Mass Preserving Multi-Scale SPH

Pixar Technical Memo #13-04

Christopher Jon Horvath
Pixar Animation Studios

Barbara Solenthaler
ETH Zurich

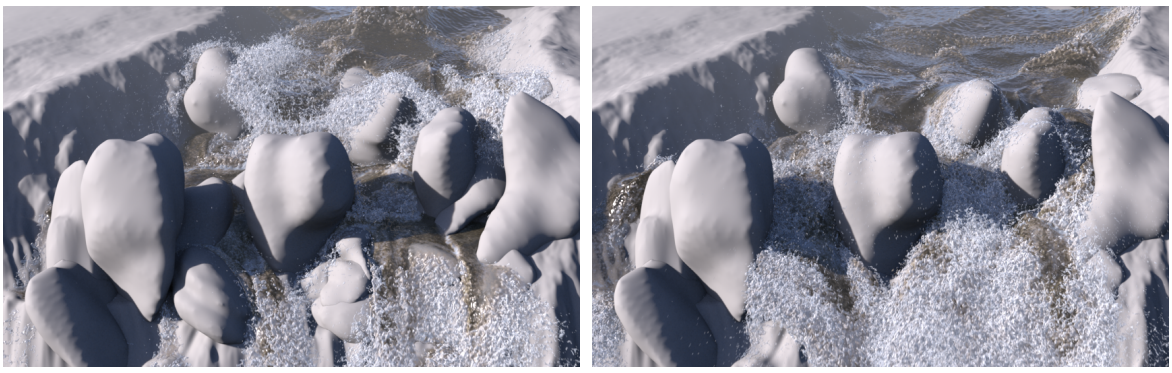

Figure 1: *Multiple frames from river splash simulated with three simultaneous levels of detail, reducing the particle count from 23 million to less than 3 million.*

Abstract

We present a method for performing very high resolution, incompressible fluid simulations at multiple resolutions of detail, simultaneously. The particle-based method supports user-defined regions of refinement and can handle complex collision boundary conditions while remaining smooth and stable. Conservation of mass in refined detail levels is handled explicitly, overcoming mass-loss problems of previous multi-scale SPH techniques. By restricting fine-resolution particles to regions defined relatively near to the fluid surface and close to an observer, the computation time and storage requirements for large simulations are significantly reduced. Compared to an equivalent single-scale simulation, our method reduces the computational costs up to a factor of twelve.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Display Algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Radiosity;

Keywords: fluid simulation, smoothed particle hydrodynamics, multi-scale

1 Introduction

The visual quality of a particle-based fluid simulation depends substantially on the number of particles that are used to discretize the fluid volume. The results of previous work show clearly how the underlying resolution affects the resulting surface complexity and the amount of dissipation, indicating that high resolution simulations are necessary to achieve the desired visual quality. Over the past years, the particle number presented in the computer graphics literature has increased considerably. However, it is still very challenging to compute simulations in the order of tens of million particles on desktop computers within a given time frame.

Adaptive methods are a very attractive technology because of the considerable savings in computation time and memory consumption as shown in, e.g., [Losasso et al. 2004] and [Chentanez and Müller 2011] for grid-based solvers. These techniques allocate resources to visually interesting regions only. This idea was adopted in the two-scale method in [Solenthaler and Gross 2011] for a particle-based solver, where two separate but coupled simulations are used, one for each resolution level. Compared to previous merging and splitting approaches, e.g., [Desbrun and Cani 1999; Adams et al. 2007], separate simulations offer more control over resource allocation and allow much larger resolution ratios.

The two-scale method has shown to significantly increase the overall efficiency. However, the method is inherently limited by its boundary emission scheme to simple, box-shaped domains. Complex collision boundaries produce instabilities as particles are densely sampled in curvature regions. Another limitation is that the single-scale SPH advantage of trivial mass conservation is lost, since particles are dynamically added and removed in the high-resolution region. Even in simple scenarios mass loss is observed, and becomes particularly problematic around complex collision geometry and in areas of splashing.

This paper is inspired by the previous two-scale model, but addresses the aforementioned limitations. We believe that the proposed components are necessary to improve the utility of the method in practice. We present a novel particle emission strategy that tracks the total mass and compensates for the volume loss, hence retaining the advantage of single-scale particle solvers. This necessarily requires an increase of emitted particles, and therefore sophisticated strategies are needed to handle dense and irregular sampling. With the proposed components, complex collision boundaries can be handled. Furthermore, we present how the initial model can be extended to arbitrary resolution levels, and demonstrate the solver's performance and level-of-detail capability in var-

ious complex examples with million of particles.

2 Previous Work

The standard SPH model, presented in [Monaghan 1992], benefits from simplicity as it primarily handles the simulation of compressible fluids. The pressures are related to the deviation from a reference density by the equation of state, and the fluid stiffness is controlled by a single parameter. While this results in fairly efficient simulations [Müller et al. 2003], the severe compressibility artifacts prevent that the original model is used for realistic water animations. Although incompressibility in SPH can be achieved by increasing the stiffness value, referred to as weakly compressible SPH (WCSPH) [Monaghan 1994; Becker and Teschner 2007], this imposes a severe time step restriction and thus prevents the model to be used for high-resolution simulations. This limitation has been addressed in PCISPH [Solenthaler and Pajarola 2009], where density errors are iteratively predicted and pressure values are adapted accordingly. Evidence shows that such an iterative scheme allows time steps that are more than a magnitude larger than in WCSPH, which is a pre-requisite for computing large particle numbers within a reasonable time frame. The basic formulation of PCISPH has been extended to allow adaptive time steps in [Ihmsen et al. 2010b]. Alternative approaches to enforce incompressibility use pressure projection schemes, e.g., [Cummins and Rudman 1999; Shao 2006; Bodin et al. 2012], or propose to apply the Poisson solve on a coarse background grid and then refine the pressure values on the particles, e.g., using PCISPH as in [Raveendran et al. 2011].

