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% 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

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, e.g. [2], [3]. For weakly compressible fluids (WCSPH), e.g. [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, e.g. PCISPH [6] and local Poisson SPH [7]. Pressure forces are modeled as constraint forces that resolve compression induced by non-pressure 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 speed-up with respect to WCSPH can be up to 55 or 23, respectively [6], [7].

- *M. Ihmsen, J. Cornelis and M. Teschner are with the Department of Computer Science, University of Freiburg.*
- *B. Solenthaler is with the Department of Computer Science, ETH Zürich.*
- *C. Horvath is with Pixar Animation Studios, Emeryville, California.*

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, e.g. [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, e.g. [11], [12], the compression due to the intermediate velocity field, e.g. [13], [14], [15], or a combination of both, e.g. [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, e.g. [18], [11], where approximations are proposed. In order 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, e.g. [19], [17], [20]. After transferring the pressure values from the grid to the particles, particle pressure values can be refined, e.g. using an EOS as in [20].

SPH projection schemes, also referred to as incompressible SPH (ISPH) methods, are currently consid-

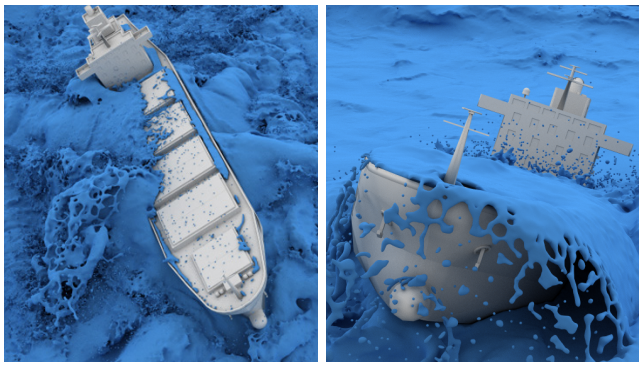

Fig. 1. A cargo ship in highly agitated water. Up to 30 million particles are processed in 44 seconds per simulation step with a time step of 0.004 seconds and a compression of less than 0.1%.

ered impractical in the context of computer graphics. As stated in, e.g. [11], [21], 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 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% 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, e.g. Figure 1.

2 Implicit Incompressible SPH (IISPH)

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

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

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

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

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, e.g. [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

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

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 non-pressure 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

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

We now search for pressure forces to resolve the compression $\rho_0 - \rho_i^{adv}$:

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

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 n equations for n unknown pressure values $\mathbf{p}(t) = \big(p_0(t), \ldots, p_{n-1}(t)\big)^{\mathrm{T}}$ and $\mathbf{b}(t)$ corresponding 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

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

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, e.g. [11], [12]. Thus, our approach is closely related to ISPH schemes which employ density constraints, e.g. [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, e.g. [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 Sec. 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

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

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 non-zero entries compared to previous projection schemes. As (6) contains a nested sum, the coefficients a_{ij} are non-zero for the neighbors j of particle i and for neighbors of neighbors of i. Typically, a particle has 30-40 neighbors [22]. Nevertheless, the system can be solved very efficiently in a matrix-free way as presented in the following.

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 $\mathbf{v}_i^{adv} = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_i^{adv}(t)}{m_i}$
    $\mathbf{d}_{ii} = \Delta t^2 \sum_j -\frac{m_j}{\rho_i^2} \nabla W_{ij}(t)$
  for all *particle* i **do**
    $\rho_i^{adv} = \rho_i(t) + \Delta t \sum_j m_j \left(\mathbf{v}_{ij}^{adv}\right) \nabla W_{ij}(t)$
    $p_i^0 = 0.5\ p_i(t-\Delta t)$
    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**
    $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^{adv} + \Delta t \mathbf{F}_i^p(t)/m_i$
    $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t+\Delta t)$

3 SOLVER

The system can be solved in various ways. For similar systems, grid approaches commonly apply SOR, e.g. [23], Conjugate Gradient, e.g. [9], or multigrid solvers, e.g. [10], while SPH approaches often employ Conjugate Gradient for PPEs, e.g. [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 Sec. 3.1, followed by a discussion of possible reasons for that outcome in Sec. 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.

