SPH Fluids in Computer Graphics

Markus Ihmsen[1], Jens Orthmann[2], Barbara Solenthaler[3], Andreas Kolb[2], Matthias Teschner[1]

[1]University of Freiburg
[2]University of Siegen
[3]ETH Zurich

Abstract

Smoothed Particle Hydrodynamics (SPH) has been established as one of the major concepts for fluid animation in computer graphics. While SPH initially gained popularity for interactive free-surface scenarios, it has emerged to be a fully fledged technique for state-of-the-art fluid animation with versatile effects. Nowadays, complex scenes with millions of sampling points, one- and two-way coupled rigid and elastic solids, multiple phases and additional features such as foam or air bubbles can be computed at reasonable expense. This state-of-the-art report summarizes SPH research within the graphics community.

Keywords: Physically-based animation, fluid animation, Smoothed Particle Hydrodynamics

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

This section starts with a compact overview of SPH fluids. Sec. 1.1 introduces the underlying equations, in particular the momentum equation. Sec. 1.2 explains how to use SPH for the interpolation of fluid quantities and for the approximation of spatial derivatives in the momentum equation. Finally, Sec. 1.3 presents a first simple SPH fluid solver and introduces its components.

The remainder of this state-of-the-art report discusses various solver components in detail. Sec. 2 explains approaches to estimate the neighborhood of a particle. Although recent research in that direction focuses on uniform grids, various realizations based on different concepts and considering different architectures such as GPUs are presented. Sec. 3 discusses various ways to compute pressure forces, either using a state equation (EOS), EOS-based iterative refinement, or pressure projection. Benefits, drawbacks, and issues with performance comparisons are discussed. Sec 4 discusses methods for boundary handling with a focus on boundaries that are represented with particles. Sec. 5 discusses variants that employ adaptive time or space discretization to optimize performance, while Sec. 6 discusses multiphase fluids. Sec. 7 discusses various techniques to reconstruct interface representations from particles and specific rendering approaches. Sec. 8 presents approaches to efficiently add small-scale detail, e.g., foam, spray, and tiny air bubbles, to pre-computed SPH simulations in order to enhance the visual quality. Finally, Sec. 9 presents a few aspects that could be addressed in future research.

1.1. Governing Equations

We consider a fluid that consists of a set of small moving fluid elements, i.e., particles. Each particle i has a mass m_i and carries attributes such as density ρ_i, pressure p_i or volume V_i. Over time t, particle positions $\mathbf{x}_i$ and the respective attributes are advected with the local fluid velocity $\mathbf{v}_i$:

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i. \tag{1}$$

As the particles move with the fluid flow, the time rate of change of the velocity $\mathbf{v}_i$ is governed by the Lagrange form of the Navier-Stokes equation (see Appendix A for a comparison of Lagrange and Euler formulations):

$$\frac{d\mathbf{v}_i}{dt} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i}. \tag{2}$$

The term $-\frac{1}{\rho_i}\nabla p_i$ describes the particle acceleration due to pressure differences in the fluid. The respective pressure

force generally dominates all forces. It is responsible to preserve the volume of the fluid. As the particle mass is constant, preserving the fluid volume corresponds to preserving the density. Small and preferably constant density deviations are important for high-quality simulations. Otherwise, a perceivable and disturbing bouncing of the free surface occurs. The term $\nu\nabla^2\mathbf{v}_i$ represents the acceleration due to friction forces between particles with different velocities. Although the kinematic viscosity ν is known, e.g., $\nu \approx 10^{-6}\mathrm{m}^2\cdot\mathrm{s}^{-1}$ for water, larger user-defined values are typically preferred to improve the stability of SPH simulations. Viscosity is also realized with alternative methods such as artificial viscosity [Mon92] or XSPH [Mon89]. The term $\frac{\mathbf{F}_i^{other}}{m_i}$ describes other accelerations such as gravity.

1.2. SPH

The SPH concept is used to interpolate fluid quantities at arbitrary positions and to approximate the spatial derivatives in Eq. (2) with a finite number of sample positions, i.e., adjacent particles.

Interpolation: A quantity A_i at an arbitrary position $\mathbf{x}_i$ is approximately computed with a set of known quantities A_j at neighboring particle positions $\mathbf{x}_j$:

$$A_i = \sum_j \frac{m_j}{\rho_j} A_j W_{ij} \tag{3}$$

with W_{ij} being a kernel function of the form

$$W_{ij} = W\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{h}\right) = W(q) = \frac{1}{h^d} f(q) \tag{4}$$

where d indicates the number of dimensions and h is the so-called smoothing length. Kernel functions should be close to a Gaussian [Mon92], but with a compact support that typically ranges from h for the bell-shaped function [Luc77] to $3h$ for the quintic spline function [Mor96]. The number of adjacent particles that are considered in the summation of Eq. (3) depends on the dimensionality d, the support of the kernel function and the particle spacing which is typically close to h [Mor96, Mon05]. The choice of the kernel function, the number and the disorder of considered particles influence the accuracy of the summation in Eq. (3). A typical function for the kernel shown in Eq. (4) in 3D would be the cubic spline [Mon92]:

$$f(q) = \frac{3}{2\pi}\begin{cases} \frac{2}{3} - q^2 + \frac{1}{2}q^3 & 0 \le q < 1 \\ \frac{1}{6}(2-q)^3 & 1 \le q < 2 \\ 0 & q \ge 2 \end{cases}. \tag{5}$$

There is, however, no consensus on the optimal kernel with respect to the trade-off between accuracy and computational cost. E.g., [APKG07, OK12, OHB*13] use a set of kernels proposed in [DC96, MCG03, MSKG05]. [ZYF10] uses a quintic spline [Mor96]. [BT07, AIS*12, SB12] use the cubic spline (Eq. (5)).

Spatial derivatives: Spatial derivatives can be computed in various ways. In order to address issues of the original formulations $\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla W_{ij}$ and $\nabla^2 A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla^2 W_{ij}$, various alternatives have been investigated and currently, the following approximations are preferred [Mon92, MFZ97]:

$$\nabla A_i = \rho_i \sum_j m_j \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2}\right) \nabla W_{ij}, \tag{6}$$

$$\nabla \cdot \mathbf{A}_i = -\frac{1}{\rho_i} \sum_j m_j \mathbf{A}_{ij} \cdot \nabla W_{ij}, \tag{7}$$

$$\nabla^2 A_i = 2 \sum_j \frac{m_j}{\rho_j} A_{ij} \frac{\mathbf{x}_{ij} \cdot \nabla W_{ij}}{\mathbf{x}_{ij} \cdot \mathbf{x}_{ij} + 0.01h^2}, \tag{8}$$

with $A_{ij} = A_i - A_j$, $\mathbf{A}_{ij} = \mathbf{A}_i - \mathbf{A}_j$, $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ and $\nabla W_{ij} = \left(\frac{\partial W_{ij}}{\partial x_{i,x}}, \frac{\partial W_{ij}}{\partial x_{i,y}}, \frac{\partial W_{ij}}{\partial x_{i,z}}\right)^T$. Eq. (6) and Eq. (8) can be used in the computation of particle accelerations in Eq. (2). Eq. (7) can, e.g., be used to predict density changes from the divergence of the velocity field based on the continuity equation [ICS*13]. While pressure forces are preferably computed with the formulation in Eq. (6), e.g. [BT07, SP09b, RWT11], there exist various alternative forms, e.g. [MCG03, APKG07, LD09].

1.3. Concept of an SPH-based Fluid Solver

The basic building blocks of SPH-based fluid solvers are: neighborhood search, pressure computation and time integration. The neighborhood search is typically accelerated by a spatial access structure, e.g. a uniform grid [THM*03, LD08, GSSP10, IABT11], with a cell size that is equal to the kernel support, e.g., $2h$ for the kernel in Eq. (5). Details are discussed in Sec. 2.

In order to compute the pressure gradient in Eq. (2), pressure p_i is computed from the density ρ_i. While Sec. 3 discusses a variety of alternatives, state equations are a simple and popular choice and, in this context, the formulation

$$p_i = k\left(\left(\frac{\rho_i}{\rho_0}\right)^7 - 1\right) \tag{9}$$

seems to be preferred [Mon94, BT07, LD09, YT10, RWT11, Mon12, YT13]. The value ρ_0 is the desired rest density of the fluid, k is a stiffness constant that scales the pressure and, thus, the pressure gradient and the respective pressure forces. In practice, a larger stiffness constant reduces the compressibility of the fluid, but demands smaller integration time steps.

Alg. 1 shows an example of a simple SPH simulation step. As the algorithm employs a state equation, it can be referred to as state equation SPH (SESPH) [ICS*13]. In an implementation, the kernel function W_{ij} has to be specified, e.g., cubic with a support of $2h$ (see Eq. (4) and Eq. (5)). The particle mass m_i has to be specified, e.g., $m_i = h^3\rho_0$ as

Algorithm 1 SPH with state equation.

for all *particle i* **do**
  find neighbors j
for all *particle i* **do**
  $\rho_i = \sum_j m_j W_{ij}$
  compute p_i using ρ_i (e.g. Eq. (9))
for all *particle i* **do**
  $\mathbf{F}_i^{pressure} = -\frac{m_i}{\rho_i}\nabla p_i$ (e.g. Eq. (6))
  $\mathbf{F}_i^{viscosity} = m_i \nu \nabla^2 \mathbf{v}_i$ (e.g. Eq. (8))
  $\mathbf{F}_i^{other} = m_i \mathbf{g}$
  $\mathbf{F}_i(t) = \mathbf{F}_i^{pressure} + \mathbf{F}_i^{viscosity} + \mathbf{F}_i^{other}$
for all *particle i* **do**
  $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \Delta t \mathbf{F}_i(t)/m_i$
  $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t+\Delta t)$

in [ICS*13] or $m_i = (\frac{2}{3}h)^3\rho_0$ as in [SB12] resulting in different numbers of neighbors that contribute to the SPH sums of a particle.

In terms of numerical integration schemes, the semi-implicit Euler, also referred to as symplectic Euler or Euler-Cromer, is regularly used, e.g. [IAAT12, SB12, ICS*13]. The time step Δt is governed by the Courant-Friedrich-Levy (CFL) condition, e.g., $\Delta t \leq \lambda \frac{h}{\|\mathbf{v}^{max}\|}$ with $\lambda \approx 0.4$ [Mon92], h being the particle diameter and $\mathbf{v}^{max}$ being the maximum velocity of all particles. For $\lambda = 1$, this constraint states that all particles move less than the particle diameter per time step. A detailed discussion of further aspects that can be incorporated into the time step analysis can be found in [IAGT10]. Boundary handling forces as discussed in Sec. 4 are not considered in Alg. 1.

2. Neighborhood Search

SPH requires the computation of sums over dynamically changing sets of neighboring particles. The search of these neighborhood sets is generally accelerated by spatial data structures that should be efficiently generated and queried, preferably in a parallelizable way. While the neighborhood search has similarities with, e.g., collision detection or intersection tests in raytracing, it is additionally characterized by the fact that more than one space cell has to be queried to find the neighbors of a particle, i.e., cells adjacent to the cell of a particle have to be accessed. The data structure should also be efficient for arbitrary, sparsely filled simulation domains with a non-uniform particle distribution.

Although hierarchical data structures such as kd-trees are used in multi-resolution scenarios with a variable kernel support [KAD*06, Kei06, APKG07, SBH09], there seems to be consensus to employ uniform grids for standard SPH with a fixed kernel support. Grids are built in $O(n)$ and particles are accessed in $O(1)$, while hierarchical data structures are typically built in $O(n \log n)$ and accessed in $O(\log n)$ [HKK07b]. Therefore, this section focuses on uniform grids.

In addition to the implementation of the grid, there are further choices that influence the efficiency of an SPH solver. E.g, it is not obvious whether to store the neighborhood set for reuse as, e.g., in [IABT11], or not as, e.g., in [HKK07b, ZSP08, GSSP10]. This mainly depends on the number of neighborhood queries per simulation step. While this number is low for non-iterative SPH solvers (see Sec. 3.1 and Sec. 3.2), it can be large for iterative solvers (see Sec. 3.3 and Sec. 3.4). Another performance issue is the frequent recomputation of neighborhood sets. This issue can be alleviated by Verlet lists [Ver67, Hie07, PH10], where a set of potential neighbors is computed within a distance that is larger than the actual kernel support. Actual neighbors are computed from the set of potential neighbors which is updated only every n-th simulation step depending on the ratio between kernel support and query distance. However, Verlet lists are prohibitively memory-intensive for complex scenarios and slower than the strategies presented in the following.

The following discussion is limited to various implementations of uniform grids in the context of SPH applications with uniform kernel support.

2.1. Uniform Grid

Space is subdivided into cubic cells and each particle is associated to one cell in the construction stage. To find all relevant neighbors, the cell of a particle and all adjacent cells are queried. If the cell size is equal to the kernel support, 3^3 cells have to be queried in $3D$, which is optimal according to [IABT11]. The number of particles associated with a cell and the number of neighboring particles depend on the initial particle distance, e.g., approximately 8 and 40, respectively, for a kernel support twice as large as the initial particle distance. While the query step is easy to parallelize, grid construction is not due to potential write conflicts (race conditions).

2.1.1. Index Sort

In order to avoid race conditions in the parallel construction of uniform grids, particles can be sorted with respect to a key that is unambiguously assigned to each cell [PDC*03, Gre08, KS09]. Instead of storing references to all particles, a cell only stores one reference to it's first particle in the sorted array. Parallel reduction is commonly employed to compute these references [KS09]. By sorting particles according to their spatial cell, particles in the same cell are close in memory which improves memory coherence. However, particles in neighboring cells are not necessarily close in memory.

2.1.2. Z-index Sort

Efficient algorithms have to enforce low memory transfers such that threads can perform the operations with almost

no latency. The transfer rate decreases for higher cache-hit rates, i.e., the percentage of accesses where the requested data is already present in the cache. The cache-hit rate of any SPH implementation can be optimized by mapping the spatial locality of particles onto memory. This can be achieved by employing a space-filling Z-curve for computing cell indices [GSSP10, IABT11].

Rather than sorting an n-dimensional space one dimension after another, a Z-curve orders the space by n-dimensional blocks of 2^n cells. This ordering preserves spatial locality due to the self-containing (recursive) block structure. Consequently, it leads to a high cache-hit rate while indices can be computed fast by bit-interleaving [PF01]. As shown in [IABT11], the Z-curve increases the cache-hit rate and, thus, improves the performance for the query and processing of particle neighbors.

