Infinite Continuous Adaptivity for Incompressible SPH

RENE WINCHENBACH, University of Siegen
HENDRIK HOCHSTETTER, University of Siegen
ANDREAS KOLB, University of Siegen

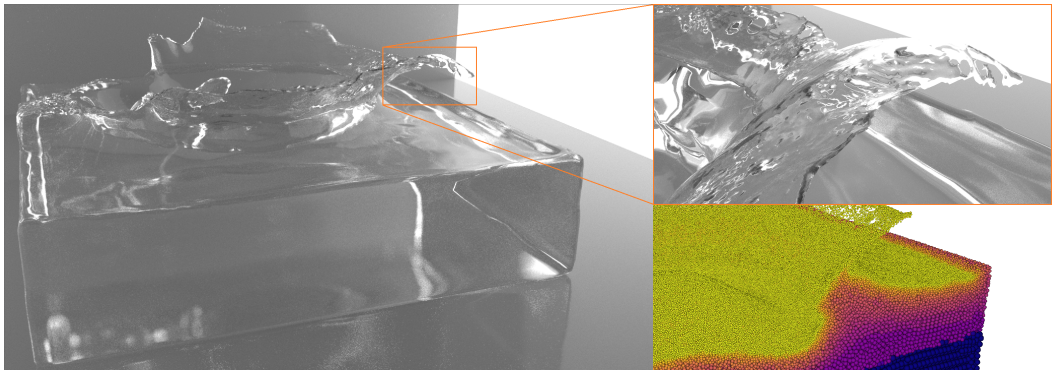

Fig. 1. Adaptive simulation of a bunny shaped drop of water falling into a tank at particle mass ratios of 1:300. The left image shows the overall splashing effect while the top right image shows the fine visible surface detail. The color mapped image shows a cut away view with volume color coded from purple to yellow. Our algorithm allows for a smooth adaptive simulation with detailed surface and low interior resolution.

In this paper we introduce a novel method to adaptive incompressible SPH simulations. Instead of using a scheme with a number of fixed particle sizes or levels, our approach allows continuous particle sizes. This enables us to define optimal particle masses with respect to, e.g., the distance to the fluid's surface. A required change in mass due to the dynamics of the fluid is properly and stably handled by our scheme of mass redistribution. This includes temporally smooth changes in particle masses as well as sudden mass variations in regions of high flow dynamics. Our approach guarantees low spatial variations in particle size, which is a core property in order to achieve large adaptivity ratios for incompressible fluid simulations. Conceptually, our approach allows for infinite continuous adaptivity, practically we achieved adaptivity ratios up to 5 orders of magnitude, while still being mass preserving and numerically stable, yielding unprecedented vivid surface detail at comparably low computational cost and moderate particle counts.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; *Massively parallel and high-performance simulations*;

Additional Key Words and Phrases: SPH Simulation, Adaptivity, Multi-Scale

ACM Reference format:
Rene Winchenbach, Hendrik Hochstetter, and Andreas Kolb. 2017. Infinite Continuous Adaptivity for Incompressible SPH. *ACM Trans. Graph.* 36, 4, Article 102 (July 2017), 10 pages.
https://doi.org/10.1145/3072959.3073713

1 INTRODUCTION

Fluid simulation has been a topic of interest for a long time and has found widespread use in computer animation. While the plausibility and vividness of simulated fluids strongly depend on the dynamics in specific regions, e.g., at the fluid surface including droplets, splashing and formation of fluid sheets, large parts of the fluid bulk are less important for the overall visual appearance of the simulation. Using a high resolution in specific flow regions, like surfaces, and a coarse resolution in other parts, like the bulk, has a huge potential to improve efficiency at no or minimal loss of visual quality.

For grid-based fluid simulation there are various methods to achieve adaptivity, e.g. using octrees [Losasso et al. 2004], non-uniform [Klingner et al. 2006] or tetrahedral meshes [Ando et al. 2013]. Grid-free, particle-based approaches, such as *Smoothed Particle Hydrodynamics (SPH)*, have quite some advantages over grid-based approaches with respect to mass preservation and modelling of free surfaces. However, only a limited amount of adaptivity has been achieved for SPH-based simulations of incompressible fluids so far. All existing methods simulate particles on a pre-defined set of *discrete particle levels*, each fixating a pre-defined particle size [Adams et al. 2007; Horvath and Solenthaler 2013; Orthmann and Kolb 2012; Solenthaler and Gross 2011]. Depending on the refinement requirements at a given spatial location, a specific level, i.e., particle size, is chosen. If the currently used particle size needs to be altered, particles can be replaced instantaneously [Adams et al. 2007] or by applying temporal blending schemes [Orthmann and Kolb 2012]. Alternatively, higher resolution can be achieved by simulating separate scales in regions which require higher resolution which either suffer from mass loss [Horvath and Solenthaler 2013; Solenthaler and Gross 2011] or only allow for very limited adaptivity [Cornelis et al. 2014].

Indirect coupling between particle levels is either limited to small particle mass ratios of 1:8 [Cornelis et al. 2014] or is not mass preserving [Horvath and Solenthaler 2013; Solenthaler and Gross 2011].

Level-based incompressible fluid simulations cannot ensure sufficiently smooth transitions between regions of different particle resolutions. Thus, direct interaction of particles of different levels can cause instabilities which so far restrict adaptivity to particle mass ratios of up to 1:64 [Orthmann and Kolb 2012].

In this paper we introduce the concept of adaptivity using arbitrary particle sizes. The main idea is to have a *continuously adaptive mass* for each particle, which is driven by the distance to important flow regions, e.g., the fluid surface. Our approach enforces a smooth transition between areas of different resolutions by directly exchanging mass between particles. As a consequence, our technique does neither suffer from mass loss like approaches with indirect coupling between different resolutions, nor from numerical instabilities due to the interaction of particles with very different sizes and masses. Thus, we allow for virtually *infinite adaptivity*.

Conceptually and technically our approach comprises the following features and contributions:

- the concept of continuously adjusted particle masses (and sizes) with restricted spatial variation, leading to technically unlimited adaptivity,
- a method to flexibly adjust particle masses not only by split/merge operations but also by redistributing mass among particles which guarantees mass preservation and prevents problems of finding merge partners,
- an efficient implicit temporal blending method which allows for stable incompressible fluid simulation, and
- proper handling of continuously varying particle radii in the SPH simulation framework.

The rest of paper is structured as follows. First we will discuss related work in Sec. 2 and recapitulate the foundations of SPH in Sec. 3. Sec. 4 gives an overview of our approach. Sec. 5 describes how to calculate a particle's distance to the surface and its desired optimal size. Our novel splitting and merging schemes and our smooth and conservative mass redistribution are described in Sec. 6. In Sec. 7 we discuss our novel concept of implicit temporal blending. Sec. 8 shows how to adjust the simulation to stably work with varying particle resolution. Sec. 9 presents results before conclusions are drawn in Sec. 10.

2 RELATED WORK

Since its introduction by Gingold and Monaghan [1977], SPH has been a very active field of research. First, stiff equations of state were employed to achieve weakly compressible fluids [Becker and Teschner 2007; Müller et al. 2003]. Later, prediction-correction-based [Solenthaler and Pajarola 2009], iterative [Macklin and Müller 2013] and implicit methods were used to enforce incompressibility [Bender and Koschier 2015; Ihmsen et al. 2014].