In order to reduce the computation time and memory consumption, hashing and sorting methods for the neighbor search which run efficiently on multi-core CPUs have been analyzed [Ihmsen et al. 2010a]. Various GPU implementations have been presented as well, mainly focusing on real-time performance with comparably small particle numbers, e.g., [Harada et al. 2007; Goswami et al. 2010].

For high-resolution simulations, level-of-detail methods have been presented reducing the overall particle count and thus the computational workload. The basic idea is to retain a high discretization only in visually important areas of the fluid. This has been achieved by merging and splitting particles dynamically, triggered by either geometrical or physical criteria [Desbrun and Cani 1999; Kitsionas and Whitworth 2002; Lastiwka et al. 2005; Adams et al. 2007; Hong et al. 2008]. With these approaches, particles of different sizes interact with each other, posing certain difficulties in conserving momentum and retaining stability in the resulting pressure field. This issue can be avoided by using distinct but coupled simulations for each resolution level, as presented by the two-scale method in [Solenthaler and Gross 2011]. In their work, large resolution differences can be simulated stably with either SPH or PCISPH, and a speed-up factor of three to six is reported compared to the single-scale solution. Inherent limitations of the two-scale method are, however, the restriction to box-shaped boundaries and gradual mass loss as the total volume is not considered in the particle emitting strategy. The novel multi-scale method presented in this paper builds upon the two-scale approach, and addresses the intrinsic problems of the original formulation.

In Eulerian fluid simulations, adaptive methods are more common. These approaches place more grid cells in visually interesting areas, as for example presented in [Losasso et al. 2004] where an octree data structure is used, or in [Feldman et al. 2005; Klingner et al. 2006] that include tetrahedral grids. Since the complexity of the surface influences the visual quality significantly, [Kim et al. 2009] tracks the surface on a higher resolution grid than the underlying simulation, resulting in high-frequency features and surface sheets. In [Chentanez and Müller 2011], a layer of tall cells is coupled with regular cubic cells on top of it, building upon the previous work of [Irving et al. 2006]. Also hybrid methods have been popular that couple, for example, a grid with fine SPH particles to achieve spray and droplets [Losasso et al. 2008], or the FLIP model [Zhu and Bridson 2005] which allows multi-resolution pressure solving on a single resolution of particles.

3 Two-Scale Overview

As the presented method builds upon the two-scale model of [Solenthaler and Gross 2011], we first give an overview of the original approach. We briefly discuss SPH and the extension to two resolution levels in Section 3.1 and illustrate the particle creation process in Section 3.2.

3.1 Solver and Resolution Levels

The fundamental equation of Smoothed Particle Hydrodynamics (SPH) [Monaghan 1992] interpolates any scalar or vector field, A_i, stored at a particle with position $\mathbf{x}_i$, from neighboring particles, j, as $A_i = \sum_j m_j/\rho_j A_j W(\mathbf{x}_i - \mathbf{x}_j, h)$. In this equation, m_j denotes the mass of particle j, ρ_j its density, and $W(\mathbf{x}_i - \mathbf{x}_j, h) = W_{ij}$ the kernel function. We use the cubic spline kernel of [Monaghan 1992] with s being the particle spacing and $h = 2s$ the kernel support, similar to [Solenthaler and Gross 2011]. A detailed introduction into SPH is given in [Monaghan 2005]. Instead of using Monaghan's equation of state to compute the pressures, incompressibility can be enforced with PCISPH [Solenthaler and Pajarola 2009] where pressures are iteratively computed until all density errors are below a threshold, e.g., 1%. Our proposed multi-scale method supports, but is not limited to, SPH and PCISPH.

The two-scale method introduced in [Solenthaler and Gross 2011] is based on SPH and PCISPH and follows the idea to simulate two different resolution levels in separate but coupled simulations. The advantages of this representation over merging and splitting approaches are discussed in the original paper. Both resolution levels are represented by a set of particles of a fixed radius per resolution. They refer to the coarse level as L, and the fine level representing only a subset of the fluid as H. The particles in L are marked as either being $active$ or $inactive$, depending if they lie within the pre-defined, fine-scale area of the fluid. A third tag, $boundary$, has been introduced between $active$ and $inactive$. This setup is illustrated in Figure 2. All fine particles in the $boundary$ layer are advected by their low-resolution parent particle until they enter the $active$ region. This serves as a boundary condition, defined by L, for the fine-scale, $active$ particles in H. Particles are dynamically added and deleted at the $boundary$ layer as discussed in Section 3.2.

3.2 Particle Creation and Total Mass

Within a single scale SPH simulation without refinement, mass conservation is trivial - each particle retains its per-particle mass, and the particles are only deleted when they leave the simulation domain or enter a sink volume. This intrinsic conservation of mass is one of the most attractive features of Lagrangian particle simulations.

With multiple refinement scales, however, particles are being dynamically added and removed from the simulation. In [Solenthaler and Gross 2011], fine children particles are emitted into the fluid boundary region only in a time step where the coarse-parent has just transitioned from $inactive$ to $boundary$. Eight fine particles are emitted in a cube configuration centered around the coarse particle, assuring to move these emitted particles outside solid objects.