In order 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

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

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

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

Note that $\sum_k \mathbf{d}_{jk}p_k$ includes pressure values p_i since i and j are neighbors. In order 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

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

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

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

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 $\mathbf{d}_{jj}$. The sum $\sum_{k\neq i} \mathbf{d}_{jk}p_k^l$ is computed 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 non-diagonal elements a_{ij} is avoided. Alg. 1 summarizes the simulation update using the proposed relaxed Jacobi pressure solve.

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 non-zero 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 $p_i^0 = 0$, e.g. [8], using $p_i^0 = p_i(t-\Delta t)$ would be intuitive in our Lagrangian approach. In our experiments, however, we 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 pre-defined 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 [24]. Although this might be interpreted as surface tension, we consider the induced cohesive effect as too exaggerated, e.g., splashes are noticeably absorbed (Figure 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.

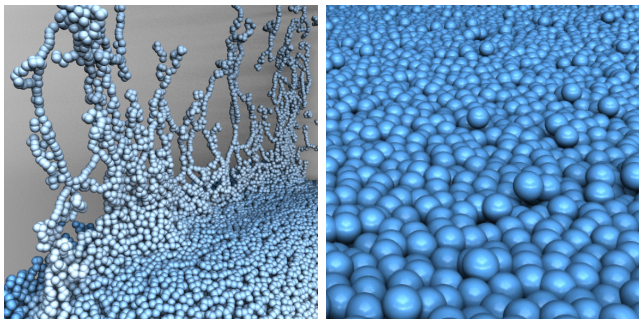

Fig. 2. Exaggerated surface tension by attracting pressure forces (left). Distorted alignment of surface particles (right) caused by clamping $b_i = min(0, \rho_0 - \rho_i^{adv})$. Particles are color coded with respect to velocity.

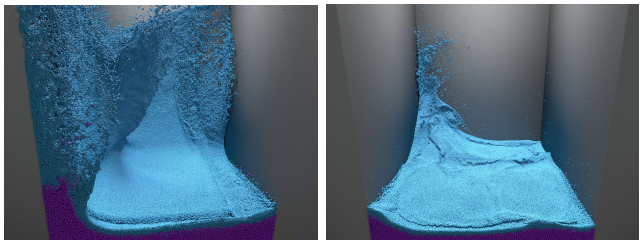

Fig. 3. A two-scale simulation with IISPH. The radii of the pink low-resolution and the blue high-resolution particles are 0.004m and $0.002m$, respectively.

Finally, in contrast to hybrid particle-mesh solvers, the proposed method builds purely on SPH. Thus, it can be integrated into sophisticated SPH frameworks, e.g. [25], as demonstrated in Figure 3.

3.2 Conjugate Gradient

We have implemented Conjugate Gradient (CG) with a diagonal pre-conditioner. 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, it is invalid for settings with non-uniform masses such as adaptively sampled SPH, e.g. [3], [26], or multi-phase simulations, e.g. [27]. Note that non-symmetry is not an issue for relaxed Jacobi.

Negative Pressure. Using relaxed Jacobi, we propose to clamp negative pressures in order 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

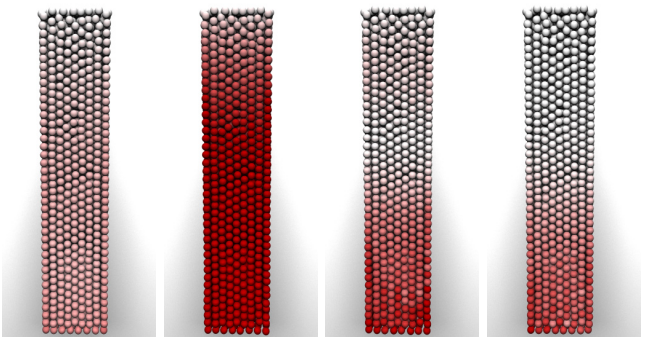

Fig. 4. Subsequent frames of a resting fluid pillar of 5 meters height, simulated at low resolution. The convergence criterion is set to a maximum density error of 1%. This leads to oscillations in the pressure field and, thus, the free surface. Pressure values are color coded.

b_i to negative values with $b_i = \min\left(0, \rho_0 - \rho_i^{adv}\right)$. 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 Figure 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 [28] that is used in all experiments.

