please send e-mail to: tcvg@computer.org, and reference IEEECS Log Number TVCG-2013-04-0100. Digital Object Identifier no. 10.1109/TVCG.2013.105.

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on November 06, 2023 at 13:34:31 UTC from IEEE Xplore. Restrictions apply.
1077-2626/14/\$31.00 © 2014 IEEE Published by the IEEE Computer Society

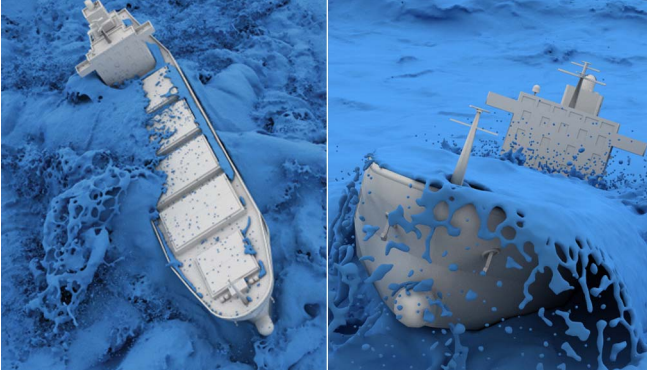


Fig. 1. A cargo ship in highly agitated water. Up to 30 million particles are processed in 44 seconds per simulation step with a time step of 0.004 seconds and a compression of less than 0.1 percent.

the stability of the time-integration scheme. This results in a significant speedup not only compared to previous ISPH approaches, but more importantly compared to PCISPH, the current state-of-the-art. Large time steps and small density deviations of down to 0.01 percent can be handled in typical scenarios. In contrast to previous projection schemes, the approach scales well with the simulation domain. This is demonstrated on a number of large-scale scenarios with up to 40 million fluid particles computed on a 16-core desktop PC, for example, Fig. 1.

2 IMPLICIT INCOMPRESSIBLE SPH (IISPH)

SSPH computes the density of particle i at time t as $\rho_i(t) = \sum_j m_j W_{ij}(t)$, where m_j denotes the mass of a particle j and $W_{ij}(t) = W(\mathbf{x}_i(t) - \mathbf{x}_j(t))$ is a kernel function with finite support. \mathbf{x}_i and \mathbf{x}_j denote the position of particles i and j . Pressure $p_i(t)$ is computed with an EOS [4] as

$$p_i(t) = \frac{\kappa \rho_0}{\gamma} \left(\left(\frac{\rho_i(t)}{\rho_0} \right)^\gamma - 1 \right), \quad (1)$$

where ρ_0 is the rest density of the fluid, κ and γ control the stiffness. As proposed in [1], momentum-preserving pressure forces are computed as

$$\mathbf{F}_i^p(t) = -m_i \sum_j m_j \left(\frac{p_i(t)}{\rho_i^2(t)} + \frac{p_j(t)}{\rho_j^2(t)} \right) \nabla W_{ij}(t). \quad (2)$$

Pressure forces in SSPH penalize compression, but do not guarantee an incompressible state at time $t + \Delta t$.

In contrast to SSPH, the proposed IISPH method computes pressure by iteratively solving a linear system. To build the system with unknown pressure values, we mainly employ a discretized form of the continuity equation and the projection (splitting) concept, for example, [8]. Additional assumptions, approximations or simplifications are avoided. The linear system can be solved efficiently with a matrix-free implementation.

2.1 Derivation

IISPH is based on a semi-implicit form of the density prediction using the time rate of change of the density. The formulation is obtained by directly discretizing the continuity equation $\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v}$ at time t using a forward difference for the time derivative of the density $\frac{\rho_i(t+\Delta t) - \rho_i(t)}{\Delta t}$.

and the SPH concept for the divergence of the velocity $\nabla \cdot \mathbf{v}_i = -\frac{1}{\rho_i} \sum_j m_j \mathbf{v}_{ij} \nabla W_{ij}$, which yields

$$\frac{\rho_i(t + \Delta t) - \rho_i(t)}{\Delta t} = \sum_j m_j \mathbf{v}_{ij}(t + \Delta t) \nabla W_{ij}(t). \quad (3)$$

This specific discretization introduces unknown relative velocities $\mathbf{v}_{ij}(t + \Delta t) = \mathbf{v}_i(t + \Delta t) - \mathbf{v}_j(t + \Delta t)$ that depend on unknown pressure forces at time t which are linear in unknown pressure values at time t .

Using a semi-implicit Euler scheme for position and velocity update, the velocity in (3) can be rewritten as: $\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_i^{adv}(t) + \mathbf{F}_i^p(t)}{m_i}$ with unknown pressure forces $\mathbf{F}_i^p(t)$ and known nonpressure forces $\mathbf{F}_i^{adv}(t)$ such as gravity, surface tension and viscosity. Following the projection concept, we consider intermediate (predicted) velocities $\mathbf{v}_i^{adv} = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_i^{adv}(t)}{m_i}$ which result in an intermediate density

$$\rho_i^{adv} = \rho_i(t) + \Delta t \sum_j m_j \mathbf{v}_{ij}^{adv} \nabla W_{ij}(t). \quad (4)$$

We now search for pressure forces to resolve the compression, i.e., the deviation from the rest density:

$$\Delta t^2 \sum_j m_j \left(\frac{\mathbf{F}_i^p(t)}{m_i} - \frac{\mathbf{F}_j^p(t)}{m_j} \right) \nabla W_{ij}(t) = \rho_0 - \rho_i^{adv}. \quad (5)$$

Note, that (5) corresponds to (3) with $\rho_i(t + \Delta t) = \rho_0$. Using (2) in (5), we get a linear system $\mathbf{A}(t)\mathbf{p}(t) = \mathbf{b}(t)$ with one equation and one unknown pressure value $p_i(t)$ for each particle. $\mathbf{b}(t)$ corresponds to the right-hand side of (5), i.e., $b_i(t) = \rho_0 - \rho_i^{adv}$. For each particle, we finally have an equation of the form

$$\sum_j a_{ij} p_j = b_i = \rho_0 - \rho_i^{adv}, \quad (6)$$

where we have skipped the time index t to improve the readability. This is also done in the following:

2.2 Discussion

Equation (4) corresponds to the prediction step of the projection, while (5) is the PPE that is used for the pressure computation. The respective pressure forces are applied in the correction step of the projection scheme: $\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i^{adv} + \Delta t \mathbf{F}_i^p(t)/m_i$. The source term states the density invariance condition which we prefer over the alternative divergence term. The divergence term tends to result in problematic compression as discussed in, for example, [11], [12]. Thus, our approach is closely related to ISPH schemes which employ density constraints, for example, [13]. However, our formulation significantly improves the convergence of the solver and the stability of the time-integration scheme due to two novel aspects discussed in the following:

Discretization of the Laplacian. Our discretization of the PPE differs from previous formulations that employ additional approximations for the Laplace operator, for example, [11], [13], [14], [16]. These ISPH methods start with a continuous form of the PPE. Then, they discretize the Laplacian and the source term. The resulting system does not consider the pressure force that is finally derived from the pressure field. Thus, there is no distinguished form of

the pressure force with a special relation to the computed pressure field. In contrast, our derivation considers the relation between pressure and pressure force. First, we do not start with a continuous PPE, but with the continuous continuity equation which is discretized. The main goal of this discretization is the introduction of $\mathbf{v}(t + \Delta t)$ which is expressed with the pressure force term used in the final velocity update. Thereby, we can finally apply the particular form of the pressure force that has been considered in the computation of the pressure field. This incorporation improves the convergence rate and the overall performance as shown in comparisons to ISPH in Section 5.

