Optimized Refinement for Spatially Adaptive SPH

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

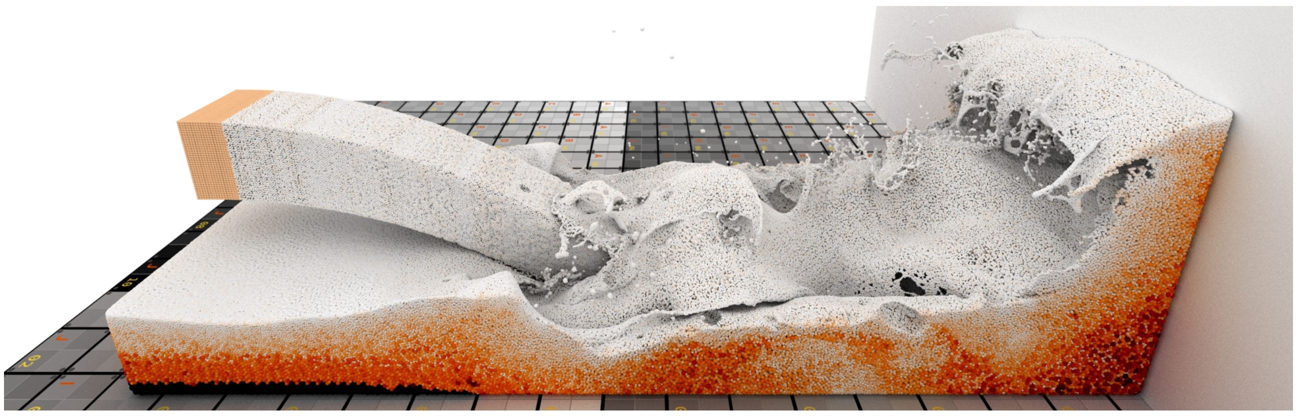

Fig. 1. An inlet, emitted with a moderate resolution, collides with a fluid volume that is agitated by a moving boundary wall. Our adaptive method can easily handle the $1:500$ adaptive volume ratio shown here, without instabilities, and can readily adapt the resolution of the inlet on the fly without causing the inlet to be decollimated. Volume color mapped from high (black) to low (white).

In this paper we propose an improved refinement process for the simulation of incompressible low-viscosity turbulent flows using Smoothed Particle Hydrodynamics, under adaptive volume ratios of up to $1:1,000,000$. We derive a discretized objective function, which allows us to generate ideal refinement patterns for any kernel function and any number of particles a priori without requiring intuitive initial user-input. We also demonstrate how this objective function can be optimized online to further improve the refinement process during simulations by utilizing a gradient descent and a modified evolutionary optimization. Our investigation reveals an inherent residual refinement error term, which we smooth out using improved and novel methods. Our improved adaptive method is able to simulate adaptive volume ratios of $1:1,000,000$ and higher, even under highly turbulent flows, only being limited by memory consumption. In general, we achieve more than an order of magnitude greater adaptive volume ratios than prior work.

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

Additional Key Words and Phrases: SPH, spatial adaptivity, optimization

ACM Reference Format:
Rene Winchenbach and Andreas Kolb. 2020. Optimized Refinement for Spatially Adaptive SPH. *ACM Trans. Graph.* 1, 1, Article 1 (January 2020), 15 pages. https://doi.org/10.1145/3363555

1 INTRODUCTION

Fluid simulations play an important role in modern computer animations, and there is an ever-increasing demand not just for more vivid and higher quality free surface fluid simulations, but also larger overall simulation domains. However, a uniform resolution in all parts of the simulation is not ideal as this requires high resolutions in regions of the fluid where the behavior is not interesting, e.g.,at the bottom of a pool. Therefore, methods are needed that can focus the computational resources to where it is most beneficial. This can be achieved using spatially adaptive methods, which have been used for grid-based simulations for a while, e.g., using octrees [Losasso et al. 2004], tetrahedral meshes [Klingner et al. 2006], or tetrahedral meshes combined with an adaptive FLIP simulation [Ando et al. 2013]. For grid-free methods, based commonly on Smoothed Particle Hydrodynamics (SPH), prior research mainly focused on weakly compressible simulations [Adams et al. 2007; Feldman and Bonet 2007] and only recent work [Winchenbach et al. 2017] has enabled incompressible flows with larger refinement ratios.

An adaptive SPH method starts by defining a desired resolution, often based on surface distance, for each particle using a sizing function. The resolution is then locally decreased by merging particles, smoothed using sharing between particles, or increased by refining a particle into multiple smaller particles. These processes can introduce errors into the simulation and in order to stabilize the simulation blending methods have been proposed [Orthmann and Kolb 2012; Winchenbach et al. 2017]. Most of the introduced refinement error is due to a reliance on some intuition [Feldman and Bonet 2007; Vacondio et al. 2013; Winchenbach et al. 2017] instead of a fundamental analytical model for particle refinement patterns.

In order to improve the refinement process, we first introduce a continuous objective function describing these processes, in Sec. 5, which can be applied to symmetric SPH formulations required in adaptive incompressible flows. Sec. 6 then demonstrates our novel discretization technique based solely on particles and derives the

Authors' addresses: Rene Winchenbach, rene.winchenbach@uni-siegen.de, University of Siegen, Hoelderlinstrasse 3 , Siegen, NRW, 57076; Andreas Kolb, andreas.kolb@uni-siegen.de, University of Siegen, Hoelderlinstrasse 3 , Siegen, NRW, 57076.

error terms, as well as their derivatives, required for efficient optimization. In Sec. 7 we then utilize our discretization technique to optimize refinement patterns regarding positions and mass distributions, both a priori for ideal environments and online using actual particle neighborhoods, using these discrete error terms. Our results show an unavoidable inherent residual refinement error, which we smooth out using improved and novel techniques to ensure stability, in Sec. 8. Finally we demonstrate the efficacy of our improved method and its properties in Sec. 10.1 by comparing it to prior work, and identifying possible factors that could further improve stability.

2 RELATED WORK

In the past, spatially adaptive SPH methods have been widely used within the CFD context, e.g.,by Vacondio et al. [2012], Feldman and Bonet [2007] and Li et al. [2015], and to some extent within computer animation, e.g.,by Solenthaler and Gross [2011], Orthmann and Kolb [2012], Horvath and Solenthaler [2013], and more recently by Winchenbach et al. [2017]. These adaptive methods use different approaches to resolve underlying stability issues, e.g.,using multiple separate simulations, temporal blending methods or, as done in most of these methods, by using a weakly compressible SPH formulation. Older methods, e.g.,Adams et al. [2007], developed before modern incompressible pressure solvers existed (starting with PCISPH [Solenthaler and Pajarola 2009]), did not have to consider as strict stability requirements. However, all these adaptive methods need particle patterns for replacing a particle of lower resolution by particles of higher resolution. This replacement, however, introduces a density error that needs to be addressed to avoid instabilities.

Solenthaler and Gross [2011] address the stability issue by utilizing separate simulations, each using a different global uniform level of resolution, where particles are directly inserted and those causing large errors are simply removed. Orthmann and Kolb [2012] apply a simple 1:2 splitting pattern and a temporal blending scheme, which significantly limits the temporal rate of resolution change possible. Vacondio et al. [2016] propose statically optimized patterns to avoid the density errors but require asymmetric support radii, which are not stable for incompressible SPH methods. Winchenbach et al. [2017] combine manually optimized refinement patterns with temporal blending to allow for adaptive incompressible SPH simulations. However, this approach does not fully solve the instability problem and strongly depends on manual parameter tuning.

Various approaches have recently been developed for SPH-based simulations in computer animation, e.g.,implicit incompressible SPH (IISPH) [Ihmsen et al. 2013], divergence-free SPH (DFSPH) [Bender and Koschier 2015] and position based fluids [Macklin and Müller 2013]. These approaches predict and correct fluid compression through predictions in each time step, but are not designed to handle sudden fluid compressions, such as those caused by particle splitting. Therefore, errors due to particle refinement need to be prevented from occurring in the first place by changing the adaptive method itself, instead of relying on an external pressure solver.

Furthermore, rigid boundary handling is an existing issue for adaptive methods [Winchenbach et al. 2017], as commonly used particle boundaries [Akinci et al. 2012] suffer from sampling problems. Fujisawa and Miura [2015] address the sampling problem by using an analytical integral formulation of boundaries. Koschier and Bender [2017] extended this approach using numerical integrals precomputed on grids. Recently, Winchenbach et al. [2020] introduced a signed distance field based boundary integral method that does not require expensive precomputation steps and ensures consistent behavior across varying particle resolutions. Band et al. [2017] utilize a moving least squares method to fit planes to particles for sufficiently flat boundaries. The method is then used to calculate accurate pressure values for boundary particles instead [Band et al. 2018]. Finally, Gissler et al. [2019] introduced an extended SPH model to simulate rigid to rigid interactions using SPH. Boundary integral based methods are a good general choice, as they avoid the sampling issue, but also introduce non-SPH representations.

Finally, temporally adaptive methods are alternative approaches to allocate computational resources where they are most beneficial. Adjusting the time step of an SPH simulation, globally, using the CFL condition has found wide adoption in SPH, with initial work by Desbrun and Gascuel [1996] and later by Ihmsen et al. [2010]. Assigning different particles, or regions, different time steps has also been proposed by Desbrun and Gascuel [1996], however, this approach has not find wide adoption, due to the complexity involved in synchronizing different time steps. Reinhardt et al. [2017] applied this concept, through regional-time-stepping, to modern pressure solvers, but is only applicable to CPU-based SPH variants.

3 FOUNDATIONS OF SPH

SPH is a numerical method to solve continuous integrals by approximating an underlying continuous system using discrete particles, see [Monaghan 1992] and for further explanations [Koschier et al. 2019]. These particles are then used to approximate continuous quantities using the basic SPH interpolation operator for a particle i utilizing all neighboring particles j, which is given by

$$A(\mathbf{x_i}) = \sum_j \frac{m_j}{\rho_j} A(\mathbf{x_j}) W_{ij}, \tag{1}$$

where m, ρ and $\mathbf{x}_i$ denote the mass, density and position respectively, and the subscript denotes indices of the particle. $W_{ij} = \hat{W}(|\mathbf{x}_{ij}|, h_{ij})$ is a radially symmetric kernel function that depends on the distance between positions $|\mathbf{x}_{ij}| = |\mathbf{x}_i - \mathbf{x}_j|$ and the support radius of the interaction h_{ij}. This term can be chosen as asymmetric, e.g.,as $h_{ij} = h_i$, which results in a gather formulation, or as $h_{ij} = h_j$, which results in a scatter formulation, or as the average support radius, $h_{ij} = \frac{h_i + h_j}{2}$, which results in a symmetric formulation. For adaptive incompressible SPH only a symmetric formulation is stable [Winchenbach et al. 2017], whereas for adaptive weakly compressible SPH (WCSPH) a common choice is the scatter formulation as this significantly simplifies many derivative terms [Vacondio et al. 2013].

Commonly used kernel functions include the cubic spline kernel [Ihmsen et al. 2013; Koschier and Bender 2017; Monaghan 2005] and the Wendland kernel functions [Vacondio et al. 2016]. The exact choice of kernel function does not influence our method, but still changes the support radius of a particle. Every kernel function has an ideal number of neighbors [Dehnen and Aly 2012], i.e., $N_H = 50$ for the cubic spline kernel, which yields the support radius for a particle by solving $\frac{4}{3}\pi h^3 = N_H V_i$, as there should be N_H particles of volume V_i contained within a spherical support radius H. The support radius H is related to the smoothing scale h through a factor $\frac{H}{h}$. Within Computer Animation, $\frac{H}{h}$ is often defined as 1 [Koschier et al. 2019] and, thus h is also often referred to as support radius. We will follow this notion in our paper.