The rigid-fluid coupling [28] employs rigid boundary particles b_i that contribute to the density of a fluid particle i. This contribution scales with the local number density $\delta_{b_i} \equiv \sum_{b_j} W_{b_i b_j}$ and is defined as $\Psi_b(\rho_{0_i}) = \rho_{0_i}/\delta_{b_i}$. Incorporating this method, the density estimation (3) is extended to

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

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

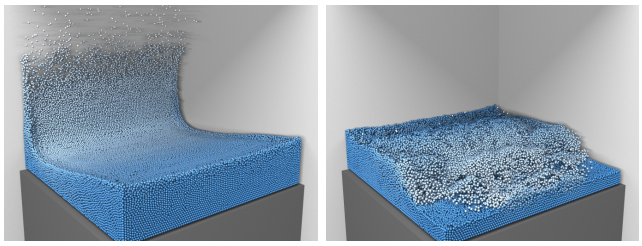

Fig. 5. Breaking dam with 100K particles.

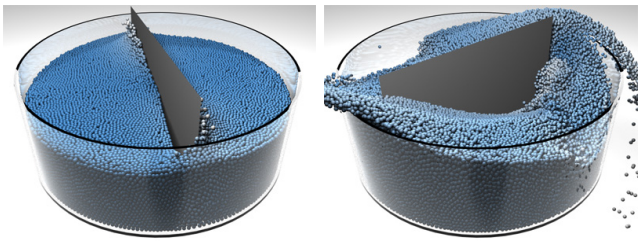

Fig. 6. Blender scenario with 90K particles.

as

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

Pressure forces correct the density field such that

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

In [28], boundary particles b do not have an individual pressure, but exert a pressure force which is dependent on the fluid pressure p_i as

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

Thus, the displacement due to pressure is computed as

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

Accordingly, the pressure update with relaxed Jacobi reads

$$p_i^{l+1} = (1-\omega) p_i^l + \omega \frac{1}{a_{ii}} \Bigg(\rho_0 - \rho_i^{adv} - \sum_j m_j \Big(\sum_j \mathbf{d}_{ij} p_j^l - \mathbf{d}_{jj} p_j^l - \sum_{k \neq i} \mathbf{d}_{jk} p_k^l \Big) \nabla W_{ij} - \sum_b \Psi_b(\rho_{0_i}) \sum_j \mathbf{d}_{ij} p_j^l \nabla W_{ib} \Bigg). \tag{16}$$

5 Results

We integrated the solvers into an SPH framework which employs the cubic spline kernel and the viscous force presented in [22]. Surface tension is modeled with [5]. Neighborhood search and fluid update are parallelized using the techniques described in [29]. The fluid surface is reconstructed with [30], [31]. Performance measurements are given for a 16-core 3.46 GHz Intel i7 with 64 GB RAM. Images were rendered with mental ray [32].

5.1 Convergence Criterion

Realistic SPH simulations require low density errors in order 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%, we recommend to maintain an average density error of less than 0.1%. Although average and maximum density error 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 Figure 4. In this scenario, the obtained average compression varied between 0.3% and 0.6% when tolerating a maximum error of 1%. We found that these artifacts can be avoided by considering the estimated average density error with a threshold of 0.1% as the convergence criterion. Please note that this criterion restricts the overall global volume change of the fluid to 0.1%, but tolerates locally delimited maximum density errors. The average density error η, also referred to as density deviation, compression or 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

	PCISPH			IISPH			PCISPH / IISPH		
		total comp. time [s]			total comp. time [s]		ratio		
Δt [s]	avg. iter.	pressure	overall	avg. iter.	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	-	-	-

TABLE 1
Comparison of IISPH with PCISPH using different time steps for a breaking dam with 100K particles (Figure 5). 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%. The particle radius was 0.025m.

