Volume xx (200y), Number z, pp. 1–13

An Efficient Hybrid Incompressible SPH Solver with Interface Handling for Boundary Conditions

Tetsuya Takahashi[1,2] Yoshinori Dobashi[3,2] Tomoyuki Nishita[2,4] Ming C. Lin[1]

[1]The University of North Carolina at Chapel Hill, USA
[2]UEI Research, Japan
[3]Hokkaido University, Japan
[4]Hiroshima Shudo University, Japan

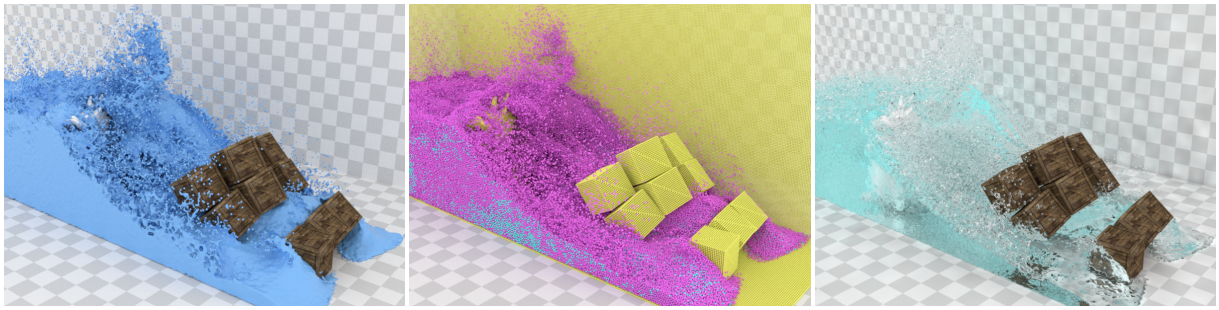

Figure 1: *A large scale corridor flood simulated with our solver, where 1.40 M fluid particles and 0.29 M solid particles are used. (Left) opaque mesh view. (Middle) particle view, where cyan, magenta, and yellow particles represent Poisson, Dirichlet, and Neumann particles, respectively (see § 4.1). (Right) final rendered view.*

Abstract

We propose a hybrid Smoothed Particle Hydrodynamics solver for efficiently simulating incompressible fluids using an interface handling method for boundary conditions in the pressure Poisson equation. We blend particle density computed with one smooth and one spiky kernel to improve the robustness against both fluid-fluid and fluid-solid collisions. To further improve the robustness and efficiency, we present a new interface handling method consisting of two components: free surface handling for Dirichlet boundary conditions and solid boundary handling for Neumann boundary conditions. Our free surface handling appropriately determines particles for Dirichlet boundary conditions using Jacobi-based pressure prediction while our solid boundary handling introduces a new term to ensure the solvability of the linear system. We demonstrate that our method outperforms the state-of-the-art particle-based fluid solvers.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Particle-based methods, such as Smoothed Particle Hydrodynamics (SPH), have been widely used to generate visual effects of fluids in computer graphics due to advantages of Lagrangian representations [IOS*14, MMCK14]. In these methods, however, enforcing fluid incompressibility has been a computational challenge, and various methods have been proposed to address this problem [KTO96, CR99, PTB*03, SL03, SP09, HLL*12, BLS12, MM13, ICS*14, KS14, BK15]. Among these methods, solving a Pressure

Poisson Equation (PPE) has been shown to be an effective approach [KTO96, CR99, SL03, PTB*03, HLL*12, ICS*14]. While it is common that this approach requires solving a sparse linear system, variations in discretization of the Laplacian operator lead to the different sparsity of the system and pressure solve. Incompressible SPH (ISPH) [CR99, SL03] and Moving Particle Semi-implicit (MPS) [KTO96, PTB*03] directly discretize the Laplacian, while a current state-of-the-art particle-based solver, Implicit Incompressible SPH (IISPH) [ICS*14] decomposes the Laplacian into divergence and gradient, and then discretizes these operators applying SPH formulations to both operators. When a (weighted) Jacobi method is used, IISPH outperforms ISPH in convergence rate and computational efficiency [ICS*14] since IISPH can propagate updated pressures faster by including farther particles with the decomposed operators. However, for ISPH and MPS, we can use a more efficient Conjugate Gradient (CG) method, which is a Krylov subspace method that generally shows faster convergence than stationary iterative methods (e.g., Jacobi and Gauss-Seidel methods). CG is inapplicable to IISPH, as pointed out in [ICS*14], due to the non-positive-definite property of the coefficient matrix (see § 6.1 for details). Additionally, the number of Jacobi iterations for IISPH increases super-linearly with time steps making the use of larger time steps ineffective. Because of these issues, ISPH and MPS can possibly offer performance advantages over IISPH by employing CG.

However, ISPH and MPS unfortunately have several issues, which make them undesirable for fluid simulation. First, ISPH uses a smooth kernel to compute the particle density in the source term. The resulting particle density is smooth and does not significantly increase although particles are almost overlapped. Consequently, ISPH with this smooth kernel is more likely to fail to prevent fluid-solid particle penetrations. On the other hand, MPS uses a spiky kernel for the density computation. Thus, as particles approach others, the resulting density rapidly increases effectively preventing fluid-solid particle penetrations. However, because of the rapid density changes, the spiky kernel tends to cause instabilities when fluid-fluid collisions occur. Second, previously proposed free surface handling methods for specifying Dirichlet boundary conditions are likely to undergo numerical instabilities mainly because of rough estimates of physical values. The density-based [KTO96, PTB*03, SL03] and surface-based [HLW*12] methods do not take into consideration the predicted density and actual pressures of neighboring particles while the ghost-particle-based method [NT14] relies on very rough estimates and assumptions based on ghost particles. Though the source-term-based method (which depends purely on the source term) considers the predicted density, this method disregards actual pressures of neighbor particles. Third, the previously used solid boundary handling for Neumann boundary conditions [KTO96, PTB*03, SL03], which treats solid particles as fluid ones, results in a much larger system of equations thereby increasing the overall computational cost, and erroneously estimates particle pressures to ensure the solvability of the linear system. Furthermore, even though this approach is used, the system can become unsolvable due to objects floating in the air.

We propose a hybrid SPH solver with a new interface handling method for addressing the three aforementioned problems. The main results of this work are as follows:

- A hybrid solver that blends the particle density computed with a smooth and a spiky kernel to take advantage of the specific properties of these kernels. This improves the robustness for both fluid-fluid and fluid-solid collisions while taking larger time steps and thus leading to more efficient simulation.
- A free surface handling method for Dirichlet boundary conditions that appropriately determines fluid particles by employing Jacobi-based pressure prediction. This takes into account not only the predicted density but also pressures of neighboring particles, to improve the robustness.
- A solid boundary handling method for Neumann boundary conditions, which introduces a new term to ensure the solvability of the linear system. This new term offers three advantages over the previous methods. First, we can completely exclude solid particles treated as unknown variables from the PPE thereby significantly reducing the size of the system and thus computational cost. Second, our solid boundary handling can avoid underestimation of particle pressures enabling the use of larger time steps. Third, we can handle objects floating in the air with the direct Laplacian discretization making it possible to simulate two-way interactions, which frequently cause objects floating in the air because of fluid-solid and solid-solid collisions.

By taking advantage of these, we demonstrate that our solver outperforms other particle-based solvers. Figure 1 illustrates a large-scale corridor flood scene involving two-way fluid-solid interactions, simulated with our method.

2. Related Work

Many particle-based methods have been proposed in Computational Fluid Dynamics (CFD) and computer graphics. We refer readers to [Mon05] for a survey on SPH in CFD and [IOS*14] for applications of particle-based methods in computer graphics. Below we focus our discussions on particle-based methods most closely related to ours.