Adaptive simulation using splitting and merging of particles was introduced by Desbrun and Cani [1999]. Adams et al. [2007] adjust particle positions after splitting to reduce the errors mainly introduced in the pressure term. As direct interactions between particles of different levels are a common source of instable simulations, Keiser et al. [2006] use particles that also carry virtual particles of neighboring resolutions which are then used in the interaction. However, none of these approaches could be shown to work with incompressible fluids. A way to work around these instabilities is to use parallel separate simulation scales in which particles of different sizes interact only indirectly through coupling forces [Horvath and Solenthaler 2013; Solenthaler and Gross 2011]. Although these methods work well with incompressible fluids, the coupling between resolution levels is rather unphysical and mass preservation is not guaranteed, as either the overall volume is not considered in the creation of particles [Solenthaler and Gross 2011] or the creation and deletion of high-resolution particles depend on random values [Horvath and Solenthaler 2013]. Orthmann and Kolb [2012] introduced a different approach to incompressible adaptive SPH by tracking the original particles after splitting and merging for a while and temporally blending the values of both resolutions to prevent abrupt changes in quantity fields. Although this allowed for stable simulations, it complicates the evaluation of quantities and the approach has only been shown to work with iterative pressure solvers and would require very expensive adjustments if used with current implicit solvers. Recently, Cornelis et al. [2014] introduced IISPH-FLIP in which the pressure is solved for low resolution SPH particles while fluid advection uses smaller FLIP particles. Although the approach works mass-preservingly, it is restricted to mass ratios of 1:8 between SPH and FLIP particles.

In adaptive approaches with direct particle interactions, instabilities are mainly due to interactions of particles of very different sizes which currently limit adaptivity to mass ratios of 1:64 between the finest and coarsest resolution [Adams et al. 2007; Orthmann and Kolb 2012]. In order to improve the stability, different geometric shapes for splitting have been proposed. While Desbrun and Cani [1999] found static 1:7-splitting to be stable, Adams et al. [2007] and Orthmann and Kolb [2012] used 1:2-patterns that are either dynamically optimized or temporally blended. In general, fix geometric refinement patterns can still cause large errors, thus, refinement patterns have been subject to optimization [Vacondio et al. 2016, 2013].

All prior approaches to adaptive SPH simulations are based on some notion of particle level, i.e., particles belong to one level of constant particle size and transitions between levels are achieved using fixed, optimized and/or blended 1:n refinement (split) and symmetric n:1 coalescing (merge) operations, preventing smooth adjustments of particle sizes. A crucial problem in the merge operation is the identification of *merge partners*, as only particles of the same level can be merged. Practically, particles often can not be merged due to lacking merge partners, which leads to isolated small particles in regions of coarse resolution and, thus, to massive numerical instabilities. Our approach of continuously varying particle masses and sizes solves all the above mentioned problems with adaptive, incompressible SPH-based fluid simulations. We guarantee smooth spatial transitions in particles sizes and, thus, we achieve numerical stability at unlimited adaptivity ratios.

3 SPH FOUNDATIONS

In SPH, quantities are interpolated from a weighted average of the surrounding particles' quantities as [Monaghan 2005]

$$A_i = \sum_j A_j \frac{m_j}{\rho_j} W_{ij}, \tag{1}$$

where $A_i = A(\boldsymbol{x}_i)$ is the interpolated quantity for particle i at position $\boldsymbol{x}_i$ and j are the neighboring particles with quantities A_j. m_j

and ρ_j denote the mass and density. $W_{ij} = W(x_{ij}, h_{ij})$ is the kernel function that weights contributions of the neighboring particles according to their distance $x_{ij} = |\mathbf{x}_i - \mathbf{x}_j|$ and their support radii h_i and h_j where the support radius is one of the most important factors in SPH. The support radius decides which particles interact how strongly with each other and is typically [Monaghan 2005] calculated for each particle i as

$$h_i = \eta \left(\frac{m_i}{\rho_i} \right)^{\frac{1}{3}}. \tag{2}$$

If two particles with different support radii interact, the support of either particle i can be used, i.e. $h_{ij} := h_i$ (gather formulation), the support of particle j , i.e. $h_{ij} := h_j$ (scatter formulation) or an average of both which yields a symmetric formulation $h_{ij} := \frac{h_i + h_j}{2}$ that is usually preferred. Throughout the text, we will assume a symmetric formulation.

The gradient of a quantity can be determined using various formulations [Monaghan 2005]. We use

$$\nabla A_i = \sum_j \left(A_j - A_i \right) \frac{m_j}{\rho_j} \nabla_i W_{ij} \tag{3}$$

as it guarantees a proper first order interpolation, where $\nabla_i W_{ij}$ is the gradient of the kernel function with respect to particle i.

4 OVERVIEW

We consider the surface to be the most important fluid region, thus, we want to reserve the highest simulation resolution to the surface and coarser resolutions to the fluid bulk, i.e., according to the distance to the surface. Therefore, we first calculate the surface distance ϕ_i for each particle i (see Sec. 5.1) and map it to the optimal mass m_i^{opt} the particle should have using a sizing function (see Sec. 5.2). We calculate the optimal mass as a continuous quantity without restrictions to levels and try to adapt particles to be as close to m_i^{opt} as possible. Depending on the ratio of a particle's current and optimal mass, we classify particles into five categories

- o: particle is close to optimal size
- s or l: particle is slightly too small or large, respectively
- S or L: particle is strongly too small or large, respectively

Particles of class L are strongly too large and thus are split into smaller particles. As introducing new particles into the simulation often causes instabilities, we use statically optimized 1:n-patterns to reduce the initial error (see Sec. 6.1). Particles of class S are strongly too small and their mass is distributed among neighbors before they get completely removed from the simulation, yielding an (n+1):n-merging of particles (see Sec. 6.2). The size of particles of classes s and l is increased or decreased by exchanging mass between neighboring particles in order to meet m_i^{opt} (see Sec. 6.3). In summary, the following operations are performed according to the particle class

- L Split (Sec. 6.1)
- S Redistribute mass and remove particle (Sec. 6.2); Receive mass from redistribution of S (Sec. 6.2)
- l Redistribute mass (Sec. 6.3)
- s Receive mass from redistribution of l or S (Secs. 6.2, 6.3)
- o Leave unchanged

To reduce errors introduced by splitting or merging particles, we propose the concept of implicit temporal blending that approximately tracks the motion of the original particles, as if they still existed, and only slowly blends to the new particle set (see Sec. 7). As changing a particle's mass alters its support radius, we derive a stable SPH formulation to handle variable support radii (see Sec. 8) in a symmetric SPH formulation. Algorithm 1 shows how our adaptive method can be incorporated into an existing SPH framework.

5 PARTICLE CLASSIFICATION

As our interest mainly lies on the surface detail, we want to simulate the surface with the finest particle resolution. Therefore, for each particle i we first determine the signed distance to the fluid surface (Sec. 5.1). The surface distance is then mapped to a desired optimal particle size m_i^{opt} and classified according to the ratio of its current mass to the optimal mass (Sec. 5.2). The mapping yields continuous optimal masses determined by the minimum m^{fine} and maximum m^{base} particle masses which are user-defined parameters to control adaptivity.

5.1 Surface Detection

To determine the surface of the fluid, we use the Level-set function proposed by Zhu and Bridson [2005] which gives an initial estimate of the distance to the surface $\tilde{\phi}_i(t)$ for particles i close to the surface, where the values are negative as particles are on the inside of the fluid. We use the propagation method proposed by Horvath and Solenthaler [2013] to iteratively propagate distance values ϕ_i to *all* particles in the simulation and clamp the resulting values to the largest distance possible $\phi_{\max}$. We adopt the approach with minor modifications to allow for adaptive particle sizes.

To avoid spurious detection of surface particles using Level-set functions, we mark particles with more than 45 neighbors as interior particles. Additionally, we do not limit the change of the surface distance for particles in the iterative step of the algorithm as this provides quicker reaction to changes. As we later derive the desired particle mass from the surface distance, particles need to have reasonably smooth distance values. In order to smooth the distance values, we apply an SPH interpolation $\phi_i = \sum_j \frac{m_j}{\rho_j} \phi_j W_{ij}$ in the end.

This algorithm efficiently calculates a stable estimate of the surface distance of all particles as shown in Fig. 3.