	PCISPH			IISPH			PCISPH / IISPH		
		total comp. time [s]			total comp. time [s]		ratio		
Δt [s]	avg. iter.	pressure	overall	avg. iter.	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	-	-	-

TABLE 2
Comparison of IISPH with PCISPH using different time steps for the blender scene with 90K particles (Figure 6). Timings are given for the whole simulation (1000 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%. The particle radius was 0.05m.

solver in computer graphics. Second, we illustrate the benefits of our PPE formulation by comparing to a state-of-the-art discretization of the PPE as used in ISPH [13]. Third, we briefly compare to constraint fluids [15] due to similarities in the concept.

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. Figure 5 shows a breaking dam with 100K particles, $r = 0.025$m, and $\eta = 0.01$%. Figure 6 shows a blender with 90K particles, $r = 0.05$m, and $\eta = 0.1$%. The performance measurements are summarized in Table 1 and Table 2.

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 speed-ups 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 vs. 3 particle loops in PCISPH), a speed-up of up to 6.2 and 5.2 is obtained. Further, the speed-ups 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 speed-up in the computation time per simulated frame also depends on other SPH components, e.g. 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 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 0.005s and 0.0025s, 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 0.004s and 0.00067s, corresponding to a factor of six with respect to the

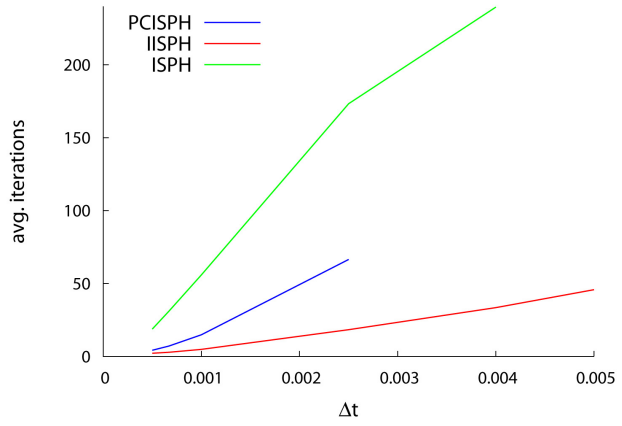

Fig. 7. Average number of iterations of PCISPH, IISPH and ISPH for different time steps in the breaking dam scenario.

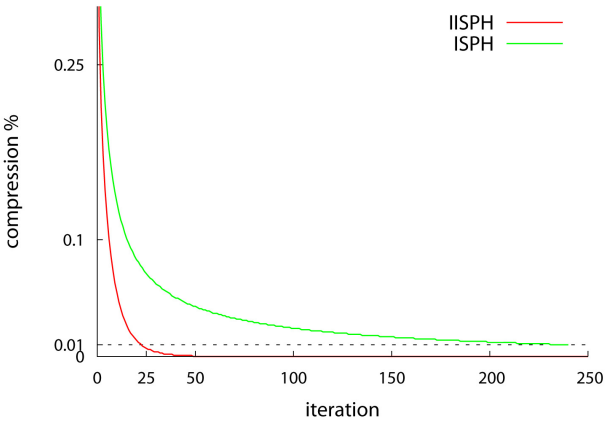

Fig. 8. Convergence of IISPH and ISPH for one simulation step of the breaking dam scenario with $\Delta t = 0.0025s$.

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 Figure 7.

For the performance comparisons (Table 1 and Table 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, e.g. [33]. 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, e.g. $\Delta t = \min\left(0.4h/\left|\mathbf{v}_i^{adv}\right|\right)$, without rolling back the simulation as required in PCISPH [33], [20]. In practice, this has a positive effect on the overall performance gain of IISPH compared to PCISPH.

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 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

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

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

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

This yields the following discretization of the PPE

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

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

Discretization of the Laplacian. IISPH takes contributions of second-ring neighbors into account which improves the convergence rate compared to ISPH. In order 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. Figure 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% 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 Figure 8 compares the convergence for one

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

TABLE 3
Performance of ISPH [13] for the breaking dam scenario (Figure 5). The tolerated error was set to 0.01%.

				avg.	comp. time / Δt	
scene	part.	r	avg. Δt	iter.	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

TABLE 4
Measurements for the large scale scenarios.