Index sort variants are considered to be one of the fastest spatial acceleration methods. However, in these schemes, the memory consumption scales with the simulation domain [Gre08]. In order to represent infinite domains with low memory consumptions, spatial hashing can be employed.

2.1.3. Hashing

In spatial hashing [THM*03], the effectively infinite domain is mapped to a finite list. The hash function that maps a position $\mathbf{x} = (x, y, z)$ to a hash table of size m has the following form:

$$c = \left[\left(\left\lfloor \frac{x}{d} \right\rfloor \cdot p_1 \right) \text{xor} \left(\left\lfloor \frac{y}{d} \right\rfloor \cdot p_2 \right) \text{xor} \left(\left\lfloor \frac{z}{d} \right\rfloor \cdot p_3 \right) \right] \% m, \tag{10}$$

with p_1, p_2, p_3 being large prime numbers that are chosen as 73856093, 19349663 and 83492791 [THM*03], respectively. Different spatial cells can be mapped to the same hash cell (hash collision), slowing down the neighborhood query. The number of hash collisions can be reduced by increasing the size of the hash table, trading memory for speed. [THM*03] suggests to reserve memory for a certain number k of entries in all hash cells on initialization in order to avoid frequent memory allocations. However, for SPH fluids, the hash table is generally sparsely filled. Thus, a significant amount of memory is unnecessarily pre-allocated. Furthermore, cells that are close in memory are necessarily not close in space which reduces the cache-hit rate for the neighborhood query. These issues are addressed by the compact hashing method.

2.1.4. Compact Hashing

Compact hashing [IABT11] uses a secondary data structure which stores a compact list of non-empty (used) cells. Hash cells just store a handle to their used cell. Memory for a used cell is allocated if it contains particles and deallocated if the cell gets empty. Thus, constant memory is consumed for the hash table and additional memory for the list of used cells. Thereby, the memory consumption scales with the number of particles and not with the simulation domain.

Generally, the hash function is designed to abolish spatial locality in order to minimize the number of hash collisions. As this would result in an increased memory transfer and longer query times compared to index sort methods, [IABT11] proposes to reorder the particles and the compact list of used cells according to a Z-curve. As particles show a high temporal locality, sorting might not be required in each simulation step. Depending on the performance of the sorting algorithm and the number of particles that move between cells, reordering can be performed in each [DRF12] to every 100th [IABT11] simulation step in order to yield the optimal performance. Interestingly, when reordering is performed in each step, the compact list just needs to store a reference to the first particle in the sorted array and the number of entries, instead of references to all particles in the cell. This further reduces the memory consumption of the compact list.

2.1.5. GPUs

Since there are almost no data dependencies, SPH methods generally map well to the streaming architecture of today's Graphic Processing Units (GPUs) [HKK07b, Gre08, ZSP08, YWH*09, GSSP10, OK12, MM13]. On GPUs, neighborhood computations can be carried out via two dual mapping operations: either by scattering particle contributions onto points of evaluation, or by gathering of particle data at sampling positions.

Scattering operations are efficiently realized in combination with the rasterization pipeline of programmable graphics hardware [KLRS04, KSW04, AIY*04, KC05, HCM06, HKK07a, HKK07b, ZSP08]. Field functions are mapped to fragment shaders which blend particle contributions into several texture slices using render-to-texture mechanisms [KC05]. The absence of acceleration structures and neighborhood queries makes scattering very attractive for interactive rendering of SPH data [vdLGS09, FGE10, FAW10].

In contrast, gathering operations better exploit parallelism in combination with generic programming APIs such as CUDA or OpenCL, which is advantageous for simulation [Gre08, GSSP10, OK12, MM13] and surface reconstruction [AIAT12, AAIT12]. However, a vital criterion for the efficiency of gathering approaches is thread coherence including coherent memory access between concurrent threads. Thus, neighborhood search is in general realized with an index sort [Gre08] or Z-index sort [GSSP10, ZGHG10], employing a data-parallel radix sort [SHG09] and stream compaction [SHZO07] as main building blocks. Furthermore, it is advantageous to avoid data transfer between CPU and GPU, favoring fully GPU-based systems which due to memory limits trade particle numbers for computation speed. However, in combination with domain decomposition techniques, multi-GPU solvers [VBDRC12, ZSL*13, RBH*13] can be utilized in order to increase the particle resolution.

3. Incompressibility

Enforcing incompressibility is essential for realistic SPH fluid simulations. While oscillations of the free surface due to compressibility are less prominent in small scenarios, oscillations become more significant for larger scenes with, e.g., millions of particles [ICS*13].

The computation of particle states with low compressibility, usually between 0.1% and 1% [MM97, SP09b, ICS*13], is one of the most expensive steps in SPH. Either the computation per time step is efficient, albeit at the expense of a small time step, or the computation is expensive for a larger time step.

This survey classifies all discussed incompressibility approaches into four classes. First, non-iterative approaches are discussed that employ an equation of state (EOS). Second, EOS solvers with splitting are outlined. Third, iterative EOS solvers are explained. Fourth, solvers based on a pressure Poisson equation are outlined.

3.1. Non-iterative EOS Solvers

State equations are used in the context of Alg. 1 to compute pressure from density. For example, $p_i = c_s^2 \rho_i$ has been proposed in [MM97]. Here, c_s is often referred to as the speed of sound. This is motivated by the fact that fluids with a Mach number greater than 0.3 are of considerable compressibility. As a given constant c_s results in a certain compressibility, the respective simulation does not represent a real fluid, but an artificial one with a reduced sound speed [Mon94]. In practice, c_s is a stiffness constant that scales the pressure and, thus, the pressure gradient and the respective pressure forces. A larger value reduces the compressibility of the fluid, but also limits the time step. Therefore, the stiffness constants for different variants of the state equation are all denoted with k in the following.

Alternatively to [MM97], pressure is often related to $\rho_i - \rho_0$, the deviation of the actual density to the rest density, i.e., the density error. Here, slightly varying forms are employed, e.g., $p_i = k(\rho_i - \rho_0)$ [DC96, MCG03] or $p_i = k(\rho_i/\rho_0 - 1)$ [APKG07]. However, as already discussed in Sec. 1, the formulation $p_i = k((\rho_i/\rho_0)^7 - 1)$ is widely used [Mon94, BT07, LD09, YT10, RWT11, YT13]. [ZYF10] proposes $p_i = k((\rho_i/\rho_0)^2 - 1)$.

Differences in terms of stability and performance between the various EOS formulations are rarely analyzed in the literature. Generally, the incorporation of any of the discussed state equations seems to be less efficient compared to iterative EOS solvers (Sec. 3.3) and pressure projection (Sec. 3.4). Nevertheless, it would be interesting to see the performance of a state equation in combination with a non-iterative splitting approach as given in Alg. 2.

3.2. Non-iterative EOS Solvers with Splitting

As an interesting alternative to Alg. 1, the pressure could be computed with the density that is obtained after advecting the particles without pressure forces. This concept is known as splitting, e.g. [Cho68, Bri08], and the basis of various iterative solvers, e.g. [PTB*03, SP09b, SBH09, HLWW12, ICS*13, MM13]. Therefore, accelerations in Eq. (2) are split, i.e., considered in two different steps. First, an intermediate velocity $\mathbf{v}^*$ is computed from all non-pressure accelerations: $(\mathbf{v}^* - \mathbf{v}(t))/\Delta t = \nu\nabla^2\mathbf{v}_i(t) + \mathbf{F}_i^{other}(t)/m_i$. Then, the acceleration from the pressure gradient is computed for the resulting velocity: $(\mathbf{v}(t+\Delta t) - \mathbf{v}^*)/\Delta t = -(1/\rho_i^*)\nabla p_i$. Pressure p_i is computed using ρ_i^*, while ρ_i^* is computed after advecting the particles with $\mathbf{v}_i^*$. The respective pressure forces are intended to project the intermediate velocity $\mathbf{v}^*$ onto a divergence-free velocity field, i.e., to minimize density deviations. Alg. 2 shows a simple implementation of this concept. While non-pressure forces compete with pressure forces in Alg. 1, Alg. 2 considers the effect of non-pressure forces in the computation of pressure forces. Alg. 1 and Alg. 2 have not been compared in the literature yet. However, since iterative SPH solvers based on splitting generally outperform Alg. 1, e.g. [SP09b, HLWW12], simple non-iterative splitting as in Alg. 2 is certainly a promising concept.

Algorithm 2 SPH with state equation and splitting.

for all *particle i* **do**
  find neighbors j
for all *particle i* **do**
  $\mathbf{F}_i^{viscosiy} = m_i\nu\nabla^2\mathbf{v}_i$ (e.g. Eq. (8))
  $\mathbf{F}_i^{other} = m_i\mathbf{g}$
  $\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t\frac{\mathbf{F}_i^{viscosity} + \mathbf{F}_i^{other}}{m_i}$
for all *particle i* **do**
  $\rho_i^* = \sum_j m_j W_{ij} + \Delta t\sum_j(\mathbf{v}_i^* - \mathbf{v}_j^*)\cdot\nabla W_{ij}$
  compute p_i using ρ_i^* (e.g. Eq. (9))
for all *particle i* **do**
  $\mathbf{F}_i^{pressure} = -\frac{m_i}{\rho_i^*}\nabla p_i$ (e.g. Eq. (6))
for all *particle i* **do**
  $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^* + \Delta t\mathbf{F}_i^{pressure}/m_i$
  $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t\mathbf{v}_i(t+\Delta t)$

3.3. Iterative EOS Solvers with Splitting

This concept is also based on the splitting concept illustrated in Alg. 2. Intermediate velocities and positions are predicted using all non-pressure forces. To minimize density errors at the intermediate state, pressure forces are calculated. As an extension to Alg. 2, these pressure forces are iteratively refined. In each iteration, the pressure forces lead to updated intermediate positions and velocities with a new density error that is considered in the force computation of the following iteration. If the density error ρ_{err} - either the average

or the maximum - is below a threshold η, the algorithm terminates. In contrast to non-iterative EOS solvers in Sec. 3.1 and Sec. 3.2, the stiffness constant in the state equation is generally computed and not user-defined. Instead, iterative EOS solvers are parameterized by a more appropriate and intuitive density error η. Smaller values for η result in more iterations, larger values are more efficient as less iterations are required.

Alg. 3 illustrates the concept. The notation is closely aligned with the iterative local Poisson approach (LPSPH) in [HLWW12], but can be mapped to other iterative EOS solvers, e.g. [SP09b] as well. While Alg. 3 is not an optimal implementation, it is a good illustration of the concept.

Algorithm 3 Iterative EOS solver with splitting.

for all *particle i* **do**
  find neighbors j
for all *particle i* **do**
  $\mathbf{F}_i^{viscosity} = m_i \nu \nabla^2 \mathbf{v}_i$ (e.g. Eq. (8))
  $\mathbf{F}_i^{other} = m_i \mathbf{g}$
  $\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_i^{viscosity} + \mathbf{F}_i^{other}}{m_i}$
  $\mathbf{x}_i^* = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i^*$
repeat
  for all *particle i* **do**
    compute ρ_i^* using $\mathbf{x}_i^*$
    compute p_i using ρ_i^*, e.g. $p_i = k(\rho_i^* - \rho_0)$
  compute ρ_{err}
  for all *particle i* **do**
    $\mathbf{F}_i^{pressure} = -\frac{m_i}{\rho_i^*} \nabla p_i$ (e.g. Eq. (6))
    $\mathbf{v}_i^* = \mathbf{v}_i^* + \Delta t \frac{\mathbf{F}_i^{pressure}}{m_i}$
    $\mathbf{x}_i^* = \mathbf{x}_i^* + \Delta t^2 \frac{\mathbf{F}_i^{pressure}}{m_i}$
until $\rho_{err} < \eta$
for all *particle i* **do**
  $\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i^*$
  $\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i^*$

In addition to [HLWW12], other approaches implement this concept to enforce incompressibility with interesting variations. In predictive-corrective SPH (PCISPH) [SP09b], pressure is accumulated instead of pressure forces. [HLWW12] recomputes particle neighbors in each iteration, [SP09b] does not. Also, different stiffness constants are used. PCISPH uses

$$k = \frac{2\rho_0^2}{m_i^2 \cdot \Delta t^2 \sum_j \nabla W_{ij}^0 \cdot \sum_j \nabla W_{ij}^0 + \sum_j (\nabla W_{ij}^0 \cdot \nabla W_{ij}^0)} \quad (11)$$

with W_{ij}^0 being kernel values for a prototype particle. So, k can be mainly precomputed and only changes with the time step. In contrast, LPSPH uses

$$k = \frac{\rho_i^* r_i^2}{2\rho_0 \Delta t^2} \quad (12)$$

with initial particle distance $2r_i$ which is derived from the pressure Poisson equation [HLWW12].

In each iteration, PCISPH and LPSPH compute intermediate positions. This is also done in position-based fluids (PBF) [MM13] and to some extent in [CBP05]. Following the splitting concept, PBF uses non-pressure forces to predict intermediate positions and velocities. Then, positions are iteratively refined to enforce incompressibility. Therefore, the state equation $p_i = k(\rho_i/\rho_0 - 1)$ with $k = 1$ is employed. Instead of accumulating pressure or pressure forces, PBF accumulates distances in each iteration with

$$\Delta \mathbf{x}_i = -\frac{1}{\rho_0} \sum_j \left(\frac{p_i}{\beta_i} + \frac{p_j}{\beta_j} \right) \nabla W_{ij}, \quad (13)$$

where $\Delta \mathbf{x}_i$ is the position change per iteration. β_i and β_j are precomputed constants. The position update in Eq. (13) is closely related to an SPH pressure force. However, PBF [MM13] avoids accumulating pressure or pressure forces that eventually update the velocity and the position.

Another closely related approach is presented in [BLS12]. Splitting is employed and then, velocities and positions are iteratively updated. The same state equation as in [MM13] is employed. In contrast to PBF [MM13] and similar to PCISPH [SP09b], pressure is accumulated. From an SPH perspective, [BLS12] is difficult to read. Density is computed with SPH using the regular kernel function W_{ij}. The kernel gradient, however, is denoted as $f_{ij}\hat{\mathbf{r}}_{ij}$. The approach solves for Lagrange multipliers λ_i that are related to the negative pressure as can be seen, e.g., from the formulation of the constraint force: $\mathbf{f}_i = \sum_j (m_i \lambda_i + m_j \lambda_j) f_{ij} \hat{\mathbf{r}}_{ij}$.