4 METHOD OVERVIEW

Spatial adaptive methods generally begin by determining the desired resolution for a particle using a sizing-function, e.g.,using the particle's surface distance, or based on visibility, and classifying particles accordingly into different categories [Winchenbach et al. 2017] and then adjusts the resolution of each particle to be closer to its desired resolution. This can, in general, be done using three distinct processes:

Merging: This process combines multiple high-resolution particles into lower resolution particle(s) to reduce the local spatial resolution. This can be done in multiple ways, e.g., merging 2 particles into 1 ($2:1$-merging), distributing one particle onto several other particles ($n:n-1$-merging) or combining many particles into one ($n:1$-merging). This process is fully constrained for $n:1$ merging due to mass and momentum conservation.

Sharing: This process changes a particle that is larger than its ideal resolution by distributing parts of its mass and quantities to nearby particles which are smaller than their ideal resolution. This process is essentially an extension of merging, where the original particle remains in the simulation. The process for interactions between two particles is also fully constrained, due to mass and momentum conservation.

Splitting: This process splits (or refines) a particle that is significantly larger than its ideal resolution into multiple smaller particles. Quantities of the inserted particles are not fully constrained, i.e.,the mass of each created particle can vary as long as the overall mass is preserved. Additionally, geometric patterns have been used in all prior methods to insert new particles into the simulation, but these often require significant manual tuning to achieve a certain level of stability. In general, different refinement methods rely on different splitting steps, e.g., $1:2$ or $1:8$. However some methods allow for arbitrary changes (up to a certain limit) of $1:n$. Some splitting methods include procedures to reduce the impact of errors introduced by the refinement process.

The merging and sharing processes are mostly limited by the search for eligible particle groups, often resulting in $2:1$ merging and $1:1$ sharing being used. These processes are fully-constrained by mass and momentum conservation, and also do not cause visually apparent instabilities. The splitting process, however, causes instabilities, i.e., due to changes to the density field, and can be optimized [Feldman and Bonet 2007]. Refining a single low resolution particle o, with mass m_o at position $\mathbf{x_o}$ and velocity $\mathbf{v_o}$, into n particles, i.e.,a $1:n$ split, has to preserve mass, kinetic energy, as well as linear and angular momentum, and should not modify the underlying density field to avoid compression. Mass-conservation can be enforced by ensuring that the total mass of the refined particles is equal to m_o, whereas momentum and kinetic energy are conserved if and only if the velocities of the refined particle are equal to $\mathbf{v_o}$ [Feldman 2006]. Consequently, the error introduced on the underlying density field can be controlled by optimizing the mass distribution and positions for the refined particles. This yields $4n$ degrees of freedom, making manual parameter tuning impractical.

The approach, to optimize the refinement, we present is independent of the number of high-resolution particles being created and of the kernel function used, and is applicable to any adaptive method that relies on particle refinement and any incompressible or weakly compressible solver. However, we first require an underlying objective function for a symmetric SPH formulation, as prior optimization approaches relied on an asymmetric scatter-based SPH formulation. Our overall refinement process is described in Algorithm 1.

Algorithm 1: An overview of the off-line a priori and the online optimization process to generate the initial refinement pattern and its local adaptation during simulation, respectively.

```
// A priori optimization of n patterns
For all refinement patterns
    Initialize pattern with random positions and uniform weights
    Optimize positions; see Sec. 6.1 and 7.1
    Optimize weight distribution; see Sec. 6.2 and 7.1
    Optimize positions and weights simultaneously 7.1
    Store pattern for refinement

// Online optimization of particle i
Determine ideal particle radius $s_i$; see Sec. 9
If particle radius $r_i \leq 2s_i$: return // no refinement
Determine pattern to be used $p = \lfloor clamp(r_i/s_i, 2, n) \rfloor$
Initialize refined particles using a priori pattern for $p$ particles
Optimize positions using gradient descent; see Sec. 6.1 and 7.2
Optimize masses using evolutionary optimization; see Sec. 7.2
Initialize blend process for refined particles; see Sec. 8.1
Insert refined particles and remove old particle
```

5 CONTINUOUS OBJECTIVE FUNCTIONS

In general, the splitting process works by replacing the original low resolution particle o with a set of higher resolution particles $\mathcal{S}$. The problem now is based on the choice of parameters for the new particles, where for n particles we get $d \cdot n$ positional parameters (for d dimensions) and n weights determining the mass distribution. The distributed mass needs to equal the original mass, e.g.,$m_o = \sum_{s\in\mathcal{S}} m_s$, due to mass conservation. However, there is no way to directly determine the patterns and prior work often employed fixed refinement patterns, e.g., 1:2 [Orthmann and Kolb 2012], 1:7 [Solenthaler and Gross 2011], or 1:13 [Vacondio et al. 2016] or for multiple n [Winchenbach et al. 2017], which are based on using intuitive shapes and often involve manual tuning.

Mass and momentum are directly conserved. The underlying fields that should be kept constant are the density and velocity fields [Feldman and Bonet 2007], with a focus on the density field, as a change to density would introduce significant instabilities. The change to the density field at any point $\mathbf{x}$ can be defined, based on the density before $\rho(\mathbf{x})$ and after $\rho^*(\mathbf{x})$ refinement [Feldman 2006]

$$\tau(\mathbf{x}) = \rho^*(\mathbf{x}) - \rho(\mathbf{x}), \tag{2}$$

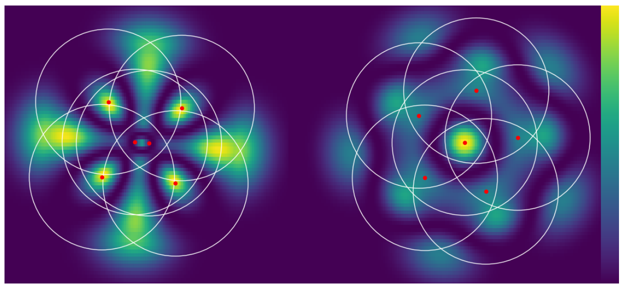
Fig. 2. The result of optimizing the refined particle positions (red) for a symmetric SPH formulation (left) and a scatter SPH formulation (right), with the density field error $\tau(\mathbf{x})$ color coded from purple to yellow.The error is visualized from 0 to $6 \cdot 10^{-4}$ for symmetric SPH on and the left from 0 to $1 \cdot 10^{-4}$ for asymmetric SPH on the right. Note that the errors in the converse terms, e.g., the right pattern evaluated utilizing the symmetric error metric, result in orders of magnitude worse behavior, highlighting the need to chose the appropriate formulation for the optimization process.

which can be evaluated over a continuous domain Ω as

$$\mathcal{E} = \int_\Omega \tau(\mathbf{x})^2 d\mathbf{x}. \tag{3}$$

This can be seen as a continuous objective function, where the ideal refinement process would cause a total error of 0. This can also be seen as enforcing density invariant refinement, i.e.,$\frac{D\rho}{Dt} = 0$ [Bender and Koschier 2015], but instead of a change over time the change occurs due to particle refinement. Feldman and Bonet [2007] use an a priori optimization process to minimize $\mathcal{E}$, using an initial refinement pattern found by intuition, i.e.,an icosahedra in 3D configuration. Fixing the relative positions of particles reduces the problem to determining a single scaling parameter and the mass distribution, which Feldmann and Bonet solve for a scatter-based formulation of SPH. However, the approach of Feldman and Bonet [2007] is not directly applicable to incompressible SPH simulations, as these require a symmetric SPH formulation to avoid instabilities [Winchenbach et al. 2017].

For a symmetric SPH formulation, the support radius of an interaction depends on the definition of a support radius $h(\mathbf{x})$ for every integration point, which could be determined using the approaches from Monaghan [2002] and Winchenbach et al. [2017] that base the support-radius on the density at this position, which also means that particles of equal resolution have varying support-radii, i.e., $V_i = V_j \;\not\Rightarrow\; h_i = h_j$. These approaches, however, introduce a coupled problem (refer to [Price 2012] for a more thorough examination), where the evaluation of the density depends on the support radius which, again depends on the density. This problem can be stated as

$$\rho_i = \sum_j m_j W\left(|\mathbf{x}_i - \mathbf{x}_j|, \frac{h_i + h_j}{2}\right), \quad h_i = \eta \sqrt[3]{\frac{m_i}{\rho_i}}, \tag{4}$$

where η is a parameter used to determine the number of neighbors for a particle, commonly chosen as $\eta \approx 2.6$ for cubic spline kernels [Winchenbach and Kolb 2019]. This term could be evaluated iteratively until the result converges, but this would require a very expensive computation for every point of integration. Nonetheless, the symmetric formulation can still be used to optimize the refined positions $\mathbf{x}_s$ for all refined particles $s \in \mathcal{S}$, even though the process is significantly slower than for a scatter formulation. The results of this optimization are shown in Fig. 2, where the generated distribution of particles is significantly different between the symmetric and scatter formulation, i.e., two particles are placed close to the position of the original particle instead of one. The patterns $\mathcal{P}_s$ and $\mathcal{P}_a$ are generated using a symmetric ($\mathcal{E}_s$) and asymmetric ($\mathcal{E}_a$) formulation of Eqn. 3, respectively. Here, $\mathcal{E}_s(\mathcal{P}_s) = 2.4 \cdot 10^{-4}$, $\mathcal{E}_a(\mathcal{P}_s) = 5.8 \cdot 10^{-4}$, $\mathcal{E}_s(\mathcal{P}_a) = 3.0 \cdot 10^{-3}$ and $\mathcal{E}_a(\mathcal{P}_a) = 3.8 \cdot 10^{-5}$. However, this does not mean that one of the generated patterns is superior to the other as they are optimized for fundamentally different SPH formulations, i.e., they are not alternative options. As such, the appropriate pattern should be chosen for each formulation and applying patterns optimized for asymmetric SPH formulations leads to a significantly worse error (by an order of magnitude).

6 DISCRETIZED OBJECTIVE FUNCTION

Minimizing Eqn. 3 by iteratively solving Eqn. 4, is not practical, especially in 3D, due to computational costs. Instead, we propose to only evaluate Eqn. 2 at the current positions of particles, which significantly reduces the computational complexity as this only requires $N_H + n$ evaluations, for n refined particles. We denote the original particle as o, the set of refined particles of o as $\mathcal{S}_o$, the set of neighbors of o, including o, as $\mathcal{N}_o$ and the discretized error term $\mathbb{E}$ for the surrounding particle positions and newly inserted particle positions as $\mathbb{E}_\mathcal{N}$ and $\mathbb{E}_\mathcal{S}$, respectively. For clarity we will drop the subscript on $\mathcal{S}$ and $\mathcal{N}$ whenever possible..

Following Vacondio et al. [2013], we can evaluate the error on neighboring particle positions, by effectively removing the old particle and inserting the refined particles, as

$$\tau_n = \sum_{s \in \mathcal{S}} m_s W_{ns} - m_o W_{no}, \forall n \in \mathcal{N}. \tag{5}$$

For a refined particle $s \in \mathcal{S}$, we can calculate the error term as the difference between the original particles density ρ_o and the density evaluated at the current position of s, which resembles the numerical SPH approximation error. This results in the following error term τ_s for a specific refined particle s:

$$\tau_s = \left(\sum_{n \in \mathcal{N}} m_n W_{sn} + \sum_{k \in \mathcal{S}} m_k W_{sk}\right) - \rho_o, \forall s \in \mathcal{S}. \tag{6}$$

The discretized error terms are then the mass weighted sum of square error terms, per particle, which yields

$$\mathbb{E}_\mathcal{N} = \sum_{n \in \mathcal{N}} m_n \tau_n^2, \quad \mathbb{E}_\mathcal{S} = \sum_{s \in \mathcal{S}} m_s \tau_s^2. \tag{7}$$

The overall minimization problem can then be defined as minimizing the positions and masses of the inserted particles, under the constraint $\sum_{s \in \mathcal{S}} m_s = m_o$ due to mass-conservation, as

$$\min_{\mathbf{x}_\mathcal{S}, m_\mathcal{S}} \mathbb{E} = \min_{\mathbf{x}_\mathcal{S}, m_\mathcal{S}} \left(\mathbb{E}_\mathcal{N} + \mathbb{E}_\mathcal{S}\right). \tag{8}$$

The partial derivatives of this minimization problem with respect to positions $\mathbf{x}_s$ and masses m_s are described in Sections 6.1 and 6.2, respectively.