5.2 Sizing Functions and Particle Classification

Using the previously determined surface distance we now calculate the desired particle mass m_i^{opt}. At the furthest distance $\phi_{\max}$ we want to use particles of mass m^{base} and at the surface we want to use particles of mass m^{fine}. Using the adaptivity factor $\alpha = \frac{m^{\text{fine}}}{m^{\text{base}}}$ we determine the optimal particle mass as a linear interpolation of the mass between m^{fine} and m^{base}

$$m_i^{\text{opt}}(\phi_i) = m^{\text{base}} \left(\frac{\min(|\phi_i|, |\phi_{\max}|)}{|\phi_{\max}|} (1 - \alpha) + \alpha \right). \tag{4}$$

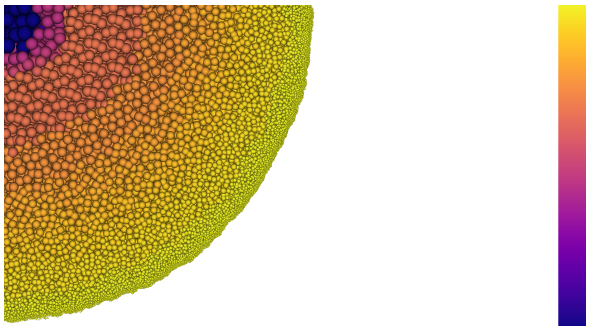

Fig. 2. This image shows a cross section of the sphere in the sphere drop scenario where we used an adaptivity ratio of 1:10000. Mass is color coded from purple m^{base} to yellow m^{fine}.

Particle i is then classified into the categories described in Sec. 4 as

$$C_i = \begin{cases} S & m_i^{\text{rel}} < 0.5 \\ s & 0.5 \leq m_i^{\text{rel}} \leq 0.9 \\ o & 0.9 < m_i^{\text{rel}} < 1.1 \\ l & 1.1 \leq m_i^{\text{rel}} \leq 2 \\ L & 2 < m_i^{\text{rel}}, \end{cases} \tag{5}$$

where $m_i^{\text{rel}} = m_i / m_i^{\text{opt}}$ denotes the relative mass.

o is chosen with a 10% margin around m_i^{opt} to prevent particles that are close to their optimal mass from causing unnecessary computational effort for redistributing comparably small amounts of mass. Using C_i, our adaptive approach tries to adjust particle masses according to the surface distance. In regions, however, where only l or only s particles are present, m_i can deviate from m_i^{opt} as no mass redistribution is applied (see Fig. 3). Note, C_i does not relate to fixed particle sizes as in level-based approaches.

We use the linear scaling of the mass as this creates a very smooth change in resolution over the simulation domain, whereas linearly changing the radius would introduce a cubic change of mass. Although both options yield the same surface detail for unchanged m^{fine}, a cubic change of mass would significantly increase the number of particles. Certain simulations, especially those of very high adaptivity ratios, see Fig. 2, could benefit from fine tuned sizing functions that can be interchanged with our linear scaling to improve surface detail even further. Additionally, we could use different inputs to the sizing function, e.g. the distance to regions of high flow vorticity, as long as the distance values are smooth, which can be achieved by the propagation method of Horvath and Solenthaler [2013].

6 ADAPTIVITY USING SPLITTING AND MERGING

In order to adjust particles of classes L and S, we use splitting and merging operations. Particles have usually been split into a fixed number of particles. We, however, want to achieve a smooth adaptive resolution. Therefore, we use pre-processed 1:n-particle refinement patterns from which we can choose the appropriate pattern to create particles close to their optimal mass (Sec. 6.1). As particle merging in fixed n:1-patterns is often impossible due to missing merge partners [Adams et al. 2007] or causes large errors that have to be corrected [Orthmann and Kolb 2012], we redistribute the mass of *a single* S particle among its n neighbors yielding (n+1):n-merge processes that overcome these difficulties (Sec. 6.2). Additionally, to allow for smooth adjustments, particles of class l can redistribute their excess mass among neighbors classified as s (Sec. 6.3).

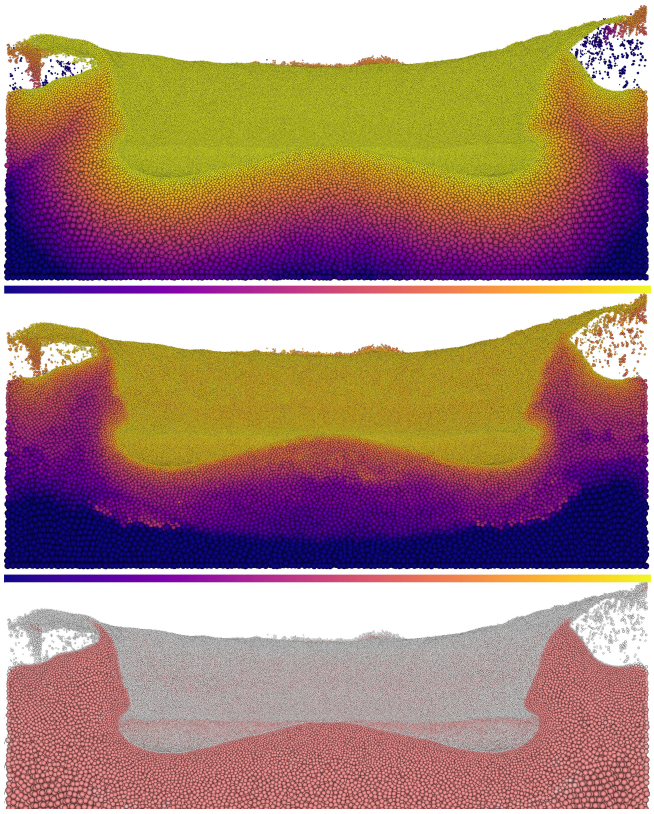

Fig. 3. Adaptive simulation of a drop of water that drips into a tank with particle mass ratios of 1:300. Surface distance (top) is color coded from 0 (yellow) to ϕ_{max} (purple). Support radius (middle) is color coded from yellow to purple. Density (bottom) is color coded from $0.75\rho_0$(black) to $1.25\rho_0$ (red). Note, our method generates a smooth distance field even with adaptive particles and adapts resolutions according to the distance with respect to the classification. We are able to generate a smooth density over the simulation even when particles of different resolutions interact. The apparent banding in the lowest resolution regions is due to a lack of possible sharing partners as the difference to the ideal mass is too small to cause splitting.

6.1 Increasing Resolution Using Particle-Splitting

We split a particle i of class L into n child particles that are close to m_i^{opt} by choosing $n = \lceil m_i / m_i^{\text{opt}} \rceil$. To allow for arbitrary 1:n-splitting we generate patterns using an optimization process. For each n we initially generate a pattern in which uniform particles are placed evenly distanced on the surface of a sphere. For patterns with $n > 4$, one of the particles is placed in the center. The particle positions of these initial patterns are then optimized similarly to Vacondio et al. [2016].

For the actual splitting process we use the optimized patterns to generate the positions of the new particles and copy all other quantities of the original L particle which is the only way to conserve kinetic energy and linear and angular momentum [Vacondio et al. 2016]. The mass of the new particles j is directly set as $m_j = m_i/n$ which conserves mass exactly.

6.2 Reducing Resolution using Particle-Merging

Particles i of class S are smaller than $m_i^{\text{opt}}/2$ and are removed from the simulation by redistributing their mass to nearby particles classified as s or S. Although our approach uses continuous particle masses and principally allows merging of arbitrary particle combinations without being restricted to particle levels, we only merge particle i with neighboring s and S particles as they also carry too little mass. We only remove one particle to get an $(n{+}1){:}n$-pattern that smoothly adjusts masses instead of merging in an n:1-pattern.

In order to find neighboring s and S particles of particle i we iterate over all neighboring particles and check if a neighboring particle j is classified as either s or S and if it doesn't already have a distribution partner. We limit the distance to be within $\frac{h_i}{2}$ to reduce long distance merging that tends to be unstable. Additionally, we check if the combined mass $m_j + \frac{m_i}{n} < m^{\text{base}}$ results in a valid size, where n is the number of partners already found plus one. If the particle is a valid partner, we mark it accordingly. If no partner is found, the initial particle i is left in its current state. Due to continuous masses, arbitrary merging combinations and the smooth transition of resolutions, in practice this is not an issue.

