Two-Scale Particle Simulation

Barbara Solenthaler
ETH Zurich

Markus Gross
ETH Zurich

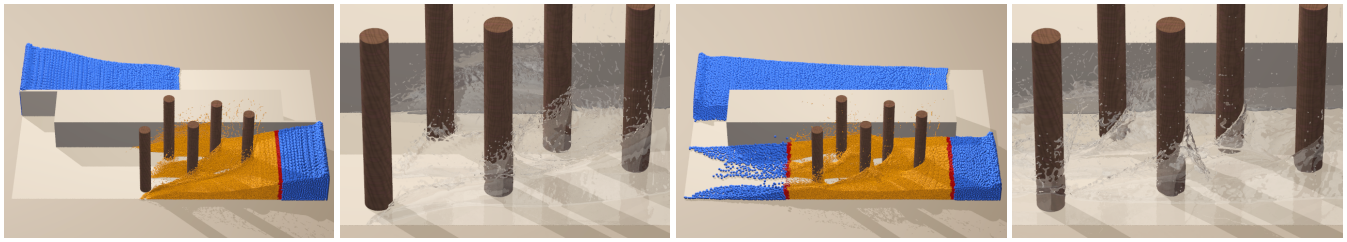


Figure 1: *With our two-scale method, computing resources can be allocated to regions where complex flow behavior emerges, like in this example around cylindrical obstacles. This region is simulated with quadrupled resolution (yellow) to get more surface details and fine-scaled splashes at impact locations. The major remaining part of the fluid is computed with low resolution (blue).*

Abstract

We propose a two-scale method for particle-based fluids that allocates computing resources to regions of the fluid where complex flow behavior emerges. Our method uses a low- and a high-resolution simulation that run at the same time. While in the coarse simulation the whole fluid is represented by large particles, the fine level simulates only a subset of the fluid with small particles. The subset can be arbitrarily defined and also dynamically change over time to capture complex flows and small-scale surface details. The low- and high-resolution simulations are coupled by including feedback forces and defining appropriate boundary conditions. Our method offers the benefit that particles are of the same size within each simulation level. This avoids particle splitting and merging processes, and allows the simulation of very large resolution differences without any stability problems. The model is easy to implement, and we show how it can be integrated into a standard SPH simulation as well as into the incompressible PCISPH solver. Compared to the single-resolution simulation, our method produces similar surface details while improving the efficiency linearly to the achieved reduction rate of the particle number.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation.

Keywords: fluid simulation, SPH, two-scale, level of detail

Links: DL PDF

1 Introduction

Fluid simulations demand a high discretization resolution in order to produce appealing visual results. Often, small-scale details like small droplets, thin sheets and surface ripples are not reproduced in the simulation. On the one hand, they are below the simulation scale, and on the other hand, numerical dissipation and smoothing dampen these effects. To cope with the increasing demand for more detailed flow structures, different methods have been proposed that follow the idea to allocate computing resources to regions where complex flow behavior emerges. Many techniques have been presented for Eulerian simulations, examples are octree data structures [Losasso et al. 2004], coupling of 2D and 3D simulations [Thürey et al. 2006], and dynamic mesh refinement [Klingner et al. 2006].

Only few works have addressed this problem in the Lagrangian context. The physical and visual quality of particle-based solvers like SPH are defined by the number of particles that are used to discretize the fluid. Generally, the more particles that are used, the smaller the damping artifacts and the more small-scale details like splashes, spray, and surface waves can be reproduced. However, doubling the resolution of a simulation increases the particle number by a factor of 8. This increases the computational cost notably since it depends linearly on the number of particles.

While much work has been done in improving the computational efficiency of the solver by using for example GPU implementations, e.g. [Goswami et al. 2010], or by speeding up incompressibility enforcement as shown in [Solenthaler and Pajarola 2009], only few works have explored level of detail techniques. [Adams et al. 2007] proposed a method where large particles are dynamically subdivided into smaller ones and small particles are merged into a larger one to adjust the resolution based on a surface feature criterion. Such an adaptive sampling can reduce the computational cost to some extent. However, difficulties exist in splitting and merging particles so that the density and force profiles are exactly reproduced. Furthermore, it has to be ensured that the spatial discretization features a smooth transition from large to small particles. This limits the maximal particle size difference inside the fluid; [Adams et al. 2007] report of maximal size difference factors of 4-8, i.e., the resolution is doubled in the best case.