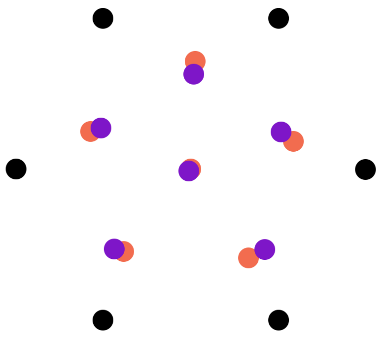
Fig. 3. The result of optimizing the positions of 6 refined particles using our discretized problem (purple) and the continuous form (red), which results in a small difference, demonstrating a good approximation through our proposed discretization.

Note that this problem has a trivial, but useless, solution where all refined particles are placed at the original particles position and a single refined particle having the mass of the original particle, i.e.,all other refined particles have zero weight. We avoid this trivial solution by iteratively optimizing positions and masses and by imposing a limit on the optimized masses. Thus, we add the constraint

$$\nexists m_s \geq 8m_i, \forall i, s \in \mathcal{S}, \tag{9}$$

which restricts the largest ratio of masses between refined particles to be at most 8. These restrictions additionally prevent degenerate optimization solutions where single particles have zero mass.

This problem can be solved in ideal, isotropic and hexagonal, particle distributions using initially random positions and mass ratios for the refined particles a priori. Due to the relatively low computational cost, it can also be solved online for an actual particle in a fluid simulation using the results of the a priori optimization as a starting point. Fig. 3 shows the result of optimizing the particle distribution for 6 particles using our proposed discretization (purple) and by using the continuous form (red), where the results are very similar (ignoring rotation and translation), demonstrating a close approximation of the underlying problem. The minimization problem is derived in detail in the supplementary material.

6.1 Spatial derivatives

Even though particle refinements can interfere with each other, leading to a global optimization problem, we refine particles separately with respect to their current neighborhood, see [Orthmann and Kolb 2012]. Thus only derivatives based on refined particles for each original particle need to be considered, and $\mathcal{N}$ can be assumed to be constant. This also allows for the optimization of all individual refinement steps in parallel. We thus need to consider the derivative of $\mathbb{E}$ with respect to the position of every inserted particle. In general a kernel function can be written as [Dehnen and Aly 2012]

$$W_{ij} = W\left(\left\|\mathbf{x}_{ij}\right\|, h_{ij}\right) = \frac{C}{h_{ij}^d}\hat{W}(q), \tag{10}$$

where $h_{ij} = \frac{h_i+h_j}{2}$, $q = \frac{\|\mathbf{x}_{ij}\|}{h_{ij}}$, C is a normalization constant, d is the spatial dimensionality of the simulation and $\hat{W}(q)$ being the kernel function, i.e.,$\hat{W}(q) = [1-q]_+^3 - 4[0.5-q]_+^3$ for the cubic spline kernel, where $[\cdot] = \max(\cdot, 0)$. The derivative of the kernel function W with respect to the position of a particle i can then be calculated as

$$\nabla_i W_{ij} = \frac{\mathbf{x_{ij}}}{|\mathbf{x_{ij}}|}\frac{C}{h^{d+1}}\frac{\partial\hat{W}(q)}{\partial q} = \hat{\mathbf{x}}_{ij}W'_{ij}, \tag{11}$$

with $W'(r,h) = \frac{C}{h^{d+1}}\frac{\partial\hat{W}(q)}{\partial q}$ and $\hat{\mathbf{x}}_{ij} = \frac{\mathbf{x_{ij}}}{|\mathbf{x_{ij}}|}$. Using some linear algebra, shown in detail in the supplementary material, the spatial derivatives of the discretized error term for neighboring particle positions $\mathbb{E}_{\mathcal{N}}$ and refined particle positions $\mathbb{E}_{\mathcal{S}}$ with respect to the position of a refined particle i is given as

$$\nabla_i\mathbb{E}_{\mathcal{N}} = \sum_{n\in\bar{\mathcal{N}}} m_n\left(\tau_i + \tau_n\right)\nabla_i W_{in}, \tag{12}$$

with $\bar{\mathcal{N}} = \mathcal{N} \setminus o$, and

$$\nabla_i\mathbb{E}_{\mathcal{S}} = \sum_{s\in\mathcal{S}} m_s\left(\tau_i + \tau_s\right)\nabla_i W_{is}. \tag{13}$$

6.2 Mass distribution derivatives

For the efficient formulation of our optimization cost function, we utilize a set of tunable weights

$$\Lambda = \left[\lambda_0, \ldots \lambda_{n-1}\right], \tag{14}$$

where $\frac{1}{\lambda_i}$ describes the individual mass ratio for each inserted particle i, with the constraint $\sum_s \frac{1}{\lambda_s} = 1$, i.e.,$m_s = m_o/\lambda_i$. Plugging these weights into the error terms τ_n (Eqn. 5) and τ_s (Eqn. 6) yields

$$\begin{aligned}\tau_n &= m_o\sum_{s\in\mathcal{S}}\frac{1}{\lambda_s}W_{ns} - m_o W_{no},\\ \tau_s &= \sum_{j\in\mathcal{N}} m_j W_{sj} + m_o\sum_{k\in\mathcal{S}}\frac{1}{\lambda_k}W_{sk} - \rho_o.\end{aligned} \tag{15}$$

To calculate the derivative of the overall error terms with respect to these weights, we first need to consider the derivative of the kernel function with respect to the support radius, which can be written as

$$\frac{\partial W_{ij}}{\partial h_i} = -\frac{1}{2}\left[\frac{d}{h_{ij}}W_{ij} + qW'_{ij}\right], \tag{16}$$

which is a term not commonly found in computer animation; see the supplementary material for further details. This can be used to evaluate the derivative of the kernel function with respect to the mass of a particle, which yields

$$\frac{\partial W(\mathbf{x}_{ij},h)}{\partial m_i} = \frac{\partial W(\mathbf{x}_{ij},h)}{\partial \mathbf{x}_{ij}}\frac{\partial \mathbf{x}_{ij}}{\partial m_i} + \frac{\partial W(\mathbf{x}_{ij},h)}{\partial h}\frac{\partial h}{\partial m_i}, \tag{17}$$

where $\frac{\partial \mathbf{x}_{ij}}{\partial m_i} = 0$. The term $\frac{\partial W(\mathbf{x}_{ij},h)}{\partial h}$ can be determined using Eqn. 16, with $h = \frac{h_i+h_j}{2}$. Using the definition of the support radius $h_i = \eta\sqrt[3]{\frac{m_i}{\rho_i}}$ [Monaghan 2005] and applying the derivative, the missing term is given as

$$\frac{\partial h_i}{\partial m_i} = \frac{1}{3}\eta\left(\frac{m_i}{\rho_i}\right)^{-\frac{2}{3}}\frac{\partial\frac{m_i}{\rho_i}}{\partial m_i} = \frac{1}{3}\eta\left(\frac{m_i}{\rho_i}\right)^{-\frac{2}{3}}\frac{\frac{\partial m_i}{\partial m_i}\rho_i - m_i\frac{\partial\rho_i}{\partial m_i}}{\rho_i^2}, \tag{18}$$

where $\frac{\partial m_i}{\partial m_i}$ is 1. Applying $\frac{\partial}{\partial m_i}$ further to the standard SPH estimate for density $\rho_i = \sum_j m_j W_{ij}$ yields

$$\frac{\partial \rho_i}{\partial m_i} = \sum_j \left(\frac{\partial m_j}{\partial m_i} W_{ij} + m_j \frac{\partial W_{ij}}{\partial m_i} \right), \tag{19}$$

where $\frac{\partial m_j}{\partial m_i} = \delta_{ij}$ (Kronecker Delta). Putting these equations together results in

$$\frac{\partial W_{ij}}{\partial m_i} = \frac{\partial W_{ij}}{\partial h} \left[\frac{h_i}{3m_i} + \frac{h_i}{3\rho_i} \left(W_{ii} - \sum_k m_k \frac{\partial W_{ik}}{\partial m_i} \right) \right], \tag{20}$$

which means that the value for $\frac{\partial W_{ij}}{\partial m_i}$ depends on itself and values of neighboring particles. In theory this could be solved iteratively, similar to the relation of density and support radius, but we opt to follow the common computer animation notion of all particles having a support radius solely based on their rest density, i.e., $h_i = \eta \sqrt[3]{\frac{m_i}{\rho_0}}$ instead of $h_i = \eta \sqrt[3]{\frac{m_i}{\rho_i}}$ and, accordingly, $V_i = V_j \implies h_i = h_j$. This means that

$$\frac{\partial W_{ij}}{\partial m_i} = \frac{h_i}{3m_i} \frac{\partial W_{ij}}{\partial h_i}. \tag{21}$$

Calculating the derivative of the kernel function with respect to a mass weight additionally requires the derivative of a weight λ_j by another weight λ_i. If in the optimization the weight of one particle is increased and all other particles are equally decreased, we can find a derivative

$$\frac{\partial}{\partial \lambda_i} \left[1 - \sum_{i \neq j} \frac{1}{\lambda_i} \right] = \begin{cases} \frac{1}{n-1} \frac{1}{\lambda_i^2}, & i \neq j \\ 0, & \text{else.} \end{cases} \tag{22}$$

This can be expressed more generally by introducing a matrix $\mathbb{M}$, describing the distribution of weights, which for Eqn. 22 yields

$$\mathbb{M}_{ij} = \begin{cases} \frac{1}{n-1}, & i \neq j \\ -1, & \text{else.} \end{cases} \tag{23}$$

Note that each column of the matrix should sum to zero to ensure mass conservation during optimization. Additionally, for refined particles the support radius does change with respect to the change of mass of a refined particle. For two particles s and j, this results in the following term

$$\frac{\partial h_{sj}}{\partial \lambda_i} = \frac{1}{2} \left(\frac{\partial h_s}{\partial \lambda_i} \mathbf{1}_{\mathcal{S}}(s) + \frac{\partial h_j}{\partial \lambda_i} \mathbf{1}_{\mathcal{S}}(j) \right), \tag{24}$$

with $\mathbf{1}$ being the indicator function defined as

$$\mathbf{1}_{\mathcal{S}}(j) = \begin{cases} 1, & j \in \mathcal{S} \\ 0, & \text{else.} \end{cases} \tag{25}$$

Finally, we can find the derivatives of the discrete error terms for neighboring positions and refined positions, respectively, are calculated as (see the supplementary material for a detailed derivation)

$$\begin{aligned} \frac{\partial \mathbb{E}_{\mathcal{N}}}{\partial \lambda_i} &= \sum_{n \in \tilde{\mathcal{N}}} m_n \left[\tau_n \frac{\mathbb{M}_{is}}{\lambda_i^2} W_{ns} + \frac{1}{\lambda_s} \frac{\partial W_{ns}}{\partial \lambda_i} (\tau_n + \tau_s) \right], \\ \frac{\partial \mathbb{E}_{\mathcal{S}}}{\partial \lambda_i} &= \sum_{k \in \mathcal{S}} m_s \left[\tau_s \frac{\mathbb{M}_{ik}}{\lambda_i^2} W_{sk} + \frac{1}{\lambda_k} \frac{\partial W_{sk}}{\partial \lambda_i} \tau_s \right], \end{aligned} \tag{26}$$