Performance: The performance of iterative EOS solvers is commonly characterized by the maximum possible time step and the required number of iterations for a specified density error. [SP09b] shows that PCISPH allows for time steps that are up to two orders of magnitude larger than in a non-iterative EOS solver [BT07]. The average number of iterations is between three and five for density errors of 0.1% and 1%, resulting in an overall speedup factor of fifty compared to [BT07]. [HLWW12] presents a slightly improved performance with a speedup of 1.5 compared to PCISPH. PBF tolerates significantly larger time steps than PCISPH, but requires more iterations, resulting in a similar overall performance. In [BLS12], speedup factors of 25 to 250 are reported compared to non-iterative EOS solvers. Typically, 5 to 15 iterations are employed, while a density error of 1% is obtained with 100 iterations. However, as discussed in [ICS*13, MM13] a thorough performance analysis of iterative EOS solvers is rather involved. This is due to the fact that these solvers do not necessarily reach their optimal performance for the largest time step. On one hand, the number of iterations grows with the time step. On the other hand, the neighborhood search has to be performed per simulation step.

Discussion: Non-iterative EOS solvers use different state

equations, in particular they use different stiffness constants. They also differ in terms of the quantity that is accumulated. Pressure, pressure forces or distances are accumulated to compute final particle positions. It would be interesting to analyze these aspects: Which stiffness constant is optimal? Which quantity should be updated? Is it an option to analyze individual stiffness constants per particle?

3.4. Pressure Projection

As an alternative to state equations, pressure can also be computed by solving a pressure Poisson equation (PPE). Following the splitting concept, intermediate velocities $\mathbf{v}_i^*$ are predicted by applying all non-pressure forces. Then, pressure p_i is computed from a discretized PPE in order to correct (project) the intermediate velocities to a divergence-free state $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^* - (1/\rho_i^*)\nabla p_i$. This technique is commonly applied in grid-based approaches, e.g. [FF01, Bri08, CM11]. In SPH approaches, the pressure Poisson equation is used with different source terms, either using the divergence of the intermediate velocity field $\mathbf{v}_i^*$, e.g. [CR99, PTB*03], as

$$\nabla^2 p_i = \frac{\rho_0}{\Delta t}\nabla\cdot\mathbf{v}_i^* \tag{14}$$

or the compression $\rho_0 - \rho_i^*$ after advecting the particles with $\mathbf{v}_i^*$, e.g. [SL03, KGS09], as

$$\nabla^2 p_i = \frac{\rho_0 - \rho_i^*}{\Delta t^2}. \tag{15}$$

Some authors propose combinations of both source terms [HA07, LTKF08]. The density can also be replaced by the number density $\delta_i = \sum_j W_{ij}$ [PTB*03, HA07, LTKF08]. The concept of pressure projection, referred to as incompressible SPH (ISPH), is illustrated in Alg. 4. It is very similar to Alg. 2, but instead of computing pressure per particle with the state equation, pressure is computed by solving the linear system that is described with the PPE, Eq. (14) or Eq. (15).

Recently, an alternative ISPH solver has been presented by [ICS*13]. The approach is referred to as implicit incompressible SPH (IISPH). Although solving a linear system seems to be expensive at first sight, IISPH outperforms PCISPH [SP09b] and a standard ISPH variant [SL03]. IISPH employs the density invariance condition as source term in Eq. (15). Further, IISPH combines a discretized form of the continuity equation and an SPH form of the pressure force to a discretized form of the PPE. The employed form of the pressure force is equal to the pressure force that is used in the velocity update. This concept is different to approaches that either directly discretize the Laplace operator, e.g. [CR99, SL03, HA07, KGS09], or to approaches that use a background grid for the discretization [LTKF08, YLHB09, RWT11].

PPEs can be solved in various ways, e.g., using successive over-relaxation (SOR) [FM96], conjugate gradient [CR99, FF01] or multigrid techniques [CM11]. In IISPH, relaxed Jacobi is employed to iteratively compute the pressure field. [ICS*13] shows that the solver can be implemented in a very efficient, matrix-free way. Only seven scalar values are stored per particle and only two particle loops are required per iteration. Further, comparatively few iterations are sufficient to obtain small density deviations of down to 0.01%. Compared to PCISPH, a speedup of up to six is presented for the computation of the pressure field.

Discussion: Although IISPH needs to solve a linear system, the actual implementation corresponds to iterative EOS solvers that accumulate pressure as in PCISPH. Basically, pressure is iteratively updated. In each iteration, PCISPH

Algorithm 4 SPH with pressure projection (Incompressible SPH).

for all *particle i* **do**
  find neighbors j
for all *particle i* **do**
  $\mathbf{F}_i^{viscosiy} = m_i \nu \nabla^2 \mathbf{v}_i$ (e.g. Eq. (8))
  $\mathbf{F}_i^{other} = m_i \mathbf{g}$
  $\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t \frac{\mathbf{F}_i^{viscosity} + \mathbf{F}_i^{other}}{m_i}$
for all *particle i* **do**
  $\rho_i^* = \sum_j m_j W_{ij} + \Delta t \sum_j (\mathbf{v}_i^* - \mathbf{v}_j^*)\cdot\nabla W_{ij}$
solve the PPE in Eq. (14) or Eq. (15)
for all *particle i* **do**
  $\mathbf{F}_i^{pressure} = -\frac{m_i}{\rho_i^*}\nabla p_i$ (e.g. Eq. (6))
for all *particle i* **do**
  $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^* + \Delta t \mathbf{F}_i^{pressure}/m_i$
  $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t+\Delta t)$

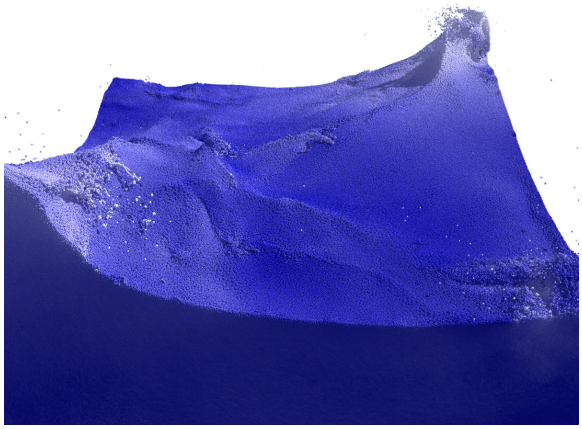

Figure 1: *A breaking dam simulated with 20 million SPH particles. The volume is preserved up to an error of 0.1%. Velocities are color-coded.*

scene	particles	neighborhood	pressure	IISPH iterations	Δt	spacing
breaking dam (Fig. 1)	20M	8.5s	21.5s	10	0.0034s	0.1m
fountain (Fig. 2)	17M	7.0s	10.0s	4	0.0008s	0.0125m
ships (Fig. 3)	19M	8.0s	24.0s	12	0.017s	0.5m

Table 1: *Measurements given for a single simulation step performed on a standard six-core desktop computer.*

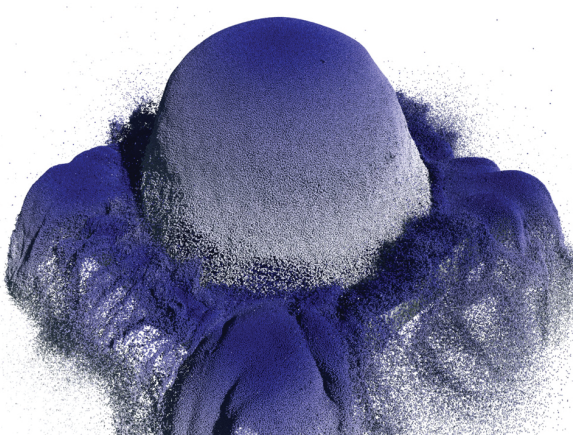

Figure 2: *A fountain simulated with 17 million SPH particles. Velocities are color-coded.*

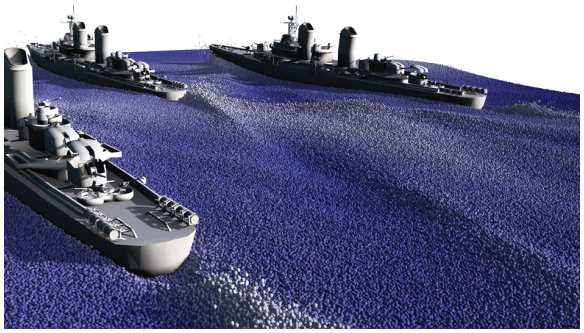

Figure 3: *Three ships sailing at a speed of 60 kilometers per hour. The 19 million SPH particles are color-coded according to velocity.*

loops three times over all particles, IISPH requires only two loops. The memory footprint of seven scalar values per particle in IISPH is negligible. In terms of the time step, [ICS*13] shows that the optimal time step does not necessarily correspond to the maximum possible time step in IISPH and PCISPH. [ICS*13] also illustrates the relation between time step and particle size. Time steps of up to 0.05s are presented for particles with a radius of 1m.

Figures 1, 2, 3 show example scenarios that have been computed with compact hashing [IABT11] and IISPH [ICS*13] on a standard six-core computer. Performance measurements are summarized in Table 1.

3.5. Performance Comparison

Existing publications indicate that iterative EOS solvers, e.g., PCISPH and PBF, are more efficient than non-iterative EOS solvers and that pressure projection, e.g., IISPH, is more efficient compared to iterative EOS solvers. As can be deduced from the detailed performance analysis in [ICS*13], a thorough comparison is rather complex. This is due to various reasons.

Criteria: There seems to be an agreement to measure the overall computation time for a scenario to account for the different characteristics of existing solvers. Non-iterative EOS solvers are fast per simulation step with rather small time steps, while iterative EOS solvers and pressure projection schemes are expensive per simulation step, but allow for large time steps. As the overall computation time of all solvers largely depends on the obtained incompressibility, average or maximum density errors are considered to specify the simulation quality.

Parameters: In non-iterative EOS solvers, the performance depends on one or more stiffness constants in the EOS. These stiffness constants govern the obtained density error, i.e. the simulation quality. Larger constants require smaller time steps, while the computation time per simulation step is constant. The desired density error can not be specified, but only tested. This is in contrast to iterative EOS solvers or pressure projection which are parameterized by a desired density error. Here, the computation time per simulation step grows for smaller specified density errors. On the other hand, larger time steps can be used. A performance comparison of non-iterative and iterative EOS solvers is only useful, if the density error in both approaches is comparable. Thus, the stiffness constant in one approach has to be mapped to the specified density error in the other approach.

Optimal Performance: As discussed in [ICS*13], SPH simulations with iterative EOS solvers or pressure projection schemes do not reach their maximum overall performance for the largest possible time step. I.e., there exists an optimal time step, where the combination of neighbor search and pressure computation reaches its best performance. While

this optimal time step is mentioned in [ICS*13], it has not been analyzed and it is not explained how to find it without testing.

Test scenarios: Depending on the test scenario, arbitrary performance differences can be obtained. E.g., for one SPH particle, a non-iterative EOS solver is the fastest solution, as iterative EOS solvers and pressure projection commonly perform a minimum number of iterations. Non-iterative EOS solvers can even be optimal for arbitrarily large scenes with millions of particles, if these particles form rather shallow water. Iterative EOS solvers and pressure projection, on the other hand, are more efficient for complex scenes. The term "complex" is rarely defined, but usually refers to large scenarios with a certain fluid depth, e.g. a breaking dam. It is more expensive to obtain a desired density error for growing fluid depths. Additionally, with growing fluid depth the tolerable density error decreases to avoid visible oscillations of the free surface, making the pressure computation even more expensive.

4. Boundary Handling

The interaction of the fluid with rigid boundaries requires special consideration in order to prevent penetration of objects. In most SPH implementations, solids are sampled with particles which exert forces on fluid particles. [Mon94, Mon05, MK09] compute distance-based penalty forces, e.g., Lennard-Jones forces which scale polynomially with the distance to the fluid particle. Although this approach has been adapted to realize the interaction of compressible SPH fluids with particle-sampled deformable meshes [MST*04, LAD08], it causes large pressure variations in the fluid which further restricts the time step for weakly compressible fluids. The main difficulty in penalty-based approaches is controlling the stiffness parameter which has to be balanced such that penetrations of rigid boundaries are avoided, while not causing pressures that are too high. Generally, these methods require small integration time steps to produce smooth pressure distributions.

In order to overcome the issues of penalty-based methods and to have more control on the boundary condition, direct forcing has been proposed in [BTT09]. In this method, one- and two-way coupling of rigid bodies and fluids are realized by computing control forces and velocities using a predictor-corrector scheme. Thereby, different slip-conditions can be modeled, while non-penetration is guaranteed. Compared to penalty-based methods larger time steps can be used.

A phenomenon that occurs in distance-based penalty schemes and in direct forcing is sticking of fluid particles to the solid boundary. This results from *particle deficiency* at the interface with the solid boundary. Here, the support domain is not sufficiently sampled and, thus, field variables cannot be well approximated with the SPH interpolation concept. Harada et al. addressed this problem for the distance-based approach [HKK07b] by employing a wall-weight function which based on the distance adds a precomputed contribution of the boundary to the fluid density. This significantly reduces the stacking of particles, but introduces irregular density distributions at the boundary as shown in [IAGT10].

In order to obtain smoother transitions at the solid interface, boundary particles should contribute to the reconstruction of field variables. Therefore, solid objects are either pre-sampled with particles, e.g., [KAD*06, SSP07, FG07, IAGT10, SB12, AIS*12], or sampled on the fly, e.g., [HA06]. Furthermore, different strategies have been proposed how boundary particles are taken into account. For example in [MM97, SSP07, IAGT10], boundary particles are treated like fluid particles which are advected by the velocity of the solid, i.e., for each boundary particle a unique density and pressure value is computed. Another strategy is to mirror quantities of neighboring fluid particles onto boundary particles, e.g., a boundary particle gets the same pressure as its neighboring fluid particle. Mirroring quantities has been successfully employed to simulate different slip conditions for straight [HA06] and curved [MM97, SB12, AIS*12] boundaries.