Source term. In previous ISPH implementations and in predictive-corrective EOS solvers, $\rho_i(t + \Delta t)$ is computed based on predicted positions as

$$\rho_i(t + \Delta t) = \sum_j m_j W(\mathbf{x}_i^* - \mathbf{x}_j^*, h), \quad (7)$$

where \mathbf{x}_i^* equals \mathbf{x}_i^{adv} in ISPH and the predicted positions during iterations in predictive-corrective EOS solvers. However, solving (7) implies a recomputation of the neighborhood. In PCISPH, this overhead is avoided by only updating distances for the current neighborhood. This introduces an error which gets more significant for larger displacements $\Delta \mathbf{x}_i = \Delta t \mathbf{v}_i^{adv} + \Delta t^2 \mathbf{F}_i^p / m_i$.

In contrast, IISPH predicts the density based on velocities (4). As known values $\nabla W_{ij}(t)$ are preferred over unknown values $\nabla W_{ij}(t + \Delta t)$ without affecting the error order of the discretization, the approximative update of the neighborhood is avoided. According to our observations, (4) tolerates significantly larger time steps and, thus, improves the robustness of the time-integration scheme.

On the other hand, the system contains a significantly larger number of nonzero entries compared to previous projection schemes. As (6) contains a nested sum, the coefficients a_{ij} are nonzero for the neighbors j of particle i and for neighbors of neighbors of i . Typically, a particle has 30-40 neighbors [23]. Nevertheless, the system can be solved very efficiently in a matrix-free way as presented in the following:

3 SOLVER

The system can be solved in various ways. For similar systems, grid approaches commonly apply SOR, for example, [24], Conjugate Gradient (CG), for example, [9], or multigrid solvers, for example, [10], while SPH approaches often employ Conjugate Gradient for PPEs, for example, [11]. We have implemented and evaluated relaxed Jacobi and Conjugate Gradient to solve our formulation. Multigrid solvers have not been considered due to their involved setup for irregular samplings.

For the proposed system, the relaxed Jacobi solver is more practical than Conjugate Gradient. We therefore describe implementation details of the relaxed Jacobi solver first in Section 3.1, followed by a discussion of possible reasons for that outcome in Section 3.2.

3.1 Relaxed Jacobi

Employing relaxed Jacobi, we iteratively solve (6) for the individual pressure values p_i as

$$p_i^{l+1} = (1 - \omega)p_i^l + \omega \frac{\rho_0 - \rho_i^{adv} - \sum_{j \neq i} a_{ij} p_j^l}{a_{ii}}, \quad (8)$$

where l denotes the iteration index and ω is called the relaxation factor.

To compute (8), we need to determine a_{ii} and $\sum_{j \neq i} a_{ij} p_j^l$ which can be efficiently computed. For extracting the coefficients, the displacement caused by the pressure force is rewritten as

$$\begin{aligned} \Delta t^2 \frac{\mathbf{F}_i^p}{m_i} &= -\Delta t^2 \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \\ &= \underbrace{\left(-\Delta t^2 \sum_j \frac{m_j}{\rho_i^2} \nabla W_{ij} \right)}_{\mathbf{d}_{ii}} p_i + \sum_j \underbrace{\left(-\Delta t^2 \frac{m_j}{\rho_j^2} \nabla W_{ij} \right)}_{\mathbf{d}_{ij}} p_j, \end{aligned} \quad (9)$$

where $\mathbf{d}_{ii} p_i$ denotes the displacement of i due to pressure value p_i and $\mathbf{d}_{ij} p_j$ the movement caused by the pressure values p_j of neighboring particles j . Plugging (9) into (5) and denoting the neighbors of j with k yields

$$\begin{aligned} \rho_0 - \rho_i^{adv} &= \\ \sum_j m_j \left(\mathbf{d}_{ii} p_i + \sum_j \mathbf{d}_{ij} p_j - \sum_k \mathbf{d}_{jk} p_k \right) \nabla W_{ij}. \end{aligned} \quad (10)$$

Note that $\sum_k \mathbf{d}_{jk} p_k$ includes pressure values p_i since i and j are neighbors. To separate p_i in this sum, we write

$$\sum_k \mathbf{d}_{jk} p_k = \sum_{k \neq i} \mathbf{d}_{jk} p_k + \mathbf{d}_{ji} p_i. \quad (11)$$

Taking these considerations into account, the right-hand side of (10) can be split up into parts that contain p_i values and other parts that contain pressures p_j and p_k as

$$\begin{aligned} \rho_0 - \rho_i^{adv} &= p_i \sum_j m_j (\mathbf{d}_{ii} - \mathbf{d}_{ji}) \nabla W_{ij} \\ &+ \sum_j m_j \left(\sum_j \mathbf{d}_{ij} p_j - \sum_{k \neq i} \mathbf{d}_{jk} p_k \right) \nabla W_{ij}. \end{aligned}$$

Now, we can compute the coefficients a_{ii} as

$$a_{ii} = \sum_j m_j (\mathbf{d}_{ii} - \mathbf{d}_{ji}) \nabla W_{ij}, \quad (12)$$

and evaluate the pressure p_i^{l+1} with

$$\begin{aligned} p_i^{l+1} &= (1 - \omega)p_i^l + \omega \frac{1}{a_{ii}} \left(\rho_0 - \rho_i^{adv} \right. \\ &\quad \left. - \sum_j m_j \left(\sum_j \mathbf{d}_{ij} p_j^l - \sum_{k \neq i} \mathbf{d}_{jk} p_k^l \right) \nabla W_{ij} \right). \end{aligned} \quad (13)$$

3.1.1 Implementation

In our implementation, \mathbf{d}_{ii} and a_{ii} are precomputed and stored. The coefficients \mathbf{d}_{ij} are computed to get a_{ii} , but not stored. In each iteration, two passes over the particles are required to update p_i^{l+1} . The first pass computes and stores $\sum_j \mathbf{d}_{ij} p_j^l$. The second pass computes p_i^{l+1} using the stored $\sum_j \mathbf{d}_{ij} p_j^l$ and \mathbf{d}_{ij} . The sum $\sum_{k \neq i} \mathbf{d}_{jk} p_k^l$ is computed