In this paper, we adopt the idea of [Adams et al. 2007], but instead of recursively subdividing particles which results in particles of different sizes that interact with each other, we rather use a hierarchy

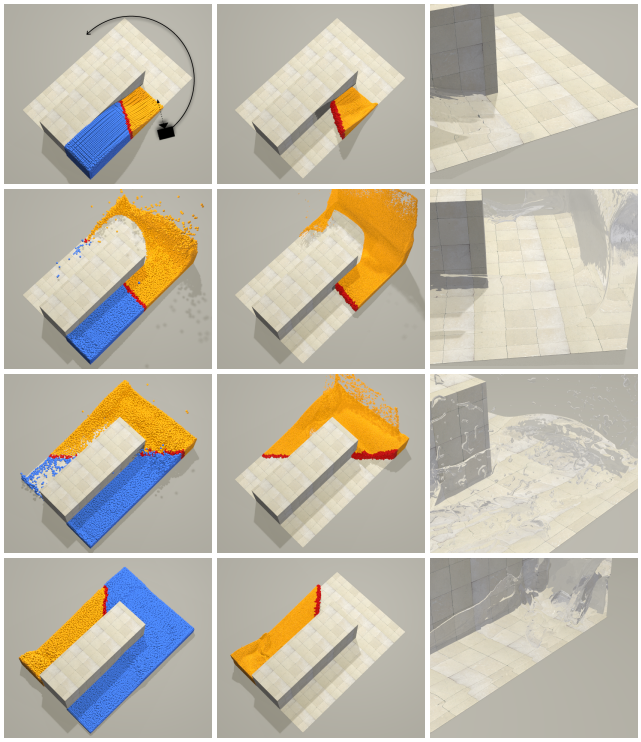


Figure 2: *In L (left), the high-resolution region (yellow) is defined according to the current view of a rotating camera. This region is then computed with quadrupled resolution (middle), and rendered according to the current view direction (right).*

of different resolution levels. This avoids the problems introduced by splitting and merging, and allows the simulation of arbitrarily large particle size differences, hence the desired resolution can be controlled. While the number of resolution levels is not fundamentally limited with our method, we focus on two scales only. For many phenomena, two scales are sufficient since the flow can often be classified into areas where small-scale surface effects are visible, and regions where larger amounts of volume are propagated without producing complex flow structures. Thus, we use two distinct but coupled simulations; one simulation that computes the whole fluid with a coarse resolution, and a high-resolution level that simulates a subset of the fluid with small particles as shown in Figure 2.

2 Related Work

In Lagrangian approaches like SPH [Müller et al. 2003; Monaghan 2005] the resulting visual quality is defined by the resolution of the simulation, i.e. the number of particles that are used to discretize the fluid. Low-resolution fluids often appear blobby because the surface particles are directly used for rendering. Furthermore, velocity differences are smoothed out very quickly in coarse simulations resulting in severe damping. The number of particles, however, is limited by the computing resources that are available. To increase efficiency, the weakly compressible SPH method (WCSPH) [Monaghan 2005] that simulates stiff fluids to keep density fluctuations below 1% has been replaced by an iterative prediction-correction scheme in [Solenthaler and Pajarola 2009]. With PCISPH, larger time steps can be used, resulting in a solver that runs at the same cost as a standard compressible SPH solver. Adaptive time stepping has been included in PCISPH to further improve the performance [Ihmsen et al. 2010b]. In both SPH and PCISPH, neighbor search is the most expensive part, thus [Ihmsen et al. 2010a] presented a parallel framework with optimized memory transfer for both simulation techniques. Parallel execution of SPH has also been explored with GPU techniques in [Harada et al. 2007; Goswami et al. 2010].