The sampling distance of boundaries influences the numerical stability and the quality of the simulation significantly. Simple objects like a cube can be equidistantly sampled with particles, but for complex shapes with convex and concave regions, irregular samplings cannot be prevented. Furthermore, sampling only the surface of solid objects with a single layer is desirable for performance reasons and for thin objects. These challenges have been addressed in [AIS*12], where inhomogeneous samplings are handled by dynamically computing the relative contribution of a boundary particle. This yields smooth reconstructions for densely sampled, complex boundaries, including dynamically changing contacts with multiple objects as illustrated in Fig. 4. Non-penetration is guaranteed even when sampling objects with a single particle layer, enabling plausible interactions with lower dimensional objects. Two-way coupling has been demonstrated even for large density ratios of up to 1000. As the approach purely builds on fluid quantities such as pressure or viscosity to handle collisions and friction, it can be easily integrated into any SPH implementation, e.g., IISPH as presented in [ICS*13]. An extension of [AIS*12] for realizing one- and two-way coupling with deformable solids and cloth has been proposed in [ACAT13].

As shown in [OHB*13], the two-way coupling concept can be employed to model transport, i.e., the exchange of quantities between rigid objects and fluids. Thereby, convection is extended by a robust surface transport.

5. Adaptivity

The resolution of a fluid has a large impact on the resulting visual quality. Surface complexity is increased and dissi-

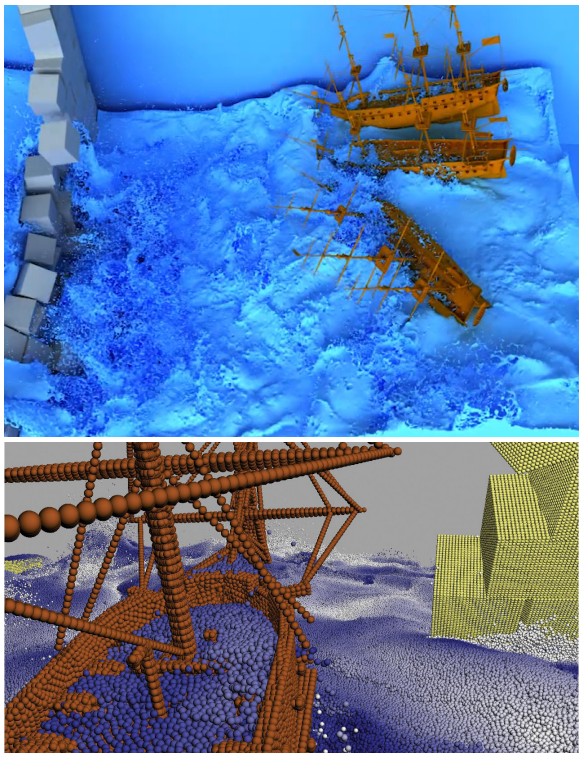

Figure 4: *Simulation of 20 million SPH particles. Two-way fluid-solid coupling is realized with [AIS*12].*

pation is reduced with increasing number of particles. Simulations in the order of tens of million particles are, however, very challenging to compute within a given time frame on a desktop computer. Therefore, adaptive sampling techniques have been explored that follow the idea to allocate resources to visually interesting regions only. Adaptive sampling mechanisms either adapt the spatial resolution or adapt the sampling distance in time. In the following, we will summarize both, space and time adaptive methods.

5.1. Adaptive Spatial Discretization

Homogeneous fluid regions are often discretized with unnecessarily large particle numbers. Space adaptive methods thus try to resample particles in order to more accurately resolve regions with larger flow activity employing non-uniform particle radii. In order not to introduce high pressure variations, sampling techniques must maintain particle regularity while staying computationally efficient. This survey classifies these level-of-detail approaches into two classes: dynamic particle refinement methods and multi-scale techniques.

5.1.1. Dynamic Particle Refinement

The idea of a particle refinement is to dynamically exchange particle sets, either globally [CPK02] or locally [FB07], in order to increase the resolution in regions of complex flow. To introduce as little sampling errors as possible, a local error functional must be minimized [FB07], which in general requires Lagrange multipliers [LB95]. However, in computer graphics, simpler but faster sampling operators are used [DC99, KW02, LQB05, KAD*06, APKG07, ZSP08, OK12]. In high resolution regions H, particles split to a fixed number of equally sized child particles, and vice versa merge to a single parent particle in low resolution regions L. Resolution levels directly contribute to each other yielding a consistent fluid representation.

However, non-uniform smoothing radii lead to difficulties in reproducing quantity fields at level boundaries [BOT01]. Sampling errors are reduced by indirect interaction between particle levels [KAD*06] or by slowly changing smoothing radii over several layers [APKG07]. Still, sampling operators do not consider contributions from neighboring particles. SPH differentials are quite sensitive to the resulting irregular particle structure [BT07]. Thus, in order to maintain integration time steps equal to non-adaptive fluid solvers, an error-bound temporal blending of resolution levels is advantageous [OK12]. An example for an adaptively sampled SPH fluid using smooth blending between particle levels is outlined in Alg. 5.

Algorithm 5 Dynamic refinement SPH.

```
while animating do
    compute physics for L and H
    blend quantities between transitioning particles
    update blend weights using an error estimation
    identify split/merge regions
    split particles in L/merge particles in H
```

To avoid temporal field discontinuities, new particles are smoothly blended in, while old particles are faded out using blend weights for all particles in transition. Blend weights are updated with respect to the introduced sampling error, leading to small transition regions as shown in Fig. 5.

5.1.2. Multi-scale Methods

As an alternative to dynamically merging and splitting particles, level-of-detail can be achieved by using multiple resolution levels which are simulated in separate but coupled simulations [SG11, HS13]. The particle radius is fixed in each level, but arbitrarily large differences of particle sizes can be used between levels. The basic idea is presented in [SG11] for two levels and is summarized in Alg. 6. The complete fluid is discretized and simulated in the coarse level L. A subset region of the fluid is simulated with higher resolution in the fine level H. The boundary condition for the subset in H is defined by boundary particles. They contribute

Algorithm 6 Two-scale SPH.

```
while animating do
    compute physics L
    determine regions in L
    transfer region information from L onto H
    add / delete boundary particles in H
    interpolate quantities from L onto boundary in H
    for nSubsteps do
        compute physics H
        advect boundary particles in H
    update parent particle
    interpolate feedback from H onto L
```

to the physics of nearby H particles but are advected by the velocity of the closest particle in L, referred to as parent particle. Boundary particles can transition into the active region of H, thus a relaxation scheme is necessary to restore the correct sampling density and to stabilize the system. Boundary particles are dynamically emitted and removed from the simulation. A feedback force transfers the velocities back from H to L. An example is illustrated in Fig. 6.

This approach is extended in [HS13] to multiple resolution levels. A different boundary emission scheme is used to ensure conservation of the total mass, hence retaining the advantage of single-scale SPH solvers. The relaxation scheme is extended to support stability with complex collision boundaries.

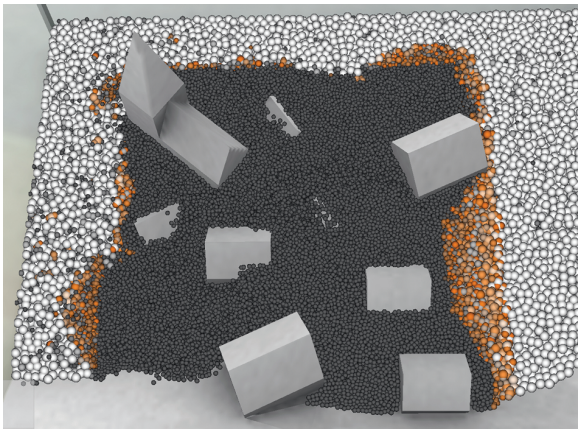

Figure 5: *In [OK12], resolution is smoothly changed via a temporal blending of particle levels (orange). An error estimation stabilizes integration time steps during blending.*

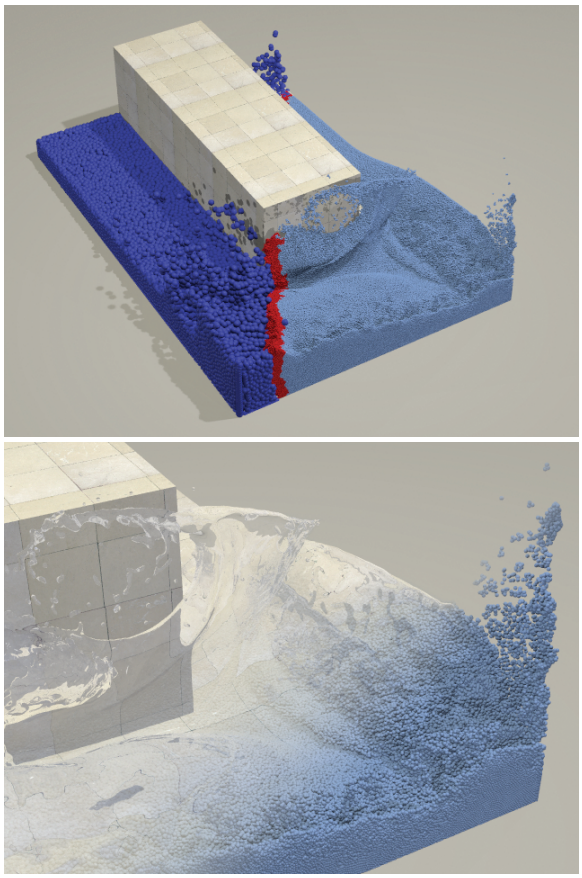

Figure 6: *Two-scale simulation [SG11]: dark blue corresponds to L, light blue to H (view frustum), and red to the boundary layer of H. Bottom row shows an overlay of the particle rendering and the surface computed with [SSP07].*

Compared to the single-scale solution, computation time is decreased by a factor of 3 to 12. The overall performance gain depends on the size of the visually interesting subset and hence the particle count in the high-resolution region.

5.2. Adaptive Time Discretization

In contrast to an adaptive space discretization, the time domain can be adaptively sampled as well. In the following, time-adaptive methods are classified into globally adaptive schemes using a single but dynamic integration time step for all particles and a locally adaptive time stepping employing individual time steps for each particle.

5.2.1. Globally Adaptive Time Steps

The convergence of SPH simulations is bound to the CFL condition (see Sec. 1). The idea of a globally adaptive time discretization [DC96, IAGT10, GP11] is to adjust the integration time step in each simulation step with respect to the

CFL condition. Thereby, the time step automatically adapts to the flow dynamics, hence avoiding complicated time step tuning.

However, in iterative EOS methods, the maximum force $\mathbf{F}^{max}$ or maximum velocity $\mathbf{v}^{max}$ is not known a priori. This might lead either to too restrictive time steps or to stability problems in case the dynamics are underestimated. To cope with this, [IAGT10] proposes to change the time step maximally by 0.2% per simulation step. In some cases, the relatively small time step change might still not be sufficient to resolve large density fluctuations, requiring a separate shock handling mechanism.

5.2.2. Locally Adaptive Time Integration

Applying the same time step globally to all particles may restrict computation speed if precise integration is only locally required. Thus, the idea of locally adaptive methods [DC96, DC99] is to use individual time steps for each particle at the cost of non-symmetric particle interactions, ignoring the action equals reaction principle. In [DC96], each particle evaluates forces at the largest possible time step which satisfies stability restrictions. Individual particle states are then synchronized at Δt intervals.

In [GP11], barely active fluid regions are completely deactivated. Particles are divided into active particles and inactive particles which are static until the velocity magnitude or distance to the fluid surface triggers reactivation. Compared to non-iterative EOS solvers with constant time stepping, a speedup of 6.7 has been reported in [GP11] for scenarios with up to 1 million particles.

6. Multiphase Fluids

The SPH algorithm can be adapted for multi-fluid flows (e.g., [MSKG05, TM05b, HA06, SP08, SB12]). In contrast to mesh-based methods where diffusion is introduced at the interface, the particle representation offers the advantage that the interface between two fluids is sharply defined. In this survey, we first discuss the literature on liquid-liquid interaction, and then present the works on modeling the liquid-air interface. Further, we outline the literature on surface tension forces.

6.1. Liquid-Liquid Interface

Multiple fluids with small density ratios, i.e., up to a factor of 10, can be simulated with a standard SPH solver (Sec. 1.2) by only changing the mass and rest density of the particles according to the fluid type [MSKG05]. The particle's rest volume remains constant, thus $m^a/\rho_0^a = m^b/\rho_0^b$ for two fluids a and b. The rest volume is invariant to changing the mass, provided that the rest density is adapted accordingly.

With the one-fluid formulation for the density and force computation, e.g., [Mon92, MCG03], discontinuities across the interface are smoothed due to the SPH nature of summing up values from neighboring particles inside a compact support. Hence, the density of a particle computed with Eq. (3) is spuriously influenced by the density of the fluid on the other side of the interface. Consequently, pressure fields and the acceleration of a particle are affected, resulting in spurious interface tension effects [Hoo98] and a large gap between the fluids [AMS*07]. The problem is intensified if larger density ratios (>10) are used, introducing instability issues which are unrelated to the time step size [CL03, SP08]. This is illustrated in Fig. 7.

Figure 7: *Standard SPH results in spurious behavior introduced by discontinuities across the interface of two fluids with density ratios (left). Adapted equations for multiple fluids according to [SP08] eliminate the artifacts (right).*

To account for the density discontinuity across the interface, the standard SPH equations can be adapted in terms of the number density $\delta_i = \sum_j W_{ij}$ [TM05b, HA06, SP08]. The density of a particle i is then computed as $\tilde{\rho}_i = m_i \delta_i$. This formulation offers the advantage that the density of particle i is not influenced by the mass of its neighbors j, even though i receives the geometric contribution W_{ij} from j. Hence, if i and j belong to different fluids, the computed density is not unphysically affected and sharp density profiles are achieved. The pressure $\tilde{p}_i$ is then computed from $\tilde{\rho}_i$ with Eq. (9).