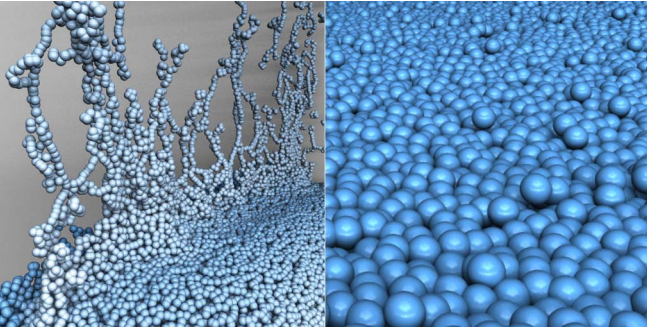


Fig. 2. Exaggerated surface tension by attracting pressure forces (left). Distorted alignment of surface particles (right) caused by clamping $b_i = \min(0, \rho_0 - \rho_i^{adv})$. Velocities are color coded, where white is maximum.

as $\sum_k \mathbf{d}_{jk} p_k^l - \mathbf{d}_{ji} p_i^l$, where the term $\sum_k \mathbf{d}_{jk} p_k^l$ can be accessed at the particle, while \mathbf{d}_{ji} is computed. Thus, an explicit computation of the nondiagonal elements a_{ij} is avoided. Algorithm 1 summarizes the simulation update using the proposed relaxed Jacobi pressure solve.

Algorithm 1. IISPH using relaxed Jacobi. l indicates the iteration.

procedure PREDICT ADVECTION

for all particle i **do**

compute $\rho_i(t) = \sum_j m_j W_{ij}(t)$

predict $\mathbf{v}_i^{adv} = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_i^{adv}(t)}{m_i}$

$\mathbf{d}_{ii} = \Delta t^2 \sum_j -\frac{m_j}{\rho_j^2} \nabla W_{ij}(t)$

for all particle i **do**

$\rho_i^{adv} = \rho_i(t) + \Delta t \sum_j m_j (\mathbf{v}_{ij}^{adv}) \cdot \nabla W_{ij}(t)$

$p_i^0 = 0.5 p_i(t - \Delta t)$

compute a_{ii} (12)

procedure PRESSURE SOLVE

$l = 0$

while $\rho_{avg}^l - \rho_0 > \eta \vee l < 2$ **do**

for all particle i **do**

$\sum_j \mathbf{d}_{ij} p_j^l = \Delta t^2 \sum_j -\frac{m_j}{\rho_j^2(t)} p_j^l \nabla W_{ij}(t)$

for all particle i **do**

compute p_i^{l+1} (13)

$p_i(t) = p_i^{l+1}$

$l = l + 1$

Procedure INTEGRATION

for all particle i **do**

$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i^{adv} + \Delta t \mathbf{F}_i^p(t) / m_i$

$\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t)$

IISPH performs two loops over all particles per iteration (PCISPH requires three loops). Moreover, the two loops do not contain data dependencies. Accordingly, the method is well suited for parallel architectures. While an equation for a particle contains up to 40^2 nonzero coefficients, the implementation only stores seven scalar values per particle, namely a_{ii} , \mathbf{d}_{ii} , and $\sum_j \mathbf{d}_{ij} p_j^l$.

We observed an optimal convergence for the relaxation factor $\omega = 0.5$ in all settings. The convergence is also optimized by initializing $p_i^0 = 0.5 p_i(t - \Delta t)$. While grid-based projection schemes commonly initialize $p_i^0 = 0$, for example, [8], using $p_i^0 = p_i(t - \Delta t)$ would be intuitive in our Lagrangian approach. In our experiments, however, we

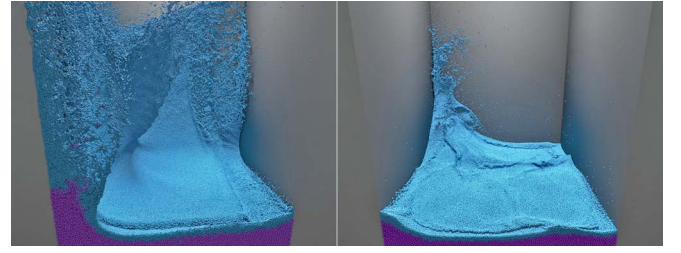


Fig. 3. A two-scale simulation based on [26] using IISPH. The radii of the purple low-resolution and the blue high-resolution particles are 0.004 and 0.002 m, respectively.

observed that the coefficient 0.5 generally provides a close to optimal convergence. The right-hand side of (13) can be rearranged to compute the density at $t + \Delta t$ including pressure forces. Thus, the compression of the fluid can be predicted in each iteration. By terminating the iterations for a compression below a predefined value η , the user can control the compression.

By enforcing $\rho_i(t + \Delta t) = \rho_0$, pressure forces induce a positive change of density for particles with a predicted density smaller than the rest density, i.e., $\rho_i^{adv} < \rho_0$. In this case, pressure forces act as attraction forces [25]. Although this might be interpreted as surface tension, we consider the induced cohesive effect as too exaggerated, for example, splashes are noticeably absorbed (Fig. 2(left)). In EOS solvers, attracting pressure forces are prohibited by clamping negative pressures to zero. We adopt the same concept and clamp negative pressures in each iteration.

Finally, in contrast to hybrid particle-mesh solvers, the proposed method builds purely on SPH. Thus, it can be integrated into sophisticated SPH frameworks, for example, [26], as demonstrated in Fig. 3.

3.2 Conjugate Gradient

We have implemented CG with a diagonal preconditioner. Although CG shows promising convergence rates, we experienced two issues which are highlighted in the following:

Symmetry. The coefficient matrix is not symmetric as each a_{ij} is scaled by m_i / ρ_i^2 . Symmetry can be enforced by assuming that $\rho_i = \rho_j = \rho_0$ and $m_i = m_j$ for all particles. While this symmetrization works for uniform masses such as adaptively sampled SPH, for example, [3], [27], or multi-phase simulations, for example, [28]. Note that nonsymmetry is not an issue for relaxed Jacobi.

Negative Pressure. Using relaxed Jacobi, we propose to clamp negative pressures to eliminate exaggerated cohesion effects. For CG, however, clamping in between the iterations leads to invalid states. We also observed instabilities in case of any change in the final pressure field, such as clamping of negative pressure values or disregarding pairwise-attracting pressure forces. Intuitively, we could disallow a positive change of density due to pressure by clamping b_i to negative values with $b_i = \min(0, \rho_0 - \rho_i^{adv})$. Unfortunately, this adaptation causes implausible alignments of single particles at the fluid surface for CG and Jacobi as demonstrated in the right-hand side of Fig. 2.

4 BOUNDARY HANDLING

IISPH does not impose special requirements on the boundary handling. All boundary handling schemes that work with PCISPH or ISPH, can also be used with IISPH. This section describes the combination of IISPH with an exemplary boundary handling approach presented in [29] that is used in all experiments.