ISPH. ISPH was originally proposed by Cummins and Rudman [CR99] introducing an idea of the pressure projection (which is commonly used in the Eulerian approach [Bri08]) into SPH. In this method, the divergence of velocity is used as a source term in the PPE. However, errors to the divergence-free velocity field can accumulate, leading to volume changes. To address this, Shao and Lo [SL03] proposed a new source term using the rest density (known as

the density invariance source term), which prevents the error accumulation, also presenting free surface handling for Dirichlet boundary conditions to keep the PPE solvable with the modified source term, which does not satisfy the compatibility condition [Bri08]. In addition, Shao and Lo [SL03] replaced a mirroring solid boundary handling method, used in [CR99], with a method that uses solid boundary particles to handle complex geometry. Nair and Tomar [NT14] proposed a new surface handling method for ISPH, adapting an idea of ghost particles, and ensured solvable linear systems without Dirichlet boundary conditions at the expense of difficult parameter tuning and inaccurately estimated pressures. Cummins and Rudman [CR99] applied a multigrid approach to the ISPH. However, this approach is imported from the grid-based approach, and cannot handle irregular domains. Additionally, their application is limited to fluid simulation without free surfaces (Dirichlet boundary condition) and complex solid objects (Neumann boundary condition).

MPS. MPS is a particle-based pressure projection method proposed in [KTO96, PTB*03]. In the algorithm level, MPS and ISPH (presented in [SL03]) are the same, and the differences lie in two aspects: (1) the (number) density computation for the source term, and (2) discretization for spatial derivatives. To compute the density, MPS uses a spiky kernel whose values rapidly increase when particles are close. This effectively prevents fluid-solid particle penetrations whereas the rapidly increased density tends to cause stability issues with fluid-fluid collision shocks. On the other hand, ISPH uses a smooth kernel, which leads to the smooth density distribution. This is more robust against fluid-fluid collisions whereas ISPH is more likely to fail to prevent fluid-solid particle penetrations. In MPS, since the gradient formulation is not anti-symmetric, and thus pressure forces computed with the gradient do not preserve fluid momentum.

IISPH. IISPH is a recently proposed particle-based method that uses the pressure projection with divergence and gradient operators instead of the Laplacian [ICS*14]. IISPH includes farther particles to faster propagate pressure updates with weighted Jacobi than ISPH and MPS. Additionally, to improve the robustness, Ihmsen et al. [ICS*14] proposed adopting a velocity-based density estimation (which uses the continuity equation with predicted particle velocities) instead of the position-based density estimation (which uses the summation approach with predicted particle positions) adopted in ISPH [SL03] and MPS. In IISPH, Ihmsen et al. used a clamping approach with weighted Jacobi for free surface handling and mirrored hydrodynamic forces [AIA*12] for solid boundary handling.

There are some techniques for accelerating the pressure solve. Kang and Sagong [KS14] and Bender and Koschier [BK15] adjusted particle velocities after fluid incompressibility is enforced to reduce the deviation of particle density at the next step. Adams et al. [APKG07] and Solenthaler and Gross [SG11] used adaptive particles to allocate more computational resources to important regions. These techniques are orthogonal and can be combined with our method for better performance.

Table 1: *Feature comparison for density blending.*

Method	Fluid-fluid collision	Fluid-solid collision
Smooth kernel [MCG03]	Robust	Weak
Spiky kernel [KTO96]	Weak	Robust
Our method	Robust	Robust

Table 2: *Feature comparison for free surface handling.*

Method	Robustness
Density-based [KTO96]	Weak
Surface-based [HLW*12]	Weak
Ghost-particle-based [NT14]	Weak
Source-term-based	Moderate
Our method	Robust

We combine different advantages of ISPH, MPS, and IISPH to achieve optimal performance with our new interface handling method, which allows us to use CG with direct Laplacian discretization. Therefore, our method outperforms the previous particle-based fluid solvers including variants of ISPH with CG. For clarity, features of previous approaches and our method are summarized in Tables 1, 2, and 3, in terms of the density blending, free surface handling, and solid boundary handling, respectively.

3. Hybrid Incompressible SPH Solver

We first explain formulations of our hybrid fluid solver, which takes advantages of ISPH, MPS, and IISPH to improve the robustness and efficiency, adopting the fluid-solid coupling method [AIA*12]. Then, we present our interface handling method in § 4.

Incompressible flows in the Lagrangian setting can be described by the continuity equation $\frac{d\rho_i}{dt} + \rho_i \nabla \cdot \mathbf{u}_i = 0$, and the Navier-Stokes equations $\frac{d\mathbf{u}_i}{dt} = -\frac{1}{\rho_i}\nabla p_i + \frac{\mathbf{F}_i^{\mathrm{v}}}{m_i} + \frac{\mathbf{F}_i^{\mathrm{ext}}}{m_i}$, where ρ_i denotes density of particle i, t time, $\mathbf{u}_i$ velocity, p_i pressure, $\mathbf{F}_i^{\mathrm{v}}$ viscosity force, m_i mass, and $\mathbf{F}_i^{\mathrm{ext}}$ external force. First, we compute density ρ_i and number density n_i by using the summation approach with a smooth kernel W_{ij} and a spiky kernel w_{ij}, respectively, with solid particles:

$$\rho_i = \sum_j m_j W_{ij} + \rho_0 \sum_s V_s W_{is}, \tag{1}$$

Table 3: *Feature comparison for solid boundary handling.*

Method	System size	Pressure	Solid floating in the air
Previous method [KTO96]	Large	Inaccurate	✗
Our method	Small	More accurate	✓

$$n_i = \sum_j w_{ij} + \frac{1}{V_0}\sum_s V_s w_{is}, \tag{2}$$

where j and s denote neighbor fluid and solid particles, respectively, ρ_0 the rest density, V_i the volume, and V_0 the rest volume defined as $V_0 = \frac{1}{\sum_j W_{ij}}$ at the initial setting. We use $V_i = \frac{m_i}{\rho_i}$ for the fluid particle volume, while we define the solid particle volume as $V_i = \frac{1}{\sum_s W_{is}}$ (see [AIA*12]). Note that we initialize particle mass by $m_i = \frac{\rho_0}{\sum_j W_{ij}}$ and compute the rest number density n_0 by $n_0 = \sum_j w_{ij}$ with the initial particle configuration to enforce $\frac{\rho_i}{\rho_0} = \frac{n_i}{n_0}$ to hold, making density and number density in the different dimensions interchangeable with just scaling while achieving the equilibrium state with the initial setup. We use kernels (including a smooth kernel) proposed in [MCG03] for SPH discretization, and a spiky kernel used in [KTO96, PTB*03].

We estimate intermediate velocity $\mathbf{u}_i^*$ with viscosity and external forces. To take advantage of both of the kernels, we blend ρ_i and n_i with scaling as

$$\tilde{\rho}_i = (1-\zeta_i)\rho_i + \zeta_i \frac{\rho_0}{n_0} n_i, \tag{3}$$

$$\zeta_i = \begin{cases} 0 & M_i < \alpha \\ \frac{M_i - \alpha}{\beta - \alpha} & \alpha \le M_i < \beta \\ 1 & \beta \le M_i \end{cases},$$

where $\tilde{\rho}_i$ denotes blended density, M_i the number of neighbor solid particles, and α and β are tunable parameters (we typically use $\alpha = 5$ and $\beta = 15$). To interpolate ρ_i and n_i in Eq. (3), we compute ζ_i such that ρ_i and n_i become dominant inside of the fluid volume and near solids, respectively. Since ρ_i and n_i are more robust against fluid-fluid and fluid-solid collisions, respectively, the blended density with ζ_i takes advantages of both kernels, and is more robust than non-blended densities. Then, we compute intermediate density ρ_i^* using the continuity equation as in [ICS*14]:

$$\rho_i^* = \tilde{\rho}_i + \Delta t \left(\sum_j m_j \mathbf{u}_{ij}^* \nabla W_{ij} + \rho_0 \sum_s V_s \mathbf{u}_{is}^* \nabla W_{is} \right), \tag{4}$$

where Δt denotes time step, and $\mathbf{u}_{ij}^* = \mathbf{u}_i^* - \mathbf{u}_j^*$.