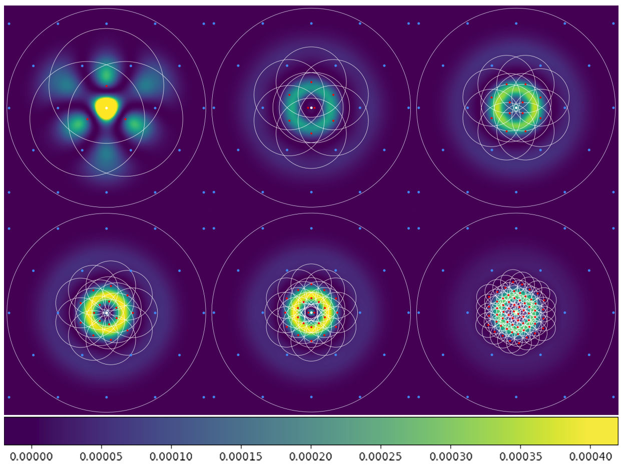

Fig. 4. Optimized patterns for 3 (top-left), 8 (top-middle), 12 (top-right), 16 (bottom-left), 24 (bottom-middle) and 64 (bottom-right) particles, in 2D for the cubic spline kernel, showing particles from $\mathcal{S}$ (red), $\mathcal{N}$ (light blue) and the removed particle o (white). The coloring indicates τ, demonstrating that the error focuses mostly in regions that are not occupied by any particle.

where

$$\begin{aligned} \frac{\partial W_{sj}}{\partial \lambda_i} &= \frac{1}{6\lambda_i^2} \left(h_s \lambda_s \mathbb{M}_{is} \mathbf{1}_{\mathcal{S}}(s) + h_j \lambda_j \mathbb{M}_{ij} \mathbf{1}_{\mathcal{S}}(j) \right) \frac{\partial W_{sj}}{\partial h_{sj}}, \\ \frac{\partial W_{sj}}{\partial h_{sj}} &= -\frac{1}{h_{sj}} \left(d W_{sj} + |\mathbf{x}_{sj}| W'_{sj} \right), W'(r,h) = \frac{C}{h^d} \frac{\partial \hat{W}(q)}{\partial q}. \end{aligned} \tag{27}$$

7 OPTIMIZATION METHODS

The main goal of the introduction of the discretized objective function and its partial derivatives in Section 6 is to determine optimal particle positions and masses for the refinement of fluids, irrespective of specific kernel functions or neighborhood requirements. The naïve solution is to start with a random distribution of positions with an equal amount of mass per inserted particle. However, optimizing from this starting point is too expensive to be done online during a simulation. Instead, we propose optimizing the refinement patterns a priori for ideal conditions, e.g.,in an isotropic hexagonal particle distribution, and use the resulting patterns as initialization for an online optimization.

7.1 A priori optimization

In order to generate a generic set of refinement patterns, we assume that an arbitrary particle o has an isotropic hexagonal neighborhood of particles $\mathcal{N}$. Therefore, we can simply apply our optimization to this ideal neighborhood and the inserted refined particles. We utilize an original particle with the properties

$$h_o = 1, \quad V_o = \frac{4}{3}\pi \frac{1}{N_h}, \quad r_o = \sqrt[3]{\frac{1}{N_h}}, \tag{28}$$

as this allows us to easily rescale the generated patterns for a particle with radius r, by scaling the generated pattern by r. We also separate the optimization for positions and masses into two distinct processes for efficiency. Using the analytical partial derivatives,

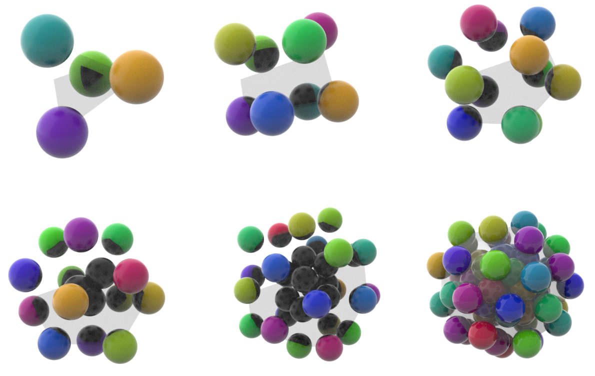

Fig. 5. This figure shows the patterns for 4, 8, 12, 16, 32 and 64 particles in 3D for the Wendland 2 kernel (N=100) with the particle color being chosen for visual distinctiveness only. The transparent object visualizes the convex hull of the particle positions.

from Sec 6.1 and 6.2, it is fairly straightforward to apply any optimization algorithm, e.g.,L-BFGS-B from SciPy [Jones et al. 2001], to optimize the positions of splitting patterns by stacking the components of the inserted particles' position $[\mathbf{x}_{0,x}, \mathbf{x}_{0,y}, ..., \mathbf{x}_{n-1,z}]$, which for n particles results in $n \cdot d$ variables. Figure 4 shows example patterns generated for 4, 8, 16 and 32 particles, respectively, in a 2D setting and Figure 5 shows patterns for 3D settings, which were all initialized with random particle distributions. Figure 4 shows that the error for the resulting configuration is mainly reduced at the particle positions, while it is rather high in between particles, where it has no practical influence.

Different kernel functions yield very similar spatial configurations, however, not all kernel functions converge equally fast, due to pairing instabilities; see Dehnen and Aly [2012] for a general discussion on the differences between kernel functions. Particles may move further from the center, even beyond the hexagonal packing distance, or move together, i.e.,they pair, during the optimization. However, in either case, which can be identified easily, we restart the optimization with a different random initialization to avoid these local minima.

For the optimization of the mass distribution we utilize the SLSQP optimization method from SciPy [Jones et al. 2001]. However, other minimizers can be used as well, as long as the optimizer can handle the required constraint, i.e.,non-negativity of masses and mass conservation.

7.2 Online optimization

Applying the optimization methods described for the a priori optimization to online optimizations would be too expensive due to computational costs, and as such we aim to use simpler methods.

To optimize the positions we use a simple gradient descent algorithm. As we target GPUs for our optimizations, we use a number of threads (e.g., 96) per particle that should be refined, where we can parallelize the evaluation of $\mathbb{E}_{\mathcal{S}}$ and $\mathbb{E}_{\mathcal{N}}$. Here $\mathcal{N}$ denotes the actual set of neighbors of o. In addition we use a simple backtracking line search algorithm in order to determine the gradient step length γ, as we start from an already good initial guess. In our implementation we use up to 32 gradient steps with 8 backtracking attempts with an initial step length of $\gamma = 0.01$ and a backtracking weight $\beta = 0.5$.

The mass distribution of the particles, however, relies on a more complex optimization. The partial derivatives of the discretized objective function with respect to the mass ratios are significantly more complex which makes them expensive to evaluate. Furthermore, the memory used to calculate the Hessian, for some non linear constrained optimization methods, severely limits performance due to memory restrictions. As such, we chose to adapt an evolutionary optimization to our problem, which preserves mass conservation, while not requiring a gradient evaluation.

To avoid explicitly enforcing the constraint $\sum_{s\in\mathcal{S}} \frac{1}{\lambda_s} = 1$, we define a set of unnormalized values

$$\Phi = [\phi_0, \dots, \phi_{n-1}], \tag{29}$$

from which the set of constrained weights is calculated as

$$\Lambda[i] = \frac{\phi_i}{\sum_{s\in\mathcal{S}} \phi_s}. \tag{30}$$

Note that Eqn. 30 enforces $\sum_s \frac{1}{\lambda_s} = 1$ by construction. To determine Φ we sample a normal distribution X, with mean $\bar{\mathbf{X}}$ and standard deviation σ, for every element as

$$\Phi[i] = \text{clamp}\left(X\left(\bar{X} = 1, \sigma^2 = 1\right), \frac{1}{2}, 2\right), \tag{31}$$

which we evaluate on every thread associated with a particle, e.g.,giving us 96 different sets. The values are clamped to avoid negative masses and very large differences between individual particle weights. We then evaluate the discretized objective function for all sets and determine the set Φ_b with the lowest error. Using this set we can then determine an updated set of weights

$$\Phi^{l+1}[i] = \text{clamp}\left(X\left(\bar{X} = \Phi_b[i], \sigma^2 = 2^{-l}\right), \frac{1}{2}, 2\right). \tag{32}$$

We repeat this process for 8 iterations as a variance $\sigma^2 \approx 0.004$ has no practical influence on the result. We also always consider $\Phi = [1, \dots, 1]$, as this set of values describes a uniform distribution of mass. Note that this process is similar to a general random optimization and evolutionary optimization techniques but with modifications to enforce a constraint and to be parallelizable. The overall online optimization process is shown in Algorithm 2.

8 ERROR SMOOTHING

Our minimization process, as will be shown later in the results in Section 10.1, cannot reduce the refinement error to zero, even for ideal particle distributions. Thus, we need to utilize further measures, which can help reduce the instabilities caused by the refinement error as they are not negligible for incompressible fluid simulations. In order to achieve this, we will first introduce a non-constant temporal blending method, followed by an extension of existing artificial viscosity methods that introduces local viscosity on newly refined particles.

Algorithm 2: Our online optimization process applied to every particle o that is refined into n particles, executed in parallel on 96 threads.

```
Initialize positions of refined particles $\mathbb{X}$ using a priori pattern; Sec. 7.1
Initialize weights of refined particles $\Lambda$ to all be $n$; Eqn. 30
$e \leftarrow \mathbb{E}_\mathcal{N} + \mathbb{E}_\mathcal{S}$ using $\mathbb{X}$ and $\Lambda$; Eqn. 7
// Optimize positions using gradient descent
For $g \in [1, 32]$
    $\gamma \leftarrow 0.01$
    For $s \in [1, 8]$
        Evaluate $\nabla_i \mathbb{E}$ with $\mathbb{X}$ for $i \in \mathcal{S}$; Eqns. 12 and 13
        $\mathbb{X}^g[i] \leftarrow \mathbb{X}[i] + \gamma \nabla_i \mathbb{E}$
        $e^g \leftarrow \mathbb{E}_\mathcal{N} + \mathbb{E}_\mathcal{S}$
        If $e^g < e$
            Update $\mathbb{X} \leftarrow \mathbb{X}^g$ and stop $s$ iteration
        $\gamma \leftarrow \gamma \cdot \beta$
// Optimize weights using evolutionary optimization
Initialize $\Phi[i] = 1, \forall i \in \mathcal{S}$
$\Lambda_b[i] \leftarrow 1/n$
$e \leftarrow \mathbb{E}_\mathcal{N} + \mathbb{E}_\mathcal{S}$ using $\Lambda_b$ and $\mathbb{X}$
For $l \in [1, 8]$
    For every thread $t$
        Sample $\Phi^l_t$ using Eqn. 32
        $\Lambda^l_t[i] = \Phi^l_t[i] / \sum_{s \in \mathcal{S}} \Phi^l_t[s]$
        $e^l_t \leftarrow \mathbb{E}_\mathcal{N} + \mathbb{E}_\mathcal{S}$ using $\Lambda^l_t$ and $\mathbb{X}$
    Find thread with lowest error $t = \inf_t e^l_t$
    If $e^l_t < e_b$
        Update $\Phi \leftarrow \Phi^l_t, \Lambda_b \leftarrow \Lambda^l_t, e_b \leftarrow e^l_t$
```

8.1 Non-constant temporal blending

The basic idea of a temporal blending method [Orthmann and Kolb 2012] is that quantities are the result of a linear blending operation between the actual quantities A_s of refined particles $s \in \mathcal{S}_o$ and an estimated quantity for the original particle $\hat{A}_o$ as

$$A_s^{\text{blended}} = (1 - \beta_s) A_s + \beta_s \hat{A}_o, \forall s \in \mathcal{S}_o, \tag{33}$$

where β describes a linear interpolation weight. In order to estimate the quantity for the original particle Winchenbach et al. [2017] track the position where the original particle would be at a new time point $t + \Delta t$ as $\boldsymbol{x}_o$ using the average velocity of all particles refined from o as

$$\boldsymbol{x}_o^{t+\Delta t} = \boldsymbol{x}_o^t + \Delta t \frac{1}{n} \sum_{s \in \mathcal{S}_o} v_s^t, \tag{34}$$

where n is the number of refined particles from o. Using this estimated position and the standard SPH estimate from Eqn. 1, we can determine an estimated quantity $\hat{A}_o$ by ignoring all particles refined from o and adding the interaction of o with itself

$$\hat{A}_o = m_o \frac{A_o}{\rho_o} W_{oo} + \sum_j \begin{cases} m_j \frac{A_j}{\rho_j} W_{oj}, & j \in \mathcal{N} \setminus \mathcal{S}_o, \\ 0, & j \in \mathcal{S}_o \end{cases}. \tag{35}$$