The rigid-fluid coupling [29] employs rigid boundary particles b_i that contribute to the density of a fluid particle i . This contribution scales with the local number density $\delta_{b_i} \equiv \sum_{b_j} W_{b_j b_i}$ and is defined as $\Psi_b(\rho_{0i}) = \rho_{0i} / \delta_{b_i}$. Incorporating this method, the density estimation (3) is extended to

$$\begin{aligned} \rho_i(t + \Delta t) = & \sum_j m_j W_{ij} + \sum_b \Psi_b(\rho_{0i}) W_{ib} \\ & + \Delta t \sum_j m_j \mathbf{v}_{ij}(t + \Delta t) \nabla W_{ij} \\ & + \Delta t \sum_b \Psi_b(\rho_{0i}) \mathbf{v}_{ib}(t + \Delta t) \nabla W_{ib}. \end{aligned} \quad (14)$$

For a weak coupling, we assume a constant rigid velocity \mathbf{v}_b throughout the pressure iterations. Therefore, we can estimate the density without pressure forces as

$$\begin{aligned} \rho_i^{adv} = & \sum_j m_j W_{ij} + \sum_b \Psi_b(\rho_{0i}) W_{ib} \\ & + \Delta t \sum_j m_j \mathbf{v}_{ij}^{adv} \nabla W_{ij} \\ & + \Delta t \sum_b \Psi_b(\rho_{0i}) (\mathbf{v}_i^{adv} - \mathbf{v}_b(t + \Delta t)) \nabla W_{ib}. \end{aligned}$$

Pressure forces correct the density field such that

$$\begin{aligned} \rho_i(t + \Delta t) = & \rho_i^{adv} + \sum_j m_j \left(\Delta t^2 \frac{\mathbf{F}_i^p}{m_i} - \Delta t^2 \frac{\mathbf{F}_j^p}{m_j} \right) \nabla W_{ij} \\ & + \sum_b \Psi_b(\rho_{0i}) \left(\Delta t^2 \frac{\mathbf{F}_i^p}{m_i} \right) \nabla W_{ib}. \end{aligned}$$

In [29], boundary particles b do not have an individual pressure, but exert a pressure force which is dependent on the fluid pressure p_i as

$$\mathbf{F}_{i \leftarrow b}^p = -m_i \Psi_b(\rho_{0i}) \frac{p_i}{\rho_i^2} \nabla W_{ib}. \quad (15)$$

Thus, the displacement due to pressure is computed as

$$\begin{aligned} \Delta t^2 \frac{\mathbf{F}_i^p}{m_i} = & \sum_j \underbrace{-\Delta t^2 \frac{m_j}{\rho_j^2} \nabla W_{ij} p_j}_{\mathbf{d}_{ij}} \\ & + \underbrace{\left(-\Delta t^2 \sum_j \frac{m_j}{\rho_i^2} \nabla W_{ij} - \Delta t^2 \sum_b \Psi_b(\rho_{0i}) \frac{1}{\rho_i^2} \nabla W_{ib} \right)}_{\mathbf{d}_{ii}} p_i. \end{aligned}$$

Accordingly, the pressure update with relaxed Jacobi reads

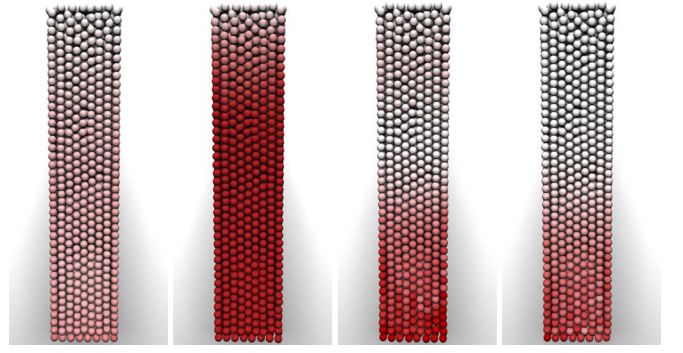


Fig. 4. Subsequent frames of a resting fluid pillar of 5 meters height, simulated at low resolution. The convergence criterion is set to a maximum density error of 1 percent. This leads to oscillations in the pressure field and, thus, the free surface. Pressure values are color coded, where red is maximum.

$$\begin{aligned} p_i^{l+1} = & (1 - \omega) p_i^l + \omega \frac{1}{a_{ii}} \left(\rho_0 - \rho_i^{adv} \right. \\ & - \sum_j m_j \left(\sum_j \mathbf{d}_{ij} p_j^l - \mathbf{d}_{jj} p_j^l - \sum_{k \neq i} \mathbf{d}_{jk} p_k^l \right) \nabla W_{ij} \\ & \left. - \sum_b \Psi_b(\rho_{0i}) \sum_j \mathbf{d}_{ij} p_j^l \nabla W_{ib} \right). \end{aligned} \quad (16)$$

5 RESULTS

We integrated the solvers into an SPH framework which employs the cubic spline kernel and the viscous force presented in [23]. Surface tension is modeled with [5]. Neighborhood search and fluid update are parallelized using the techniques described in [30]. The fluid surface is reconstructed with [31], [32]. Performance measurements are given for a 16-core 3.46 GHz Intel i7 with 64-GB RAM. Images were rendered with mental ray [33].

5.1 Convergence Criterion

Realistic SPH simulations require low density errors to avoid perceivable volume changes, i.e., oscillations of the free surface. While the original PCISPH method considered the maximum density error with a threshold of 1 percent, we recommend to maintain an average density error of less than 0.1 percent. Although average and maximum density errors are closely related, their ratio varies throughout a simulation, i.e., if a maximum density error is preserved, the average density error and therefore the fluid volume might change over time. Such variations of the average density error result in oscillations of the pressure field and in perceivable simulation artifacts such as jumping of the free surface. This issue is particularly perceivable in simulations with growing water depth as illustrated in Fig. 4. In this scenario, the obtained average compression varied between 0.3 and 0.6 percent when tolerating a maximum error of 1 percent. We found that these artifacts can be avoided by considering the estimated average density error with a threshold of 0.1 percent as the convergence criterion. Please note that this criterion restricts the overall global volume change of the fluid to 0.1 percent, but tolerates locally delimited maximum density errors. The average density

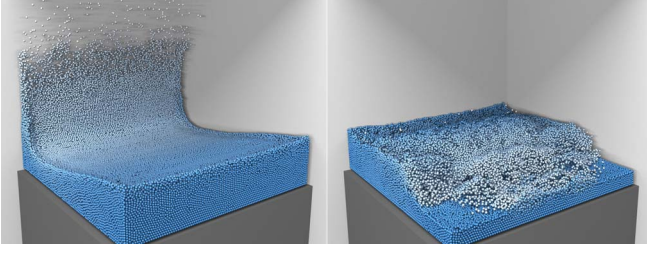


Fig. 5. Breaking dam with 100K particles.

volume compression, is consistently used as convergence criterion in all experiments with all solvers.