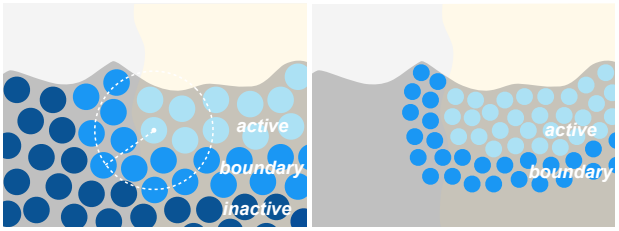

Figure 2: *A two-scale example with the coarse and fine resolution levels. The fluid volume is shown in grey, and a user-specified refinement volume in light yellow. Active particles are light blue, boundary particles are blue, and inactive particles are dark blue. In this illustration, and in all our multi-scale examples, only particles which are sufficiently close to the surface are refined.*

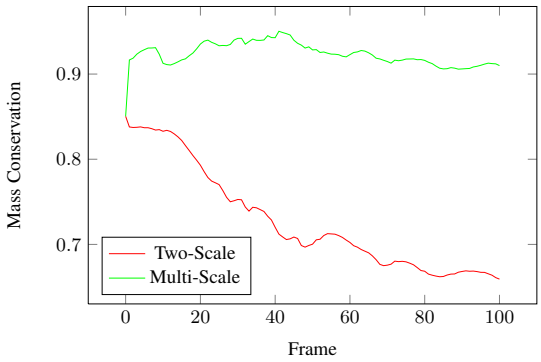

Figure 3: *Mass conservation comparison. With the original two-scale model, a mass loss of 22% is observed in our dam break example, while the total mass is conserved with the proposed emission strategy.*

Applying the same strategy to complex geometries results in instabilities as extensive oversampling near the solid is introduced. Oversampled particles can also be observed at the *boundary* layer of the fine level. Since those particles are advected, the dense sampling is less critical in terms of stability. However, mass loss is observed as it may happen that too many particles in H are deleted if their parent in L changes from *boundary* to *inactive*. We see gradual mass loss occurring over the course of a long simulation, especially during very dynamic and splashy parts of the evolution. In our dam break test, we observe a mass loss in one refinement scale of 22% over six seconds of simulated time (see Figure 3, red curve, and Figure 10, top row).

4 Multi-Scale Method

We have replaced many critical components of the two-scale model that are necessary to simulate multiple resolution levels for complex environments that are typical in production scenarios. We first discuss in Section 4.1 how we change the method to support arbitrary numbers of levels. In Section 4.2, we then present a robust fluid boundary emission which conserves the total mass and maintains boundary integrity. As the emission strategy produces oversampled particle regions, we present an elaborated relaxation in Section 4.3. We introduce a particle-dependent relaxation coefficient, which is considered in the SPH equations and the final displacement control. With this, the rest volume of a particle is gradually approached, eliminating any instabilities caused by the irregular sampling. In order to keep our refinement confined to regions near the fluid surface,

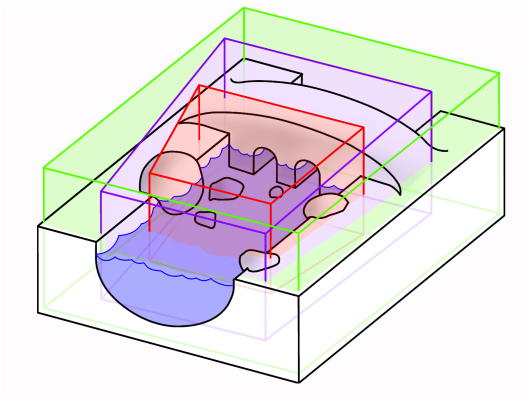

Figure 4: *Example system definition. The solid bridge and riverbed objects in white, a fluid initial state in blue. The simulation domain bounds are in green, with two frustum-shaped regions of refinement in purple and red.*

we define an improved and temporally smooth fluid surface depth calculation based on the equations for a level set in Section 4.4. Lastly, we show how the top-level time-step for the system is organized in Section 4.5, which recursively computes all scales of detail starting from the coarsest.

4.1 Terminology and Level Definition

In contrast to the two-scale model, our method is not restricted to two resolution levels only. We label these levels, referred to as states in the following, explicitly as $(L_0, L_1, ..., L_{n-1})$. Hence, for any state L_r, the coarse-parent state is L_{r-1} and the fine-child state is L_{r+1}. The coarsest state, L_0, has a user-specified particle radius, r_0, and each subsequent refined state's particles are always half the radius of the previous, coarser state's, thus $r_r = 0.5 r_{r-1}$. This restriction on the refinement ratio, which is not applied by the two-scale method in [Solenthaler and Gross 2011], reduces the implementation complexity considerably.

At the coarsest state, the simulation environment is defined by a static domain bound, a fluid initial state volume, static and moving collision volumes, fluid sources, and fluid sinks. Each subsequent refined state is additionally defined by a refinement volume, which is user-defined and may be dynamically changing. Refinement volumes can be designed by the user to represent, for example, frustums of visibility relative to an observer or areas of interest around a particular collision object, as illustrated in Figure 4. Each of the refined states other than the coarsest are only defined inside of their respective refinement volumes, and are furthermore restricted to only a relatively shallow depth beneath the surface of the fluid interface, as illustrated in Figure 2, which significantly decreases the particle count.

As in the two-scale method, the particles of state L_r can be classified at a specific time according to the refinement volume of the fine-children state L_{r+1}. Particles which are inside the refinement volume and sufficiently close to the fluid surface are marked as *active*. Particles which are not marked *active*, but are within a distance of $4r$ of any neighboring *active* particle, are marked as *fboundary*, and the remaining particles are marked *inactive*. Following the two-scale approach, each refined particle has a single particle in the next coarser state that it is coupled to, called the *coarse-parent*. We mark each refined particle according to the classification of its coarse-parent as *pActive*, *pFboundary*, or *pInactive*. We additionally allow for refined particles to have no coarse-parent if they're sufficiently

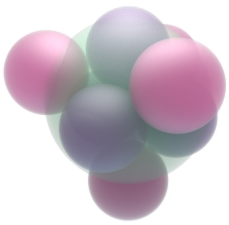

Figure 5: *Children emission pattern. Purple and rose spheres are the inner and outer tetrahedral points, respectively, and the green sphere is the coarse point.*

isolated, and these orphaned particles are marked *pUndefined*.

4.2 Mass Conserving Particle Emission

To solve the problem of mass conservation, we consider the entire fluid boundary region as a target for emission of fine particles, and we emit or delete fine particles from each coarse *fboundary* particle to achieve a target fine particle density. Emission of new particles is restricted to the fluid boundary region, in order to prevent visible mass changes within the refinement volume.