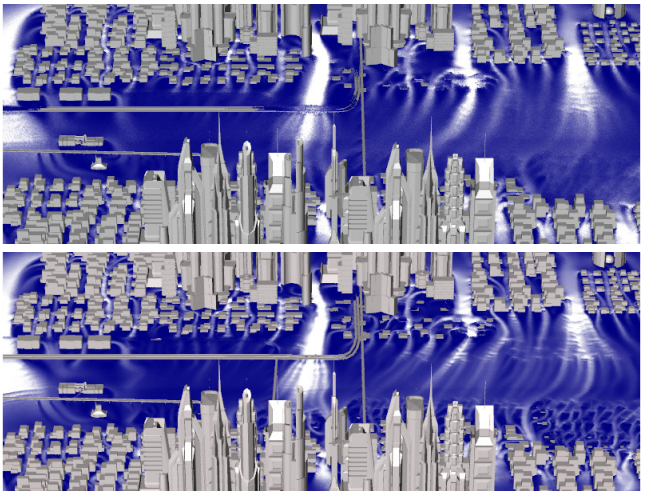

Fig. 9. A city with an area of $5 \cdot 10^6 \text{m}^2$ is flooded at low resolution (top), 6 million particles, and high resolution (bottom), 40 million particles. Velocities are color coded.

simulation step, Figure 7 compares the convergence rate for varying time steps using the same setup. The experiments show in all cases that the 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% for a time step of 0.004s. 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% for the largest time step of 0.005s.

Source term. IISPH employs velocity projection in the source term. In order 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 0.001s, 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 0.001s for the breaking dam scenario with an overall computation time of 1535s where 1123s is represented by the pressure computation. For the same time step, IISPH computes the pressure field in 164s which is 6.8 times faster than ISPH. IISPH reaches the best overall performance at a time step of 0.004s with 379s. This is an overall speed up of 4 compared to the ISPH formulation presented in [13].

Comparison to Constraint Fluids

Another related approach for incompressible SPH is presented in [15] and [21]. 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 $r = 0.01$m is presented. For a time step of 0.0083s and a fixed number of 15 pressure iterations, a volume compression of 17% 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% 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, e.g., for ISPH in [11] and for constraint fluids in [21]. In contrast, the performance of IISPH scales well with the size of the

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%.

simulation domain as the convergence and the costs per particle are invariant to the number of particles. This is verified on a large scale scenario where a city with an area of $5 \cdot 10^6 \text{m}^2$ is flooded at two different resolutions (Figure 9). In the low resolution, we simulated up to 6 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 5.7s to update the low-resolution simulation and 38s 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 (Figure 10), 10 million particles of volume radius $r = 0.075$m were simulated with an average time step of 0.0054s. On average, the simulation update took 13.6s where the pressure solve represents 6.5s. For the larger scenes, Street (Figure 11) and Cargo (Figure 1), strong turbulences were generated by high-velocity inflows. The Street scene with 28 million particles and $r = 0.025$m was updated in 41.8s where 23.1s were spent on solving the pressure. This is similar to the performance of the Cargo scene, 30 million particles and $r = 0.05$m, where a simulation step is computed in 44.2s with 24.9s represented by the computation of the pressure field.

6 Conclusion

We presented a discretization of the PPE that can be solved efficiently, handles large timesteps and scales well for large-scale scenarios. The straight-forward 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. The paper presented detailed comparisons with PCISPH, the fastest pressure solver. It also compared to the closely related ISPH approach [13] and to constraint fluids [15]. These comparisons did not only show the superiority of our approach, but also illustrated novel aspects that have not been discussed before. E.g., when embedded in an SPH framework, the maximum time step that can be handled by an iterative pressure solver generally does not correspond to the most efficient setting.

This paper mainly focused on the discretization of the PPE, the efficient implementation of the employed relaxed Jacobi solver and the comparisons with PCISPH and ISPH. While various other aspects such as neighborhood search, the time-stepping scheme and the linear solver also influence the performance of SPH frameworks, a discussion of those aspects was beyond the scope of this paper.