5.2 Performance Comparisons

We present comparisons with three existing solvers. First, we illustrate the performance of IISPH by comparing to PCISPH, the current state-of-the-art pressure solver in computer graphics. Second, we illustrate the benefits of our PPE formulation by comparing to the discretization of the PPE used in ISPH [13]. Third, we briefly compare to constraint fluids [15] due to similarities in the concept.

5.2.1 Comparison to PCISPH

We compare IISPH to PCISPH in two scenarios with different particle radii r and different average density errors η to illustrate the effect of these parameters on the time step and the convergence rate of the solvers. Fig. 5 shows a breaking dam with 100K particles, $r = 0.025$ m, and $\eta = 0.01\%$. Fig. 6 shows a blender with 90K particles, $r = 0.05$ m, and $\eta = 0.1\%$. The performance measurements are summarized in Tables 1 and 2.

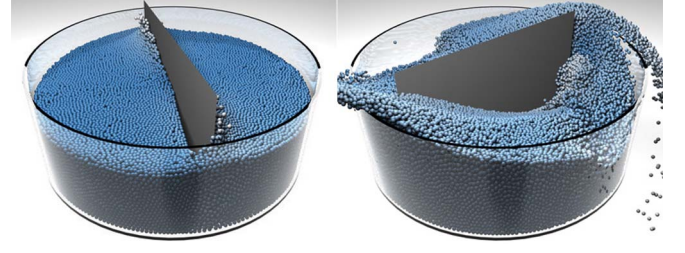


Fig. 6. Blender scenario with 90K particles.

Pressure solve. Compared to PCISPH, IISPH computes the pressure field up to 6.2 times faster for the breaking dam and up to 5.2 times faster in the blender scene. These speedups are a combination of an improved convergence and an improved efficiency per iteration. Regarding the convergence, IISPH requires up to 3.6 and 3.1 times less iterations compared to PCISPH. Combined with the improved efficiency per iteration (2 particle loops in IISPH versus 3 particle loops in PCISPH), a speedup of up to 6.2 and 5.2 is obtained. Further, the speedups grow for larger time steps. This indicates that the convergence of IISPH scales better compared to PCISPH for growing time steps. As shown in the accompanying video, the simulation results are in good agreement.

SPH computation time per simulated frame. IISPH computes the pressure field up to six times faster than PCISPH. However, the overall speedup in the computation time per simulated frame also depends on other SPH components, for example, the neighborhood query. The neighborhood query adds roughly constant costs per simulation step. However, the total costs per simulated frame for finding the neighbors decrease with larger time steps as less simulation

TABLE 1
Comparison of IISPH with PCISPH Using Different Time Steps for a Breaking Dam with 100K Particles (Fig. 5)

Δt [s]	PCISPH			IISPH			PCISPH / IISPH		
	avg. iter.	total comp. time [s]		avg. iter.	total comp. time [s]		ratio		
		pressure	overall		pressure	overall	iterations	pressure	overall
0.0005	4.3	540	1195	2.2	148	978	2.0	3.6	1.2
0.00067	7.2	647	1145	2.9	149	753	2.5	4.3	1.5
0.001	14.9	856	1187	4.9	164	576	3.0	5.2	2.1
0.0025	66.5	1495	1540	18.4	242	410	3.6	6.2	3.8
0.004	-	-	-	33.5	273	379	-	-	-
0.005	-	-	-	45.8	297	383	-	-	-

Timings are given for the whole simulation (300 frames). The largest ratio in the pressure computation time and the lowest total computation times are marked bold. The maximum volume compression was set to 0.01 percent. The particle radius was 0.025 m.

TABLE 2
Comparison of IISPH with PCISPH Using Different Time Steps for the Blender Scene with 90K Particles (Fig. 6)

Δt [s]	PCISPH			IISPH			PCISPH / IISPH		
	avg. iter.	total comp. time [s]		avg. iter.	total comp. time [s]		ratio		
		pressure	overall		pressure	overall	iterations	pressure	overall
0.0025	4.2	416	978	2.0	106	709	2.1	3.9	1.4
0.005	15.4	688	964	4.9	132	435	3.1	5.2	2.2
0.01	-	-	-	13.2	182	338	-	-	-
0.02	-	-	-	78.8	510	588	-	-	-

Timings are given for the whole simulation (1,000 frames). The largest ratio in the pressure computation time and the lowest total computation times are marked bold. The maximum volume compression was set to 0.1 percent. The particle radius was 0.05 m.

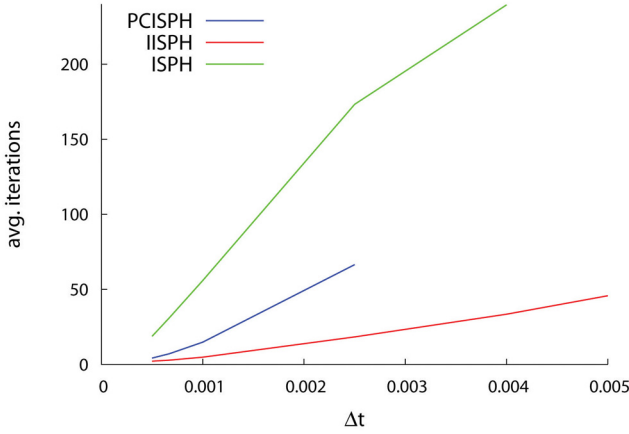


Fig. 7. Average number of iterations of PCISPH, IISPH, and ISPH for different time steps in the breaking dam scenario.

steps are performed per simulated frame. Interestingly, increasing costs for the pressure solve and decreasing costs for the neighborhood search result in the fact, that IISPH and PCISPH do not necessarily achieve their best overall performance for the largest possible time step. For the breaking dam, IISPH, and PCISPH handle time steps of up to 0.005 and 0.0025 s, respectively. The factor of two with respect to the maximum time step is, however, practically irrelevant for the overall computation time, as both solvers achieve their best performance for time steps of 0.004 and 0.00067 s, corresponding to a factor of six with respect to the optimal time step and a factor of three with respect to the overall performance gain. The situation in the blender scenario is analogous.

Time step. As discussed in [6], PCISPH scales pressure values by a global stiffness value δ which is time-step dependent. So, dependent on the time step, δ might be too large and cause overshooting, or it might be too small and negatively influence the convergence rate. The performance and stability of PCISPH are, thus, heavily influenced by the time step. For IISPH, the time step has less effect on the performance. The influence of the time step on the number of iterations is illustrated in Fig. 7.

For the performance comparisons (Tables 1 and 2), we kept the time step fix for each simulation run. However, in practice, adaptive time-stepping schemes are desirable. For PCISPH, the coupling of the time step size with the convergence rate and the stability is an issue that requires sophisticated techniques to realize adaptive time steps, for example, [34]. In contrast, changing time steps do not negatively affect the robustness of IISPH. The time step can just be set according to the CFL condition, for example, $\Delta t = \min(0.4h/|\mathbf{v}_i^{adv}|)$, without rolling back the simulation as required in PCISPH [34], [21]. In practice, this has a positive effect on the overall performance gain of IISPH compared to PCISPH.