In the coarse-parent state L_{r-1}, we can count the number of particles marked *active* and *fboundary*, which we label N^a_{r-1} and N^{fb}_{r-1}, respectively. In this state L_r, we should expect to find exactly $8(N^a_{r-1} + N^{fb}_{r-1})$ particles. Given the total number of particles in L_r, N_r, the change in number of particles can be computed as

$$\Delta N_r = 8(N^a_{r-1} + N^{fb}_{r-1}) - N_r.$$

We count the number of existing *pFboundary* particles at this scale, N^{pfb}_r, add the change in number of particles ΔN_r, and then compute a desired average number of children per coarse *fboundary* parent:

$$\text{NumChildren}^{avg}_{r-1} = (N^{pfb}_r + \Delta N_r)/N^{fb}_{r-1}. \quad (1)$$

For each coarse *fboundary* particle, we compare the desired average number of children to the number of existing children, and then either delete or emit new particles to achieve the desired number of $\text{NumChildren}^{avg}_{r-1}$, as computed in Equation (1). For a specific coarse particle i, the change in number of children is given as

$$\Delta\text{NumChildren}^i_{r-1} = \text{NumChildren}^{avg}_{r-1} - \text{NumChildren}^i_{r-1}. \quad (2)$$

If $\Delta\text{NumChildren}^i_{r-1}$ is positive, we attempt to emit new children, and otherwise we delete existing children.

Emitting new children particles is done by taking a fixed emission pattern consisting of eight points, as seen in Figure 5, centering it on the coarse parent particle, and randomly rotating it. We only allow for a maximum of eight newly emitted fine points per each coarse parent point, per iteration. Each point in the emission pattern is considered a candidate point, and is tested against existing fine particles. If any existing fluid particle overlaps a candidate particle by more than $\epsilon_{sepf}\ r$, or any existing solid particle overlaps a candidate particle by more than $\epsilon_{seps} r$, the candidate is not emitted. For the case that not as many points as desired can be emitted in the current time step, this procedure is repeated in the next time step. In our implementation, we use $\epsilon_{sepf} = 0.5$ and $\epsilon_{seps} = 0.1$. Figure 6 shows a two-dimensional projection of the fine particle emission process.

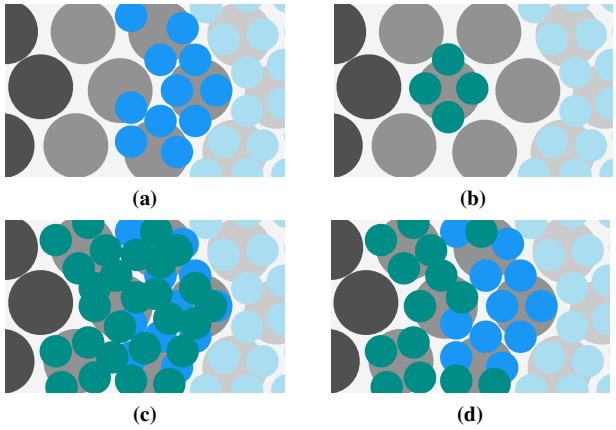

Figure 6: *Emission strategy. The coarse active, fboundary, and inactive particles are shown in light gray, gray, and dark gray. Fine particles pActive, pFboundary, and emitCandidate are illustrated in light blue, blue, and green. (a) Existing fine particles over coarse parents. (b) Emission candidates for a single coarse boundary particle. (c) All emission candidates. (d) Final emitted pFboundary particles.*

Deleting fine child particles does not have any spatial constraints, and therefore is more likely to occur than the corresponding emission of particles. In order to achieve an even deletion and emission of particles, we test a random roll for each deletion candidate against a deletion probability. Based on measurements of the simulations, a deletion probability of $p_{del} = 0.6$ has proven to balance the emission and deletion correctly.

4.3 Relaxation

The presented emission strategy typically results in over-concentrated regions of particles in the $pFboundary$ layer. At this stage, this is not critical as particles are still advected by their parent in the next coarser state. When a particle enters $pActive$, however, high pressure forces are computed to counterbalance the compression. For a smooth transition between the layers, we apply a relaxation scheme similar to [Solenthaler and Gross 2011]. As our emission strategy creates much more particles in order to compensate the mass loss, and also due to the inclusion of complex boundaries, an elaborated relaxation process is required.

Relaxation Coefficient. We introduce a relaxation coefficient ξ_i for each particle, which represents how much sampling regularity we expect from that particle. When the relaxation coefficient is zero, as for *pFboundary* particles, it signifies that we expect zero sampling regularity from this particle, and therefore zero pressure force to correct overlapping. Conversely, when the relaxation coefficient is one, as for most *pActive* and all *pUndefined* particles, it signifies that the particle is no longer near the fluid boundary, nor recently part of the fluid boundary, and is expected to be evenly sampled.

The relaxation coefficient ξ_i considers the age and the proximity to the boundary, given by two coefficients ξ^{age}_i and ξ^{prox}_i, and is defined as

$$\xi_i = \begin{cases} 0 & \text{if } pFboundary, \\ 1 & \text{if } pUndefined, \\ \min(\xi^{age}_i, \xi^{prox}_i) & \text{if } pActive. \end{cases} \quad (3)$$

The first relaxation coefficient, ξ_i^{age}, corresponds to the normalized number of simulation steps for which a particle has been marked *pActive*, which is similar to the timestep-based relaxation in [Solenthaler and Gross 2011]. The second coefficient, ξ_i^{prox}, is defined by proximity to the fluid boundary region, with a value of 0 in or at the fluid boundary, and gradually easing to 1 at a distance equal to the kernel support radius s.

Blending Volumes. We compute an adjusted sampling volume V^{eq} for each particle, using ξ_i to blend between a sampling volume that is fully normalized against the particle's neighborhood at the beginning of the time step, versus the regular, constant SPH volume $V_0 = m/\rho_0$. Note that for incompressible fluids $\rho_i = \rho_0$. This condition may be violated near the boundary due to over-dense sampling.