In [SP08], $\tilde{\rho}_i$ and $\tilde{p}_i$ are substituted into the pressure term of the Navier Stokes equations, resulting in the pressure force $\mathbf{F}^{pressure} = -\nabla\tilde{p}/\delta$. Employing the quotient rule [Mon92] yields $\nabla\tilde{p}/\delta = \nabla(\tilde{p}/\delta) + \tilde{p}/\delta^2\nabla\delta$. After applying the SPH rule and replacing V by $1/\tilde{\rho}$, the pressure force equation is written as

$$\mathbf{F}_i^{pressure} = -\sum_j \left(\frac{\tilde{p}_j}{{\delta_j}^2} + \frac{\tilde{p}_i}{{\delta_i}^2}\right)\nabla W_{ij}. \qquad (16)$$

Though derived differently, Eq. (16) is applied in a similar form in [TM05b, HA06]. The viscosity force for multi-fluid systems can be derived similarly [SP08]. Note that, applied to a single fluid, the equations correspond to the standard one-phase flow formulation. By applying them to multiple fluids, however, spurious tension effects are avoided and stability is improved. The system can be further extended with an incompressibility condition [HA07].

Alternatively, the continuity equation can be used to compute the density in multi-phase flows [OS03, MR13]. The latter work is based on a Lagrangian with the continuity equation as a constraint. The continuity equation evolves the density over time by $d\rho/dt = -\rho\nabla \cdot \mathbf{v}$ [Mon94]. This formulation avoids the inherent issues of the summation equation, and hence is also applicable to free surface problems. They typically require higher-order time stepping schemes and careful considerations of time step sizes to avoid accumulation of integration errors and thus drift from true mass conservation [SP08, SB12]. In [GAC*09], the density summation equation is used in combination with a Shepard kernel to accurately preserve the discontinuity at the interface. The presented kernel is based on the volume distribution and the rate of change of the volume estimated by the continuity equation.

In the computer graphics literature, the density summation equation is preferred over evolving the density with the continuity equation. Avoiding error accumulation and thus drift from the rest volume is especially relevant for applications that require large time steps and robust results even if a comparably large time frame is simulated.

6.2. Liquid-Air Interface

In contrast to the liquid-liquid interface (Sec. 6.1), the problems at the liquid-air interface are intensified due to the lack of air particles. Thus, the modified equations for multi-phase fluids have no effect. Instead of discretizing the surrounding air completely, methods have been proposed to seed air particles only in areas close to the free surface, e.g., [MSKG05, IBAT11, SB12].

One-way coupling of air and water particles is presented in [SB12]. A narrow layer of air particles is dynamically generated, and quantities, such as the mass and velocity, are extrapolated. The air density is set to ρ_0 considering the $p = 0$ surface boundary condition. Air particles only contribute to the density field and otherwise do not interact with fluid particles. They are passively advected with the velocity field of the fluid. Thus, particle resampling is necessary to reduce drift away from the liquid. However, large smoothing radii are required in order to smooth sampling errors. Air particles in [SB12] are only employed to account for particle deficiency and, thus, to correct the density estimates at the free surface. However, air particles are not used to model the entrainment of air.

Although large density ratios can be modeled with [SP08], small buoyant volumes are damped as soon as the fluid approaches the equilibrium state [LSRS99, SP08]. It is argued that in this situation pressure forces arrange the particles in a stable lattice configuration which is difficult to break up. In order to model fast rising air bubbles, a buoyancy force can be added to counteract this effect, artificially simulating the high density ratio between water and air, e.g., [MSKG05, CPPK07, IBAT11]. In these methods, particles are dynamically generated and deleted at the liquid-air interface which can be computed based on the gradient of the color field ∇c [MSKG05]. In contrast to [MSKG05], two-way coupling of air and water particles in [IBAT11] is not realized via the pressure field, but according to the velocity field. This allows to simulate realistic drag and lift effects at comparatively large time steps which would cause numerical instabilities for both, standard SPH [MSKG05] and the modified multi-phase method [SP08].

Besides convection, physically-based formulation of reaction-diffusion processes at free surfaces requires stable modeling of phase singularities. The key concept of the implicit definition as given by [OHB*13] is to assign to each particle a value estimating its surface area, yielding a narrow band of surface particles as shown in Fig. 8. Like this, quantity fields defined on the surface can be consistently represented, enabling anisotropic diffusion effects and robust handling of thin fluid sheets as they arise in combination with multiple rigid objects [AIS*12].

6.3. Surface Tension

In this survey, we distinguish between macroscopic and microscopic views of the surface tension model.

6.3.1. Macroscopic Models

The macroscopic models, also referred as the continuum surface force model (CSF) [BKZ92], are based on a color field c which has a jump across the interface of two phases [BKZ92, Mor00, MCG03, MSKG05]. The color value of a particle i is interpolated with (3). Then, ∇c is computed, yielding the surface normal $\mathbf{n}$ pointing into the fluid. It can either be computed with the original SPH formulation for the gradient [MCG03, MSKG05], by applying Eq. (6) [GAC*09], or by considering the color difference c_{ij} between neighboring particles to get more accurate estimates of the surface normal [Mor00, SP08]. The curvature κ is given as $\kappa = -\nabla^2 c$. The resulting tension force then acts perpendicular to the interface of two phases, trying to minimize the curvature κ. In [MCG03, MSKG05, SP08], forces are only considered if the length of the normals are large enough. In order to ensure that tension forces are acting only on the interface of multiple liquids and not at the free surface, the color field has to be normalized [SP08].

Alternatively, the surface tension can be expressed as the divergence of the stress tensor as in [HA06, GAC*09]. The stress tensor $\mathbf{T}$ is again given by the color field and can be computed as $\mathbf{T} = \sigma 1/|\nabla c|(1/3\mathbf{I}|\nabla c|^2 - \nabla c \nabla c^T)$ [HA06]. In [GAC*09], the formulation provides surface tension effects only between the two liquids but not at the surface.

However, the approximation of the Laplacian with SPH for computing surface normals is error prone. As shown

in [YWTY12], this can be alleviated by computing the curvature from the reconstructed surface mesh. The surface normals can then be interpolated to adjacent particles. While this method improves the quality, it relies on an explicit representation of the fluid mesh in each simulation step which decreases the overall performance.

6.3.2. Microscopic Models

The microscopic models consider cohesion forces between particles to imitate attractive forces between molecules [NP00, TM05a, BT07]. In contrast to the macroscopic formulations (Sec. 6.3.1), the computation of the curvature and thus the second derivative is avoided which is particularly sensitive to particle disorder. In [NP00], the van der Waals equation of state is used to compute the cohesive pressure. The range of the attractive forces is increased to $4h$ to obtain stable liquid drops. Alternatively, the attractive force can be computed within a distance h as in [BT07]. The force is given as $\mathbf{F}_i^{surface} = -\sigma \sum_j m_j \mathbf{x}_{ij} \cdot W_{ij}$. In contrast, in [TM05a, AAT13], the cohesion force is modeled as a combined force of short-range repulsive forces and long-range attractive forces.

Recently, it has been demonstrated that neither the microscopic nor the macroscopic model alone is sufficient to model all effects at the same time [AAT13], i.e., large surface tension, minimization of the surface curvature, no clustering of particles and zero dissipation of momentum. This has been realized in [AAT13], using a combined approach which is further enhanced by an adhesion force that plausibly models wetting effects.

7. Surface Reconstruction and Rendering

In the SPH literature, various techniques have been presented to reconstruct surfaces from a set of particles, both for offline and real-time applications. The main challenge thereby is to efficiently achieve smooth surfaces while capturing fine surface details such as droplets, thin sheets, and capillary waves. In this overview, we first discuss scalar field definitions and polygonization in Sec. 7.1, then we outline explicit methods and direct rendering techniques in Sec. 7.2, and present screen-space approaches in Sec. 7.3. We conclude the section with volume rendering approaches, discussed in Sec. 7.4.

7.1. 3D Scalar Fields

In Eulerian simulations, the liquid surface is typically represented as the zero level set of the scalar level set function, which is then advected by the fluid flow. In contrast, Lagrangian simulations usually define the surface by superimposing kernel functions of the individual particles to define a scalar density field. The isosurface of this 3D field is then polygonized with Marching Cubes [LC87]. The grid resolution has a large influence on the surface quality and efficiency. Literature suggests that a cell size of half the particle spacing is required to capture very fine details [AIAT12].

Another important aspect for the surface quality, especially for the surface smoothness, is the definition of the implicit function. Several scalar field formulations are presented in the literature. The earliest and probably simplest approach are blobbies, also known as metaballs, presented in [Bli82]. Each particle is approximated by a Gaussian potential, and the superposition of these functions define a density field. While this approach is comparably simple, irregular particle distributions result in visible surface bumps. These artifacts can be reduced by weighting the contribution of each particle by its volume [MCG03].

Smoother surfaces are preferably computed by defining the scalar field based on the weighted average of nearby particle positions $\bar{\mathbf{x}}$ [ZB05, APKG07, SSP07, OCD13, AIAT12, AAIT12, OHB*13]. A signed distance field is then given by the implicit function $\phi(\mathbf{x}) = |\mathbf{x} - \bar{\mathbf{x}}| - r$, where $\mathbf{x}$ is the query point, r is the particle radius, i.e. half the particle spacing, and $\phi(\mathbf{x}) = 0$ defining the on-surface points [ZB05]. As shown in [AIAT12], optimal results are achieved if the weighted value $\bar{\mathbf{x}}$ is computed within an influence radius of 4 times the particle spacing. Such a comparably large influence radius, however, intensifies the problem of artifacts in concave regions observed with the original formulation of [ZB05]. This issue can be addressed by modulating the surface distance r, either based on an eigenvalue analysis of $\nabla_{\mathbf{x}}(\bar{\mathbf{x}})$ [SSP07], or by a position-based decay function [OCD13]. The formulation of [SSP07] is also used in [AIAT12, AAIT12], but the scalar field is only reconstructed in a narrow band around the surface, reducing the memory footprint and computation time. The formulation of [ZB05] can be extended to reconstruct smooth surfaces for spatially adaptive simulations with variably sized particles [APKG07] (see Sec. 5.1.1). For this, the distance to the surface is tracked for each particle and considered in the scalar field computation. A temporally smooth transition of surface particles stabilizes singularities like splashes or thin sheets, as proposed in [OHB*13].

In contrast to previous approaches which all use isotropic weighting kernels, anisotropic kernels, determined by a Principal Component Analysis over particle neighbors, can be employed. Therefore, a single pass of Laplacian smoothing of particle positions is added which facilitates the reconstruction of sharp features and edges [YT10]. Alternatively, the thin-plate energy of the surface can be optimized, while constraining the surface to lie within an inner and outer bound given by the particle positions [BGB11].

7.2. Explicit Methods and Direct Rendering

Instead of constructing a new mesh in each rendering step, explicit meshes can be used which are advected over time

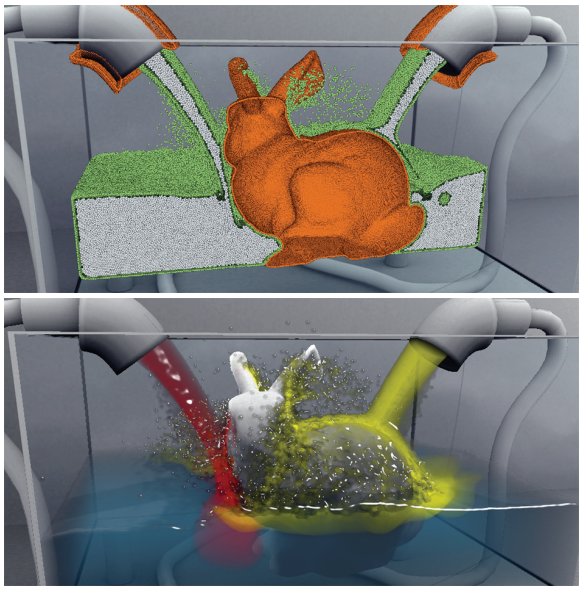

Figure 8: *Implicit surface model (top) proposed by [OHB*13] and the corresponding volume rendering (bottom) similar to [OKK10]. A discretization of the surface delta-function results in a narrow band of surface particles (green) enabling quantity transport between bulk (white), surface and rigids (orange).*

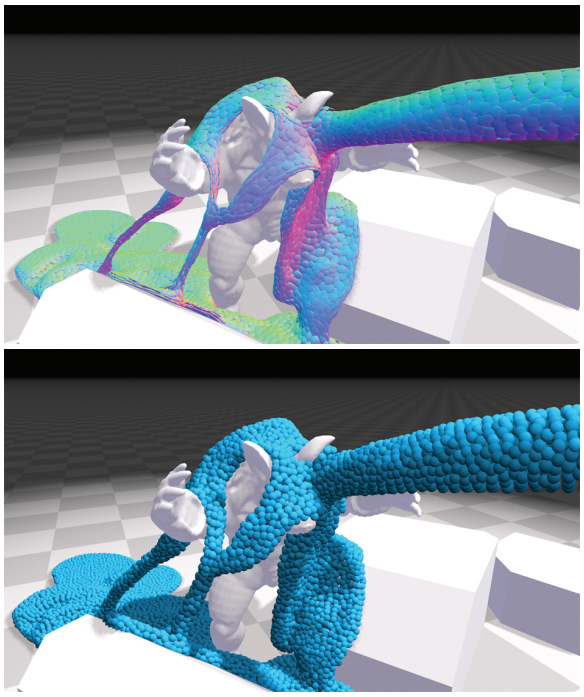

Figure 9: *Ellipsoid point splatting as used in [MM13] computed based on anisotropic kernel weights [YT13]. The bottom row shows the corresponding particle set.*

and thus result in temporal-coherent structures. While surface tracking techniques are usually used in the Eulerian context (e.g., [EFFM02]), only few work exist in the SPH literature. In [PTB*03], a level set surface is defined, which acts as an initialization for the consecutive rendering step. The grid resolution used for the level set is independent from the particle resolution. In [DC98], an active surface model is used that is attracted to the isosurface of an implicit function. A grid is constructed in each rendering frame, thus remaining memory intensive. In contrast, the surface mesh is only created initially in [YWTY12], and mesh vertices are advected by the nearby particle velocities. The mesh surface is periodically projected onto an implicit function to avoid drift from the particles.

Alternatively, the polygonization step and thus exhaustive memory consumption can be avoided by rendering the isosurface directly, for example, by computing ray-isosurface intersections with metaballs [KSN08, ZSP08] or distance fields [GSSP10] on the GPU, or intersections with spheres on the CPU [GIK*07]. For interactive simulations, point splatting approaches [ZPvBG01] are employed to render the particle sets using either isotropic weighting kernels in [MCG03] or anisotropic kernels in [MM13], see Fig. 9.