Next, we solve the PPE to obtain a pressure field that enforces fluid incompressibility. Assuming that the density change in the continuity equation is caused by the pressure forces, we formulate the PPE for the fluid domain Ω as $\nabla^2 p_i = \frac{\rho_0 - \rho_i^*}{\Delta t^2}$ in Ω with Dirichlet boundary condition $p_i = 0$ on F and Neumann boundary condition $\frac{dp_i}{d\hat{\mathbf{n}}_i} = 0$ on S, where F and S denote domain boundaries in contact with free surfaces and solid objects, respectively, and $\hat{\mathbf{n}}_i$ is the normal to S. In particle-based methods, negative pressures which can occur because of the positive source term if the PPE is solved without special cares produce attractive forces between particles leading to tensile instability [Mon00]. Thus, we need to satisfy $p_i \ge 0$ while solving the PPE, and this is a Linear Complementarity Problem (LCP), e.g., solved to avoid particle adhesion [AW09, NGL10, AO11, IWT13, GB13, ICS*14].

After the PPE is solved by using a CG solver, satisfying the boundary conditions and constraint on pressure with our interface handling (see § 4), we compute pressure forces, and then integrate particle velocities and positions using the semi-implicit Euler method.

We summarize the algorithm for our hybrid incompressible SPH solver in Algorithm 1. An algorithm for solving the PPE (line 9 in Algorithm 1) is given in Algorithm 2 (§ 4.6).

Algorithm 1 Hybrid incompressible SPH solver

```
1:  for all particle i do
2:      find neighbor particles
3:  for all fluid particle i do
4:      compute ρ_i and n_i with Eqs. (1) and (2)
5:  for all particle i do
6:      compute intermediate velocity u_i^*
7:  for all fluid particle i do
8:      estimate ρ_i^* with Eq. (4)
9:  solve the PPE (Algorithm 2)
10: for all fluid particle i do
11:     compute pressure force F^p
12: for all fluid particle i do
13:     integrate velocity u_i^{t+1} and position x_i^{t+1}
```

4. Interface Handling

In this section, we concisely describe boundary conditions in the PPE (§ 4.1). Next, we clarify problems on Dirichlet (§ 4.2) and Neumann (§ 4.3) boundary conditions in the ISPH and MPS setting, and then explain our interface handling consisting of free surface handling (§ 4.4) and solid boundary handling (§ 4.5). Finally, we give an algorithm for solving the PPE (§ 4.6).

We use our solid boundary handling method for Neumann boundary conditions in the PPE and employ the work of Akinci et al. [AIA*12] to resolve collisions between fluid and solid particles. [AIA*12] is also suitable for our free surface handling method since it allows for accurate density estimates even with complex geometry. Our free surface handling is for Dirichlet boundary conditions and is orthogonal to other free surface handling methods that consider air domains, e.g., [SB12, HWZ*14]. Thus, these methods can be combined with ours.

In our experiments, quantities computed with SPH and MPS discretization for the Laplacian operator are almost the same and are virtually interchangeable. However, since the gradient formulation in MPS is not anti-symmetric, we use SPH discretization for consistency to compute the spatial derivatives.

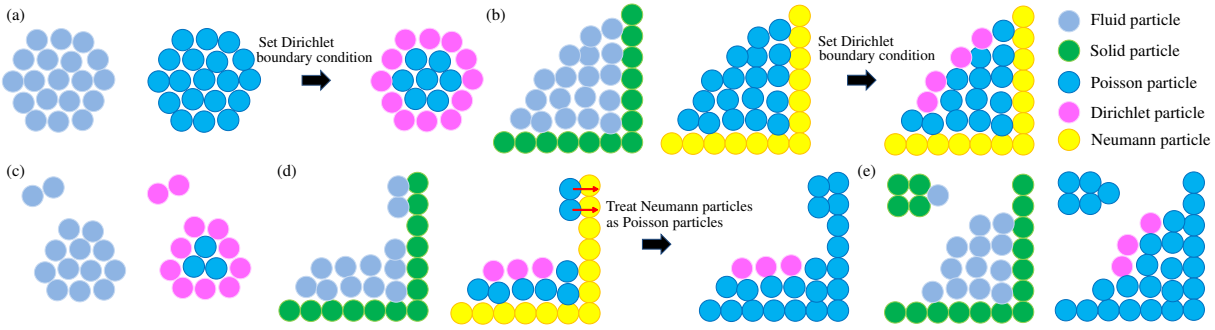

Figure 2: *Illustration of particle configurations. Red arrows represent particle velocities. (a) Configuration of fluid particles, unsolvable configuration of Poisson particles, and solvable configuration of Poisson and Dirichlet particles due to the newly set Dirichlet boundary condition, from left to right. (b) Configuration of fluid and solid particles, unsolvable configuration of Poisson and Neumann particles, and solvable configuration of Poisson, Dirichlet, and Neumann particles. (c) Configuration of fluid particles, and solvable configuration of Poisson and Dirichlet particles. (d) Configuration of fluid and solid particles, unsolvable configuration of Poisson, Dirichlet, and Neumann particles, and solvable configuration of Poisson and Dirichlet particles due to the new Poisson particles converted from the Neumann particles. (e) Configuration of fluid and solid particles, and unsolvable configuration of Poisson and Dirichlet particles due to the group of isolated Poisson particles without Dirichlet particles.*

4.1. Boundary Conditions in PPE

To aid in our description, we call fluid and solid particles included in the PPE as unknown variables *Poisson particles*, fluid particles used for Dirichlet boundary condition *Dirichlet particles*, and solid particles for Neumann boundary condition *Neumann particles*.

To solve the PPE, we can discretize $\nabla^2 p_i$ using the SPH formulation taking solid particles into account with the assumption of virtually existing solid particle's pressure p_s:

$$\begin{aligned}\nabla^2 p_i &= \nabla^2 p_i^{\text{fluid}} + \nabla^2 p_i^{\text{solid}} \\ &= \sum_j a_{ij}(p_i - p_j) + \sum_s a_{is}(p_i - p_s), \end{aligned} \tag{5}$$

where $a_{ij} = V_{ij}\hat{W}_{ij}$, $V_{ij} = (V_i + V_j)/2$, $\hat{W}_{ij} = 2\frac{\mathbf{x}_{ij}\cdot\nabla W_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2}$, $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ ($\mathbf{x}_i$: particle position), and h denotes kernel radius. $a_{ij} < 0$ because of the kernel definition [MCG03]. For Neumann boundary condition ($\frac{dp_i}{d\mathbf{n}_i} = 0$), $p_i = p_s$ must be satisfied, and thus the PPE becomes

$$\nabla^2 p_i = \sum_j a_{ij}(p_i - p_j) = c_i,$$

where $c_i = \frac{\rho_0 - \rho_i^*}{\Delta t^2}$. Unlike the Eulerian fluid simulation, which uses the divergence of velocity as a source term, the density invariance source term generally does not satisfy the compatibility condition (even if the source term is not blended), i.e., $\sum_{i\in\Omega^{\mathrm{P}}} c_i \neq 0$, where Ω^{P} denotes a set of Poisson particles. Therefore, this form of the PPE without Dirichlet boundary conditions is unsolvable because of the rank deficient coefficient matrix (see [Bri08] for more details). Figure 2 (a) illustrates a particle configuration without solid particles for an unsolvable setting consisting of Poisson particles only, and a solvable setting due to Dirichlet particles. Figure 2 (b) illustrates a similar situation with solid Neumann particles.

4.2. Problems on Dirichlet Boundary Condition

Unlike Eulerian fluid simulation, negative pressures almost always work negatively, causing the tensile instability [Mon00] in particle-based fluid simulation. Thus, it is essential to avoid negative pressures, and formally, we need to solve an LCP. With (weighted) Jacobi method, this step can be easily achieved by clamping negative pressures to zero in each iteration, whereas CG does not allow us to perform such an operation in its iterations [ICS*14]. More expensive LCP solvers have been used in the Eulerian methods [NGL10, GB13]. By contrast, in Lagrangian particle-based methods, we approximately solve the LCP to avoid using the costly LCP solvers by solving the PPE with Dirichlet boundary conditions, which are set to particles whose pressures should be negative after the PPE is solved (See Figure 2). Previously, the density-based [KTO96, PTB*03, SL03] or surface-based [HLW*12] method has been used to determine Dirichlet particles. However, since predicted density and pressures of fluid particles are neglected, these methods tend to fail to appropriately determine Dirichlet particles, i.e., fluid particles whose pressures should be positive (negative) after the PPE is solved could be erroneously treated as Dirichlet (Poisson) particles consequently leading to stability issues. The source-term-based method takes the predicted density into account, and thus its stability is improved. However, disregarded neighbor particles' pressures still negatively affect the stability of the simulation.