5.2.2 Comparison to ISPH

IISPH differs from ISPH presented in [13] in the discretization of the Laplacian and the source term. Both aspects improve the performance of IISPH. While our discretization of the Laplacian improves the convergence rate compared

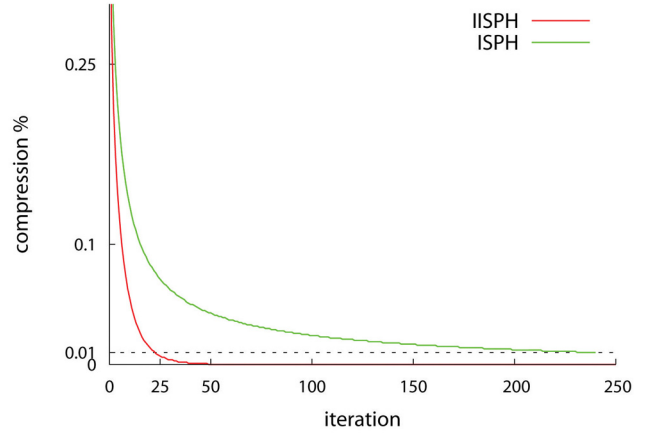


Fig. 8. Convergence of IISPH and ISPH for one simulation step of the breaking dam scenario with $\Delta t = 0.0025$ s.

to [13], our source term allows for larger time steps compared to the source term of [13]. We first summarize the ISPH formulation of [13], followed by an analysis of the benefits of the proposed IISPH discretization of the Laplacian and the source term.

ISPH [13]. Here, the Laplacian is approximated as proposed in [11] with

$$\nabla^2 p_i = \sum_j m_j \frac{8}{(\rho_i + \rho_j)^2} \left(\frac{p_{ij} \mathbf{x}_{ij} \nabla W_{ij}}{\mathbf{x}_{ij} \cdot \mathbf{x}_{ij} + \epsilon^2} \right), \quad (17)$$

where $\epsilon = 0.1h$ avoids singularities. For the density invariant scheme, the source term is computed with $(\rho_0 - \rho^*)/\Delta t^2$, where ρ^* is computed based on the intermediate particle positions $\mathbf{x}_i^{adv} = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i^{adv}$ as

$$\rho^* = \sum_j m_j W(\mathbf{x}_i^{adv} - \mathbf{x}_j^{adv}). \quad (18)$$

This yields the following discretization of the PPE

$$\nabla^2 p_i = (\rho_0 - \rho_i^*)/\Delta t^2. \quad (19)$$

Finally, pressure forces are computed with the symmetric pressure force (2).

Discretization of the Laplacian. IISPH takes contributions of second-ring neighbors into account that improves the convergence rate compared to ISPH. To verify this, we compared the left-hand side of (5) with the approximate discretization of the Laplacian in ISPH (17). In this comparison, we only vary the discretization of the Laplacian. The source term is taken from (5) in both cases. Fig. 8 illustrates the convergence of both implementations for one simulation step of the breaking dam scenario with equivalent initial particle configurations. IISPH achieves an estimated compression of 0.01 percent after 23 iterations, while ISPH takes 231 iterations. As we only use different discretizations of the Laplacian in both settings, the experiment indicates that the specific form of the Laplacian in IISPH significantly improves the convergence rate compared to ISPH. While Fig. 8 compares the convergence for one simulation step, Fig. 7 compares the convergence rate for varying time steps using the same setup. The experiments show in all cases that the

TABLE 3
Performance of ISPH [13] for the Breaking Dam
Scenario (Fig. 5)

Δt [s]	avg. iter.	total comp. time [s]	
		pressure	overall
0.0005	18.8	759	1588
0.00067	31.1	959	1563
0.001	55.9	1123	1535

The tolerated error was set to 0.01 percent.

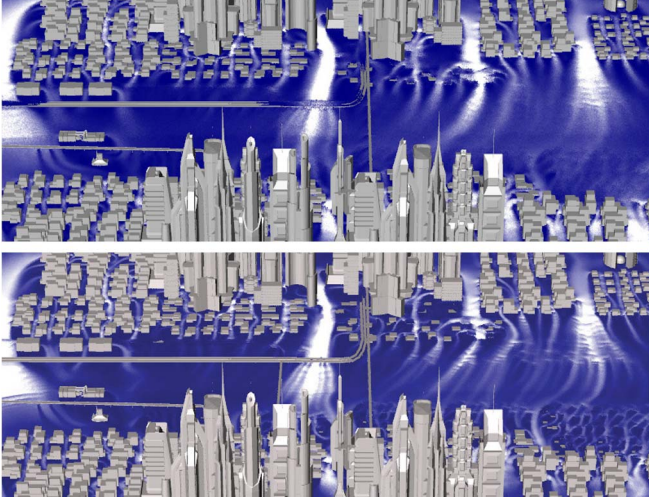


Fig. 9. A city with an area of $5 \cdot 10^6 \text{m}^2$ is flooded at low resolution (top), 6 million particles, and high resolution (bottom), 40 million particles. Velocities are color coded.

convergence rate of the IISPH discretization is superior to the ISPH discretization.

Furthermore, the approximation of the Laplacian (17) employed in ISPH does not accurately consider the pressure force, but uses different approximations for the pressure gradient and the Laplacian. Therefore, the predicted level of compression when solving the PPE might significantly differ from the real compression at time $t + \Delta t$. For the breaking dam scene, the obtained density error of ISPH is 0.29 percent for a time step of 0.004 s. In contrast, the error introduced by the semi-implicit formulation employed in IISPH is much smaller. IISPH could even obtain a compression of 0.011 percent for the largest time step of 0.005 s.

Source term. IISPH employs velocity projection in the source term. To illustrate the positive effect of this choice on the time step, we have compared velocity projection (4) with position projection (18) as proposed in [13]. For the comparison, we use the proposed IISPH formulation in (5) for the Laplacian. For the breaking dam scenario, the setting with position projection could only handle time steps up to 0.001 s, whereas four times larger time steps can be handled with velocity projection.

Performance. Finally, we compare the performance of ISPH [13] to IISPH on the breaking dam scenario. The timings of ISPH are given in Table 3, while the timings of IISPH are given in Table 1. ISPH reaches the best performance at a time step of 0.001 s for the breaking dam scenario with an overall computation time of 1535 s, where 1123 s is represented by the pressure computation. For the same time step, IISPH computes the pressure field in 164 s

TABLE 4
Measurements for the Large-Scale Scenarios

scene	part.	r	avg. Δt	avg. iter.	comp. time / Δt	
					pressure	total
City	6M	1.000m	0.0500s	5.2	2.5s	5.7s
	40M	0.500m	0.0250s	4.1	15.8s	38.2s
Island	10M	0.075m	0.0054s	13.5	6.5s	13.6s
Cargo	30M	0.050m	0.0040s	16.1	24.9s	44.2s
Street	28M	0.025m	0.0025s	17.3	23.1s	41.8s



Fig. 10. A scripted island rises out of the water. 10 million SPH fluid particles. One simulation step is computed in 14 seconds with a time step of 0.0054 seconds and a compression of less than 0.1 percent.

which is 6.8 times faster than ISPH. IISPH reaches the best overall performance at a time step of 0.004 s with 379 s. This is an overall speedup of 4 compared to the ISPH formulation presented in [13].