Winchenbach et al. [2017] additionally apply a clamping operation to estimated density values $\hat{\rho}_o$. The blending weight as described by Orthmann and Kolb [2012] and Winchenbach et al. [2017] is given as

$$\beta_s^{t+1} = \beta_s^t + \Delta\beta, \quad \beta^0 = 1, \tag{36}$$

where a constant change in blend weight per time step t is utilized. The initial blend weight is 1, with a fixed change of blend weight per time step of $\Delta\beta = -\frac{1}{\Theta}$. Here Θ is usually chosen to be 10 and denotes the number of time steps over which blending occurs. Instead we propose to utilize the following blend weight β for any particle i as

$$\beta_i = \text{clamp}\left(\frac{\Delta t^0}{2\Delta t} \left[1 - \frac{t_i}{\Theta \Delta t^0} \right], 0, \frac{1}{2} \right) \tag{37}$$

which bases the blend weight β_i on a value describing the lifetime of a particle t_i, the current time step Δt, the time step at the time of refinement Δt^0, as well as the number of blend steps Θ. For a fixed time step this results in a linearly decreasing weight, per time step, as prior methods, but instead of starting with an initial blend weight of 1 we start with a blend weight of $\frac{1}{2}$. However, if the time step changes during the blending process, i.e., $\frac{\partial \Delta t}{\partial t} \neq 0$, the blend weight gets adjusted as well. For increasing time steps, $\frac{\partial \Delta t}{\partial t} > 0$ the blend weight decreases more slowly. For example with $\Delta t^0 = 0.1$ and $\Theta = 10$, changing the time step to $\Delta t = 0.2$ halfway through the blend process causes the blend weight to change from $\beta = \frac{1}{4}$ to $\beta = \frac{1}{8}$. For decreasing time steps $\frac{\partial \Delta t}{\partial t} < 0$ the blend weight instead reverts to $\beta = \frac{1}{2}$. This is motivated by the fact that $\frac{\partial \Delta t}{\partial t} > 0$ indicates a stabilizing overall simulation, whereas $\frac{\partial \Delta t}{\partial t} < 0$ indicates a destabilizing simulation.

8.2 Local viscosity

In general, the refinement error alters the local density in an incompressible fluid, causing visually noticeable instabilities. This divergence can be smoothed by introducing a higher artificial viscosity. However, increasing the artificial viscosity on a global level for all particles prevents the simulation of an overall relatively inviscid liquid. To avoid this problem, we only introduce additional artificial viscosity locally, which only affects particles that are in the process of being blended, e.g., those with $\beta_i > 0$. XSPH [Monaghan 2002] uses artificial viscosity to modify the velocity of a particle i, which is given as

$$v_i^{\text{new}} = v_i + \sum_j c \frac{m_j}{\rho_j} v_{ij} W_{ij}, \tag{38}$$

where c is the viscosity factor. We propose a modified c^{new} given by

$$c^{\text{new}} = c \begin{cases} 1, & i \in \mathcal{S}_o \wedge j \in \mathcal{S}_o \\ \left(1 + 0.5 \frac{\beta_i + \beta_j}{2}\right), & \text{else} \end{cases}, \tag{39}$$

which can similarly be applied to a traditional artificial viscosity formulation [Monaghan 2005] by changing the corresponding viscosity factor ν. This term still results in the same global viscosity applied to all particle interactions c, but introduces an additional local viscosity $c\frac{(\beta_i + \beta_j)}{4}$ that only affects interactions of particles with newly refined particles, as β is 0 for any non-blending particle, and excludes interactions between refined particles belonging to the same original particle. This additional term has a maximal magnitude of $0.5c$, i.e.,it increases viscosity locally by at most 50%. Applying this artificial viscosity in our experiments reduces any instabilities that remain after our optimized refinement process, without noticeably changing the global behavior.

9 SIZING-FUNCTIONS

In adaptive SPH a sizing function determines the ideal particle volume $V(\boldsymbol{x})$ at a location $\boldsymbol{x}$ and is commonly defined using a distance function $d(\boldsymbol{x})$ from the region of interest, e.g.,the fluid surface. Usually, the sizing function specifies a smooth gradient from a base volume V_{base} at a maximum distance $d_{\max}$ to the finest volume $V_{\text{fine}} = \frac{1}{\alpha} V_{\text{base}}$ at the fluid surface with a desired adaptive volume ratio α. Winchenbach et al. [2017] proposed a linear sizing function that scales the volume directly with the distance, i.e.,$V \propto d$, as

$$V(\boldsymbol{x}) = \left[\frac{1}{\alpha} + \frac{d(\boldsymbol{x})}{d_{\max}}\left(1 - \frac{1}{\alpha}\right)\right] V_{\text{base}}. \tag{40}$$

Unless otherwise noted we use this formulation for all of our results as the linear sizing of volume yields high surface-resolutions at moderate overall particle counts. However, other sizing functions might also be used. A practical problem that arises for Eqn. 40 is that very high adaptive volume ratios, e.g.,$1 : 4,000$, can result in a very thin sheets of high-resolution particles at the surface, which does not provide the expected improvement in quality. This effect, however, can be avoided by either using deep fluid volumes to increase the thickness of the highest-resolution sheet, additional factors in the sizing, e.g.,camera visibility or closeness to an object of interest (see the **torus** scene), or by using a different sizing function. A straight forward replacement for the sizing function in Eqn. 40 is to scale the particle radius linearly, resulting in a cubic scaling of particle volume based on surface distance. This sizing function can be defined as

$$V(\boldsymbol{x}) = \frac{4}{3}\pi\left[\left(\frac{1}{\sqrt[3]{\alpha}} + \frac{d(\boldsymbol{x})}{d_{\max}}\left(1 - \frac{1}{\sqrt[3]{\alpha}}\right)\right) r_{\text{base}}\right]^3. \tag{41}$$

However, this sizing function generates significantly more particles for the same adaptive volume ratio, when compared to Eqn. 40, resulting in a reduction of achieved adaptive volume ratios by a factor 100. For example, for a scene with an achievable adaptive volume ratio of $1 : 1,000,000$ using Eqn. 40, using Eqn. 41 would result in an achievable adaptive volume ratio of $1 : 10,000$, due to computational resource limitations.

10 RESULTS AND DISCUSSION

All simulations were run on a single Nvidia GeForce RTX 2080ti GPU with 11 GiB of VRAM, a 32 core AMD Ryzen 3970x with 64 GiB of RAM. Pressure solving was done using DFSPH [Bender and Koschier 2015] with XSPH [Monaghan 2005] for artificial viscosity, with fluid air phase interactions based on Gissler et al. [2017], surface tension effects model from Akinci et al. [2013], with the vorticity refinement method of Bender et al. [2017], dynamically adjusted time steps based on the CFL condition [Ihmsen et al. 2013] and the data handling model from Winchenbach and Kolb [2019]. In all examples we set DFSPH to a density error of 0.01% and a divergence error of 0.1%. Renderings were done using a custom ray tracing program, with surface extraction based on the work of Yu and Turk [2013]. Surface distance calculations were based on a modified approach of Horvath and Solenthaler [2013]. We use the overall adaptive method of Winchenbach et al. [2017] in our evaluations, although our approach is not restricted to this adaptivity approach. We implemented our approach in the open source SPH framework openMaelstrom [Winchenbach 2019] using the boundary handling approach of Winchenbach et al. [2020]. For the a priori optimizations we use SciPy [Jones et al. 2001]

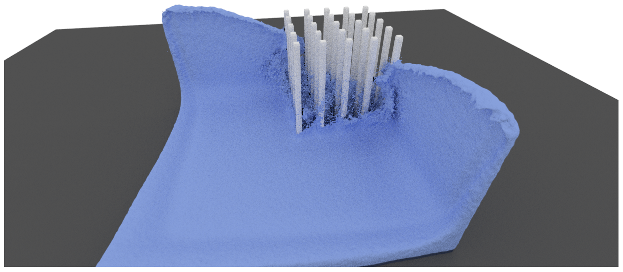
Fig. 6. The **corner dam break** scene with an extracted fluid surfaced based on [Yu and Turk 2013] for an adaptive volume ratio of $1 : 100$.

10.1 Test Scenes

We evaluated our approach in seven scenes. The **inlet** scene involves a fluid inlet, surrounded by a box, emitting fluid into a basin, which is agitated by a moving wall on the opposing side of the inlet stream; see Fig. 1. The **corner dam break** scene involves an initial fluid volume located in one corner of the simulation domain colliding with a regular obstacle in the opposing corner of the domain; see Fig. 6. The **double dam break** scene involves two fluid volumes, initially located in opposing corners of the simulation domain, colliding with a simple cubical rigid object placed in the center; see Fig. 11. The **simple dam break** scene involves a fluid volume in a simple box shaped domain with no additional obstacles; see Fig. 12. The **stream** scene involves a fluid inlet in a simulation with no gravity; see Fig. 14. The **hemisphere dam break** scene involves a fluid volume colliding with a hemisphere on the floor; see Fig. 15. The **moving sphere** scene involves a solid sphere slowly being moved through a resting fluid volume; see Fig. 17. Finally, the **torus** scene involves a torus rotating about its vertical axis in a resting basin of liquid; see Fig. 16. We chose a basic particle radius of $r = 0.5m$ in all of our scenes, however our method would also work at different basic particle resolutions.

10.2 A priori position optimization

To evaluate our a priori optimization process, we optimized refinement patterns for 2 to 32 particles using the cubic-spline, quintic-spline, Wendland 2 and Wendland 4 kernels, using an L-BFGS-B optimizer [Byrd et al. 1995]; see Fig. 7. The total error shows similar behavior for different kernel functions, i.e.,the refinement pattern for two particles has high error, with patterns around five to ten particles having lower errors, and increasing error ratios on higher particle counts. In our evaluation the cubic-spline function shows the largest error, an order of magnitude higher than the quintic-spline function, and the Wendland 4 Kernel to have similar behavior to the quintic-spline function. Additionally, the error on the neighboring particles is the main component of the overall error, i.e.,the error on the refined particles is two orders of magnitude

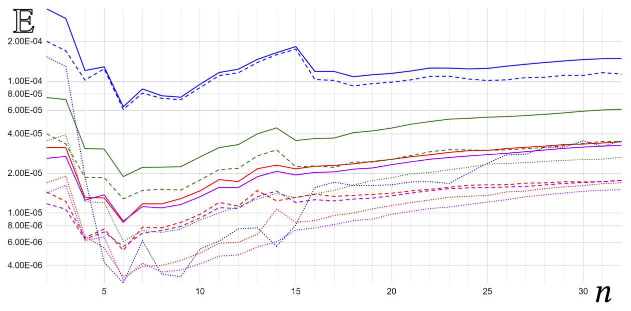

Fig. 7. The logarithmic error terms $\mathbb{E}$ (solid), $\mathbb{E}_{\mathcal{N}}$ (dashed) and $\mathbb{E}_{\mathcal{S}}$ (dotted) for a $1:n$ split using a cubic-spline (blue), quintic-spline (red), Wendland 2 (green) and Wendland 4 (purple) kernel functions in 3D after optimizing positions only.