The ghost-particle-based method [NT14] can change an unsolvable system consisting of Poisson and Neumann particles only (without Dirichlet particles) to a solvable one by setting Dirichlet boundary conditions to ghost particles, assuming that they virtually exist outside of the fluid domain. When the particle uniformity is disturbed, however, the linear system can be not diagonally-dominant, and thus not positive-definite. Consequently, we fail to solve the system using CG. Although taking larger diagonal entries can ensure the diagonal dominance of the linear system, resultant pressures can be much smaller failing to prevent particle penetrations with mirrored hydrodynamic forces [AIA*12].

4.3. Problems on Neumann Boundary Condition

When Poisson particles have a channel to at least one Dirichlet particle directly or via other Poisson particles, the solvability of the PPE is ensured. In liquid simulations, however, fluid particles can be separated from the fluid bulk that has Dirichlet particles (see Figures 2 (c) and (d)). If there is no neighboring solid particle, we can appropriately set Dirichlet boundary conditions with a free surface handling method ensuring the solvability of the system, as illustrated in Figure 2 (c). In this case, although pressure forces of Dirichlet particles do not resolve their collisions because of the zero-pressures, artificial viscosity force, which is necessary to stabilize particle behaviors, can prevent fluid particle penetrations. On the other hand, if fluid particles have neighboring solid particles (see Figure 2 (d)), we cannot set Dirichlet boundary condition ($p_i = 0$) to the fluid particles except the cases, where fluid particles do not move toward solid particles. This is because pressures of fluid particles must be valid for [AIA*12], and we need to treat the fluid particles as Poisson particles. In this case, the resulting linear system is unsolvable because of isolated groups of Poisson particles without Dirichlet particles (see Figure 2 (d)). To avoid this unsolvable configuration, we tested pressures computed with an equation of state instead of setting $p_i = 0$ and artificial viscosity excluding Poisson particles that contact with Neumann particles from the system, we could not determine appropriate parameters and experienced significant fluid energy dissipations.

This problem was partially addressed in [KTO96, PTB*03, SL03] by treating solid particles as Poisson particles in the PPE (see Figure 2 (d)). This approach can connect separated fluid Poisson particles to Dirichlet particles via solid Poisson particles which are originally Neumann particles, and thus can change an unsolvable PPE to a solvable one. However, this approach brings about several other problems. First, solid Poisson particles newly included in the PPE significantly increase the size of the system and thus computational cost and memory usage. Second, this approach tends to underestimate pressures requiring smaller time steps to ensure no fluid-solid particle penetrations because solid Poisson particles newly generate shorter channels between fluid Poisson particles and Dirichlet particles limiting the pressures of the fluid Poisson particles to lower values. Most importantly, this approach cannot handle solid objects floating in the air, which frequently occur in the two-way solid-fluid coupling (see Figure 1 and accompanying video), since solid particles of these objects may not have channels to Dirichlet particles (see Figure 2 (e)).

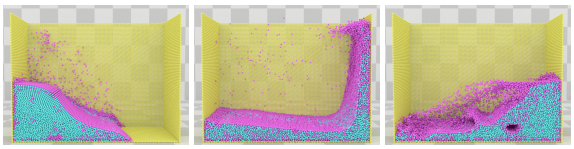

Figure 3: *Cutaway views for a dam break scene. Dirichlet particles are appropriately set near free surfaces and cavities inside of the fluid.*

4.4. Free Surface Handling

We aim to approximately solve the LCP by appropriately setting Dirichlet boundary conditions to particles, whose pressures would be negative after the PPE is solved. To estimate particle pressures, we compute a pseudo particle pressure $\tilde{p}_i$ using the projected Jacobi method based on Eq. (5):

$$\tilde{p}_i^{l+1} = \max\left(0, \frac{c_i + \sum_j a_{ij}\tilde{p}_j^l + \tilde{p}_i^l \sum_s a_{is}}{\sum_j a_{ij} + \sum_s a_{is}}\right), \tag{6}$$

where l denotes iteration count, and the initial pseudo pressures are set as $\tilde{p}_i^0 = \gamma p_i^t$ with a tunable parameter γ. Then, if $\tilde{p}_i^l = 0$, we treat the fluid particle i as a Dirichlet particle, otherwise a Poisson particle. We use a projected Jacobi method, not Jacobi method, since Jacobi method excessively increases the number of Dirichlet particles propagating negative pressures. Note that since our Jacobi-based prediction includes pressures of neighboring particles and the source term, which is based on the predicted density ρ_i^* and thus density ρ_i, we can appropriately set Dirichlet boundary conditions to particles, e.g., on free surfaces and near cavities inside of fluid, as shown in Figure 3.

Since repeatedly applying Eq. (6) means solving the PPE using the projected Jacobi method as in [ICS*14], we can obtain more accurate particle pressures to determine Dirichlet particles with many iterations. However, using many iterations did not significantly improve the stability. Additionally, since all particle pressures are zero or positive, repeatedly applying Eq. (6) generally increases pseudo pressures and thus decreases the number of Dirichlet particles. Whereas Dirichlet particles can cause stability issues if they are excessive and inappropriately set, they can reduce the number of Poisson particles in the system improving computational efficiency and memory usage and accelerate the convergence of CG because of their fixed pressure values and the decreased system size. Taking these into account,

we use Eq. (6) only once with $\gamma = 0.5$ not to excessively decrease the number of Dirichlet particles.

Since negative pressures can still occur with Dirichlet particles specified using our free surface handling, we clamp negative pressures to zero to prevent the tensile instability [Mon00] after the PPE is solved.

It is worth noting that an ad-hoc technique, clamping the positive source term to zero, can completely prevent negative pressures. However, using this technique with our method entirely and excessively increases particle pressures leading to stability problems or surface artifacts [ICS*14].

4.5. Solid Boundary Handling

To handle solid objects including isolated ones without treating solid particles as Poisson particles in the PPE, we aim to compute the Laplacian without including the term $p_i - p_s$, ensuring the solvability of the linear system even with Neumann boundary condition $\frac{dp_i}{d\mathbf{n}_i} = 0$, i.e., $p_i = p_s$ applied. Inspired by IISPH, we decompose the Laplacian operator for solid particles into divergence and gradient as

$$\nabla^2 p_i^{\text{solid}} \approx \nabla \cdot \nabla p_i^{\text{solid}} = -\sum_s V_{is} \frac{\nabla p_i + \nabla p_s}{2} \cdot \nabla W_{is}. \quad (7)$$

Assuming $\nabla p_i = \nabla p_s$ when $p_i = p_s$ (Neumann boundary condition) within a local domain because of the smoothed distribution of physical quantities in the SPH setting [SP09], we obtain

$$\sum_s V_{is} \frac{\nabla p_i + \nabla p_s}{2} \cdot \nabla W_{is} = \nabla p_i \cdot \sum_s V_{is} \nabla W_{is}. \quad (8)$$

By using the SPH formulation, we can compute ∇p_i by

$$\nabla p_i = \sum_j V_{ij} \frac{p_i + p_j}{2} \nabla W_{ij} + \sum_s V_{is} \frac{p_i + p_s}{2} \nabla W_{is}.$$

Again, because of the smoothed pressures of fluid particles within their local domain [SP09], we assume $p_i = p_j$. Combining this assumption and Neumann boundary condition, we obtain

$$\nabla p_i = p_i \left(\sum_j V_{ij} \nabla W_{ij} + \sum_s V_{is} \nabla W_{is} \right). \quad (9)$$

We did not gain any improvement in the robustness even if the PPE is solved with $p_i \neq p_j$, and the matrix construction cost significantly increased because of the coefficient of p_j. With Eqs. (7), (8), and (9), we obtain

$$\nabla^2 p_i^{\text{solid}} = b_i p_i,$$
$$b_i = -\left(\sum_j V_{ij} \nabla W_{ij} + \sum_s V_{is} \nabla W_{is} \right) \cdot \sum_s V_{is} \nabla W_{is}. \quad (10)$$

Since our solid boundary handling introduces a new term $b_i p_i$ into the PPE when fluid Poisson particles are in contact with Neumann particles, we can change an unsolvable system to a solvable one. For example, when there are two fluid particles i and j touching each other and a solid particle s touching at least one of i and j, the PPE for i and j without our solid handling can be written as $a_{ij}(p_i - p_j) = c_i$ and $a_{ji}(p_j - p_i) = c_j$, respectively, where $a_{ij} = a_{ji} \neq 0$. Therefore, $c_i + c_j = 0$ must hold to ensure the solvability of the PPE for the compatibility condition [Bri08]. In general, however, $c_i + c_j \neq 0$ with the density invariance source term, and thus the linear system is unsolvable. On the other hand, with our solid boundary handling, we can write the PPE as $a_{ij}(p_i - p_j) + b_i p_i = c_i$ and $a_{ji}(p_j - p_i) + b_j p_j = c_j$, where at least one of $b_i \neq 0$ and $b_j \neq 0$ holds, and therefore this system is solvable ($p_i = \frac{a_{ij}(c_i + c_j) + b_j c_i}{a_{ij}(b_i + b_j) + b_i b_j}$ and $p_j = \frac{a_{ij}(c_i + c_j) + b_i c_j}{a_{ij}(b_i + b_j) + b_i b_j}$).