A different strategy to optimize the performance is to allocate computing resources to regions of the fluid where complex flow behavior emerges. This is employed in [Adams et al. 2007] where adaptive particle sizes have been used to better resolve surface features. Particles are dynamically split and merged similar to the method presented in [Desbrun and Cani 1999]. Sampling criteria based on physical quantities have been used as well, mainly in the CFD community to get more accurate simulations [Kitsionas and Whitworth 2002; Lastiwka et al. 2005]. Adaptive particles have been integrated in a GPU SPH solver in [Zhang et al. 2008] as well as in the FLIP model in [Hong et al. 2008]. Although the adaptive sampling increases the efficiency of the simulation, the interaction of differently sized particles leads to difficulties. Merging and splitting, for example, can introduce pressure oscillations because of neighbor locations that change abruptly. This problem is reduced in [Adams et al. 2007] by replacing a large particle by only two smaller ones with optimized positions. Another issue is that smaller particles have smaller neighborhood radii than large particles. This often leads to asymmetric visibilities between two particles, violating the momentum conservation. We avoid these problems by computing each resolution level with distinct simulations. This fundamental idea has been applied in [Debunne et al. 2001] to simulate elastic objects. Instead of recursively subdividing a mesh where parts of the mesh interact through common nodes and edges, they use several levels of different mesh resolutions.

Similar to the particle number in Lagrangian models, the quality of Eulerian simulations depend on the grid resolution. Most solvers are based on [Stam 1999], and large grids are used to resolve small-scale details and to reduce the amount of numerical diffusion. Adaptive methods like octree data structure [Losasso et al. 2004] and non-uniform meshes [Klingner et al. 2006] have been presented to reduce the computational cost in areas of low flow complexity. Often, the fluid surface is tracked on a higher resolution grid than the underlying fluid simulation, e.g. [Kim et al. 2009]. In addition, hybrid methods that couple an underlying grid simulation with particles at the surface to get small-scale splashes and droplets have been presented in [Kim et al. 2006; Losasso et al. 2008]. Although these techniques represent the surface with more details, the fluid simulation relies on a low-resolution base simulation and small-scale details that can be resolved are limited. This restriction is avoided in [Lentine et al. 2010] where the basic fluid resolution is kept high but the expensive pressure projection is computed on a coarse grid to improve the efficiency. 3D grid simulations have also been coupled with 2D heightfield methods to reduce the accuracy vertically [Irving et al. 2006] or far away from interaction processes [Thürey et al. 2006]. Less physical but efficient approaches to handle high-resolution grids include model reduction techniques presented in [Treuille et al. 2006; Wicke et al. 2009].

3 SPH Summary

Our two-scale method is based on the particle-based fluid solver SPH [Monaghan 2005]. SPH smoothes quantities over a neighborhood with radius h by using a kernel $W(\mathbf{x}_{ij}, h)$ to weight the contributions according to the distance $\mathbf{x}_{ij}$ between two particles i and j. A smoothed, physical quantity $< q >$ of a particle i can thus be computed by summing up the contributions of the neighboring