The sampling volume is computed with the SPH summation as $V_i = 1/\sum_j W_{ij}$, and is then used to blend between the sampled volume and the rest volume, given as

$$V_i^{eq} = (1 - \xi_i)V_i + \xi_i V_0. \tag{4}$$

Using V_i^{eq}, we can rewrite the original equations of ([Monaghan 1992]) for the density and acceleration due to pressure as

$$\rho_i = \rho_0 \sum_j V_j^{eq} W_{ij}, \tag{5}$$

$$\frac{\partial \mathbf{v}_i^p}{\partial t} = -\sum_j V_j^{eq} (\frac{p_i}{\rho_i} + \frac{p_j}{\rho_j}) \nabla W_{ij}. \tag{6}$$

Displacement Restriction. Similar to [Solenthaler and Gross 2011], we carefully blend the velocity computed from physics $\hat{\mathbf{v}}_i(t + \Delta t)$ and the velocity interpolated from the coarse parent state $\tilde{\mathbf{v}}_i(t + \Delta t)$ with

$$\mathbf{v}_i(t + \Delta t) = (1 - \xi_i)\tilde{\mathbf{v}}_i(t + \Delta t) + \xi_i \hat{\mathbf{v}}_i(t + \Delta t). \tag{7}$$

We always use the physically computed velocity when updating the position of particles, but save the mixed velocity for integration into the next time step.

We restrict the final displacement due to pressure forces for particles undergoing relaxation by applying an intra-scale velocity constraint, which limits the difference between $\tilde{\mathbf{v}}$ and $\hat{\mathbf{v}}$. The maximum velocity difference is given as

$$||\tilde{\mathbf{v}}_i - \hat{\mathbf{v}}_i|| \leq \frac{\epsilon_i^{constraint} r}{\Delta t}, \tag{8}$$

with $\epsilon_i^{constraint} = \xi_i + 1$. Since we do not modify the computed velocity if differences are less than the maximum allowable difference, there is no general damping effect. We use a tapering function to limit the magnitude of the velocity difference smoothly, avoiding visible and aesthetically unpleasing velocity clamps. The tapering function is designed to be equal to the input x until some threshold τ which we set to 0.6 and then to smoothly and asymptotically approach 1:

$$taper(x, \tau) = \begin{cases} x & x \leq \tau, \\ \tau + (1 - \tau)\tanh(\frac{x-\tau}{1-\tau}) & x > \tau. \end{cases} \tag{9}$$

4.4 Fluid Surface Detection

Our system calculates refinement only to a shallow depth beneath the fluid-air interface, which in our implementation is nine times the particle diameter, in order to maximize resource economy (see for example Figure 7). The performance of the system is intimately connected to the accuracy of the surface identification, as refined particles will be emitted and deleted unnecessarily if the region is identified noisily.

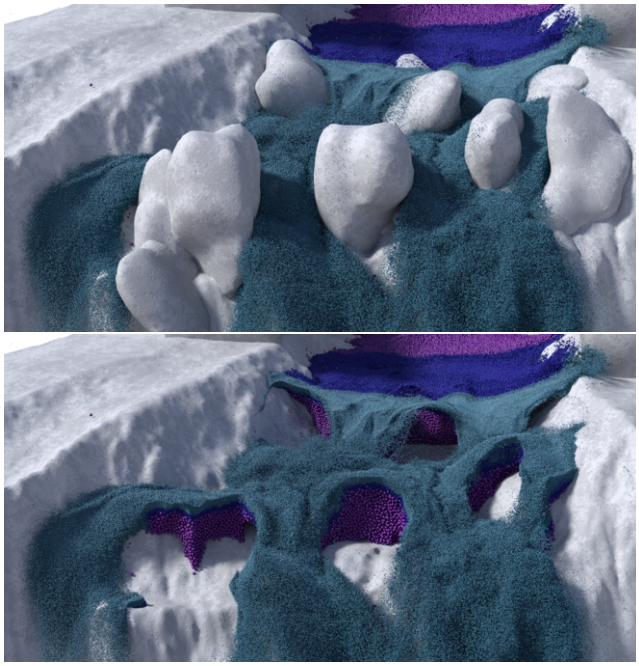

Figure 7: *3-scale river simulation. In the lower image, rocks are invisible to reveal the refinement.*

Various methods have been proposed in the literature for finding particles which are near the surface, e.g., [Solenthaler et al. 2007; Akinci et al. 2012a], but their results are temporally noisy, resulting in significant extra work for the multi-level system. Therefore, we base our detection on the following equations for a level set [Zhu and Bridson 2005], which we evaluate at each particle position $\mathbf{x}_i$:

$$\begin{array}{lcl} \tilde{\phi}_i & = & ||\mathbf{x}_i - \bar{\mathbf{x}}_i|| - \bar{r}_i \\ \bar{\mathbf{x}}_i & = & \sum_j w_{ij}\mathbf{x}_j \\ \bar{r}_i & = & \sum_j w_{ij} r_j \\ w_{ij} & = & W(\mathbf{x}_i - \mathbf{x}_j, h)/\sum_j W(\mathbf{x}_i - \mathbf{x}_j, h). \end{array} \tag{10}$$

We use this measure to calculate an initial value for $\tilde{\phi}_i$, which we then limit the difference of in a time step, also incorporating the cutoff for a small number of neighbors:

$$\begin{array}{lcl} \phi_i(t) & = & \begin{cases} -0.85r & \text{if num neighbors} < 5 \\ \phi_i^{min}(t) & \text{if} \tilde{\phi}_i(t) < \phi_i^{min}(t) \\ \phi_i^{max}(t) & \text{if} \tilde{\phi}_i(t) > \phi_i^{max}(t) \\ \tilde{\phi}_i(t) & \text{otherwise} \end{cases} \\ \phi_i^{min}(t) & = & \phi_i(t - \Delta t) - \Delta\phi^{max} r \\ \phi_i^{max}(t) & = & \phi_i(t - \Delta t) + \Delta\phi^{max} r. \end{array} \tag{11}$$

The constant $\Delta\phi^{max} r$ is the maximum amount of change, proportional to the particle radius, of the level set from the previous time step, and we set it to 1.5 in our simulations. We then mark any particle which has a level set value greater than or equal to $-0.85r$ as *tmpFluidSurface*, and all other particles as *tmpFluidInterior*.