It is worth noting that b_i is generally negative when particle i is compressed toward solid particles, and thus our solid boundary handling is likely to lead to smaller pressures near solid objects, as p_i without our solid handling can be computed by $p_i = \frac{c_i + \sum_j a_{ij} p_j}{\sum_j a_{ij}}$, and $p_i = \frac{c_i + \sum_j a_{ij} p_j}{\sum_j a_{ij} + b_i}$ with our solid handling, where a_{ij} is also negative because of the kernel definition [MCG03]. However, the blended density can increase particle pressures near solid objects because of the spiky kernel, and effectively prevent particle penetrations.

Our solid boundary handling is decoupled from pressures of neighbor fluid and solid particles. This decoupling allows us to significantly simplify our implementation and efficiently compute contributions from solid particles without multiple access to fluid and solid particles and matrix structures specialized for sparse linear systems. In addition, our method does not increase the size of the linear system nor CG iterations, unlike the solid handling used in [KTO96, PTB*03, SL03].

Our solid boundary handling is numerically equivalent to implicitly adding Dirichlet boundary condition to the system, to make the system solvable. Thus, in this perspective, our method and the ghost-particle-based method [NT14] share the same idea. However, our method cannot handle isolated fluid Poisson particles with no channels to Dirichlet particles (this case is handled by setting Dirichlet boundary condition to such particles with our free surface handling), and needs the fluid Poisson particles to be touching Neumann particles to ensure the solvability. On the other hand, the ghost-particle-based method can handle fluid Poisson particles without Neumann and Dirichlet particles at the expense of difficult parameter tuning and underestimated pressures.

4.6. PPE Solve

Algorithm 2 summarizes an algorithm for solving the PPE. Since our free surface and solid boundary handling can be incorporated into the originally existing particle loops, the additional cost is trivial.

Algorithm 2 PPE solve

for all fluid particle i **do**
  compute source term c_i
  determine Dirichlet particles with Eq. (6)
  if i is Dirichlet particle **then**
    $p_i = 0$
  else
    for all fluid particle j **do**
      compute a_{ij} for the matrix
      compute $V_{ij}\nabla W_{ij}$
    for all solid particle s **do**
      compute $V_{is}\nabla W_{is}$
    compute b_i with Eq. (10) for the matrix
solve the linear system using a CG solver
for all fluid particle i **do**
  if $p_i < 0$ **then**
    $p_i = 0$

5. Results

We implemented our algorithm with artificial viscosity [Mon92], and used incomplete Cholesky CG for solving the linear system. In our method, we use $h = 2d$ (d: initial particle spacing). Light blue and green particles represent fluid and solid particles, respectively. Cyan, magenta, and yellow particles represent Poisson, Dirichlet, and Neumann particles, respectively. We measured computational time for a PC with 4-core 3.40 GHz CPU and RAM 16.0 GB, excluding the surface reconstruction and rendering from the measurement. N^{f} and N^{s} denote the number of fluid and solid particles, respectively. N^{P}, N^{D}, and N^{N} denote the averaged number of Poisson, Dirichlet, and Neumann particles, respectively. l^{avg} denotes the averaged number of CG iterations per simulation step. t^{p} and t^{f} denote averaged computational time for solving the PPE and for the frame, respectively. Particles are rendered at 60 Hz, and the result of the last simulation step in the frame is used for the color code of particle rendering.

5.1. Convergence Criterion

While the residual is commonly used in the Eulerian fluid simulation, a density-based criterion, based on the positive density error $(\rho_i^{\mathrm{err}})^l = \max(0, \rho_i^l - \rho_0)$ (ρ_i^l: estimated density after l iterations), which represents the level of fluid compression, is traditionally used as a convergence criteria in the SPH fluid simulations [SP09, ICS*14]. Ihmsen et al. [ICS*14] proposed using the average density error, not the maximum one, to keep fluid volumes constant. For the intuitiveness and fair comparison with previous methods, we use the density-based criterion with the averaged positive density error.

In each CG iteration, we can obtain the residual r_i^l without explicitly computing it, following its definition given as $r_i^l = \frac{\rho_0 - (\rho_i^*)^l}{\Delta t^2} - \nabla^2 p_i^l$. Since the estimated density ρ_i^l can be computed by $\rho_i^l = (\rho_i^*)^l + \Delta t^2 \nabla^2 p_i^l$ [ICS*14], we can obtain the positive density error with the residual r_i^l as $(\rho_i^{\mathrm{err}})^l = \max(0, \rho_i^l - \rho_0) = \max(0, -r_i^l \Delta t^2)$. Then, we compute the averaged, positive density error $(\hat{\rho}^{\mathrm{err}})^l = \frac{1}{N}\sum_{i\in\Omega^P}(\rho_i^{\mathrm{err}})^l$ (N: the number of Poisson particles). If $(\hat{\rho}^{\mathrm{err}})^l/\rho_0 < \eta$ (η: error threshold), we terminate the CG iterations. We use $\eta = 0.01\%$ in all the scenarios.

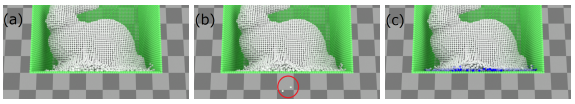

Figure 4: *Stability test with a bunny drop, where averaged particle spacing is 1.50×10^{-2} m. Smooth kernel with (a) $\Delta t = 1.03 \times 10^{-3}$ s and (b) $\Delta t = 1.40 \times 10^{-3}$ s, where particle penetrations occur as noted by a red circle. (c) Blended density with $\Delta t = 1.40 \times 10^{-3}$ s.*

5.2. Density Blending

We tested our density blending scheme by comparing our method with the density blending and without it (i.e., our method using a smooth or spiky kernel only). Particles are color-coded based on ζ_i (when ρ_i (n_i) computed with the smooth (spiky) kernel is dominant, particle colors approach white (blue)).

Comparison with the smooth kernel. Figure 4 compares our method using the density blending and a smooth kernel only. When the time step is large, our method with the smooth kernel suffers from particle penetrations, as shown in Figure 4 (b). Our method with density blending can prevent penetrations because of the rapidly increased pressure by the spiky kernel, as shown in Figure 4 (c), taking 1.36x larger time steps.

Comparison with the spiky kernel. Figure 5 compares our method using the density blending and a spiky kernel only, where very large time steps are chosen to clearly show the differences in the robustness. When the time step is large, our method with the spiky kernel can become unstable and causes particle penetrations, as shown in Figure 5 (b). Our method with density blending does not cause particle penetrations because of the blended smooth kernel, as shown in Figure 5 (c), taking 1.46x larger time steps.