5.2.3 Comparison to Constraint Fluids

Another related approach for incompressible SPH is presented in [15], [22], and [35]. Similar to the proposed method, incompressibility is enforced by imposing density constraints. In contrast to ISPH and IISPH, the incompressibility condition is not derived from the continuity equation, but from constraint dynamics. Density constraints are enforced through the use of Lagrange multipliers and not by employing standard SPH formulations for pressure and pressure force.

In [15], a setup of a fluid pillar with 10K particles of radius $r = 0.01 \text{ m}$ is presented. For a time step of 0.0083 s and a fixed number of 15 pressure iterations, a volume compression of 17 percent has been reported. We simulated this test case with IISPH using the parameters given in [15]. For the same number of pressure iterations per frame, IISPH enforced a volume compression of 1.2 percent which is 14 times less compression than stated for the constraint-based method. This indicates an improved convergence of the IISPH pressure solver compared to [15].

5.3 Large-Scale Scenarios

Poor performance scaling has been reported for all previous global SPH methods, for example, for ISPH in [11] and for constraint fluids in [22]. In contrast, the performance of IISPH scales well with the size of the simulation domain as the convergence and the costs per particle are invariant to the number of particles. This is verified on a large-scale



Fig. 11. Street flood. This scene contains up to 28 million fluid particles. IISPH computes one simulation step in 42 seconds with an average time step of 0.0025 seconds and a compression of less than 0.1 percent.

scenario, where a city with an area of $5 \cdot 10^6 \text{ m}^2$ is flooded at two different resolutions (Fig. 9). In the low resolution, we simulated up to six million particles with a radius of 1 meter. The high resolution contains up to 40 million particles with radius 0.5 meter. The factor of 6.7 in the number of particles instead of 8 stems from the shallow parts of the simulation. The IISPH solver took 5.7 s to update the low-resolution simulation and 38 s for the high resolution, see Table 4. This is a performance ratio of 6.7 which shows the perfect linear scaling of the proposed method.

The practicability of the approach is further demonstrated by three large-scale scenarios with particle counts ranging from 10 to 30 million including two-way coupling with rigid objects. Performance measurements are given in Table 4.

In the Island scene (Fig. 10), 10 million particles of volume radius $r = 0.075 \text{ m}$, were simulated with an average time step of 0.0054 s. On average, the simulation update took 13.6 s, where the pressure solve represents 6.5 s. For the larger scenes, Street (Fig. 11) and Cargo (Fig. 1), strong turbulences were generated by high-velocity inflows. The Street scene with 28 million particles and $r = 0.025 \text{ m}$ was updated in 41.8 s, where 23.1 s were spent on solving the pressure. This is similar to the performance of the Cargo scene, 30 million particles and $r = 0.05 \text{ m}$, where a simulation step is computed in 44.2 s with 24.9 s represented by the computation of the pressure field.

6 CONCLUSION

We presented a discretization of the PPE that can be solved efficiently, handles large time steps and scales well for large-scale scenarios. The straightforward derivation of our scheme shows that its accuracy and fast convergence is based on the fact that only few approximations are employed. As our approach is closely related to ISPH, we thoroughly discussed similarities and differences. Further, we compared IISPH with PCISPH and constraint fluids [15]. These comparisons indicate that IISPH yields higher convergence rates than the previous schemes, independent of the size of the domain.

Although the convergence rate of IISPH is superior to PCISPH and ISPH, the required number of iterations still grows faster than linear with the time step. To improve performance for larger time steps, we plan to investigate more advanced solvers, for example, multigrid [11]. In this context, we would like to eliminate the exaggerated cohesion

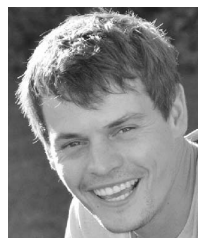
caused by IISPH with Conjugate Gradient. This might be addressed by employing a robust scheme for correcting the erroneous SPH density values at the free surface.

The IISPH formulation builds purely on established SPH discretizations and accounts for physical values such as pressure. Therefore, it might be possible to map the proposed discretization to simulate other materials, for example, granular material [36], [37] or deformable objects [38]. The robustness of IISPH and the alternative formulation for predicting the density might also improve the stability of multiphase simulations with high-density contrasts and could lead to an alternative formulation for multiresolution SPH fluids. This might be a potential direction for future work.

REFERENCES

- [1] J. Monaghan, "Smoothed Particle Hydrodynamics," *Ann. Rev. Astronomy and Astrophysics*, vol. 30, pp. 543-574, 1992.
- [2] M. Müller, D. Charypar, and M. Gross, "Particle-Based Fluid Simulation for Interactive Applications," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pp. 154-159, 2003.
- [3] B. Adams, M. Pauly, R. Keiser, and L. Guibas, "Adaptively Sampled Particle Fluids," *ACM Trans. Graphics*, vol. 26, no. 3, pp. 48-54, 2007.
- [4] J. Monaghan, "Simulating Free Surface Flows with SPH," *J. Computational Physics*, vol. 110, no. 2, pp. 399-406, 1994.
- [5] M. Becker and M. Teschner, "Weakly Compressible SPH for Free Surface Flows," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pp. 209-217, 2007.
- [6] B. Solenthaler and R. Pajarola, "Predictive-Corrective Incompressible SPH," *ACM Trans. Graphics*, vol. 28, pp. 40:1-40:6, 2009.
- [7] X. He, N. Liu, H. Wang, and G. Wang, "Local Poisson SPH for Viscous Incompressible Fluids," *Computer Graphics Forum*, vol. 31, pp. 1948-1958, 2012.
- [8] R. Bridson, *Fluid Simulation for Computer Graphics*. A K Peters / CRC Press, 2008.
- [9] N. Foster and R. Fedkiw, "Practical Animation of Liquids," *Proc. ACM SIGGRAPH*, pp. 23-30, 2001.
- [10] N. Chentanez and M. Müller, "Real-Time Eulerian Water Simulation Using a Restricted Tall Cell Grid," *ACM Trans. Graphics*, vol. 30, pp. 82:1-82:10, 2011.
- [11] S. Cummins and M. Rudman, "An SPH Projection Method," *J. Computational Physics*, vol. 152, no. 2, pp. 584-607, 1999.
- [12] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R. Whitaker, "Particle-Based Simulation of Fluids," *Computer Graphics Forum*, vol. 22, pp. 401-410, 2003.
- [13] S. Shao and Y. Lo, "Incompressible SPH Method for Simulating Newtonian and Non-Newtonian Flows with a Free Surface," *Advances in Water Resources*, vol. 26, no. 7, pp. 787-800, 2003.
- [14] A. Khayyer, H. Gotoh, and S. Shao, "Enhanced Predictions of Wave Impact Pressure by Improved Incompressible SPH methods," *Applied Ocean Research*, vol. 31, no. 2, pp. 111-131, 2009.
- [15] K. Bodin, C. Lacoursire, and M. Servin, "Constraint Fluids," *IEEE Trans. Visualization and Computer Graphics*, vol. 18, no. 3, pp. 516-526, Mar. 2012.