We then iteratively propagate this level set solution out to adjacent particles, as follows. For each particle that is marked *tmpFluidInterior*, we search its neighbors in order of increasing proximity until

we find a neighbor that is marked *tmpFluidSurface* or *tmpNearFluidSurface*. If a neighbor at a given index j is found, we mark the particle *tmpNearFluidSurface* and then compute its level set value as follows:

$$\phi_i(t) = max(\phi_j(t) - dist(\mathbf{x}_i, \mathbf{x}_j), \phi^{min}), \qquad (12)$$

where ϕ^{min} represents the maximal level set value, which in our case is $-9(2r)$, corresponding to nine particle diameters. As before, we limit the maximum change of the level set from the previous time step, see Equation (11). This propagation sweep is continued until there are no changes in the level set values, and typically requires 8 or 9 iterations.

4.5 Implementation

The main time step for a detail level L_r is shown in Algorithm 1, which recursively calls the time step on its fine-children state L_{r+1}. Thus, a single call to the time step at level L_0 will fully update all levels of detail in the simulation.

We utilize the compact hash map acceleration structure described in [Ihmsen et al. 2010a]. We have a separate map for each resolution level, each having access to the structures at other levels.

Collision objects, whether static or moving, are represented at each level of detail as particles, just like the fluid, with the same radius and volume. We take the same approach as in [Akinci et al. 2012b]. Collision objects may be unmoving (static), rigidly transformed, or deforming.

5 Performance and Results

We demonstrate the effectiveness of our multi-scale method on a dam break and a more complex river example. In both simulations, we use a rather large particle count, with 2.7 and 23.7 million particles in their single-scale representations, respectively. All examples use PCISPH and restrict the maximal volume compression to less than 1%, as suggested in [Solenthaler and Pajarola 2009]. Computations are run on an Intel Xeon E5-2680 with 16-cores and 60GB of RAM.

Level-of-detail. Our multi-scale implementations are designed to restrict the fine-resolution particles to regions defined relatively near to the fluid surface and the observer. This is illustrated with the particle renderings for the river scene in Figure 7 and for the dam break in Figure 8. In these images, the resolution states are color-coded with purple corresponding to the coarsest state L_0, blue to L_1, and cyan to the finest state L_2. The resulting, reconstructed surfaces shown in Figures 1 and 8 show clearly that it is sufficient to calculate refinement only to a shallow depth without sacrificing fine splashes and complex flow structures emerging at the surface. This is emphasized by the close-up views in Figure 9.

The inclusion of the view frustum information allows to model camera-dependent level-of-detail, as shown in the river example where the complexity is decreased with increasing distance to an observer. The reduced resolution in distant areas is unnoticed as seen in Figure 1. Note that our implementation is not limited to three scales but can handle an arbitrary number of resolution levels. This allows an artist to gradually control the desired resolution at given distances, depending on the targeted surface complexity or the available computation time. Another notable advantage of camera-dependent level-of-detail is that it allows to initialize particles on a very coarse level outside the viewing frustum, for example in big water tanks. As soon as the particles enter the visible area they can be smoothly refined until the desired resolution is reached.

Algorithm 1: SubTimeStep(L_r, t, $t + \Delta t$)

```
Data: Fluid state L_r
Result: Increment L_r and all refined levels {L_{r+1}, ...}
        from time t to time t + Δt
begin
    UpdateCompactHashMaps();
    DeleteFluidFromSinks( t );
    EmitFluidInSources( t );
    if has coarse parent L_{r-1} then
        UpdateCoarseParentIndices();
        TransferRegionsFromCoarse();
        EmitPfboundaryParticles();
    ReUpdateCompactHashMaps();
    FindNeighborhoods();
    RemoveSignificantOverlaps();
    CalculateFluidSurfaceDepth( Δt );
    if has fine children L_{r+1} then
        CalculateRefinement( t );
        CalculateFineFeedback();
    if has coarse parent L_{r-1} then
        CalculateRelaxation( t );
        InterpolateCoarseToFine();
        CalculateFluidRestVolumes();
    CalculateExternalForces( t );
    CalculateViscousTensionAndPressureForces( t );
    IntegrateVelocity( t, t + Δt );
    if has coarse parent L_{r-1} then
        ConstrainVelocityToCoarse();
    Output( L_r, t );
    if has fine children L_{r+1} then
        SubTimeStep( L_{r+1}, t, t + Δt/2 );
        IntegratePosition( t, t + Δt/2 );
        SubTimeStep( L_{r+1}, t + Δt/2, t + Δt );
        IntegratePosition( t + Δt/2, t + Δt );
    else
        IntegratePosition( t, tΔt );
```

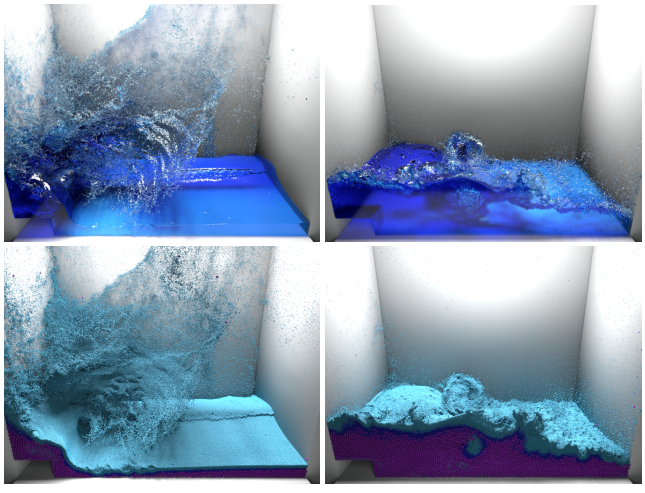

Figure 8: *Mesh (top) and particle renderings (bottom) of the 3-scale dam break simulation.*