5.3. Free Surface Handling

We tested our free surface handling to demonstrate its robustness with a simple scene, where a cubed fluid is dropped from a lower position to make the free surface handling crucial in determining time steps (see Figure 6 (a) for the initial setup). We compared our method

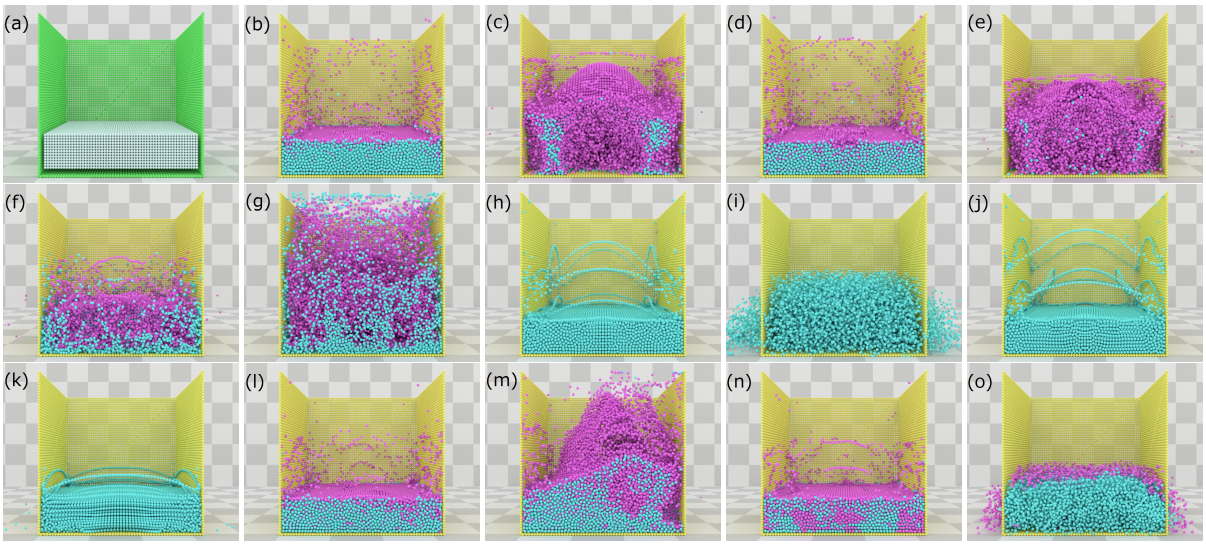

Figure 6: *Comparison for free surface handling, where averaged particle spacing is* 1.50×10^{-2} *m. Different frames are chosen for illustration. (a) Initial setup. Density-based with (b)* $\Delta t = 0.75 \times 10^{-3}$ *s (stable) and (c)* $\Delta t = 1.68 \times 10^{-3}$ *s (unstable). Density-based (clamped) with (d)* $\Delta t = 0.75 \times 10^{-3}$ *s (stable) and (e)* $\Delta t = 1.68 \times 10^{-3}$ *s (unstable). (f) Surface-based with* $\Delta t = 0.35 \times 10^{-3}$ *s (unstable). (g) Surface-based (clamped) with* $\Delta t = 0.35 \times 10^{-3}$ *s (unstable). Ghost-particle-based with (h)* $\Delta t = 0.42 \times 10^{-3}$ *s (stable) and (i)* $\Delta t = 1.68 \times 10^{-3}$ *s (unstable). Ghost-particle-based (clamped) with (j)* $\Delta t = 0.52 \times 10^{-3}$ *s (stable) and (k)* $\Delta t = 1.68 \times 10^{-3}$ *s (unstable). Source-term-based with (l)* $\Delta t = 1.20 \times 10^{-3}$ *s (stable) and (m)* $\Delta t = 1.68 \times 10^{-3}$ *s (unstable). (n) Ours with* $\Delta t = 1.68 \times 10^{-3}$ *s (stable). (o) Ours (clamped) with* $\Delta t = 1.68 \times 10^{-3}$ *s (unstable).*

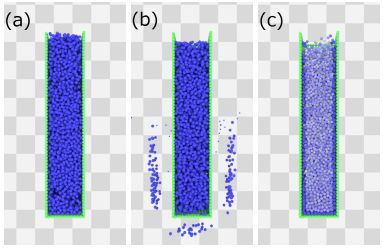

Figure 5: *Stability test with a fluid column, where averaged particle spacing is* 1.50×10^{-2} *m. Spiky kernel with (a)* $\Delta t = 0.73 \times 10^{-3}$ *s and (b)* $\Delta t = 1.07 \times 10^{-3}$ *s, where particle penetrations occur. (c) Blended density with* $\Delta t = 1.07 \times 10^{-3}$ *s.*

with the density-based [KTO96, PTB*03, SL03], surface-based [HLW*12], ghost-particle-based [NT14], and source-term-based method, which treats fluid particles as Dirichlet particles if $\rho_0 - \rho_i^* \geq 0$, combining the clamping approach [ICS*14]. For the comparisons, we used our solid boundary handling. Figure 6 illustrates the results of our method and others.

Density-based method. While the density-based method with a small time step as in Figure 6 (b) generates plausible fluid behaviors, the method suffered from a stability problem with a large time step seen in Figure 6 (c). The clamping did not improve nor deteriorate the robustness of this method (see Figure 6 (d) and (e)).

Surface-based method. The surface-based method sets Dirichlet particles only near surfaces, and consequently particle pressures are likely to be excessively high inside of the fluid volume. Thus, this method failed to perform a stable simulation even with a small time step as shown in Figure 6 (f). Since the clamping approach increases particle pressures, this technique did not improve the stability as shown in Figure 6 (g).

Ghost-particle-based method. Because of the underestimated pressures, a small time step was necessary to ensure no particle penetrations as in Figure 6 (h). This method failed with a large time step as in Figure 6 (i). The increased pressure using the clamping slightly improved the robustness of this method, allowing for the use of a larger time step as seen in Figure 6 (j). The clamping was insufficient to take a very larger time step as in Figure 6 (k). For all the scenes, we observed the volume change (fluid oscillation) because of underestimated pressures even though the convergence criterion is satisfied.

Source-term-based method. The source-term-based method can perform a stable simulation with a moderately large time step as in Figure 6 (l) while the simulation becomes unstable with a large time step as in Figure 6 (m).

The clamping approach is ineffective for this method because Dirichlet particles determined by the source term are excluded from the system.

Our method. Our method can perform a stable simulation with a large time step as in Figure 6 (n), while the clamping introduced stability issues due to increased pressures (see Figure 6 (o)).

Table 4 shows simulation conditions and performance for Figures 6 (b), (j), (l), and (n), where the density-based, ghost-particle-based, source-term-based, and our method generate plausible fluid behaviors with their best performance, respectively. Since our method can take larger time steps than the others, ours outperformed the others regardless of the more iterations required for convergence, amortizing the increased cost of the pressure solve. When we need additional computations, e.g., for viscosity, stress, temperature, and surface tension, taking larger time steps can further increase the performance gain of our method.

The number of Poisson (Dirichlet) particles with our method is smaller (larger) than that of the source-term-based method in Table 4. However, this is due to different time steps, and the number of Poisson and Dirichlet particles was 34.4k (> 32.0k) and 7.8k (< 10.3k), respectively, when the same time step as in Figure 6 (l) was used with our method.

5.4. Solid Boundary Handling

We tested our solid handling method to demonstrate its robustness and efficiency with a dam break scene with a solid dragon (see Figure 7 (a) for the initial setup). Note that the dragon must be connected to the surrounding cube representing the simulation domain to ensure the solvability of the linear system for [KTO96, PTB*03, SL03]. We compare these methods in Figure 7, adopting our free surface handling for both methods, and show simulation conditions and performance in Table 5.

While both methods can generate plausible fluid behaviors, the increased number of Poisson particles with the previous method leads to a larger size of the linear system, requiring more computations. Additionally, the previous method erroneously underestimates particle pressures, and the averaged maximum pressure was 1.88×10^4 and 2.32×10^4 kg/(m·s^2) in Figures 7 (d) and (f), respectively, with the same time step $\Delta t = 1.09 \times 10^{-3}$ s. Thus, the previous method [KTO96, PTB*03, SL03] needed to use smaller time steps than those for our method to prevent particle penetrations. Consequently, our method outperformed the previous method by a factor of 2.20.

Additionally, our method can handle solid objects floating in the air (see Figure 8).