- [16] X. Hu and N. Adams, "An Incompressible Multi-Phase SPH Method," *J. Computational Physics*, vol. 227, no. 1, pp. 264-278, 2007.
- [17] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw, "Two-Way Coupled SPH and Particle Level Set Fluid Simulation," *IEEE Trans. Visualization and Computer Graphics*, vol. 14, no. 4, pp. 797-804, July 2008.
- [18] J. Morris, P. Fox, and Y. Zhu, "Modeling Low Reynolds Number Incompressible Flows Using SPH," *J. Computational Physics*, vol. 136, no. 1, pp. 214-226, 1997.
- [19] J. Liu, S. Koshizuka, and Y. Oka, "A Hybrid Particle-Mesh Method for Viscous, Incompressible, Multiphase Flows," *J. Computational Physics*, vol. 202, no. 1, pp. 65-93, 2005.
- [20] G. Yue, C. Li, S. Hu, and B. Barsky, "Simulating Gaseous Fluids with Low and High Speeds," *Computer Graphics Forum*, vol. 28, no. 7, pp. 1845-1852, 2009.
- [21] K. Raveendran, C. Wojtan, and G. Turk, "Hybrid Smoothed Particle Hydrodynamics," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pp. 33-42, 2011.
- [22] M. Ellero, M. Serrano, and P. Español, "Incompressible Smoothed Particle Hydrodynamics," *J. Computational Physics*, vol. 226, no. 1, pp. 1731-1752, 2007.
- [23] J. Monaghan, "Smoothed Particle Hydrodynamics," *Reports on Progress in Physics*, vol. 68, no. 8, pp. 1703-1759, 2005.
- [24] N. Foster and D. Metaxas, "Realistic Animation of Liquids," *Graphical Models and Image Processing*, vol. 58, no. 5, pp. 471-483, 1996.
- [25] H. Schechter and R. Bridson, "Ghost SPH for Animating Water," *ACM Trans. Graphics*, vol. 31, no. 4, 2012.
- [26] B. Solenthaler and M. Gross, "Two-Scale Particle Simulation," *ACM Trans. Graphics*, vol. 30, no. 4, pp. 72:1-72:8, 2011.
- [27] J. Orthmann and A. Kolb, "Temporal Blending for Adaptive SPH," *Computer Graphics Forum*, vol. 31, no. 8, pp. 2436-2449, 2012.
- [28] B. Solenthaler and R. Pajarola, "Density Contrast SPH Interfaces," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pp. 211-218, 2008.
- [29] N. Akinci, M. Ihmsen, B. Solenthaler, G. Akinci, and M. Teschner, "Versatile Rigid-Fluid Coupling for Incompressible SPH," *ACM Trans. Graphics*, vol. 30, no. 4, pp. 72:1-72:8, 2012.
- [30] M. Ihmsen, N. Akinci, M. Becker, and M. Teschner, "A Parallel SPH Implementation on Multi-Core CPUs," *Computer Graphics Forum*, vol. 30, no. 1, pp. 99-112, 2011.
- [31] B. Solenthaler, J. Schläfli, and R. Pajarola, "A Unified Particle Model for Fluid-Solid Interactions," *Computer Animation and Virtual Worlds*, vol. 18, no. 1, pp. 69-82, 2007.
- [32] G. Akinci, M. Ihmsen, N. Akinci, and M. Teschner, "Parallel Surface Reconstruction for Particle-Based Fluids," *Computer Graphics Forum*, vol. 32, no. 1, pp. 99-112, 2012.
- [33] "Mental Ray 3.9 [Software]," NVIDIA ARC, <http://www.mentalimages.com>, 2011.
- [34] M. Ihmsen, N. Akinci, M. Gissler, and M. Teschner, "Boundary Handling and Adaptive Time-stepping for PCISPH," *Proc. Workshop Virtual Reality Interaction and Physical Simulation Conf. (VRIPHYS '10)*, pp. 79-88, 2010.
- [35] M. Macklin and M. Mueller, "Position Based Fluids," *ACM Trans. Graphics*, vol. 32, pp. 1-5, 2013.
- [36] I. Alduán and M.A. Otaduy, "SPH Granular Flow with Friction and Cohesion," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pp. 25-32, 2011.
- [37] M. Ihmsen, A. Wahl, and M. Teschner, "High-Resolution Simulation of Granular Material with SPH," *Proc. Eurographics Association (VRIPHYS '12)*, pp. 53-60, 2012.
- [38] M. Becker, M. Ihmsen, and M. Teschner, "Corotated SPH for Deformable Solids," *Proc. Eurographics Workshop Natural Phenomena*, pp. 27-34, 2009.



Markus Ihmsen received diploma degree in computer science from the University of Freiburg in 2007. He is a research assistant and a PhD candidate in the computer graphics group at the University of Freiburg. His research interest includes physically based simulation. In particular, he is focusing on uid simulation.



Jens Cornelis earned his MS degree in applied computer science at the University of Freiburg in 2012. He is a research assistant and a PhD candidate at the computer graphics group at the University of Freiburg. His current research interest include the field of fluid animation.



Barbara Solenthaler received the PhD degree in computer science from the University of Zurich in 2009, before joining the ETH Computer Science Department. She is a senior research scientist at the Computer Graphics Laboratory at ETH Zurich, Switzerland. Her research interests include computer graphics with particular focus on the development of new methods for physics-based simulations, medical applications and computer-aided, and game-based learning.



usability and aesthetic refinement.

Christopher Horvath is a technical director at Pixar Animation Studios, in Emeryville, California. He has previously held lead technical and creative roles at Weta Digital, in Wellington, New Zealand, Industrial Light & Magic, in San Francisco, California, and was a cofounder of Tweak Films, a visual effects and software company also in San Francisco, California. His research interests include hierarchical fluid simulations, with an emphasis on production



robotics, medical simulation, and entertainment technology. He has served on program committees of major graphics conferences including Eurographics, Pacific Graphics, IEEE Vis, and ACM Siggraph/Eurographics SCA. He serves as an associate editor for Computer Graphics Forum and Computers & Graphics.

Matthias Teschner received the PhD degree in electrical engineering from the University of Erlangen-Nuremberg in 2000. He is professor of computer science and head of the computer graphics group at the University of Freiburg. From 2001 to 2004, he was research associate at Stanford University and at the ETH Zurich. His research interests include physically based simulation, computer animation, rendering, and computational geometry with applications in

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.