In terms of visual quality, a current limitation of our system is the limited functionality of our meshing toolkit for incorporating partial

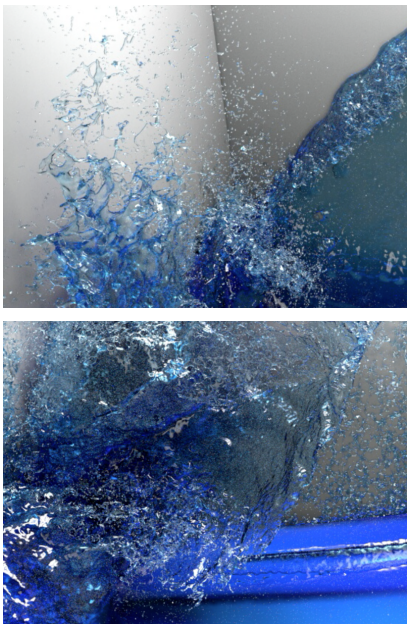

Figure 9: *Close-up views of the dam break simulation showing fine details.*

contributions from particles at different scales. This leads to mesh artifacts at the refinement region edge in the river scene that are not present in the particle renderings. A more robust meshing solution, currently the subject of active development, will solve this issue.

Performance Comparison. The river example demonstrates the solver in a high-resolution, complex scenario. The single-scale reference solution consists of a maximum of 23.7 million particles, resulting in an average per-frame computation time of 4869.23 seconds, i.e., 81 minutes approximately. Apparently, computation times of this magnitude are not acceptable in production environments where simulations have to finish within a given time. In contrast, using three resolution states, the total particle count is decreased by a factor of 8.1 to 2.9 million particles, with 370k, 713k, 1.8m particles in states L_0, L_1, and L_2, respectively (see Table 1). The result is an overall speed-up factor of 12, which is - with our implementation - even more than the particle count factor because of additional overhead associated with such high-resolution examples. In addition, the memory consumption is reduced by a factor of 2 from 16g to 7g. We observe less gain in the memory consumption because the refined states have the solid boundary objects rasterized throughout the entire refinement volume requiring a large amount of storage.

Modest performance gains are achieved in the dam break example regarding computation time (factor of 3) as well as memory consumption (factor of 1.2), see Table 1. The lesser improvement is because the water is very shallow and the entire domain is refined. Thus, the performance gain scales with the fluid depth, or more generally, with the reduction factor of the overall particle count. The small difference in memory consumption is related to the large overhead, relative to the refinement scale, of the solid collision objects in this scene.

Comparison of Mass Conservation. Our method conserves the total mass even for long simulations compared to [Solenthaler and Gross 2011] as illustrated by the green curve in Figure 3. The impact on the visual result is demonstrated in Figure 10 where our multi-scale result is compared to the original two-scale approach.

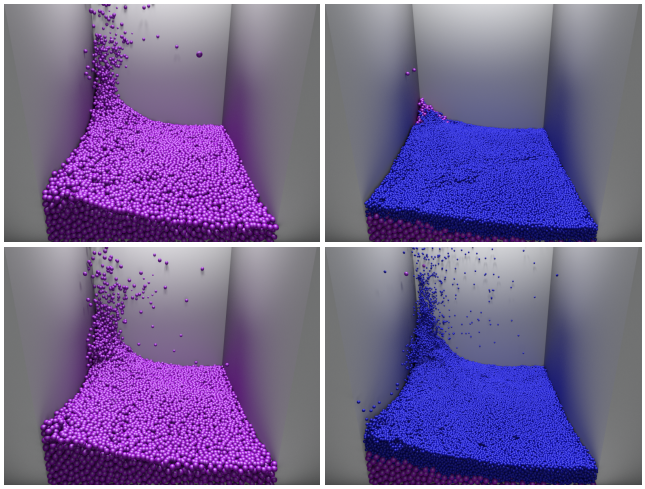

Figure 10: *Comparison of our mass-conserving multi-scale method (below) and the original 2-scale approach (above). Our method conserves the mass of the coarse simulation (left) in the refined particles (right).*

6 Conclusion and Future Work

We have presented a multi-scale method that conserves the total mass and maintains stability even if complex solid geometries are included. The emission scheme tracks the total mass and gradually compensates for the volume loss. The resulting over densely particle sampling is gradually resolved without degrading the stability of the solver. By adding these critical components, the multi-scale method could become a valuable tool for computing production-quality scenarios, in particular, if complex collision environments are included. The total particle count is decreased, and with this memory consumption and computation time.

Although our results show a significant speed-up over the single scale simulation, the potential is not yet fully exploited. Performance is affected as the relaxation constraints impede the PCISPH pressure solve, which results in slower convergence for those particular areas. The same effect was already observed in the previous two-scale approach. In contrast to the two-scale method, the calculation of interpolated data between multiple levels is slower and further decreases the total performance gain.

An inherent problem of SPH is the fluid-air interface as incorrect fluid quantities at the free surface lead to artifacts. This was addressed by the ghost fluid method presented in [Schechter and Bridson 2012], where a narrow layer of ghost particles is used around the free surface. The presented multi-scale approach may offer a new way to model air particles as the entire domain could be simulated efficiently as a multi-scale fluid.

References

ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. 2007. Adaptively sampled particle fluids. *ACM Trans. Graph. (SIGGRAPH Proc.) 26*, 3, 48–54.

AKINCI, G., IHMSEN, M., AKINCI, N., AND TESCHNER, M. 2012. A parallel surface reconstruction for particle-based fluids. *Computer Graphics Forum*.