An S particle i with n partners distributes $m_n = \frac{m_i}{n}$ to every partner j. Therefore, we iterate over all partners that have previously been marked as merge partners of i and calculate their new masses and other quantities A, i.e. position, velocity and surface distance, using a mass weighted average as

$$\begin{aligned} m_j^* &= m_j + m_n \\ A_j^* &= \frac{A_i m_n + A_j m_j}{m_j^*}. \end{aligned} \tag{6}$$

Once all partners have been updated, the original particle i is removed from the simulation. By redistributing mass, conservation of mass is guaranteed in our method and by using mass weighted average positions and velocities, we also conserve linear and angular momentum [Vacondio et al. 2013].

6.3 Mass Redistribution

As l particles i have less than $2m_i^{\text{opt}}$ mass, splitting would at least create two particles with masses less than m_i^{opt}. Instead of splitting these particles, we propose to redistribute the excess mass $m_{\text{ex}} = m_i - m_i^{\text{opt}}$ to nearby s particles. This redistribution smoothly adjusts the resolution and prevents l particles from possibly turning into L particles that later would have to be split.

We use a similar mass redistribution process like for merging in Sec. 6.2, however, only the excess mass m_{ex} is redistributed to neighboring s particles j as

$$\begin{aligned} m_i^* &= m_i - m_{\text{ex}} \\ m_j^* &= m_j + \frac{m_{\text{ex}}}{n}. \end{aligned} \tag{7}$$

As the initial l particle i remains and only part of its mass is redistributed, we only update quantities A_j (positions, velocities and surface distances) of the partner particles j that receive mass as

$$A_j^* = \frac{\frac{m_{\text{ex}}}{n} A_i + m_j A_j}{\frac{m_{\text{ex}}}{n} + m_j} \tag{8}$$

in order to conserve momentum. As this process is very similar to the merging in Sec. 6.2, both processes can be combined in a single step that finds partners for both l and S particles.

ALGORITHM 1: Algorithm Overview. New steps required for our method are marked in orange.

RESORT AND NEIGHBORLIST STEP
- Sort Particles
- Build neighborlists for all particles [Winchenbach et al. 2016]

DENSITY COMPUTATION
- Compute $\rho_i = \sum_j m_j W_{ij}$ for all particles
- Blend density for blending particles (see Sec. 7)

PARTICLE INTEGRATION
- Calculate advection forces (Surface Tension, Viscosity, etc.) F_i^{adv}
- Blend velocity for blending particles (see Sec. 7)
- Enforce incompressibility using modified IISPH (see Sec. 8)
- Update position of all particles

ADAPT RESOLUTION
- Detect Surface (see Sec. 5.1)
- Apply sizing function to classify particles (see Sec. 5.2)
- Create particles using splitting (see Sec. 6.1)
- Find partners for merging and redistribution (see Sec. 6.3)
- Merge particles and redistribute mass

7 IMPLICIT TEMPORAL BLENDING

Both adding and removing particles through splitting and merging introduces errors even though positions are updated using weighted averaging and optimized refinement patterns are applied. Previously this error has been reduced by temporally blending in new particles [Orthmann and Kolb 2012] which however involves the simulation of both particle resolutions and severely changes the standard SPH interpolation. We thus propose an *implicit temporal blending* for S and L particles which passively advects the original particle set.

Our algorithm stores the position $\boldsymbol{x}_O$ of the original particle O for the new particles, in case of splitting, or for the particles that received mass, in case of merging. These particles use $\boldsymbol{x}_O$ to compute a density ρ_O for the original particle O that ignores all split particles with the same parent. In the first time step ρ_O yields exactly the density as if the old particle still existed and no particles were added. We calculate the density by blending the new particle density ρ_i with the approximate old density ρ_O as

$$\rho_i^{\text{blended}} = (1 - \beta_i)\rho_i + \beta_i \rho_O, \tag{9}$$

where β_i denotes the temporal blend weight of particle i. This blended density is used for all calculations the new particles are

involved in. Compared to explicit temporal blending no changes to SPH formulations are necessary. We blend the density as all other quantities in an SPH Simulation can be derived from it and improving the quality of the density improves the overall results.

In order to provide a better estimate for the next step, the velocity $\vec{\boldsymbol{v}}_O$ of the original particle is calculated as the mean velocity of all new particles with the same original particle O, or all particles that received mass from the same original particle O, and $\boldsymbol{x}_O$ is updated according to $\vec{\boldsymbol{v}}_O$. We blend the velocities similarly to the density as

$$\vec{\boldsymbol{v}}_i^{\text{blended}} = (1-\beta_i)\vec{\boldsymbol{v}}_i + \beta_i \vec{\boldsymbol{v}}_O. \tag{10}$$

For our blending process we initialize the particle blend weight for particles generated by splitting as $\beta_i = 0.5$ and for particles updated by merging processes as $\beta_i = 0.2$. We chose a smaller initial blend weight for merged particles as they introduce a smaller error. As the tracked position $\boldsymbol{x}_O$ becomes less accurate over time and to fully utilize the new resolution we update the blend weight with a constant rate of $\Delta\beta_i = -0.1$ per timestep until $\beta_i = 0$. While $\beta_i > 0$ the particle does not participate in split, merge or redistribute interactions as it is still in the process of transitioning resolution.

8 ADAPTING TO VARIABLE SUPPORT RADII

For particles with uniform masses, the support radius is only influenced by the density of a particle (cf. Eq. 2) and this influence can typically be ignored [Hernquist and Katz 1989]. Changing support radii due to mass changes in our adaptive simulation, however, cannot simply be ignored when calculating derivatives without causing instabilities. Pressure solvers like IISPH [Ihmsen et al. 2014] and DFSPH [Bender and Koschier 2015] rely on the time rate of change of density to enforce incompressibility and the accuracy of this derivative is central to their performance.

For our SPH framework with varying radii support, the time rate of change of density can be expressed as

$$\frac{d\rho_i}{dt} = \frac{1}{\Omega_i}\sum_j m_j \vec{\boldsymbol{v}}_{ij} \nabla_i W_{ij}, \tag{11}$$

where

$$\Omega_i = \begin{cases} 1 + \frac{h_i}{3\rho_i} m_i \frac{\partial W_{ii}}{\partial h_{ii}} & , C_i^{t-1} = l \\ 1 + \frac{h_i}{3\rho_i} \sum_j m_j \frac{\partial W_{ij}}{\partial h_{ij}} & , \text{else} \end{cases} \tag{12}$$

is a corrective factor that depends on the classification C_i^{t-1} of particle i at the last (discrete) time step $t-1$. See App. A.1 for a comprehensive derivation.

This corrective factor is also applied to the pressure force resulting in a corrected pressure force term as

$$F_i^P = \sum_j \frac{m_j}{\rho_j}\left(\frac{P_i}{\rho_i^2 \Omega_i} + \frac{P_j}{\rho_j^2 \Omega_j}\right)\nabla_i W_{ij}. \tag{13}$$

Both Eq. 11 and 13 are necessary to achieve stable incompressible simulations at resolution interfaces and for varying support radii. For our simulations we modified the IISPH algorithm by carrying the Ω_i term through the derivation resulting in simple adjustments to the overall algorithm. The derivation of the pressure term and the application to IISPH are covered in detail in App. A.2 and A.3 .