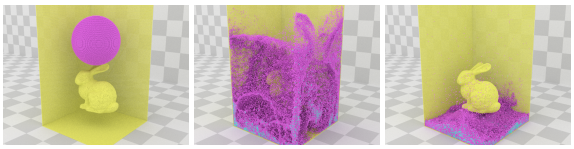

Figure 8: *A fluid drop with a floating solid bunny.*

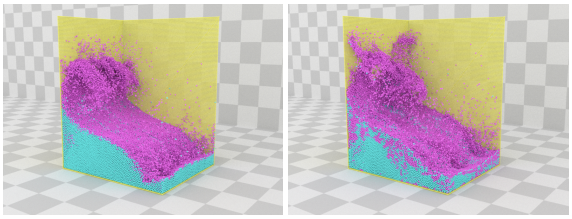

Figure 9: *Comparison of fluid behaviors. (Left) IISPH. (Right) Our method.*

5.5. Comparison with IISPH

We compare our method with IISPH [ICS*14] using a dam break scenario, as shown in Figure 9 for the fluid behaviors and Figure 10 for their performance of the pressure solve. While both methods generate comparable results, our method outperformed by a factor of 3.78 in the pressure solve (IISPH used 86.73 s while ours used 22.95 s) because of the fast convergence of CG.

5.6. Time Step Scaling

To compare the performance with different time steps, we experimented with a dam break scene, shown in Figure 9. For this comparison, averaged particle spacing was 1.80×10^{-2} m. Table 6 shows performance results with different time steps. With our method, the number of iterations increases *sublinearly* according to time steps. The sublinearity offers an advantage that our method generally achieves the

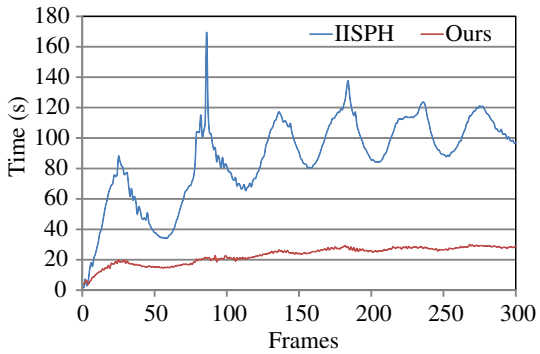

Figure 10: *Comparison of pressure solve time for Figure 9. Our method outperforms IISPH by a factor of 3.78.*

Table 4: *Performance comparison for Figure 6. Ours achieves the best performance, taking the largest time step due to the improved robustness.*

Figure	Method	N^{f}	N^{s}	N^{P}	N^{D}	N^{N}	$\Delta t(s)$	l^{avg}	t^{p} (s)	t^{f} (s)
6 (b)	Density-based	42.2k	21.8k	40.0k	2.3k	21.8k	0.75×10^{-3}	2.92	4.79	6.64
6 (j)	Ghost-particle-based (clamped)	42.2k	21.8k	42.2k	0.0k	21.8k	0.52×10^{-3}	0.99	5.95	8.85
6 (l)	Source-term-based	42.2k	21.8k	32.0k	10.3k	21.8k	1.20×10^{-3}	1.96	2.20	3.23
6 (n)	Ours	42.2k	21.8k	28.3k	14.0k	21.8k	1.68×10^{-3}	4.66	2.16	2.97

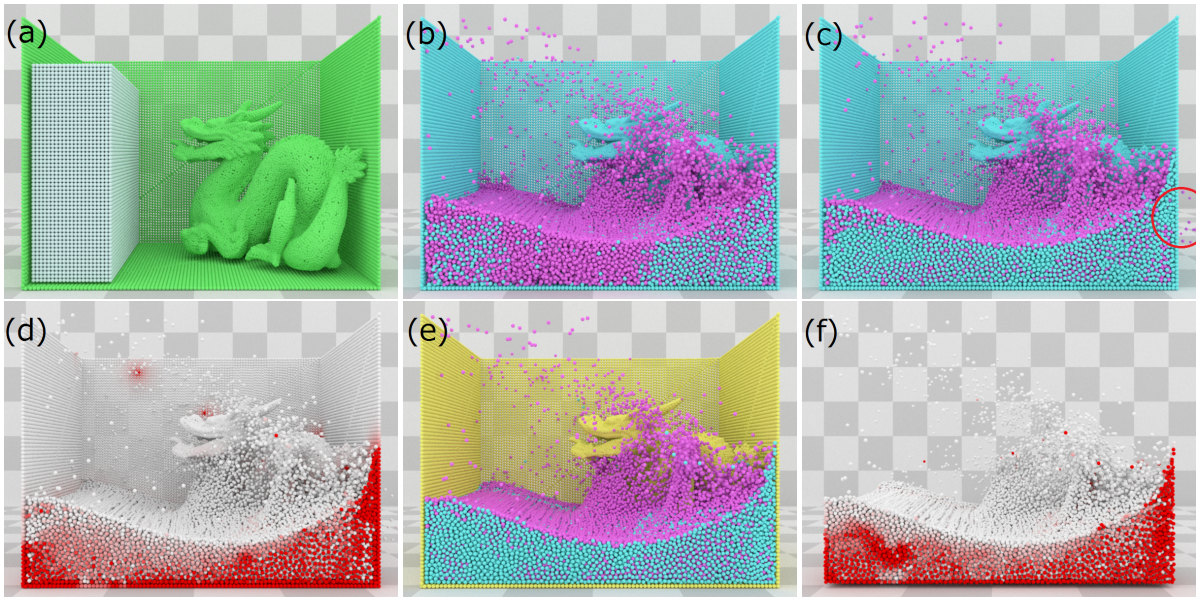

Figure 7: *Comparison for solid boundary handling, where averaged particle spacing is 1.29×10^{-2} m. (a) Initial setup. Previous method with (b) $\Delta t = 0.51\times10^{-3}$ s and (c) with $\Delta t = 1.09\times10^{-3}$ s, where particle penetrations occur as noted by a red circle; and (d) with particles color coded based on their pressures. (e) Our method with $\Delta t = 1.09\times10^{-3}$ s, and (f) with particles color coded based on their pressures. Red and white particles represent high and low pressures, respectively.*

Table 6: *Performance results with different time steps. With our method, l^{avg} increases sublinearly w.r.t. time steps.*

$\Delta t(s)$	l^{avg}	$t^{\mathrm{p}}(s)$	$t^{\mathrm{f}}(s)$
0.20×10^{-3}	0.99	12.27	24.93
1.60×10^{-3}	6.69	2.96	4.69

best performance with the largest available time steps, simplifying a process of finding optimal time steps. By contrast, the number of iterations for IISPH with weighted Jacobi increases superlinearly [ICS*14], and thus IISPH needs to take into account various factors, e.g., pressure solve and neighbor search, to determine time steps for the optimal performance of IISPH.

6. Discussions and Limitations

In this section, we discuss applicability of CG to IISPH (§ 6.1), and limitations of our solver (§ 6.2).

6.1. CG for IISPH

In [ICS*14], the authors attempted to solve the PPE using CG with the assumption of uniformly constant particle mass and density to make the system symmetric. With the IISPH discretization, however, the system can be not diagonally dominant and thus not positive definite.

The left hand side of the PPE in the IISPH can be written as $-\frac{m^2}{\rho_0^2}\sum_j\left(\sum_j(p_i+p_j)\nabla W_{ij}-\sum_k(p_j+p_k)\nabla W_{jk}\right)\nabla W_{ij}$ (j: first-ring neighbors and k: second-ring neighbors), including second-ring neighbors. According to [TDF*15], we can separately write terms on p_i, p_j, and p_k with $\omega_{ij}=\sum_j\nabla W_{ij}$ as $-\frac{m^2}{\rho_0^2}\|\omega_{ij}\|^2 p_i$,

Table 5: *Performance comparison for Figure 7. Ours outperforms the previous method by taking a 2.14x larger time step, and achieves the performance gain by a factor of 2.20.*

Figure	Method	N^{f}	N^{s}	N^{P}	N^{D}	N^{N}	$\Delta t(s)$	l^{avg}	t^{p} (s)	t^{f} (s)
7 (b)	Previous method	76.6k	57.1k	90.6k	43.1k	0.0k	0.51×10^{-3}	2.26	16.62	26.37
7 (e)	Our method	76.6k	57.1k	59.1k	17.5k	57.1k	1.09×10^{-3}	4.39	6.99	11.97

$-\frac{m^2}{\rho_0^2}\sum_j \nabla W_{ij}(\omega_{ij}-\omega_{jk})p_j$, and $-\frac{m^2}{\rho_0^2}\omega_{ij}\sum_k \nabla W_{jk}p_k$, respectively, and the diagonal component is the coefficient of p_i, which is $-\frac{m^2}{\rho_0^2}\omega_{ij}\cdot(\omega_{ij}+\sum_j \nabla W_{ji})$ taking contributions of k into account (particle k can be particle i). When particle i has a completely uniform neighbor particles ($\omega_{ij}=0$) and j does not have ($\omega_{jk}\neq 0$), this system is not diagonally dominant. Therefore, this system cannot be solved with CG.