AKINCI, N., IHMSEN, M., AKINCI, G., SOLENTHALER, B., AND TESCHNER, M. 2012. Versatile rigid-fluid coupling for incom-

Scene	#particles (total)	particle radius	avg comp. time [s] (per-frame)	memory [g]
Dam break, 1-scale	2.76m	0.003	383.09	2.24
Dam break, 3-scale	42.4k / 139.9k / 651.1k *	0.012 / 0.006 / 0.003 *	120.67	1.87
River, 1-scale	23.77m	0.125	4869.23	16.59
River, 3-scale	370.4k / 713.2k / 1.84m *	0.125 / 0.25 / 0.5 *	405.31	7.07

*: L_0, L_1, L_2

Table 1: *Performance comparison of 1-scale and 3-scale for the dam break and the river scene.*

pressible sph. *ACM Trans. Graph. (SIGGRAPH Proc.) 31*, 4, 62:1–62:8.

BECKER, M., AND TESCHNER, M. 2007. Weakly compressible SPH for free surface flows. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 209–217.

BODIN, K., LACOURSIERE, C., AND SERVIN, M. 2012. Constraint fluids. *IEEE Transactions on Visualization and Computer Graphics 18*, 3, 516–526.

CHENTANEZ, N., AND MÜLLER, M. 2011. Real-time eulerian water simulation using a restricted tall cell grid. *ACM Trans. Graph. (SIGGRAPH Proc.) 30*, 4, 82:1–82:10.

CUMMINS, S. J., AND RUDMAN, M. 1999. An SPH projection method. *J. Comput. Phys. 152*, 2, 584–607.

DESBRUN, M., AND CANI, M. P. 1999. Space-time adaptive simulation of highly deformable substances. Tech. rep., INRIA Nr. 3829.

FELDMAN, B. E., OBRIEN, J. F., AND KLINGNER, B. M. 2005. Animating gases with hybrid meshes. *ACM Trans. Graph. (Proc. of SIGGRAPH) 24*, 3, 904–909.

GOSWAMI, P., SCHLEGEL, P., SOLENTHALER, B., AND PAJAROLA, R. 2010. Interactive SPH simulation and rendering on the GPU. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 55–64.

HARADA, T., KOSHIZUKA, S., AND KAWAGUCHI, Y. 2007. Smoothed Particle Hydrodynamics in complex shapes. In *Spring Conference on Computer Graphics*, 235–241.

HONG, W., HOUSE, D. H., AND KEYSER, J. 2008. Adaptive particles for incompressible fluid simulation. *Vis. Comput. 24*, 535–543.

IHMSEN, M., AKINCI, N., BECKER, M., AND TESCHNER, M. 2010. A parallel SPH implementation on multi-core CPUs. *Computer Graphics Forum 30*, 1, 99–112.

IHMSEN, M., AKINCI, N., GISSLER, M., AND TESCHNER, M. 2010. Boundary handling and adaptive time-stepping for PCISPH. In *Proc. of VRIPHYS*, 79–88.

IRVING, G., GUENDELMAN, E., LOSASSO, F., AND FEDKIW, R. 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph. (SIGGRAPH Proc.) 25*, 805–811.

KIM, D., SONG, O.-Y., AND KO, H.-S. 2009. Stretching and wiggling liquids. *ACM Trans. Graph. (SIGGRAPH ASIA Proc.) 28*, 5, 1–7.

KITSIONAS, S., AND WHITWORTH, A. 2002. Smoothed Particle Hydrodynamics with particle splitting, applied to self-gravitating collapse. *MNRAS 330*, 1, 129–136.

KLINGNER, B. M., FELDMAN, B. E., CHENTANEZ, N., AND O'BRIEN, J. F. 2006. Fluid animation with dynamic meshes. *ACM Trans. Graph. (SIGGRAPH Proc.) 25*, 820–825.

LASTIWKA, M., QUINLAN, N., AND BASA, M. 2005. Adaptive particle distribution for Smoothed Particle Hydrodynamics. *Int. J. Numer. Meth. Fluids 47*, 1403–1409.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.) 23*, 3, 457–462.

LOSASSO, F., TALTON, J., KWATRA, J., AND FEDKIW, R. 2008. Two-way coupled SPH and particle level set fluid simulation. *IEEE TVCG 14*, 4, 797–804.

MONAGHAN, J. J. 1992. Smoothed Particle Hydrodynamics. *Annu. Rev. Astron. Physics 30*, 543.

MONAGHAN, J. J. 1994. Simulating free surface flows with SPH. *J. Comput. Phys. 110*, 399–406.

MONAGHAN, J. J. 2005. Smoothed Particle Hydrodynamics. *Rep. Prog. Phys. 68*, 1703–1759.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 154–159.

RAVEENDRAN, K., WOJTAN, C., AND TURK, G. 2011. Hybrid smoothed particle hydrodynamics. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 33–42.

SCHECHTER, H., AND BRIDSON, R. 2012. Ghost sph for animating water. *ACM Transactions on Graphics (SIGGRAPH Proc.) 31*, 4.

SHAO, S. 2006. Incompressible SPH simulation of wave breaking and overtopping with turbulence modelling. *Int. J. Numer. Meth. Fluids 50*, 597–621.

SOLENTHALER, B., AND GROSS, M. 2011. Two-scale particle simulation. *ACM Trans. Graph. (SIGGRAPH Proc.) 30*, 4, 81:1–81:8.

SOLENTHALER, B., AND PAJAROLA, R. 2009. Predictive-corrective incompressible SPH. *ACM Trans. Graph. (SIGGRAPH Proc.) 28*, 3, 1–6.

SOLENTHALER, B., ZHANG, Y., AND PAJAROLA, R. 2007. Efficient refinement of dynamic point data. In *Proc. of the Eurographics Symposium on Point-Based Graphics*, 65–72.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Trans. Graph. (Proc. of SIGGRAPH) 24*, 3, 965–972.