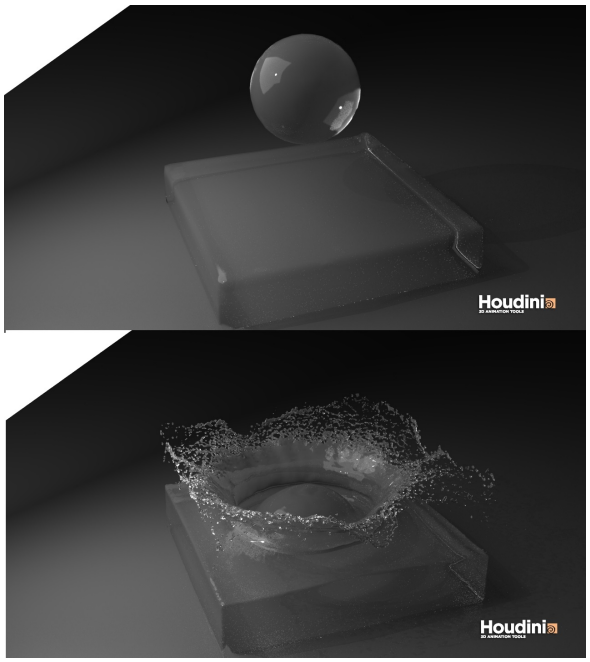

Fig. 4. This image shows the sphere drop scenario rendered using a ray tracer at an adaptivity ratio of 1:250. We used this scenario in various configurations for performance and quality assessment. The top image is the inital configuration and the bottom image the resulting initial splash.

9 RESULTS AND DISCUSSION

To compare our adaptive simulation approach with simulations of fixed resolutions, and to show the capabilities of our method, we set up different scenarios. We used a *sphere drop scenario* where we drip a sphere of liquid into a basin (see Fig. 4) to compare performance and visual quality, a *bunny drop scenario* where instead of a sphere we used a bunny shaped object (see Fig. 1), a *dam break scenario* as a common case, and a *double dam break scenario* (see Fig. 7) to show higher adaptive ratios. We assess the quality of our approach by visually comparing adaptive and non-adaptive simulation results. The performance is assessed by comparing run times of simulations with similar resolution.

All simulations were run on a single Nvidia Titan X GPU with 12 GiB of VRAM, an Intel i7-5930k and 16 GiB of RAM. We use the surface tension of Akinci et al. [2013], the artificial viscosity of Monaghan [2005], boundaries are represented as signed distance fields [Harada et al. 2007]. Fluid renderings were achieved using Houdini and mantra with no modifications required to surface extraction or ray-tracing techniques.

Qualitative comparison. To compare the quality of the results we used the *drop scenario*, see Fig. 4. This scenario provided challenging dynamics on the initial impact of the fluid and difficulties of handling the boundary interactions for higher resolutions. Additionally, the quality of the crowning is a simple way of judging the quality of the result and indicates possible negative impacts. Fig. 8 shows the comparison setup we used. When comparing the high resolution fixed result with the adaptive result, we see little difference in the

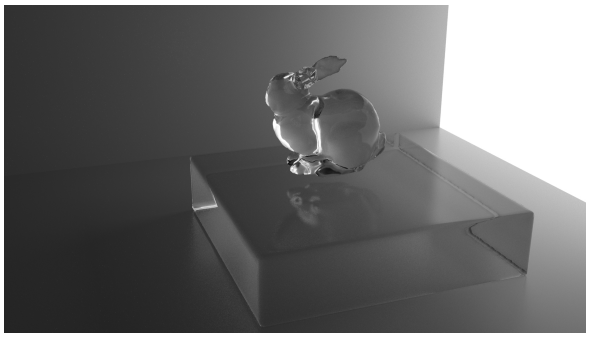

Fig. 5. This image shows the bunny drop scenario rendered using a ray tracer at an adaptivity ratio of 1:250. The image is at the initial start of the simulation before the splash. The splashing behaviour can be seen in Fig. 1.

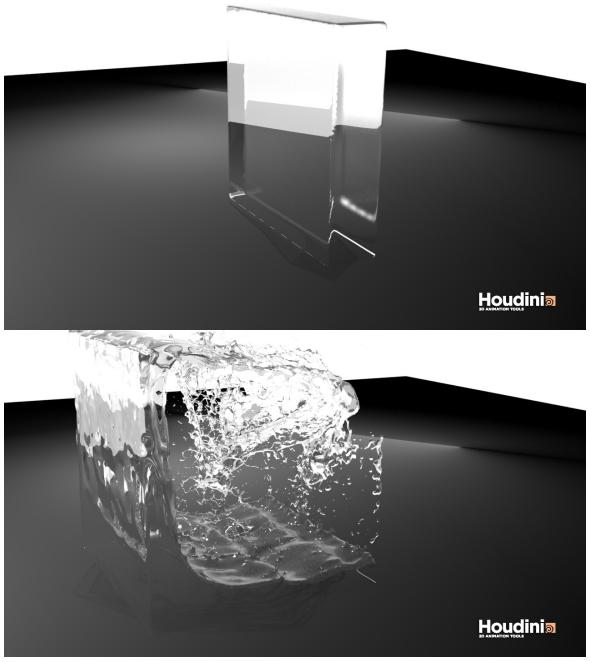

Fig. 6. This image is a traditional dam break scenario rendered using a ray tracer at an adaptivity ratio of 1:250. The top image shows the initial volume used in the scenario and the bottom image shows the splashing behaviour.

crowning, whereas the result is significantly better in comparison to the low resolution result. Overall the behavior was very similar on comparable surface resolutions but differed slightly due to the dependence of certain parameters, e.g. viscosity, on particle resolution. Overall, the fluid behavior was very similar.

Quantitative comparison. Figure 9 shows the performance and particle counts for the sphere drop scene over 30*s* simulated time using different adaptivity ratios and fixed resolutions. Comparing the results for $\alpha = 1/256$ and $m = m^{\text{base}}/8$ we see a comparable number of particles over the course of the simulation. Although our adaptive simulation was slower than the fixed resolution by a factor of 0.7x, we achieved far more detailed surface dynamics due to the

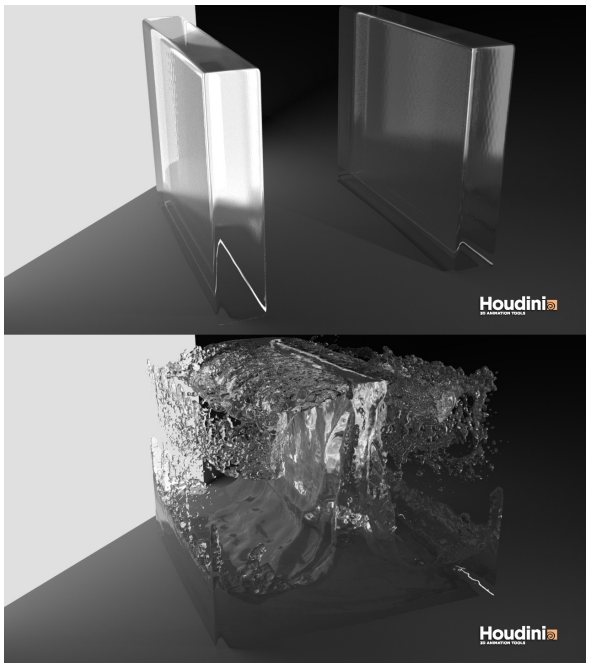

Fig. 7. This image shows the initial state of our double dam break scenario at the top and the behaviour after the collision of the two fluid volumes. An adaptive ratio of $\alpha = 1/500$ and we used a ray tracer to render the images.

higher surface resolution. When comparing the results for $\alpha = 1/32$ and $m = m^{\text{base}}/32$ which have the same surface resolution, we see a reduction in the average number of particles from $7.4M$ to $1.9M$ particles and a total speedup of 7.3x.

Scaling. Considering scaling we saw the largest performance gains for large resolution differences. Due to our sizing function we can roughly estimate the number of particles for an adaptive compact fluid volume as $n_\alpha = n_{\text{base}} \left(1 + \log_2 \frac{1}{\alpha}\right)$, where n_{base} is the number of equivalent uniform particles of mass m^{base}. With an adaptivity ratio of $\alpha = 1/1024$ the average memory and computational requirements increased by 11x compared to $m = m^{\text{base}}$, like our estimate predicted. Simulating $m = m^{\text{base}}/1024$, i.e. using the finest resolution, would have required approximately 90x more memory than our adaptive approach which could not not fit into memory. However the average does not take into account the large amount of surface particles that can be generated as spray on thin fluid details and as such the momentary particle counts could be significantly different.