It is worth noting that Jacobi method also cannot solve a non-diagonally-dominant system, and thus *weighted* Jacobi method is used in [ICS*14].

6.2. Limitations

Our density blending and interface handling methods significantly improve the stability, and the improved robustness of our solver can be comparable to IISPH depending on scenarios. However, our solver can be less robust than IISPH when very large time steps and high resolutions are used, since IISPH considers pressure forces with farther particles in the PPE, generating more reliable pressures. Although there are many situations where our method is more advantageous (e.g., when we cannot use very large time steps because of fast moving particles), the weaker robustness is a limitation and should be investigated to make our solver more robust. Additionally, IISPH can be advantageous for relatively smaller scenarios consisting of shallow water only, with a soft constraint on volume changes, because fewer Jacobi iterations can be sufficient to converge, and thus our method cannot benefit from CG, which shows a fast, superlinear convergence in the latter phase.

Our density blending uses two tunable parameters. Although we were able to use constant values (as suggested in Section 3) in our scenarios, when the distributions of solid particles are highly irregular, it would be necessary to take into account actual contributions from solid particles. Other than blending, it might also be possible to design a new shape of kernels to address this issue.

7. Conclusions and Future Work

We proposed a hybrid incompressible SPH solver with a new interface handling method. Our density blending improves the stability for both fluid-fluid and fluid-solid collisions. Our free surface handling improves the robustness by appropriately setting Dirichlet particles with Jacobi-based pressure prediction while our solid boundary handling introduces a new term to ensure the solvability of the linear system even with objects floating in the air, without sacrificing memory and computational efficiency. Several examples demonstrated the effectiveness of our method.

In general, our solver becomes more efficient as we take larger time steps unlike IISPH with weighted Jacobi since required iterations sublinearly increase according to time steps. Adopting a more effective preconditioner would further accelerate our method, solving the linear system with fewer iterations even if larger time steps make the system ill-conditioned. Additionally, such a preconditioner might allow us to handle larger scale scenarios with deep water depth, for which both of our solver with CG and IISPH with weighted Jacobi show a poor scalability.

It is interesting to adopt different approaches to ensure the solvability of the linear system. We plan to use ghost SPH [SB12] to generate particles outside of fluid volumes, which can be used for Dirichlet boundary condition. Though this is similar to [NT14], we can use better sampled particles with the ghost SPH, and this would further improve the robustness. To simulate fluids with our method in a closed container, i.e., Neumann boundary condition only (without free surfaces, i.e., Dirichlet boundary condition), adopting the source term correction approach described in [Bri08] would be promising.

Acknowledgements

This work is supported in part by JASSO for Study Abroad, JST CREST, U.S. National Science Foundation, and UNC Arts and Sciences Foundation. We would like to thank Matthias Teschner and anonymous reviewers for their valuable suggestions and comments. We also thank Pavel Krajcevski for helping to proofread the final version of this paper.

References

[AIA*12] AKINCI N., IHMSEN M., AKINCI G., SOLENTHALER B., TESCHNER M.: Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics 31*, 4 (2012), 62:1–62:8. 3, 4, 6

[AO11] ALDUÁN I., OTADUY M. A.: SPH granular flow with friction and cohesion. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), pp. 25–32. 4

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Transactions on Graphics 26*, 3 (2007). 3

[AW09] ADAMS B., WICKE M.: Meshless approximation methods and applications in physics based modeling and animation. In *Eurographics 2009 Tutorials* (2009), pp. 213–239. 4

[BK15] BENDER J., KOSCHIER D.: Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2015), pp. 147–155. 3

[BLS12] BODIN K., LACOURSIERE C., SERVIN M.: Constraint fluids. *IEEE Transactions on Visualization and Computer Graphics 18*, 3 (2012), 516–526. 1

[Bri08] BRIDSON R.: *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press, 2008. 2, 3, 5, 7, 12

[CR99] CUMMINS S. J., RUDMAN M.: An SPH projection method. *Journal of Computational Physics 152*, 2 (1999), 584–607. 1, 2, 3

[GB13] GERSZEWSKI D., BARGTEIL A. W.: Physics-based animation of large-scale splashing liquids. *ACM Transactions on Graphics 32*, 6 (2013), 185:1–185:6. 4, 5

[HLL*12] HE X., LIU N., LI S., WANG H., WANG G.: Local poisson SPH for viscous incompressible fluids. *Computer Graphics Forum 31*, 6 (2012), 1948–1958. 1, 2

[HLW*12] HE X., LIU N., WANG G., ZHANG F., LI S., SHAO S., WANG H.: Staggered meshless solid-fluid coupling. *ACM Transactions on Graphics 31*, 6 (2012), 149:1–149:12. 2, 3, 5, 9

[HWZ*14] HE X., WANG H., ZHANG F., WANG H., WANG G., ZHOU K.: Robust simulation of sparsely sampled thin features in SPH-based free surface flows. *ACM Transactions on Graphics 34*, 1 (2014), 7:1–7:9. 4

[ICS*14] IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 426–435. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

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

[IWT13] IHMSEN M., WAHL A., TESCHNER M.: A Lagrangian framework for simulating granular material with high detail. *Computers & Graphics 37*, 7 (2013), 800–808. 4

[KS14] KANG N., SAGONG D.: Incompressible sph using the divergence-free condition. *Computer Graphics Forum 33*, 7 (2014), 219–228. 1, 3

[KTO96] KOSHIZUKA S., TAMAKO H., OKA Y.: A particle method for incompressible viscous flow with fluid fragmentations. *Computational Fluid Dynamics Journal 4*, 1 (1996), 29–46. 1, 2, 3, 4, 5, 6, 7, 9, 10

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 154–159. 3, 4, 5, 7

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Transactions on Graphics 32*, 4 (2013), 104:1–104:5. 1

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

[Mon92] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics 30* (1992), 543–574. 8

[Mon00] MONAGHAN J.: SPH without a tensile instability. *Journal of Computational Physics 159*, 2 (2000), 290 – 311. 4, 5, 7

[Mon05] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Reports on Progress in Physics* (2005). 2

[NGL10] NARAIN R., GOLAS A., LIN M. C.: Free-flowing granular materials with two-way solid coupling. *ACM Transactions on Graphics 29*, 6 (2010), 173:1–173:10. 4, 5

[NT14] NAIR P., TOMAR G.: An improved free surface modeling for incompressible SPH. *Computers & Fluids 102*, 10 (2014), 304 – 314. 2, 3, 6, 7, 9, 12

[PTB*03] PREMOZE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R. T.: Particle-based simulation of fluids. *Computer Graphics Forum 22*, 3 (2003), 401–410. 1, 2, 3, 4, 5, 6, 7, 9, 10

[SB12] SCHECHTER H., BRIDSON R.: Ghost SPH for animating water. *ACM Transactions on Graphics 31*, 4 (2012), 61:1–61:8. 4, 12

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *ACM Transactions on Graphics 30*, 4 (2011), 81:1–81:8. 3

[SL03] SHAO S., LO E. Y.: Incompressible SPH method for simulating newtonian and non-newtonian flows with a free surface. *Advances in Water Resources 26*, 7 (2003), 787 – 800. 1, 2, 3, 5, 6, 7, 9, 10

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Transactions on Graphics 28*, 3 (2009), 40:1–40:6. 1, 7, 8

[TDF*15] TAKAHASHI T., DOBASHI Y., FUJISHIRO I., NISHITA T., LIN M. C.: Implicit formulation for SPH-based viscous fluids. *Computer Graphics Forum 34*, 2 (2015), 493–502. 11