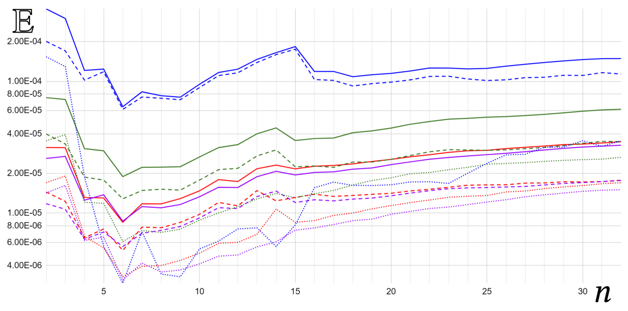

Fig. 8. The logarithmic error terms $\mathbb{E}$ (solid), $\mathbb{E}_{\mathcal{N}}$ (dashed) and $\mathbb{E}_{\mathcal{S}}$ (dotted) for a $1:n$ split using a cubic-spline (blue), quintic-spline (red), Wendland 2 (green) and Wendland 4 (purple) kernel functions in 3D after optimizing masses (with pre-optimized positions).

smaller. Interestingly, the 1:13 splits results in icosahedron-shaped refinement patterns for all kernel functions, with one particle at the center, which is the configuration manually defined by Vacondio et al. [2016].

10.3 A priori mass optimization

To evaluate the mass optimization process we use the patterns optimized a priori for positions only, see Sec. 10.2, and used a constrained trust-region optimizer [Byrd et al. 1987] to optimize the masses, without changing positions; see Fig. 8. The results are almost identical to the position optimization alone and do not show significant overall improvement for the kernels evaluated here.

10.4 A priori simultaneous optimization

After the a priori optimization of both positions and masses, we further optimized the refinement patterns, using a constrained trust-region and SLSQP optimizer [Kraft 1988], by simultaneously optimizing positions and masses; see Fig. 9. The results show an overall reduction of the error by up to a factor of 2, mostly reducing the error on the neighboring particles and not on the refined particles themselves. The overall refinement patterns stay in very similar overall spatial configurations and get only slightly modified during the combined optimization.

We tried two different initialization schemes for the combined optimization, i.e., starting from pre-optimized spatial layouts and random initialization. While the pre-optimized initialization results in stable, but potentially local minima, the combined optimization did not robustly converge when initialized with random positions and masses, regardless of the optimization algorithm used. Furthermore, comparing our results against the prior refinement patterns of Winchenbach et al. [2017] (see Fig. 10), we can observe a significant reduction in the error terms. Overall, our refinement patterns provide an improvement of about two orders of magnitude, regarding both the error on refined and neighboring particles, and yield comparable errors across all refinement ratios.

10.5 Online optimization

Starting with the a priori optimized refinement patterns we used the **double dam break** scene to evaluate the errors during an SPH simulation, using the cubic-spline kernel and refinement patterns for 2 to 16 particles. In general, using the a priori refinement patterns worked reasonably well in most cases. We, however, frequently observed instabilities, particularly in the fluid interior that resulted in local compression, which can destabilize the entire simulation. Applying our proposed online optimization avoids virtually all of these interior instabilities, however, instabilities caused by boundary interactions and by fully constrained particle merging remain (see also Sec. 4). Overall, applying the online optimization to practical particle configurations during a simulation results in slightly higher error values (in average an additional error of $\mathbb{E} \approx 0.01$) compared to the a priori optimization under ideal particle configurations. This

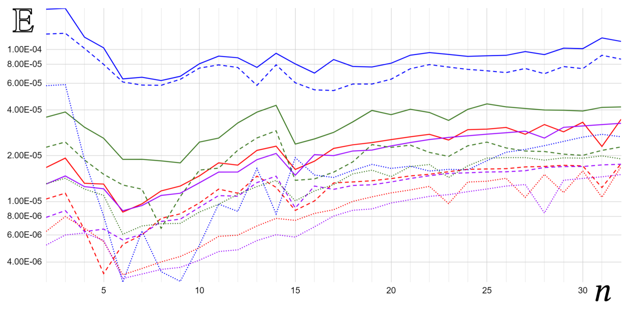

Fig. 9. The logarithmic error terms $\mathbb{E}$ (solid), $\mathbb{E}_{\mathcal{N}}$ (dashed) and $\mathbb{E}_{\mathcal{S}}$ (dotted) for a $1:n$ split using a cubic-spline (blue), quintic-spline (red), Wendland 2 (green) and Wendland 4 (purple) kernel function in 3D after simultaneous optimization (with pre-optimized positions and masses).

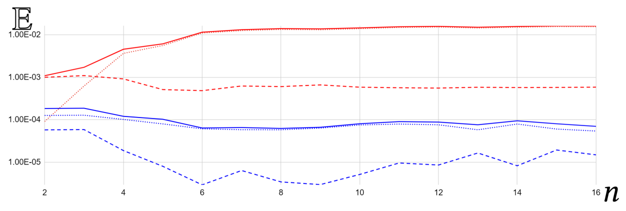

Fig. 10. The logarithmic error terms $\mathbb{E}$ (solid), $\mathbb{E}_{\mathcal{N}}$ (dashed) and $\mathbb{E}_{\mathcal{S}}$ (dotted) for a $1:n$ split for a cubic-spline kernel using our approach (blue) and manually tuned refinement patterns [Winchenbach et al. 2017].

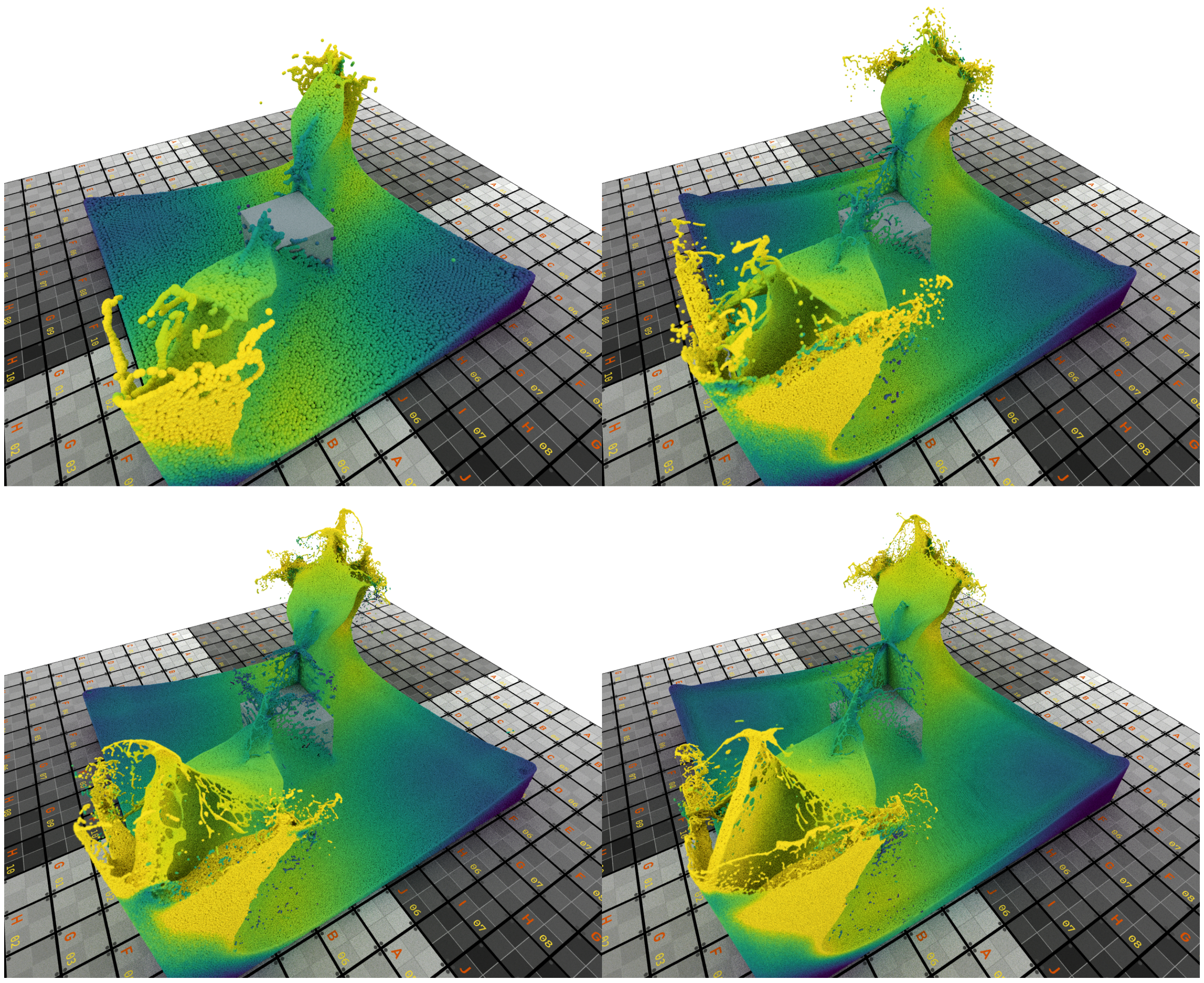

Fig. 11. The **double dam break** scene we used to evaluate performance and online optimization, particle velocity color coded from purple (0 m/s) to yellow (30 m/s). Top left shows a low resolution simulation ($r = 0.5$m), bottom left an adaptive resolution simulation ($r_{\text{base}} = 0.5$m, 128:1 adaptive), top right a uniform simulation with the same particle count as the adaptive simulation ($r = 0.218$m) and bottom right a high-resolution simulation ($r = 0.1$m).

additional error is not reduced when using the same optimization methods as a priori. Furthermore, outside of removing instabilities, the online optimization provides significant visual benefits on the fluid surface as using the exact same pattern on the surface leads to visibly repeating particle patterns on the surface, which are also visible in some surface extraction approaches. In the gravity-free **stream** scene, our method is capable of producing a smooth fluid surface in a difficult scenario and provides a smooth resolution gradient from low to high resolution, whereas prior methods were not able to stably simulate this scene, when enforcing incompressibility; see Fig. 14. Additionally, this improved behavior on the surface of an inlet flow allows us to emit particles at a fixed low resolution, as done in the **inlet** scene, and only refining the particles once they become visible. Moreover, evaluating the overall energy of the **simple dam break** scene, see Fig. 13, shows a significantly reduced loss of energy when using all aspects of our method, compared to prior work. Whilst using the online optimization alone, without local viscosity, does not provide significant benefits in this regard, i.e., the blue and green lines are fairly close, using our online optimization allows for the utilization of our local viscosity scheme that provides a significant reduction in dampening. Using the blending process, and refinement patterns from Winchenbach et al. [2017] results in a mostly stable simulation, but causes some instabilities. For example, at 12 seconds (see Fig. 13) a momentary increase of the total energy can be observed, which was caused by an instability induced by a particle refinement. Reducing the impact of these momentary increases in energy can be achieved by increasing the

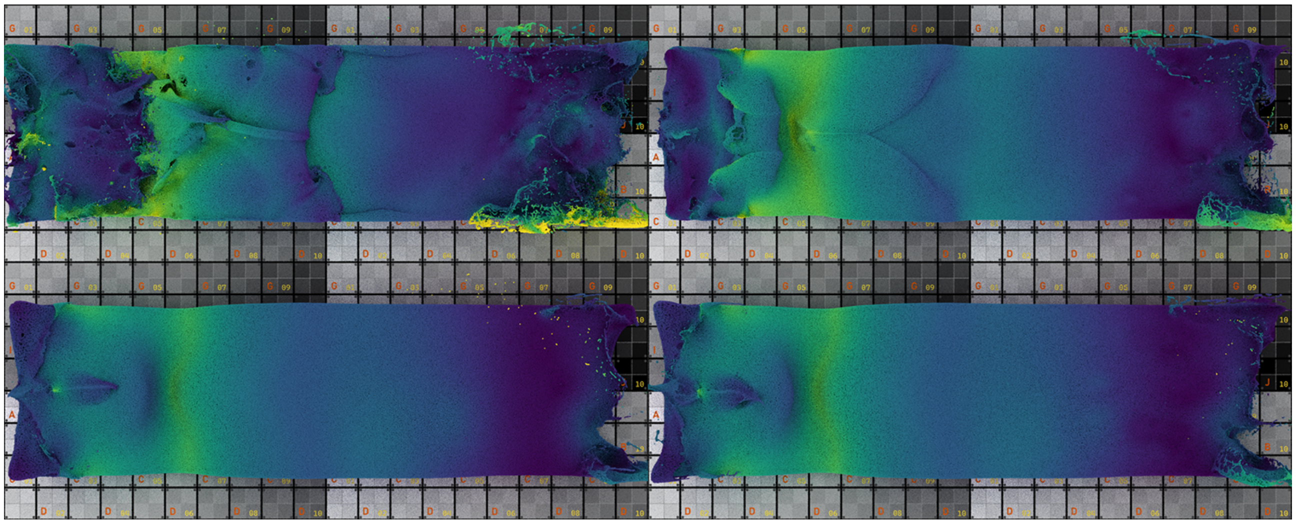

Fig. 12. The **simple dam break** scene we used to evaluate the impact of our proposed blending, local viscosity and online optimization approaches, particle velocity color coded from purple (0 m/s) to yellow (30 m/s). Top left uses our blending, local viscosity and online optimization, top right uses our blending and online optimization, bottom left uses our blending and bottom right uses the approach of [Winchenbach et al. 2017]. Velocity color coded.