Stability. For simulations with highly dynamic behaviour like a double dam break (see Fig. 6) adaptivity factors of $\alpha = 1/1024$ could stably be used.

Even for adaptivity factors of $\alpha = 1/100000$ we found stable behaviour considering the algorithm, however, the computational cost increased significantly past roughly $\alpha = 1/2048$ due to the uniform cell grid we currently use to search for neighbors. Additionally resolution dependant parameters, e.g. viscosity, made choosing the

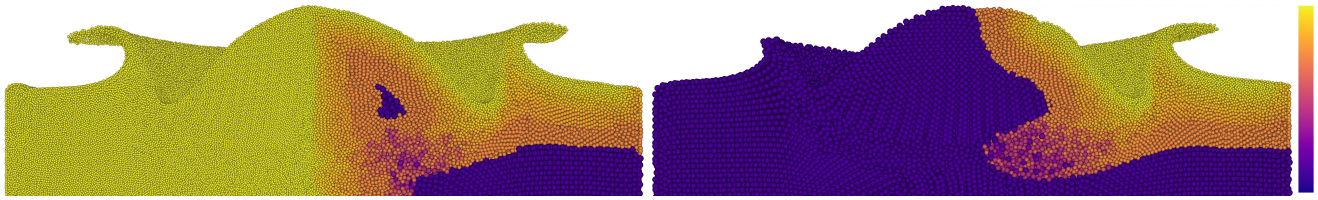

Fig. 8. This image shows our qualitative comparision. For the left image we changed the sizing function to ensure all particles on the left to be of the highest resolution $m_i = m^{\text{base}}/32$, and for the right image we changed the sizing function to ensure all particles on the left to be of the lowest resolution $m_i = m^{\text{base}}$. Volumes are color coded from purple m^{base} to yellow $m^{\text{base}}/32$.

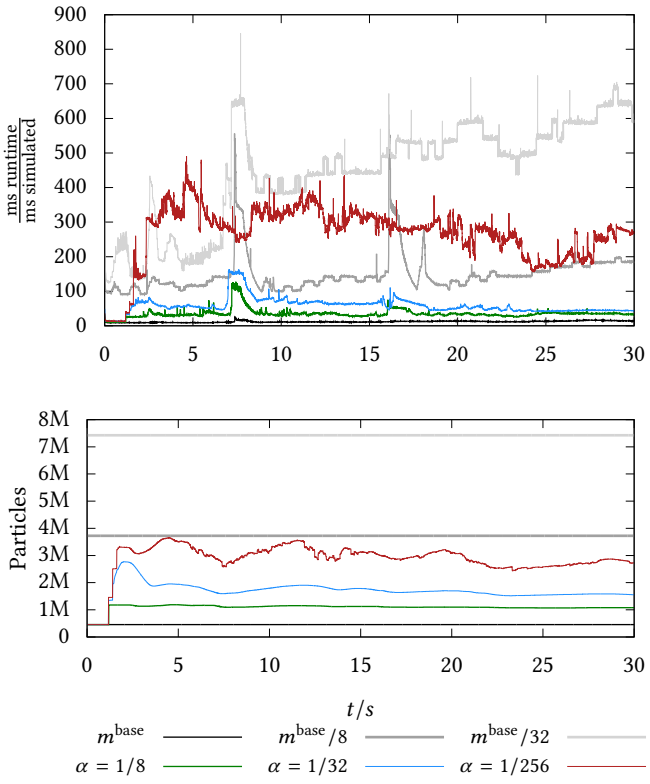

Fig. 9. These graphs show the performance of the sphere drop scenario with and without adaptivity. The top graph shows the computation time in ms required per ms simulation time. The bottom graph shows the number of particles over the course of the scenario for all the different cases. We used 3 fixed resolutions m^{base}, $m^{\text{base}}/8$ and $m^{\text{base}}/32$ and four adaptive simulations where m^{base} was the same as for the non adaptive tests.

right parameters difficult. Disregarding the problems due to the data structure we found no limit to our adaptivity.

Limitations. Although similar, there always were differences between adaptive and equivalent fixed high resolution simulations which mostly depended on viscosity and surface tension parameters. While we could tune parameters to make the fluids behave more similar, different fluid behavior is a common problem if different time steps or particle sizes are used [Peer et al. 2015]. When using highly adaptive simulations, the use of rigid particles [Akinci et al. 2012] becomes difficult if the rigid is only sampled at one resolution. Additionally uniform cell grid structures become restrictive at higher adaptivity ratios.

10 CONCLUSIONS

We have presented a novel method for adaptive incompressible SPH simulations. Our method uses continuous particle masses that are smoothly and mass-conservingly adjusted which allows for infinitely adaptive simulations. We use the surface distance to calculate the optimal mass of each particle that we try to achieve. To adjust particle masses, we allow for splitting of particles, merging and redistribution of mass among neighbors yielding a fine-grained control of particle mass. Our novel merging scheme, in which a particle's mass is redistributed among its neighbors, alleviates previous problems of finding merge partners and the smooth redistribution of mass and the proposed implicit temporal blending are able to reduce errors in quantity fields. With respect to changing support radii a simple update to certain equations significantly improved the stability. We are able to stably simulate highly dynamic fluids with particle mass ratios of 5 orders of magnitude between the finest and coarsest resolution, yielding large speedups and reductions in memory compared to similar uniform simulations.

A CORRECTIVE TERMS

A.1 Time Rate of Change of Density

In order to calculate the time rate of change of density we differentiate the standard SPH estimate for density $\rho_i = \sum_j m_j W_{ij}$ with respect to time as

$$\frac{d\rho_i}{dt} = \sum_j m_j \frac{dW_{ij}}{dt} + \underbrace{\sum_j \frac{dm_j}{dt} W_{ij}}_{\lambda_i}, \tag{14}$$

where λ_i is the term introduced by the time rate of change of mass of a particle which for adaptive methods is generally non-zero as merging, splitting, and mass redistribution change the mass of individual particles although the total mass stays constant due to mass-conservation. Using the total time derivative of the kernel

$$\frac{dW_{ij}}{dt} = \frac{\partial W_{ij}}{\partial x_{ij}} \frac{dx_{ij}}{dt} + \frac{\partial W_{ij}}{\partial h_{ij}} \frac{dh_{ij}}{dt}, \tag{15}$$

we re-factor Eq. 14 into terms containing the time rate of change of mass, terms containing the time rate of change of support radius

and the remaining terms as

$$\frac{d\rho_i}{dt} = \sum_j m_j \frac{\partial W_{ij}}{\partial x_{ij}} \frac{dx_{ij}}{dt} + \lambda_i + \sum_j m_j \frac{\partial W_{ij}}{\partial h_{ij}} \frac{dh_{ij}}{dt}. \tag{16}$$

The kernel derivative with respect to the distance x_{ij} can be calculated as

$$\frac{\partial W_{ij}}{\partial x_{ij}} \frac{dx_{ij}}{dt} = \vec{\boldsymbol{v}}_{ij} \nabla_i W_{ij}, \tag{17}$$

where $\vec{\boldsymbol{v}}_{ij} = \vec{\boldsymbol{v}}_i - \vec{\boldsymbol{v}}_j$ is the velocity difference of the interacting particles and $\nabla_i W_{ij} = \frac{\partial W_{ij}}{\partial \boldsymbol{x}_i}$ denotes the kernel derivative with respect to particle i [Monaghan 2005]. Due to the symmetric SPH formulation, the kernel derivative with respect to the support h_{ij} can be calculated as

$$\frac{\partial W_{ij}}{\partial h_{ij}} \frac{dh_{ij}}{dt} = \frac{1}{2} \left(\frac{dh_i}{dt} + \frac{dh_j}{dt} \right) \frac{\partial W_{ij}}{\partial h_{ij}}. \tag{18}$$