7.3. Screen Space Approaches

In contrast to the rendering approaches in world space, much work has focused on screen space methods targeted at real-time applications. As these algorithms operate in 2D, they are much more efficient than their 3D counterparts. All methods have in common that they extract and smooth a depth map from the 3D point cloud in screen space. In [MHHR07], the depth map is smoothed by a binomial filter, and then used to extract a mesh in screen-space with Marching Squares for the particles that are visible to the camera. Instead of generating an intermediate mesh, the depth buffer can be directly rendered [vdLGS09, Gre10, BSW10]. For this, each particle is first rendered as a sphere, and then a smoothing filter is applied to the depth values to get a continuous surface. A Gaussian filter is used in [Gre10], with bilateral weights to preserve silhouette edges. The efficiency is further improved by separating the filter at the cost of introducing artifacts at the fluid surface. Alternatively to Gaussian smoothing, changes in curvature between the particles can be smoothed [vdLGS09]. The method is extended in [BSW10] to achieve smoothing independent of the camera distance.

7.4. Volume Rendering

While surface reconstructions conveys the geometric shape of the fluid, the depth of the fluid volume or the mass flow inside of it cannot be perceived by sole surface rendering alone. For this kind of information volume rendering techniques are commonly employed, as shown in Fig. 8. The idea is to evaluate a physically-based model for light transport by treating the quantity field defined by the particles as a participating medium interacting with rays of light [EHK*06, HLSR08]. Conceptually, for each viewing ray, emission-adsorption values are integrated from the camera towards the fluid volume.

In the context of SPH, mainly hybrid splatting-slicing approaches have been proposed [vdLGS09, FGE10, FAW10] using texture-based volume rendering. Particle contributions are scattered [Wes90] onto view-aligned [NMM*06] or axis-aligned texture slices [SP09a] by exploiting rasterization units of GPUs. Texture slices are then composed front-to-back to yield the final image by exploiting the hardware-based rendering pipeline.

In combination with more generic rendering architectures, ray casting or ray tracing is commonly used, e.g. for surface [SSP07, GSSP10] as well as volume rendering [OKK10, JFSP10, IAAT12]. Memory coherence for unstructured point data is usually enforced by space subdivision schemes in object-space, i.e. the simulation domain is subdivided using kd-trees [ZRL*08, LLRR08, JFSP10] or octrees [GGG08, ZGHG10, FAW10]. However, ray casting of larger particle numbers at interactive rates requires sophisticated pre-processing limited to offline applications [FSW09, RTW13]. Caching mechanisms [OKK10] and sampling operators [FAW10] as discussed in Sec. 5 can reduce the sampling overhead to some extent. A different strategy is to use a perspective view-aligned grid with a logarithmic distribution of samples [FAW10] in order to adapt the number of samples in viewing direction to the image plane resolution.

8. Secondary Simulation

The realistic animation of ocean scenes with breaking waves is challenging. Highly turbulent flows demand a high resolution to capture small-scale details like splashes, while for the major part of the fluid a lower resolution is sufficient. This can be partly addressed by employing multi-scale methods as discussed in Sec. 5.1, however, the interplay of fluid and air has to be explicitly modeled in order to capture the formation of diffuse material, perceived as foam, spray and tiny air bubbles.

In order to simulate fluid-air mixtures, multi-phase approaches as discussed in Sec. 6 can be employed. However, in order to avoid numerical instabilities, small time steps and expensive computations, implicit models are favored, where air-water mixtures are determined based on heuristics and an artificial buoyancy force is applied to model the high-density contrast. Implicit modeling is convenient to capture even small-scale phenomena. In the context of air bubble animations, this has been demonstrated for grid-based [HLYK08, PAKF13] and particle-based solvers [MSKG05, GH04, IBAT11], whereas implicit mist models and spray models have been coupled to grid-based fluid solvers [TRS06, LTKF08, CM11]. The first, pure Lagrangian method which simulates spray, foam and air-bubbles in an unified way has been presented in [IAAT12].

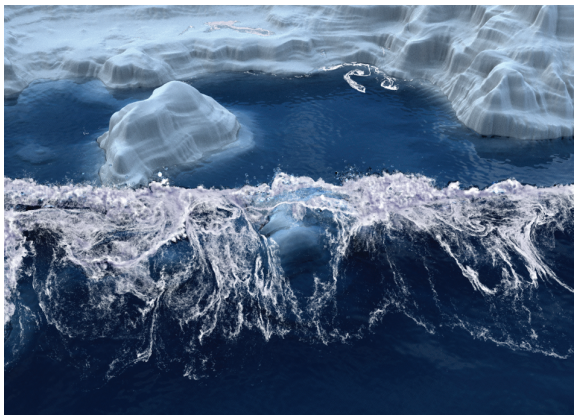

Figure 10: *IISPH [ICS*13] simulation of breaking waves with whitewater [IAAT12]. The fluid volume is discretized with 16 million SPH particles and up to 25 million secondary particles representing foam, bubbles and spray. The volume radius of SPH particles was set to* 1.8*cm.*

In [IAAT12], the influence of diffuse material onto the water phase is neglected, which allows to simulate whitewater effects in a secondary simulation on top of pre-computed fluid data. Physically-motivated criteria are formulated for locating regions where water mixes with air, determined by the potential of a fluid particle to trap air, its likelihood to be at the crest of a wave and its kinetic energy. The combined potential and the volume ratio determines the number of generated diffuse particles. Based on their location, particles are classified as foam, spray or bubbles, which determines how the fluid affects the motion of diffuse particles. As no forces are computed between diffuse particles, diffuse-diffuse neighbor-sets are not computed, while large time steps up to the frame rate can be used. This makes the approach efficient to compute, e.g., [IAAT12] reports a computation time of 2 hours for a 20 second video with 15 million diffuse particles using a standard desktop PC. Further, [MM13] showed that the model of [IAAT12] can be employed to enhance interactive fluid simulations. Examples are given in Fig. 10 and Fig. 11.

An alternative method to model diffuse material for SPH

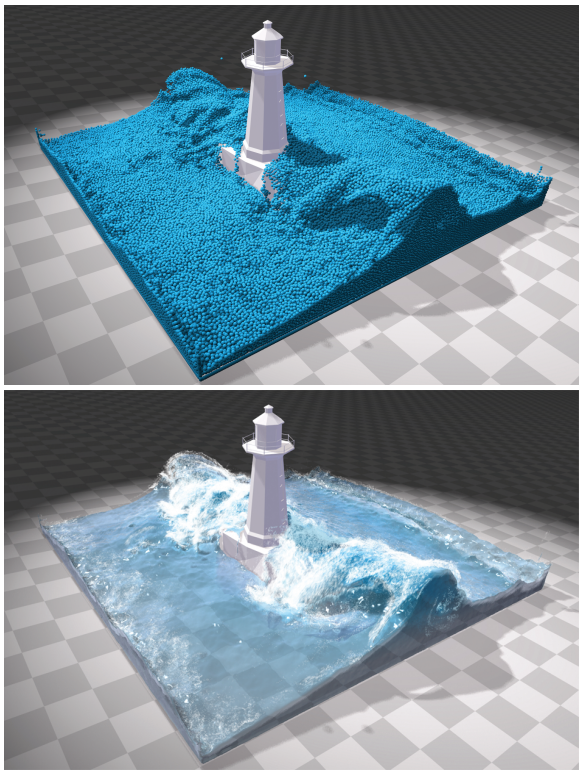

Figure 11: *PBF [MM13] simulation of waves breaking around a lighthouse. The top row shows the base simulation with 300k particles. The bottom row shows the final frame. The surface is reconstructed using ellipsoid point splatting and screen space filtering [vdLGS09]. The foam is simulated with [IAAT12]. The computation of the final frame took 60ms on the NVIDIA GTX 680.*

fluids is presented in Bagar et al. [BSW10]. This model uses the Weber number to classify fluid regions into water and foam for real-time rendering. In this method, fluid particles are rendered as foam particles when the Weber number exceeds a threshold. This improves the realism of real-time simulations significantly. However, in contrast to [IAAT12], the simulation only relies on fluid particles, i.e., no foam particles are represented, which restricts the level of detail to the resolution of the underlying fluid.

9. Conclusions and Future Development

This survey showed that with the recent improvements on incompressibility, efficiency, and flexibility, SPH has outgrown its infancy and emerged to a powerful and versatile fluid model. Particular challenges of SPH, such as searching neighbors efficiently, preserving volume, locally adapting resolutions, and achieving smooth surfaces, have been addressed. The simulation of millions of particles on desktop computers is now possible, enabling the Lagrangian method to keeping up with the visual quality of its Eulerian counterpart.

Such comparably high-resolution simulations, however, introduce new challenges. A main limitation is the time step restriction, which represents a current bottleneck. Further, evaluation of iterative incompressibility models, such as PCISPH, IISPH, and PBF, suggest that the largest possible time step does not necessarily result in the best overall performance, which renders the time stepping even more challenging. Future work is certainly necessary to automatically determine the optimal time step size.

In contrast to high-resolution offline simulations, real-time applications have gained less attention, although it seems that particles are still the preferred discretization element, e.g., to model coarse 3D fluids in games. The main issue with coarse simulations is that substantial dissipation is introduced, resulting in viscous behavior and preventing splashes and surface structures to evolve. Literature suggests that this is due to larger support radii in coarse simulations, i.e., the viscosity force which smoothes neighboring velocities over larger regions. While various formulations for the viscosity force have been presented that reduce dissipation, more work is certainly necessary to better understand and to prevent the energy loss in coarse real-time simulations. A concession is also made between efficiency and surface quality, as splashes appear large and blobby, and surfaces comparably bumpy. Recent work on secondary effects, such as foam layers and bubbles, show their potential to dramatically improve the visual appearance. Also, multi-scale methods provide the potential to counteract these effects.

Further, the liquid-air interface has not gained sufficient attention in the SPH literature, yet. Although first attempts have been presented to address the problems of particle deficiency at the free surface, they introduce substantial additional costs and increase the complexity of the solver. A better handling of the free surface is therefore necessary, which accounts for the particle deficiency and accurately models the air pressure, while not drastically affecting memory consumption, performance, and simplicity of the solver.

While the particle representation has proven to be beneficial in many aspects, such as trivial advection, interaction with complex boundaries, free surface tracking, and modeling multiple liquids, all presented variants of fully Lagrangian models come at the cost of tedious parameter tuning. Although these parameters allow an artist to control the desired effects, more work on automated parameter setting is necessary to facilitate the scene setup and fluid initialization.

Appendix A: Lagrangian and Eulerian Approaches

In Lagrangian fluid approaches, a sample point $\mathbf{x}_i$ is advected with the local flow velocity of the fluid $\mathbf{v}_i$: $\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i$. This is in contrast to Eulerian approaches, where a sample point $\mathbf{x}_i$

is not necessarily advected, but commonly fixed in space and aligned to a grid.

Sample positions, particles and grid cells: Lagrangian and Eulerian fluid approaches compute the velocity field at sample positions. These sample positions represent a small fluid volume and a small mass. In Lagrangian approaches, these sample positions are often referred to as particles to indicate that these small volumes or masses move with the flow. In Eulerian approaches, sample positions represent small volumes or masses that are not advected, e.g., cubic cells fixed in space.

Navier-Stokes equation: The differential form of the fluid momentum equation, i.e. Navier-Stokes equation, is commonly derived with the time rate of change of the velocity of a small volumetric fluid element that moves with the flow. The respective operator $\frac{D}{Dt}$ is often referred to as material derivative. Using this operator, the Navier-Stokes equation can be written as

$$\frac{D\mathbf{v}_i}{Dt} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i}, \tag{17}$$

regardless of whether i is a Lagrangian or Eulerian sample. Nevertheless, the material derivative translates to different terms depending on the approach. In a Lagrangian setting, we simply have $\frac{D\mathbf{v}_i}{Dt} = \frac{d\mathbf{v}_i}{dt}$. Here, the material derivative of the velocity corresponds to the time rate of change of the velocity of the sample point, i.e. particle, that is advected with its velocity. Note that $\frac{D\mathbf{v}_i}{Dt} = \frac{d\mathbf{v}_i}{dt}$ is only valid in combination with $\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i$. In Eulerian approaches, the material derivative has to be expressed with derivatives at samples i with fixed positions. Here, we get $\frac{D\mathbf{v}_i}{Dt} = \frac{\partial\mathbf{v}_i}{\partial t} + \mathbf{v}_i\cdot\nabla\mathbf{v}_i$ for the material derivative of the velocity. Note that $\frac{\partial\mathbf{v}_i}{\partial t}$ is the time derivative of the velocity at a fixed Eulerian sample. This term is different to $\frac{d\mathbf{v}_i}{dt}$ which is the time derivative of an advected Lagrangian sample.

To summarize, the Navier-Stokes equation for *advected Lagrangian sample points* $\mathbf{x}_i$ with $\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i$ can be written as

$$\frac{d\mathbf{v}_i}{dt} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i} \tag{18}$$

describing the *time rate of change of the velocity of a moving sample point or particle* $\mathbf{x}_i$. For *fixed Eulerian sample points* $\mathbf{x}_i$, the Navier-Stokes equation reads

$$\frac{\partial\mathbf{v}_i}{\partial t} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i} - \mathbf{v}_i\cdot\nabla\mathbf{v}_i \tag{19}$$

describing the *time rate of change of the velocity* $\mathbf{v}_i$ *at a fixed position* $\mathbf{x}_i$.

Discussion: It is not possible to generally compare performance and accuracy of Lagrangian and Eulerian methods. This paragraph, however, discusses two aspects in this context that sometimes tend to be discussed in a rather prejudiced way.

Lagrange approaches are expensive as they determine particle neighbors in each simulation step, while Eulerian approaches efficiently work with a constant neighborhood. The neighborhood search constitutes computational overhead in Lagrangian approaches. On the other hand, it is rarely discussed that the handling of the convective derivative $\mathbf{v}_i\cdot\nabla\mathbf{v}_i$ is overhead in Eulerian approaches. E.g., the particles in FLIP methods are a numerical technique to account for the convective derivative at Eulerian sample positions of the underlying grid.