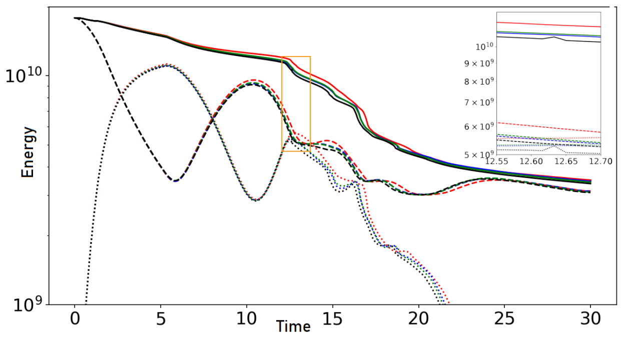

Fig. 13. The overall energy for the **simple dam break** scene over 30 simulated seconds. The dashed line indicates potential energy, the dotted line kinetic energy and the solid line indicates total energy. The graph compares the prior approach from [Winchenbach et al. 2017] (black) against our approach with our blending, local viscosity and online optimization (red), with our blending and online optimization (blue) and with only our blending (green). The top right section shows the orange region closer up.

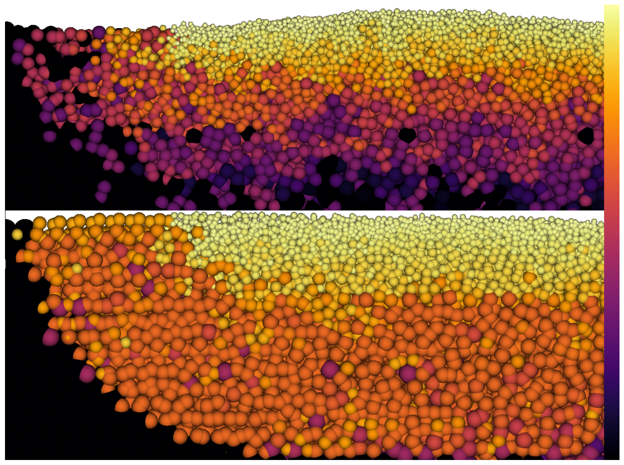

Fig. 14. The gravity-free **stream** scene, particle volume color coded from black to yellow. The prior approach [Winchenbach et al. 2017] (top) is not able to stably simulate this scenario, whilst our improved method (bottom) only produces some slight irregular particle distributions.

artificial viscosity, however, this causes an overall significant loss of detail and a noticeably viscous fluid behavior; see Fig. 12 bottom right and top right. Note that increasing the artificial viscosity only reduces the impact of these instabilities but does not prevent them completely; see Fig. 15 top row. Furthermore, the **moving sphere** scene (see close-up Fig. 17) highlights the smooth transition of particles between resolutions, even as they are close to a boundary object, using our method.

10.6 Influence of local viscosity and blending

In order to evaluate the influence of our blending scheme and the local viscosity approach we use the **simple dam break** scene. We visually compare the overall flow behavior of using our blending with local viscosity and online optimization, our blending and online optimization, our blending and the blending approach of Winchenbach et al. [2017] (using their refinement patterns). Comparing our blending against the prior approach, we only observe a small difference (Fig. 12 bottom left and bottom right). Adding the online optimization allows us to lower the overall viscosity of the simulation, as it becomes more stable, resulting in more flow details. However, adding the local viscosity allows us to reduce the global artificial viscosity by a factor of 2, resulting in a significant increase of more surface details. In the **hemisphere dam break** scene the

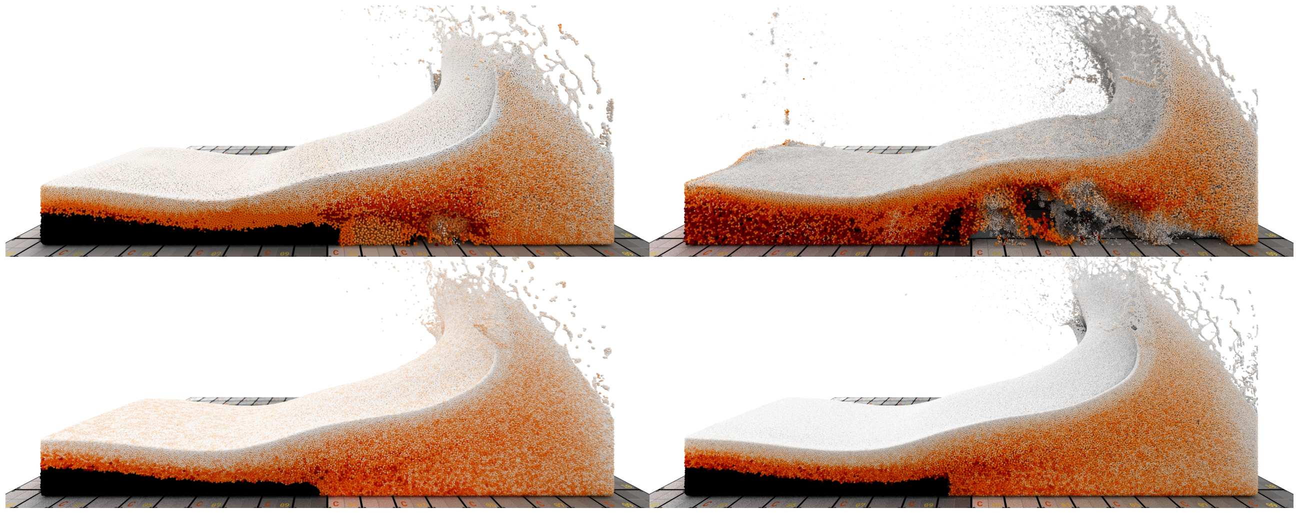

Fig. 15. The **hemisphere dam break** scene, particle volume color coded from black to white. The top row uses the prior approach, while the bottom row uses our approach, with the left column using a 1 : 32 adaptive volume ratio and the right column using a 1 : 1000 adaptive volume ratio. Without an online optimization process (top row), instabilities appear at the boundary that get more pronounced as the adaptive volume ratio increases and would require significant added viscosity to reduce. Using our approach (bottom row), with its online optimization, yields a stable simulation for both adaptive volume ratios.

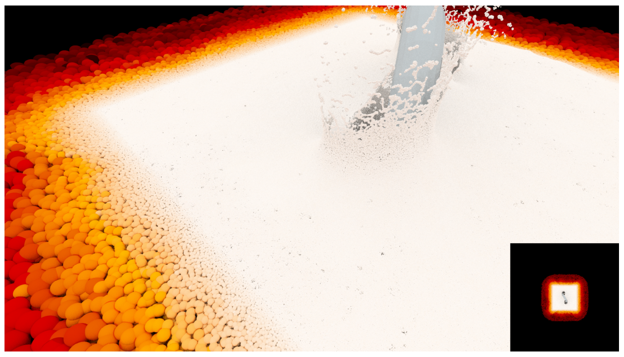

Fig. 16. The **torus** scene, particle volume color coded black to white, with the bottom right showing the overall simulation domain. Our method can simulate an adaptive volume ratio of up to 1 : 1, 000, 000, allowing for fine details close to the torus but limiting overall computational resources.

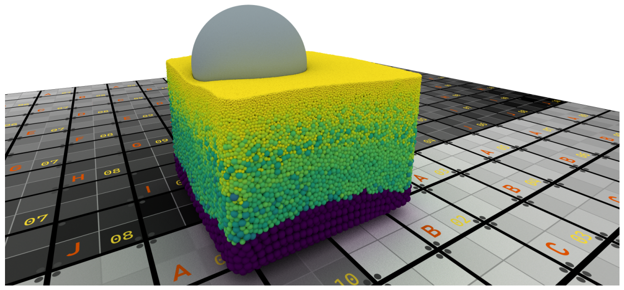

Fig. 17. The **moving sphere** scene, particle volume color coded from high (purple) to low (yellow). Here a sphere slowly moves through a pool of liquid with an adaptive volume ratio of 1 : 20, 000.

Variant	n_{fluid}	radius	ratio	Δt	Frame	Adaptive
		$/m$		$/ms$	$/s$	$/ms$
Low	168K	0.5	1:1	8.0	0.15	
Average	1.9M	0.218	1:1	5.8	1.55	
High	21M	0.1	1:1	2.0	45.0	
Adaptive	1.9M	0.5	1:128	3.8	1.77	325

Table 1. Quantitative comparison for the **double dam break** scene; see Fig. 11. All performance numbers are average values with respect to 30 seconds simulation time and timings refer 1/60s of simulation time.

induced instabilities from the refinement process using prior approaches are too high to be compensated by an increased artificial viscosity and, thus, fully destabilize the simulation at higher adaptive volume ratios; see Fig. 15 top right. In contrast to this, our online optimization process ensures a low refinement error and, accordingly, enables much higher adaptive volume ratios in difficult scenarios.

10.7 Performance

To evaluate the performance and efficiency of our method we first use the **double dam break** scene; see Fig. 11. In this scene, we compare the adaptive simulation with a volume ratio of 128:1, a fixed resolution simulation with the same number of particles as the adaptive simulation, on average, and a high-resolution simulation at approximately the finest resolution of the adaptive simulation. The simulated time is 30 sec; see Table 1 for the quantitative results. Overall, our method provides comparable performance to a simulation of equal particle count, with some overhead due to the usage of an adaptive method. Note that the time per frame of our method minus the time spent on adaptivity related methods is less than the time per frame of the average resolution simulation due to a lower time step requiring fewer pressure solver iterations per frame. Compared to the high-resolution variant, our method with moderate adaptive volume ratios provides a significant speed-up of approximately 25 times. Accordingly, the speed-up will become significantly higher, at higher adaptive volume ratios. However, due to computational resource limitations, we were not able to provide a similar comparison against higher uniform resolution. Overall, the surface appearance of our method is similar to the high-resolution one, i.e.,considering the tearing of thin fluid sheets, but at orders of magnitude lower computational costs, even at moderate adaptive volume ratios. Furthermore, in the **torus** scene, our method can focus computational resources in small areas of interest, allowing for much greater detail without requiring hundreds of millions of particles in areas that are not of interest. However, scenes of very high adaptive volume ratios are difficult to render as the adaptivity induces a highly uneven particle distribution that causes raytracing acceleration structures, e.g., kd-trees, to be very unbalanced and, thus, inefficient. Accordingly, even when rendering particles as spheres, the computational requirements increase linearly with higher adaptive volume ratios, i.e., $O(\alpha)$, and make rendering very high ratios computationally difficult. For example, with our computational resources, the **hemisphere dam break** scene required 4 hours to render a sequence of 30 seconds at an adaptive volume ratio of $1:1,000$, the **moving sphere** scene took 2.5 days to render at a ratio of $1:20,000$ for a 20 second sequence, whereas the **torus** scene required multiple hours for a single frame at a ratio of $1:1,000,000$.