This results in an updated Eq. 16 as

$$\frac{d\rho_i}{dt} = \sum_j m_j \vec{\boldsymbol{v}}_{ij} \nabla_i W_{ij} + \lambda_i + \frac{1}{2} \sum_j m_j \left(\frac{dh_i}{dt} + \frac{dh_j}{dt} \right) \frac{\partial W_{ij}}{\partial h_{ij}}. \tag{19}$$

Here $\sum_j m_j \vec{\boldsymbol{v}}_{ij} \nabla_i W_{ij}$ is the standard formulation of the time rate of change of density for constant support [Monaghan 2005]. The λ_i term is due to the inclusion of time-varying mass and the last term is due to the varying support radii. Considering the third term we calculate the support as a function of density, see Eq. 2, and thus

$$\frac{dh_i}{dt} = \frac{\partial h(\rho_i)}{\rho_i} \frac{d\rho_i}{dt} = -\frac{h_i}{3\rho_i} \frac{d\rho_i}{dt}. \tag{20}$$

Using this equation we could evolve the last term of Eq. 19. However, if we directly use this term, the rate of change of a particle i would depend on the rates of change of neighboring particles. Resolving such a dependency would require an iterative process until a stable result is reached, which is not desirable due to its high computational effort.

In order to resolve this issue, we make the assumption that, due to our mass redistribution and smooth resolution changes, neighboring particles receive similar amounts of mass and change similarly which is the case for particles that receive mass. However for particles that distribute mass (case l) the change of the neighbors is of opposite sign. Let C_i^t denote the classification of particle i at the (discrete) time t, we thus assume

$$\frac{dh_j}{dt} \approx \begin{cases} -\frac{dh_i}{dt} & , C_i^{t-1} = l \\ \frac{dh_i}{dt} & , \text{else.} \end{cases} \tag{21}$$

Considering these two cases, we first cover the result for the else case. Assuming $\frac{dh_i}{dt} \approx \frac{dh_j}{dt}$ yields

$$\frac{1}{2} \sum_j m_j \left(\frac{dh_i}{dt} + \frac{dh_i}{dt} \right) \frac{\partial W_{ij}}{\partial h_{ij}} = \frac{dh_i}{dt} \sum_j m_j \frac{\partial W_{ij}}{\partial h_{ij}} \tag{22}$$

for the third term of Eq. 19 that includes the time rate of change of support radii. Using Eq. 20, Eq. 19 results in

$$\frac{d\rho_i}{dt} = \sum_j m_j \vec{\boldsymbol{v}}_{ij} \nabla_i W_{ij} + \lambda_i + \left(-\frac{h_i}{3\rho_i} \sum_j m_j \frac{\partial W_{ij}}{\partial h_{ij}} \right) \frac{d\rho_i}{dt}. \tag{23}$$

Re-factoring all terms containing $\frac{d\rho_i}{dt}$ to the left hand side yields

$$\frac{d\rho_i}{dt} \underbrace{\left(1 + \frac{h_i}{3\rho_i} \sum_j m_j \frac{\partial W_{ij}}{\partial h_{ij}} \right)}_{\Omega_i} = \sum_j m_j \vec{\boldsymbol{v}}_{ij} \nabla_i W_{ij} + \lambda_i \tag{24}$$

and moving the corrective factor Ω_i over we get the final equation for the else case

$$\frac{d\rho_i}{dt} = \frac{1}{\Omega_i} \left(\sum_j m_j \vec{\boldsymbol{v}}_{ij} \nabla_i W_{ij} + \lambda_i \right). \tag{25}$$

For the case $C_i^{t-1} = l$ we refer back to Eq. 19. By splitting the third term into terms that contain i and terms that don't, we get

$$\frac{1}{2} \sum_{j \neq i} m_j \left(\frac{dh_i}{dt} + \frac{dh_j}{dt} \right) \frac{\partial W_{ij}}{\partial h_{ij}} + \frac{1}{2} m_i \left(\frac{dh_i}{dt} + \frac{dh_i}{dt} \right) \frac{\partial W_{ii}}{\partial h_{ii}}. \tag{26}$$

Due to our assumption that $\frac{dh_j}{dt} \approx -\frac{dh_i}{dt}$ for $j \neq i$, the sum-term becomes zero and using Eq. 20 we get the full equation for $\frac{d\rho_i}{dt}$ as

$$\frac{d\rho_i}{dt} = \sum_j m_j \vec{\boldsymbol{v}}_{ij} \nabla_i W_{ij} + \lambda_i + \left(-\frac{h_i}{3\rho_i} m_i \frac{\partial W_{ii}}{\partial h_{ii}} \right) \frac{d\rho_i}{dt}, \tag{27}$$

which can be re-factored to the same form of Eq. 25 with a different corrective factor $\Omega_i = 1 + \frac{h_i}{3\rho_i} m_i \frac{\partial W_{ii}}{\partial h_{ii}}$.

So far, we carried λ_i through all equations to provide the full derivation. However, we found that including this term did not provide any measurable improvement to the stability or behaviour of our SPH framework, thus, in practice we drop the term.

A.2 Corrected Pressure Forces

In order to derive the pressure forces we start with the Lagrangian for the non-dissipative motion of a fluid in a potential $\Phi(\boldsymbol{x})$ per unit mass in SPH form [Monaghan 2005] as

$$\mathcal{L} = \sum_j m_j \left(\frac{1}{2} \vec{\boldsymbol{v}}_j \cdot \vec{\boldsymbol{v}}_j - u_j - \Phi_j \right), \tag{28}$$

where $u_j = u(\rho_j, s_j)$ is the thermal energy per unit mass and s_j is the entropy. Here, the Euler-Lagrange equation $\frac{d}{dt}(\frac{d\mathcal{L}}{d\vec{\boldsymbol{v}}_i}) - \frac{d\mathcal{L}}{d\boldsymbol{x}_i} = 0$ is used to derive the equations of motion. The first term of the Euler-Lagrange equation results in $\frac{d\mathcal{L}}{d\vec{\boldsymbol{v}}_i} = m_i \vec{\boldsymbol{v}}_i$. We assume that the entropy of each element of fluid remains constant, though each particle can have a different entropy [Monaghan 2005]. From the first law of thermodynamics it follows that $\frac{\partial u_j}{\partial \rho_j} = \frac{P_j}{\rho_j^2}$ which yields

$$\frac{dm_i}{dt} \vec{\boldsymbol{v}}_i + m_i \frac{d\vec{\boldsymbol{v}}_i}{dt} = -\sum_j m_j \left(\frac{P_j}{\rho_j^2} \frac{\partial \rho_j}{\partial \boldsymbol{x}_i} - \frac{\partial \Phi_j}{\partial \boldsymbol{x}_i} \right). \tag{29}$$

Similar to the time rate of change of density, the term containing $\frac{dm_i}{dt}$ also showed no appreciable difference if it was included, thus, we drop it. From the standard SPH estimate for density we can calculate $\frac{\partial \rho_j}{\partial \boldsymbol{x}_i}$ by applying $\frac{\partial}{\partial \boldsymbol{x}_i}$, which similar to Monaghan [2005] results in

$$\frac{\partial \rho_j}{\partial \boldsymbol{x}_i} = \sum_k m_k \left(\nabla_j W_{ji} \left[\delta_{ji} - \delta_{jk} \right] + \frac{\partial W_{jk}}{\partial h_{jk}} \frac{dh_{jk}}{d\boldsymbol{x}_i} \right), \tag{30}$$

where δ_{ij} is the Kronecker delta. The second term can be developed identically to the similar term in the time rate of change of density resulting in

$$\frac{d\rho_j}{d\boldsymbol{x}_i} = \frac{1}{\Omega_j}\sum_k m_k \nabla_j W_{ji}\left[\delta_{ji} - \delta_{jk}\right]. \tag{31}$$