References

[1] J. Monaghan, "Smoothed particle hydrodynamics," *Ann. Rev. Astron. Astrophys.*, vol. 30, pp. 543–574, 1992.

[2] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003, pp. 154–159.

[3] B. Adams, M. Pauly, R. Keiser, and L. Guibas, "Adaptively sampled particle fluids," *ACM Trans. on Graphics (SIGGRAPH Proc.)*, vol. 26, no. 3, pp. 48–54, 2007.

[4] J. Monaghan, "Simulating free surface flows with SPH," *Journal of Comp. Phys.*, vol. 110, no. 2, pp. 399–406, 1994.

[5] M. Becker and M. Teschner, "Weakly compressible SPH for free surface flows," in *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2007, pp. 209–217.

[6] B. Solenthaler and R. Pajarola, "Predictive-corrective incompressible SPH," *ACM Trans. on Graphics (SIGGRAPH Proc.)*, 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," in *Proc. SIGGRAPH 2001*, 2001, pp. 23–30.

[10] N. Chentanez and M. Müller, "Real-time eulerian water simulation using a restricted tall cell grid," *ACM Trans. on Graphics (SIGGRAPH Proc.)*, vol. 30, pp. 82:1–82:10, 2011.

[11] S. Cummins and M. Rudman, "An SPH projection method," *Journal of Comp. 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 (Eurographics Proc.)*, 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 TVCG*, vol. 18, no. 3, pp. 516–526, 2012.

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%.

[16] X. Hu and N. Adams, "An incompressible multi-phase SPH method," *Journal of Comp. Phys.*, 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 TVCG*, vol. 14, no. 4, pp. 797–804, 2008.

[18] J. Morris, P. Fox, and Y. Zhu, "Modeling low Reynolds number incompressible flows using SPH," *Journal of Comp. Phys.*, 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," *Journal of Comp. Phys.*, vol. 202, no. 1, pp. 65–93, 2005.

[20] K. Raveendran, C. Wojtan, and G. Turk, "Hybrid Smoothed Particle Hydrodynamics," in *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2011, pp. 33–42.

[21] M. Ellero, M. Serrano, and P. Español, "Incompressible smoothed particle hydrodynamics," *Journal of Computational Physics*, vol. 226, no. 1, pp. 1731–1752, 2007.

[22] J. Monaghan, "Smoothed particle hydrodynamics," *Reports on Progress in Physics*, vol. 68, no. 8, pp. 1703–1759, 2005.

[23] N. Foster and D. Metaxas, "Realistic animation of liquids," *Graph. Models Image Process.*, vol. 58, no. 5, pp. 471–483, 1996.

[24] H. Schechter and R. Bridson, "Ghost SPH for animating water," *ACM Trans. on Graphics (SIGGRAPH Proc.)*, vol. 31, no. 4, 2012.

[25] B. Solenthaler and M. Gross, "Two-scale particle simulation," *ACM Trans. on Graphics (SIGGRAPH Proc.)*, vol. 30, no. 4, pp. 72:1–72:8, 2011.

[26] J. Orthmann and A. Kolb, "Temporal blending for adaptive SPH," *Computer Graphics Forum*, vol. 31, no. 8, pp. 2436–2449, 2012.

[27] B. Solenthaler and R. Pajarola, "Density contrast SPH interfaces," in *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2008, pp. 211–218.

[28] N. Akinci, M. Ihmsen, B. Solenthaler, G. Akinci, and M. Teschner, "Versatile rigid-fluid coupling for incompressible SPH," *ACM Trans. on Graphics (SIGGRAPH Proc.)*, vol. 30, no. 4, pp. 72:1–72:8, 2012.

[29] 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.

[30] 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.

[31] 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.

[32] NVIDIA ARC, "mental ray 3.9 [software]," *http://www.mentalimages.com*, 2011. [Online]. Available: http://www.mentalimages.com

[33] M. Ihmsen, N. Akinci, M. Gissler, and M. Teschner, "Boundary Handling and Adaptive Time-stepping for PCISPH," in *Proc. VRIPHYS*, 2010, pp. 79–88.