Finally, at very high adaptive volume ratios, the neighborhood search becomes computationally increasingly expensive. For example, at a ratio of $1:1,000$, a cell contains approximately $12,000$ particles, compared to 12 particles per cell in homogenous resolutions. The data-structure approach of Winchenbach and Kolb [2019] resolves these problems in most situations; however, due to symmetric interactions of particles, required to ensure stability, the number of particles queried to find the actual neighbors of a particle can still be significantly higher than for homogenous resolutions. This problem can reduced by ensuring a large enough distance between low and high-resolution particles, e.g.,by setting $d_{\max}$ sufficiently high in Eqn. 40. Furthermore, limiting the number of neighbors per particle, see [Winchenbach et al. 2016] and [Winchenbach and Kolb 2019], can further reduce the problem for actual SPH evaluations, however the neighborhood search is still a computationally expensive operation.

10.8 Clamping mass distributions

When using an adaptive method, the user specifies a desired volume ratio between the volume of the smallest V_{fine} to the volume of the largest V_{base} particles. However, this desired ratio is almost never exactly achieved. For example, a particle with $V = \frac{1}{350}V_{\text{base}}$ and a desired ratio of $1:1000$ might be split into 3 particles, which results in particles of volume $\frac{1}{1150}V_{\text{base}}$, exceeding the desired ratio. Optimizing the mass distribution exacerbates this effect as the optimization process creates particles of very different volumes. In Sec. 7.2, we proposed to clamp the weights ϕ between 0.5 and 2, which limits the variation in particle sizes. In the simulation shown in Fig. 6 the clamped optimization process results in an effective ratio of $1:1250$ instead of the desired $1:512$ ratio. Not clamping the weights resulted in an effective ratio of $1:7500$. Consequently, this substantial difference in smallest particle volume requires a significantly smaller time step, due to the CFL condition.

10.9 Limitations

The adaptive method of Winchenbach et al. [2017], which we base our contributions on, already demonstrated scaling problems of certain parameters, e.g., the surface tension parameters used by Akinci et al. [2013] and parameters used for surface extraction by Yu and Turk [2013]. That is, these parameters are heavily dependent on particle sizes, causing visual discontinuities in the surface extraction. Additionally, Winchenbach et al. [2017] described a problem with boundary handling methods based on particle representations, due to size differences, which can be avoided by using non-particle-based methods, i.e., [Koschier and Bender 2017]. Moreover, particle merging can lead to instabilities, especially close to boundaries, if applied with SPH solvers commonly used in computer animation. Furthermore, if the sizing function is based on the surface distance of a particle, i.e., using the surface-distance method of Horvath and Solenthaler [2013], the stability of these methods plays an important role in the stability of the overall method. Accordingly, some artifacts may arise due to a non-stable sizing function, i.e., particles sitting on the surface of a boundary might not be properly detected as surface particles. Investigating this is beyond the scope of this paper.

Additionally, rendering a simulated fluid surface is an important aspect in computer animation. However, existing surface extraction methods are not designed for varying particle resolutions. They often involve parameters that significantly depend on the particle resolution and lead to visual artifacts such as missing details in high-resolution areas, lumpy surfaces in low resolution areas or visible changes in areas of varying resolution. Because of this, and since we explicitly need to investigate the varying particle resolution, we opt for displaying particle-based renderings and only provide an example of a surface extraction for parameters chosen for the high-resolution surface; see Fig. 6. For very high adaptive volume ratios even particle-based renderings become impractical; see Fig. 16.

Please note that in the images we use linearly color coded quantities. Thus, a change from the highest particle volume to a particle with half the volume, i.e., a 1:2 refinement, results in a visual discontinuity. Nonlinear color mapping could resolve this discontinuity to some degree, but makes the results more difficult to interpret.

11 CONCLUSIONS

We presented an optimization approach for particle refinement patterns, based on a novel discretized objective function that describes the error introduced by the particle refinement for symmetric SPH formulations. This allows us to significantly improve stability and removes the need for user intuition and parameter tuning, and is applicable to arbitrary refinement ratios using any kernel function. Our optimization approach works both a priori, to generate refinement patterns for ideal particle distributions, and online, to optimize the refinement pattern during a simulation with respect to the specific particle neighborhood. We also presented an improved non-linear blending process that, together with a novel local artificial viscosity formulation, that removes the impact of residual refinement errors. Our improved process allows for the simulation of highly adaptive incompressible SPH flows, even in highly turbulent and low-viscosity situations. Currently, our approach is mostly limited by other methods it relies upon, i.e., surface extraction methods.

REFERENCES

Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J Guibas. 2007. Adaptively sampled particle fluids. In *ACM Transactions on Graphics (TOG)*, Vol. 26. Acm, 48.

Nadir Akinci, Gizem Akinci, and Matthias Teschner. 2013. Versatile surface tension and adhesion for SPH fluids. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 182.

Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 62.

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2013. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 103.

Stefan Band, Christoph Gissler, Andreas Peer, and Matthias Teschner. 2018. MLS pressure boundaries for divergence-free and viscous SPH fluids. *Computers & Graphics* 76 (2018), 37–46.

Stefan Band, Christoph Gissler, and Matthias Teschner. 2017. Moving least squares boundaries for SPH fluids. In *Proceedings of the 13th Workshop on Virtual Reality Interactions and Physical Simulations*. Eurographics Association, 21–28.

Jan Bender and Dan Koschier. 2015. Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics symposium on computer animation*. ACM, 147–155.

Jan Bender, Dan Koschier, Tassilo Kugelstadt, and Marcel Weiler. 2017. A micropolar material model for turbulent SPH fluids. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 1–8.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing* 16, 5 (1995), 1190–1208.

Richard H Byrd, Robert B Schnabel, and Gerald A Shultz. 1987. A trust region algorithm for nonlinearly constrained optimization. *SIAM J. Numer. Anal.* 24, 5 (1987), 1152–1170.

Walter Dehnen and Hossam Aly. 2012. Improving convergence in smoothed particle hydrodynamics simulations without pairing instability. *Monthly Notices of the Royal Astronomical Society* 425, 2 (2012), 1068–1082.

Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation'96*. Springer, 61–76.

Jonathan Feldman. 2006. Dynamic refinement and boundary contact forces in smoothed particle hydrodynamics with applications in fluid flow problems.

Jonathan A. Feldman and Javier Bonet. 2007. Dynamic refinement and boundary contact forces in SPH with applications in fluid flow problems. *Internat. J. Numer. Methods Engrg.* 72, 3 (2007), 295–324.

Makoto Fujisawa and Kenjiro T Miura. 2015. An efficient boundary handling with a modified density calculation for SPH. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 155–162.

Christoph Gissler, Stefan Band, Andreas Peer, Markus Ihmsen, and Matthias Teschner. 2017. Generalized drag force for particle-based simulations. *Computers & Graphics* 69 (2017), 1–11.

Christoph Gissler, Andreas Peer, Stefan Band, Jan Bender, and Matthias Teschner. 2019. Interlinked SPH Pressure Solvers for Strong Fluid-Rigid Coupling. *ACM Transactions on Graphics (TOG)* 38, 1 (2019), 5.

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

Markus Ihmsen, Nadir Akinci, Marc Gissler, and Matthias Teschner. 2010. Boundary Handling and Adaptive Time-stepping for PCISPH. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2010)*, Kenny Erleben, Jan Bender, and Matthias Teschner (Eds.). The Eurographics Association. https://doi.org/10.2312/PE/vriphys/vriphys10/079-088

Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. 2013. Implicit incompressible SPH. *IEEE transactions on visualization and computer graphics* 20, 3 (2013), 426–435.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001. SciPy: Open source scientific tools for Python. http://www.scipy.org/ [Online; accessed 2019-04-05].

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/

Dan Koschier and Jan Bender. 2017. Density maps for improved SPH boundary handling. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 1.

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. 2019. Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids. In *Eurographics 2019 - Tutorials*, Wenzel Jakob and Enrico Puppo (Eds.). The Eurographics Association, 1–41. https://doi.org/10.2312/egt.20191035

Dieter Kraft. 1988. A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt* (1988).

Chen Li, ChangBo Wang, and Hong Qin. 2015. Novel adaptive SPH with geometric subdivision for brittle fracture animation of anisotropic materials. *The Visual Computer* 31, 6-8 (2015), 937–946.

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating water and smoke with an octree data structure. In *ACM transactions on graphics (TOG)*, Vol. 23. ACM, 457–462.

Miles Macklin and Matthias Müller. 2013. Position based fluids. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 104.

Joseph J Monaghan. 1992. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics* 30, 1 (1992), 543–574.

Joseph J Monaghan. 2002. SPH compressible turbulence. *Monthly Notices of the Royal Astronomical Society* 335, 3 (2002), 843–852.

Joseph J Monaghan. 2005. Smoothed particle hydrodynamics. *Reports on progress in physics* 68, 8 (2005), 1703.

Jens Orthmann and Andreas Kolb. 2012. Temporal blending for adaptive SPH. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 2436–2449.

Daniel J Price. 2012. Smoothed particle hydrodynamics and magnetohydrodynamics. *J. Comput. Phys.* 231, 3 (2012), 759–794.

Stefan Reinhardt, Markus Huber, Bernhard Eberhardt, and Daniel Weiskopf. 2017. Fully asynchronous SPH simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 1–10.

Barbara Solenthaler and Markus Gross. 2011. Two-scale particle simulation. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 81.

Barbara Solenthaler and Renato Pajarola. 2009. Predictive-corrective incompressible SPH. In *ACM transactions on graphics (TOG)*, Vol. 28. ACM, 40.

Renato Vacondio, Benedict Rogers, and Peter K. Stansby. 2012. Accurate particle splitting for smoothed particle hydrodynamics in shallow water with shock capturing. *International Journal for Numerical Methods in Fluids* 69, 8 (2012), 1377–1410.

Renato Vacondio, Benedict Rogers, Peter K. Stansby, and Paolo Mignosa. 2016. Variable resolution for SPH in three dimensions: Towards optimal splitting and coalescing for dynamic adaptivity. *Computer Methods in Applied Mechanics and Engineering* 300 (2016), 442–460.

Renato Vacondio, Benedict Rogers, Peter K. Stansby, Paolo Mignosa, and Jonathan A. Feldman. 2013. Variable resolution for SPH: a dynamic particle coalescing and splitting scheme. *Computer Methods in Applied Mechanics and Engineering* 256 (2013), 132–148.

Rene Winchenbach. 2019. *openMaelstrom*. http://www.cg.informatik.uni-siegen.de/openMaelstrom Accessed: 2020-08-04.

Rene Winchenbach, Rustam Akhunov, and Andreas Kolb. 2020. Semi-Analytic Boundary Handling Below Particle Resolution for Smoothed Particle Hydrodynamics. In *Proceedings of ACM SIGGRAPH Asia 2020*. 173:1–173:17. https://doi.org/10.1145/3414685.3417829

Rene Winchenbach, Hendrik Hochstetter, and Andreas Kolb. 2016. Constrained Neighbor Lists for SPH-Based Fluid Simulations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Zurich, Switzerland) *(SCA '16)*. Eurographics Association, 49–56.

Rene Winchenbach, Hendrik Hochstetter, and Andreas Kolb. 2017. Infinite Continuous Adaptivity for Incompressible SPH. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 102:1–102:10.

Rene Winchenbach and Andreas Kolb. 2019. Multi-Level-Memory Structures for Adaptive SPH Simulations. In *Vision, Modeling and Visualization*. The Eurographics Association. https://doi.org/10.2312/vmv.20191323

Jihun Yu and Greg Turk. 2013. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Transactions on Graphics (TOG)* 32, 1 (2013), 5.