Using these results in Eq. 29 and removing the derivative of the potential similarly to Monaghan [1992] yields a corrected formulation of the pressure force as

$$\vec{F}_i^P = -m_i \sum_j m_j \left(\frac{P_i}{\Omega_i \rho_i^2} + \frac{P_j}{\Omega_j \rho_j^2}\right)\nabla_i W_{ij}. \tag{32}$$

A.3 IISPH Modifications

Adapting the IISPH method requires only a slight modification. In the original paper by Ihmsen et al. [2014], Eq. 9 in Sec. 3.1 calculates

$$\Delta t^2 \frac{\vec{F}_i^P}{m_i} = \underbrace{\left(-\Delta t^2 \sum_j \frac{m_j}{\rho_i^2}\nabla_i W_{ij}\right)}_{d_{ii}} P_i + \sum_j \underbrace{-\Delta t^2 \frac{m_j}{\rho_j^2}\nabla_i W_{ij}}_{d_{ij}} P_j. \tag{33}$$

We modify the equation to

$$\begin{aligned}\Delta t^2 \frac{\vec{F}_i^P}{m_i} &= -\Delta t^2 \sum_j m_j \left(\frac{P_i}{\Omega_i \rho_i^2} + \frac{P_j}{\Omega_j \rho_j^2}\right)\nabla_i W_{ij} = \\ &\underbrace{\left(-\Delta t^2 \sum_j \frac{m_j}{\Omega_i \rho_i^2}\nabla_i W_{ij}\right)}_{d_{ii}} P_i + \sum_j \underbrace{-\Delta t^2 \frac{m_j}{\Omega_j \rho_j^2}\nabla_i W_{ij}}_{d_{ij}} P_j\end{aligned} \tag{34}$$

by including our corrective factors Ω. There are no further changes required as the pressure force is only computed using the terms d_{ii} and d_{ij} that include the corrective factors.

REFERENCES

Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. 2007. Adaptively sampled particle fluids. In *ACM Trans. Graph. (Proc. ACM SIGGRAPH 2007)*, Vol. 26. ACM, 48:1–48:7. DOI: https://doi.org/10.1145/1276377.1276437

Nadir Akinci, Gizem Akinci, and Matthias Teschner. 2013. Versatile Surface Tension and Adhesion for SPH Fluids. *ACM Trans. Graph.* 32, 6 (2013), 182:1—-182:8. DOI: https://doi.org/10.1145/2508363.2508395

Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans. Graph.* 31, 4 (2012), 1–8. DOI: https://doi.org/10.1145/2185520.2335413

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2013. Highly Adaptive Liquid Simulations on Tetrahedral Meshes. *ACM Trans. Graph.* 32, 4, Article 103 (July 2013), 10 pages. DOI: https://doi.org/10.1145/2461912.2461982

Markus Becker and Matthias Teschner. 2007. Weakly compressible SPH for free surface flows. *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symp. Comput. Animat.* (2007), 209–217. http://portal.acm.org/citation.cfm?id=1272690.1272719$\delimiter"026E30F$nhttp://dl.acm.org/citation.cfm?id=1272719

Jan Bender and Dan Koschier. 2015. Divergence-Free Smoothed Particle Hydrodynamics. *Proc. 2015 ACM SIGGRAPH/Eurographics Symp. Comput. Animat.* 1 (2015). DOI: https://doi.org/10.1145/2786784.2786796

Jens Cornelis, Markus Ihmsen, Andreas Peer, and Matthias Teschner. 2014. IISPH-FLIP for incompressible fluids. *Comput. Graph. Forum* 33, 2 (may 2014), 255–262. DOI: https://doi.org/10.1111/cgf.12324

Mathieu Desbrun and Marie-Paule Cani. 1999. *Space-Time Adaptive Simulation of Highly Deformable Substances.* Research Report RR-3829. INRIA. https://hal.inria.fr/inria-00072829

R. A. Gingold and J. J. Monaghan. 1977. Smoothed particle hydrodynamics-theory and application to non-spherical stars. *Monthly Notices of the Roy. Astronomical Soc.* 181 (1977), 375–389.

Takahiro Harada, Seiichi Koshizuka, and Yoichiro Kawaguchi. 2007. Smoothed Particle Hydrodynamics on GPUs. *Computer Graphics International* (2007), 63–70.

Lars Hernquist and Neal Katz. 1989. TREESPH - A unification of SPH with the hierarchical tree method. *Astrophys. J. Suppl. Ser.* 70 (1989), 419. DOI: https://doi.org/10.1086/191344

Christopher Jon Horvath and Barbara Solenthaler. 2013. *Mass Preserving Multi-Scale SPH.* Technical Report. Emeryville, CA.

Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. 2014. Implicit incompressible SPH. *IEEE Trans. Vis. Comput. Graph.* 20, 3 (2014), 426–435. DOI: https://doi.org/10.1109/TVCG.2013.105

Richard Keiser, Bart Adams, Philip Dutré, Leonidas Guibas, and Mark Pauly. 2006. *Multiresolution particle-based fluids.* Technical Report 520. Department of Computer Science, ETH Zurich. 10 pages.

Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. 2006. Fluid Animation with Dynamic Meshes. In *Proceedings of ACM SIGGRAPH 2006.* 820–825. http://graphics.cs.berkeley.edu/papers/Klingner-FAD-2006-08/

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating Water and Smoke with an Octree Data Structure. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 457–462. DOI: https://doi.org/10.1145/1015706.1015745

Miles Macklin and Matthias Müller. 2013. Position based fluids. *ACM Trans. Graph.* 32, 4 (2013), 1. DOI: https://doi.org/10.1145/2461912.2461984

J. J. Monaghan. 1992. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics* 30, 1 (1992), 543–574. DOI: https://doi.org/10.1146/annurev.astro.30.1.543 arXiv:arXiv:1007.1245v2

J J Monaghan. 2005. Smoothed particle hydrodynamics. *Reports Prog. Phys.* 68, 8 (2005), 1–34. arXiv:astro-ph/0507472v1

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-Based Fluid Simulation for Interactive Applications. *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animat.* 5 (2003), 154–159.

Jens Orthmann and Andreas Kolb. 2012. Temporal blending for adaptive SPH. *Computer Graphics Forum* 31, 8 (2012), 2436–2449.

Andreas Peer, Markus Ihmsen, Jens Cornelis, and Matthias Teschner. 2015. An implicit viscosity formulation for SPH fluids. *ACM Trans. Graph.* 34, 4 (jul 2015), 114:1–114:10. DOI: https://doi.org/10.1145/2766925

Barbara Solenthaler and Markus Gross. 2011. Two-scale particle simulation. *ACM Trans. Graph.* 30, 4 (2011), 1. DOI: https://doi.org/10.1145/2010324.1964976

B. Solenthaler and R. Pajarola. 2009. Predictive-corrective incompressible SPH. *ACM Trans. Graph.* 28, 3 (2009), 1. DOI: https://doi.org/10.1145/1531326.1531346

R. Vacondio, B.D. Rogers, P.K. Stansby, and P. Mignosa. 2016. Variable resolution for SPH in three dimensions: Towards optimal splitting and coalescing for dynamic adaptivity. *Comput. Meth. Appl. Mechanics and Eng.* 300 (2016), 442 – 460. DOI: https://doi.org/10.1016/j.cma.2015.11.021

R. Vacondio, B.D. Rogers, P.K. Stansby, P. Mignosa, and J. Feldman. 2013. Variable resolution for SPH: A dynamic particle coalescing and splitting scheme. *Comput. Meth. Appl. Mechanics and Eng.* 256 (2013), 132 – 148. DOI: https://doi.org/10.1016/j.cma.2012.12.014

Rene Winchenbach, Hendrik Hochstetter, and Andreas Kolb. 2016. Constrained Neighbor Lists for SPH-based Fluid Simulations. In *Proc. Eurographics/ACM SIGGRAPH Symp. Comput. Animat.* Eurographics Association.

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Transactions on Graphics* 24, 3 (2005), 965. DOI: https://doi.org/10.1145/1073204.1073298