Lagrangian approaches are compressible, while Eulerian methods are incompressible. Many SPH fluids in Graphics are compressible which basically refers to the fact that the fluid density is a function of time. A typical issue in such methods are oscillations of the fluid volume that adversely affect the simulation quality. Incompressible Eulerian techniques, on the other hand, have no notion of the fluid density. They start with the rest density at sample positions and compute a divergence-free velocity field in each simulation step to preserve the rest density. Issues in such formulations are artificial viscosity [Sta99], mass loss [CM13] or mass gain [GB13]. So, all compressible or incompressible approaches have issues with volume changes. In compressible formulations, these volume changes are due to density fluctuations with perfect mass preservation. In incompressible techniques, volume changes are due to mass changes with perfect density preservation. Please also note that Eulerian techniques also work with compressible formulations, while Lagrangian techniques can be applied to incompressible formulations as, e.g., discussed in [ICS*13].

References

[AAIT12] AKINCI G., AKINCI N., IHMSEN M., TESCHNER M.: An efficient surface reconstruction pipeline for particle-based fluids. In *Proceedings VRIPHYS* (2012), pp. 53–60. 4, 14

[AAT13] AKINCI N., AKINCI G., TESCHNER M.: Versatile surface tension and adhesion for SPH fluids. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 32*, 6 (2013), 182:1–182:8. 14

[ACAT13] AKINCI N., CORNELIS J., AKINCI G., TESCHNER M.: Coupling elastic solids with smoothed particle hydrodynamics fluids. *Comp. Anim. Virtual Worlds 24* (2013), 195–203. 9

[AIAT12] AKINCI G., IHMSEN M., AKINCI N., TESCHNER M.: Parallel surface reconstruction for particle-based fluids. *Computer Graphics Forum 32*, 1 (2012), 99–112. 4, 14

[AIS*12] AKINCI N., IHMSEN M., SOLENTHALER B., AKINCI G., TESCHNER M.: Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics (Proceedings SIGGRAPH) 30*, 4 (2012), 72:1–72:8. 2, 9, 10, 13

[AIY*04] AMADA T., IMURA M., YASUMORO Y., MANABE Y., CHIHARA K.: Particle-based fluid simulation on GPU. In *ACM Workshop on General-Purpose Computing on Graphics Processors* (2004), pp. 1–6. 4

[AMS*07] AGERTZ O., MOORE B., STADEL J., POTTER D., MINIATI F., READ J., MAYER L., GAWRYSZCZAK A., KRAVTSOV A., MONAGHAN J., NORDLUND A., PEARCE F., QUILIS V., RUDD D., SPRINGEL V., STONE J., TASKER E.,

TEYSSIER R., WADSLEY J., WALDER R.: Fundamental differences between SPH and grid methods. *Mon. Not. R. Astron. Soc. 380*, 3 (2007), 963–978. 12

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L.: Adaptively sampled particle fluids. In *ACM Transactions on Graphics (Proceedings SIGGRAPH)* (2007), vol. 26, pp. 48:1–48:7. 2, 3, 5, 10, 14

[BGB11] BHATACHARYA H., GAO Y., BARGTEIL A.: A level-set method for skinning animated particle data. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, USA, 2011), SCA '11, ACM, pp. 17–24. 14

[BKZ92] BRACKBILL J. U., KOTHE D. B., ZEMACH C.: A continuum method for modeling surface tension. *Journal of Computational Physics 100*, 2 (1992), 335–354. 13

[Bli82] BLINN J.: A generalization of algebraic surface drawing. *ACM Transactions on Graphics 1*, 3 (1982), 235–256. 14

[BLS12] BODIN K., LACOURSIRE C., SERVIN M.: Constraint fluids. *IEEE Transactions on Visualization and Computer Graphics 18*, 3 (2012), 516–526. 6

[BOT01] BØRVE S., OMANG M., TRULSEN J.: Regularized smoothed particle hydrodynamics: A new approach to simulating magnetohydrodynamic shocks. *The Astrophysical Journal Supplement Series 561* (2001), 345–367. 10

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

[BSW10] BAGAR F., SCHERZER D., WIMMER M.: A layered particle-based fluid model for real-time rendering of water. *Computer Graphics Forum (Proceedings EGSR 2010) 29*, 4 (2010), 1383–1389. 15, 17

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), pp. 209–217. 2, 5, 6, 10, 14

[BTT09] BECKER M., TESSENDORF H., TESCHNER M.: Direct forcing for Lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics 15*, 3 (2009), 493–503. 9

[CBP05] CLAVET S., BEAUDOIN P., POULIN P.: Particle-based viscoelastic fluid simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 219–228. 6

[Cho68] CHORIN A. J.: Numerical solution of the Navier-Stokes equations. *Mathematics of Computation 22*, 104 (1968), 745–762. 5

[CL03] COLAGROSSI A., LANDRINI M.: Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *J. Comp. Phys. 191*, 2 (2003), 448–475. 12

[CM11] CHENTANEZ N., MÜLLER M.: Real-time Eulerian water simulation using a restricted tall cell grid. *ACM Transactions on Graphics (Proceedings SIGGRAPH) 30* (2011), 82:1–82:10. 7, 16

[CM13] CHENTANEZ N., MUELLER M.: Mass-conserving Eulerian liquid simulation. *IEEE Transactions on Visualization and Computer Graphics 99*, PrePrints (2013), 1. 18

[CPK02] CHANIOTIS A. K., POULIKAKOS D., KOUMOUTSAKOS P.: Remeshed smoothed particle hydrodynamics for the simulation of viscous and heat conducting flows. *J. Comput. Phys. 182* (2002), 67–90. 10

[CPPK07] CLEARY P., PYO S., PRAKASH M., KOO B.: Bubbling and frothing liquids. *ACM Transaction on Graphics (Proceedings SIGGRAPH) 26*, 3 (2007), 97. 13

[CR99] CUMMINS S., RUDMAN M.: An SPH projection method. *Journal of Computational Physics 152*, 2 (1999), 584–607. 7

[DC96] DESBRUN M., CANI M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Eurographics Workshop on Computer Animation and Simulation (EGCAS)* (1996), Springer-Verlag, pp. 61–76. 2, 5, 11, 12

[DC98] DESBRUN M., CANI M.-P.: Active implicit surface for animation. In *Graphics Interface* (1998), Canadian Human-Computer Communications Society, pp. 143–150. Published under the name Marie-Paule Cani-Gascuel. 15

[DC99] DESBRUN M., CANI M.-P.: *Space-Time Adaptive Simulation of Highly Deformable Substances*. Tech. Rep. 3829, INRIA, 1999. 10, 12

[DRF12] DURAND M., RAFFIN B., FAURE F.: A packed memory array to keep moving particles sorted. In *VRIPHYS - Ninth Workshop on Virtual Reality Interactions and Physical Simulations* (2012), The Eurographics Association, pp. 69–77. 4

[EFFM02] ENRIGHT D., FEDKIW R., FERZIGER J., MITCHELL I.: A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics 183*, 1 (2002), 83–116. 15

[EHK*06] ENGEL K., HADWIGER M., KNISS J., REZK-SALAMA C., WEISKOPF D.: *Real-time Volume Graphics*. A. K. Peters, Ltd., Natick, MA, USA, 2006. 16

[FAW10] FRAEDRICH R., AUER S., WESTERMANN R.: Efficient high-quality volume rendering of SPH data. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1533–1540. 4, 16

[FB07] FELDMAN J., BONET J.: Dynamic refinement and boundary contact forces in SPH with applications in fluid flow problems. *Int. J. Numerical Methods in Engineering 72*, 3 (2007), 295–324. 10

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proceedings SIGGRAPH 2001* (2001), pp. 23–30. 7

[FG07] FALAPPI S., GALLATI M.: SPH simulation of water waves generated by granular landslides. In *Proceedings of 32nd Congress of IAHR (International Association of Hydraulic Engineering & Research* (2007). 9

[FGE10] FALK M., GROTTEL S., ERTL T.: Interactive image-space volume visualization for dynamic particle simulations. In *Proceedings of The Annual SIGRAD Conference* (2010). 4, 16

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5 (1996), 471–483. 7

[FSW09] FRAEDRICH R., SCHNEIDER J., WESTERMANN R.: Exploring the millennium run – scalable rendering of large-scale cosmological datasets. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1251–1258. 16

[GAC*09] GRENIER N., ANTUONO M., COLAGROSSI A., TOUZÉ D. L., ALESSANDRINI B.: An Hamiltonian interface SPH formulation for multi-fluid and free surface flows. *Journal of Computational Physics 228*, 22 (2009), 8380 – 8393. 13

[GB13] GERSZEWSKI D., BARGTEIL A. W.: Physics-based animation of large-scale splashing liquids. *ACM Trans. on Graphics (Proc. ACM SIGGRAPH Asia) 32*, 6 (2013), 1–6. 18

[GGG08] GUENNEBAUD G., GERMANN M., GROSS M.: Dynamic sampling and rendering of algebraic point set surfaces. *Computer Graphics Forum 27*, 2 (2008), 653–662. 16

[GH04] GREENWOOD S. T., HOUSE D. H.: Better with bubbles: enhancing the visual realism of simulated fluid. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), pp. 287–296. 16

[GIK*07] GRIBBLE C. P., IZE T., KENSLER A., WALD I., PARKER S. G.: A coherent grid traversal approach to visualizing particle-based simulation data. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 758–768. 15

[GP11] GOSWAMI P., PAJAROLA R.: Time adaptive approximate SPH. In *Proceedings Eurographics Workshop on Virtual Reality Interaction and Physical Simulation* (2011). 11, 12

[Gre08] GREEN S.: Particle-based fluid simulation. "http://developer.download.nvidia.com/presentations/2008/GDC/GDC08_ParticleFluids.pdf", 2008. 3, 4

[Gre10] GREEN S.: Screen space fluid rendering for games. "http://developer.download.nvidia.com/presentations/2010/gdc/Direct3D_Effects.pdf", 2010. 15

[GSSP10] GOSWAMI P., SCHLEGEL P., SOLENTHALER B., PAJAROLA R.: Interactive SPH simulation and rendering on the GPU. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), pp. 55–64. 2, 3, 4, 15, 16

[HA06] HU X., ADAMS N.: A multi-phase SPH method for macroscopic and mesoscopic flows. *Journal of Computational Physics 213*, 2 (2006), 844–861. 9, 12, 13

[HA07] HU X., ADAMS N.: An incompressible multi-phase SPH method. *Journal of Computational Physics 227*, 1 (2007), 264–278. 7, 12

[HCM06] HEGEMAN K., CARR N. A., MILLER G. S. P.: Particle-based fluid simulation on the GPU. In *Proceedings of the 6th international conference on Computational Science - Volume Part IV* (Berlin, Heidelberg, 2006), ICCS'06, Springer-Verlag, pp. 228–235. 4

[Hie07] HIEBER S.: *Particle-Methods for Flow-Structure Interactions*. PhD thesis, Swiss Federal Institute of Technology, 2007. 3

[HKK07a] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Sliced data structure for particle-based simulations on GPUs. In *Proceedings of the Computer graphics and interactive techniques in Australia and Southeast Asia* (2007), pp. 55–62. 4

[HKK07b] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics on GPUs. In *Proceedings of Computer Graphics International* (2007), pp. 63–70. 3, 4, 9

[HLSR08] HADWIGER M., LJUNG P., SALAMA C. R., ROPINSKI T.: Advanced illumination techniques for GPU volume raycasting. In *ACM SIGGRAPH ASIA 2008 Courses* (2008), SIGGRAPH Asia '08. 16

[HLWW12] HE X., LIU N., WANG H., WANG G.: Local Poisson SPH for viscous incompressible fluids. *Computer Graphics Forum 31* (2012), 1948–1958. 5, 6

[HLYK08] HONG J.-M., LEE H.-Y., YOON J.-C., KIM C.-H.: Bubbles alive. *ACM Transactions on Graphics (Proceedings SIGGRAPH) 27* (2008), 48:1–48:4. 16

[Hoo98] HOOVER W.: Isomorphism linking smooth particles and embedded atoms. *Physica A: Statistical Mechanics and its Applications 260*, 3 (1998), 244–254. 12

[HS13] HORVATH C. J., SOLENTHALER B.: Mass preserving multi-scale SPH. Pixar Technical Memo 13-04, Pixar Animation Studios, 2013. 10, 11

[IAAT12] IHMSEN M., AKINCI N., AKINCI G., TESCHNER M.: Unified spray, foam and bubbles for particle-based fluids. *The Visual Computer 30*, 1 (2012), 99–112. 3, 16, 17

[IABT11] IHMSEN M., AKINCI N., BECKER M., TESCHNER M.: A parallel SPH implementation on multi-core CPUs. *Computer Graphics Forum 30*, 1 (2011), 99–112. 2, 3, 4, 8

[IAGT10] IHMSEN M., AKINCI N., GISSLER M., TESCHNER M.: Boundary handling and adaptive time-stepping for PCISPH. In *Proceedings VRIPHYS* (2010), pp. 79–88. 3, 9, 11, 12

[IBAT11] IHMSEN M., BADER J., AKINCI G., TESCHNER M.: Animation of air bubbles with SPH. In *Computer Graphics Theory and Applications GRAPP* (2011), pp. 225–234. 13, 16

[ICS*13] IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics* (2013). doi:10.1109/TVCG.2013.105. 2, 3, 5, 6, 7, 8, 9, 16, 18

[JFSP10] JANG Y., FUCHS R., SCHINDLER B., PEIKERT R.: Volumetric evaluation of meshless data from smoothed particle hydrodynamics simulations. In *Proceedings of the 8th IEEE EG International Conference on Volume Graphics* (2010), VG'10, Eurographics Association, pp. 45–52. 16

[KAD*06] KEISER R., ADAMS B., DUTRÉ P., GUIBAS L., PAULY M.: *Multiresolution Particle-Based Fluids*. Tech. rep., ETH Zurich, 2006. 3, 9, 10

[KC05] KOLB A., CUNTZ N.: Dynamic particle coupling for GPU-based fluid simulation. In *Proc. Symp. Sim. Technique* (2005), pp. 722–727. 4

[Kei06] KEISER R.: *Meshless Lagrangian Methods for Physics-Based Animation of Solids and Fluids*. PhD thesis, ETH Zürich, 2006. 3

[KGS09] KHAYYER A., GOTOH H., SHAO S.: Enhanced predictions of wave impact pressure by improved incompressible SPH methods. *Applied Ocean Research 31*, 2 (2009), 111 – 131. 7

[KLRS04] KOLB A., LATTA L., REZK-SALAMA C.: Hardware-based simulation and collision detection for large particle systems. In *Proceedings Graphics Hardware* (2004), ACM/Eurographics, pp. 123–131. 4

[KS09] KALOJANOV J., SLUSALLEK P.: A parallel algorithm for construction of uniform grids. In *Proceedings of the ACM conference on High Performance Graphics* (2009), pp. 23–28. 3

[KSN08] KANAMORI Y., SZEGO Z., NISHITA T.: GPU-based fast ray casting for a large number of metaballs. *Computer Graphics Forum 27*, 2 (2008), 351–360. 15

[KSW04] KIPFER P., SEGAL M., WESTERMANN R.: UberFlow: a GPU-based particle engine. In *Proc. ACM SIGGRAPH/EUROGRAPHICS Conf. on Graphics Hardware* (2004), pp. 115–122. 4

[KW02] KITSIONAS S., WHITWORTH A.: Smoothed particle hydrodynamics with particle splitting, applied to self-gravitating collapse. *Monthly Notices of the Royal Astronomical Society 330*, 1 (2002), 129–136. 10

[LAD08] LENAERTS T., ADAMS B., DUTRÉ P.: Porous flow in particle-based fluid simulations. *ACM Transactions on Graphics (Proceedings SIGGRAPH) 27*, 3 (2008), 49:1–49:8. 9

[LB95] LAPENTA G., BRACKBILL J.: Control of the number of particles in fluid and MHD particle in cell methods. *Computer Physics Communications 87*, 1-2 (1995), 139 – 154. 10

[LC87] LORENSEN W., CLINE H.: Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, USA, 1987), ACM Press, pp. 163–169. 14

[LD08] LAGAE A., DUTRÉ P.: Compact, fast and robust grids

for ray tracing. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering) 27*, 4 (2008), 1235–1244. 2

[LD09] LENAERTS T., DUTRÉ P.: Mixing fluids and granular materials. *Computer Graphics forum 28*, 2 (March 2009), 213–228. 2, 5

[LLRR08] LINSEN L., LONG T., ROSENTHAL P., ROSSWOG S.: Surface extraction from multi-field particle volume data using multi-dimensional cluster visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1483–1490. 16

[LQB05] LASTIWKA M., QUINLAN N. J., BASA M.: Adaptive particle distribution for smoothed particle hydrodynamics. *Int. J. Numerical Methods in Fluids 46*, 10-11 (2005), 1403–1409. 10

[LSRS99] LOMBARDI J. C., SILLS A., RASIO F. A., SHAPIRO S. L.: Tests of spurious transport in smoothed particle hydrodynamics. *J. Comput. Phys. 152*, 2 (1999), 687–735. 13

[LTKF08] LOSASSO F., TALTON J., KWATRA N., FEDKIW R.: Two-way coupled SPH and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (2008), 797–804. 7, 16

[Luc77] LUCY L.: A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal 82* (1977), 1013–1024. 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 154–159. 2, 5, 12, 13, 14, 15

[MFZ97] MORRIS J., FOX P., ZHU Y.: Modeling low Reynolds number incompressible flows using SPH. *Journal of Computational Physics 136*, 1 (1997), 214–226. 2

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Commun. 18*, 2 (2007), 109–118. 15

[MK09] MONAGHAN J., KAJTAR J.: SPH particle boundary forces for arbitrary boundaries. *Computer Physics Communications 180*, 10 (2009), 1811–1820. 9

[MM97] MORRIS J., MONAGHAN J.: A switch to reduce SPH viscosity. *Journal of Computational Physics 136*, 1 (1997), 41–50. 5, 9

[MM13] MACKLIN M., MUELLER M.: Position based fluids. *ACM Transactions on Graphics (Proceedings SIGGRAPH) 32* (2013), 1–5. 4, 5, 6, 15, 16, 17

[Mon89] MONAGHAN J. J.: On the problem of penetration in particle methods. *Journal of Computational Physics 82* (1989), 1–15. 2

[Mon92] MONAGHAN J.: Smoothed particle hydrodynamics. *Ann. Rev. Astron. Astrophys. 30* (1992), 543–574. 2, 3, 12

[Mon94] MONAGHAN J.: Simulating free surface flows with SPH. *Journal of Computational Physics 110*, 2 (1994), 399–406. 2, 5, 9, 13

[Mon05] MONAGHAN J.: Smoothed particle hydrodynamics. *Reports on Progress in Physics 68*, 8 (2005), 1703–1759. 2, 9

[Mon12] MONAGHAN J.: Smoothed particle hydrodynamics and its diverse applications. *Annual Review of Fluid Mechanics 44*, 1 (2012), 323–346. 2

[Mor96] MORRIS J.: *Analysis of Smoothed Particle Hydrodynamics with Applications*. PhD thesis, Monash University, 1996. 2

[Mor00] MORRIS J.: Simulating surface tension with smoothed particle hydrodynamics. *Int. J. Numer. Meth. Fluids 33*, 3 (2000), 333–353. 13

[MR13] MONAGHAN J., RAFIEE A.: A simple SPH algorithm for multi-fluid flow with high density. *International Journal for Numerical Methods in Fluids 71*, 5 (2013), 537–561. 13

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 237–244. 2, 12, 13, 16

[MST*04] MÜLLER M., SCHIRM S., TESCHNER M., HEIDELBERGER B., GROSS M.: Interaction of fluids with deformable solids. *Computer Animation and Virtual Worlds 15*, 34 (2004), 159–171. 9

[NMM*06] NEOPHYTOU N., MUELLER K., MCDONNELL K., W.HONG, GUAN X., QIN H., KAUFMAN A.: GPU-accelerated volume splatting with elliptical RBFs. In *IEEE VGTC Conference on Visualization* (2006), EUROVIS'06, pp. 13–20. 16

[NP00] NUGENT S., POSCH H.: Liquid drops and surface tension with smoothed particle applied mechanics. *Phys. Rev. E. 62*, 4 (2000), 4968–4975. 14

[OCD13] ONDERIK J., CHLÁDEK M., DURIKOVIC R.: SPH with small scale details and improved surface reconstruction. In *Proceedings of the 27th Spring Conference on Computer Graphics* (2013), SCCG '11, pp. 29–36. 14

[OHB*13] ORTHMANN J., HOCHSTETTER H., BADER J., BAYRAKTAR S., KOLB A.: Consistent surface model for SPH-based fluid transport. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2013), SCA '13, ACM, pp. 95–103. 2, 9, 13, 14, 15

[OK12] ORTHMANN J., KOLB A.: Temporal blending for adaptive SPH. *Computer Graphics Forum 31*, 8 (2012), 2436–2449. 2, 4, 10, 11

[OKK10] ORTHMANN J., KELLER M., KOLB A.: Topology-caching for dynamic particle volume raycasting. In *Proceedings Vision, Modeling and Visualization (VMV)* (2010), pp. 147–154. 15, 16

[OS03] OTT F., SCHNETTER E.: A modified SPH approach for fluids with large density differences, 2003. 13

[PAKF13] PATKAR S., AANJANEYA M., KARPMAN D., FEDKIW R.: A hybrid lagrangian-eulerian formulation for bubble generation and dynamics. In *Proceedings Eurographics/ACM Siggraph Symposium on Computer Animation* (2013). 16

[PDC*03] PURCELL T. J., DONNER C., CAMMARANO M., JENSEN H. W., HANRAHAN P.: Photon mapping on programmable graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware* (2003), pp. 41–50. 3

[PF01] PASCUCCI V., FRANK R. J.: Global static indexing for real-time exploration of very large regular grids. In *Proceedings of the ACM/IEEE Conference on Supercomputing* (2001), pp. 2–2. 4

[PH10] PELFREY B., HOUSE D.: Adaptive neighbor pairing for smoothed particle hydrodynamics. *Advances in Visual Computing 6454* (2010), 192–201. 3

[PTB*03] PREMOZE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R.: Particle-based simulation of fluids. *Computer Graphics Forum (Proceedings of Eurographics) 22* (2003), 401–410. 5, 7, 15

[RBH*13] RUSTICO E., BILOTTA G., HERAULT A., NEGRO C. D., GALLO G.: Advances in multi-GPU smoothed particle hydrodynamics simulations. *IEEE Transactions on Parallel and Distributed Systems 99*, PrePrints (2013), 1. 4

[RTW13] REICHL F., TREIB M., WESTERMANN R.: Visualization of big SPH simulations via compressed Octree grids. In *IEEE International Conference On Big Data* (2013), pp. 71 – 78. 16

[RWT11] RAVEENDRAN K., WOJTAN C., TURK G.: Hybrid smoothed particle hydrodynamics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), pp. 33–42. 2, 5, 7

[SB12] SCHECHTER H., BRIDSON R.: Ghost SPH for animating water. *ACM Transactions on Graphics (Proceedings SIGGRAPH) 31*, 4 (2012), 61:1–61:8. 2, 3, 9, 12, 13

[SBH09] SIN F., BARGTEIL A. W., HODGINS J. K.: A point-based method for animating incompressible flow. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), pp. 247–255. 3, 5

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *ACM Transactions on Graphics (Proceedings SIGGRAPH) 30*, 4 (2011), 72:1–72:8. 10, 11

[SHG09] SATISH N., HARRIS M., GARLAND M.: Designing efficient sorting algorithms for manycore GPUs. *Parallel and Distributed Processing Symposium, International 0* (2009), 1–10. 4

[SHZO07] SENGUPTA S., HARRIS M., ZHANG Y., OWENS J. D.: Scan primitives for GPU computing. In *Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware* (Aire-la-Ville, Switzerland, Switzerland, 2007), GH '07, Eurographics Association, pp. 97–106. 4

[SL03] SHAO S., LO Y.: Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Advances in water resources 26*, 7 (2003), 787–800. 7

[SP08] SOLENTHALER B., PAJAROLA R.: Density contrast SPH interfaces. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), pp. 211–218. 12, 13

[SP09a] SCHLEGEL P., PAJAROLA R.: Layered volume splatting. In *Proc. Int. Symp. on Visual Computing (ISVC)* (2009), vol. 5876 of *Lecture Notes in Computer Science*, Springer, pp. 1–12. 16

[SP09b] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Transactions on Graphics (Proceedings SIGGRAPH) 28* (2009), 40:1–40:6. 2, 5, 6, 7

[SSP07] SOLENTHALER B., SCHLÄFLI J., PAJAROLA R.: A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds 18*, 1 (2007), 69–82. 9, 11, 14, 16

[Sta99] STAM J.: Stable fluids. *Proceedings of Computer graphics and interactive techniques* (1999), 121–128. 18

[THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANETS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. In *Proceedings of Vision, Modeling, Visualization (VMV)* (2003), pp. 47–54. 2, 4

[TM05a] TARTAKOVSKY A., MEAKIN P.: Modeling of surface tension and contact angles with smoothed particle hydrodynamics. *Physical Review E 72*, 2 (2005), 026301. 14

[TM05b] TARTAKOVSKY A., MEAKIN P.: A smoothed particle hydrodynamics model for miscible flow in three-dimensional fractures and the two-dimensional Rayleigh-Taylor instability. *J. Comp. Phys. 207* (2005), 610–624. 12

[TRS06] THÜREY N., RÜDE U., STAMMINGER M.: Animation of open water phenomena with coupled shallow water and free surface simulations. *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), 157–164. 16

[VBDRC12] VALDEZ-BALDERAS D., DOMINGUEZ J. M., ROGERS B. D., CRESPO A. J.: Towards accelerating smoothed particle hydrodynamics simulations for free-surface flows on multi-GPU clusters. *ArXiv e-prints* (oct 2012). 4

[vdLGS09] VAN DER LAAN W. J., GREEN S., SAINZ M.: Screen space fluid rendering with curvature flow. In *Proc. Symp. Interactive 3D Graphics & Games* (2009), pp. 91–98. 4, 15, 16, 17

[Ver67] VERLET L.: Computer experiments on classical fluids. i. thermodynamical properties of Lennard-Jones molecules. *Physical Review 159*, 1 (1967), 98–103. 3

[Wes90] WESTOVER L.: Footprint evaluation for volume rendering. *Computer Graphics 24*, 4 (1990), 367–376. 16

[YLHB09] YUE G., LI C., HU S., BARSKY B.: Simulating gaseous fluids with low and high speeds. *Computer Graphics Forum 28*, 7 (2009), 1845–1852. 7

[YT10] YU J., TURK G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, 2010), SCA '10, Eurographics Association, pp. 217–225. 2, 5, 14

[YT13] YU J., TURK G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Trans. Graph. 32*, 1 (Feb. 2013), 5:1–5:12. 2, 5, 15

[YWH*09] YAN H., WANG Z., HE J., CHEN X., WANG C., PENG Q.: Real-time fluid simulation with adaptive SPH. *Comput. Animat. Virtual Worlds 20* (2009), 417–426. 4

[YWTY12] YU J., WOJTAN C., TURK G., YAP C.: Explicit mesh surfaces for particle based fluids. *EUROGRAPHICS 2012 30* (2012), 41–48. 14, 15

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Transactions on Graphics (Proceedings SIGGRAPH) 24* (2005), 965–972. 14

[ZGHG10] ZHOU K., GONG M., HUANG X., GUO B.: Data-parallel Octrees for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics 17*, 5 (2010), 669–681. 4, 16

[ZPvBG01] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Surface splatting. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 371–378. 15

[ZRL*08] ZHOU K., REN Z., LIN S., BAO H., GUO B., SHUM H.: Real-time smoke rendering using compensated ray marching. *ACM Trans. on Graphics 27* (2008), 36:1–36:12. 16

[ZSL*13] ZHANG F., SHEN X., LONG X., HU L., ZHAO B.: A particle model for fluid simulation on the multi-graphics processing unit. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields 26*, 4 (2013), 397–414. 4

[ZSP08] ZHANG Y., SOLENTHALER B., PAJAROLA R.: Adaptive sampling and rendering of fluids on the GPU. In *Proceedings Symposium on Point-Based Graphics* (2008), pp. 137–146. 3, 4, 10, 15

[ZYF10] ZHU B., YANG X., FAN Y.: Creating and preserving vortical details in SPH fluid. *Computer Graphics Forum 29*, 7 (2010), 2207–2214. 2, 5