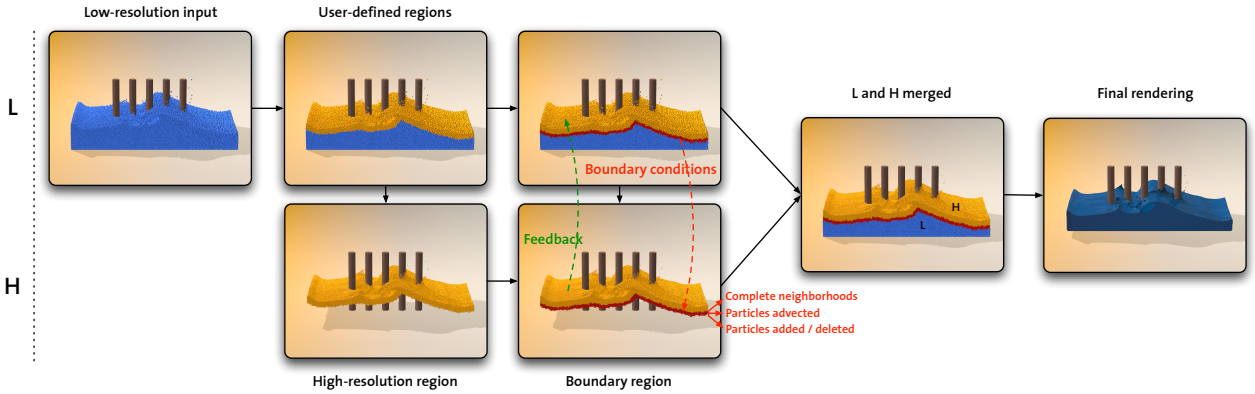


Figure 3: *Method overview. In our two-scale algorithm, a fluid subset (yellow particles) determined in the low-resolution level (L) is additionally simulated with higher resolution (H). Appropriate boundary conditions given by L are defined (red particles), and a feedback force from H onto L is included to get corresponding flows. The particles of both simulations can then be merged for the final rendering.*

particles j

$$< q_i >= \sum_j \frac{m_j}{\rho_j} q_j W(\mathbf{x}_{ij}, h), \tag{1}$$

where m_j is the mass of particle j and ρ_j its density. In our implementation, we use the SPH force equations for multiple fluids proposed in [Solenthaler and Pajarola 2008], and the kernels given in [Müller et al. 2003]. We have additionally integrated our method into the incompressible PCISPH solver presented in [Solenthaler and Pajarola 2009] to keep density variations below 1%. The main difference between SPH / WCSPH and PCISPH is that pressure values are set in a different way: While in SPH / WCSPH pressures are defined by the equation of state, PCISPH iteratively adapts pressure values according to the predicted density error of the particles.

4 Two-Scale Model

Our method, illustrated in Figure 3, uses two simulations with different resolution scales, a low-resolution L and a high-resolution H. The resolution difference is a user-defined value and can be chosen arbitrarily large. We have defined the particle size difference to be a multiple of 8 in each scaling step. This results in a regular particle sampling as discussed in Section 4.3. In our simulations we typically use a particle size difference of a factor of 8 (doubled resolution) or 64 (quadrupled resolution). Larger resolution differences can be chosen as well, but, as we discuss in Section 6, our experiments have shown that these sizes work best to keep the influence of the damping effects from L small.

The coarse level L (blue particles) acts as the base simulation and computes the physics for the whole fluid. In L we determine a subset of the fluid that we want to simulate with higher resolution (yellow particles). We show how this region can be determined in Section 4.1. This subset region defines the second simulation. An additional particle layer (red particles) is used to model the boundary conditions for H. These particles are advected by the flow field of L, see Section 4.2, and are dynamically added and deleted as described in Section 4.3. When a boundary particle enters the active, yellow region, care is taken that the physical quantities change smoothly, see Section 4.4. Since we get more flow details in H we include a feedback force from H onto L, this is described in Section 4.5. The particles of H and L can then be merged for the final rendering. The adapted SPH and PCISPH Algorithm as well as parameter settings are given in Section 5.

4.1 High-Resolution Region

The high-resolution region H can be defined by any type of sampling condition and can change dynamically during the simulation. Since we aim for better visual quality we use geometry driven criteria in all our examples, this is insofar important since the surface particles are typically used to reconstruct the fluid surface. However, physics-driven conditions and combined criteria can be incorporated as well.

Spatial conditions are straightforward to define, an example is shown in Figure 1 where a region around obstacles is defined to be simulated with higher resolution to get more surface details at impact locations. Camera information can be additionally included to change the region according to the field of view as shown in Figure 2. Often, it is desirable to allocate computing resources to the surface of the fluid to get more surface details and fine-scaled splashes as in Figure 3. We define a particle to be at the surface if the distance to the center of mass of its neighborhood $\mathbf{x}_{i,cm} = \mathbf{x}_i - \sum_j \mathbf{x}_j / \sum_j 1$ is above a threshold as described in [Solenthaler et al. 2007]. Isolated particles are detected separately, they are defined by having empty neighborhoods. We use flood fill to extract several layers of particles that are close to the surface. The interface between multiple fluids can be determined similarly, the only difference is that $\mathbf{x}_{i,cm}$ is based on particles of the same fluid type only. An example where the interface region is sampled with higher resolution is shown in Figure 4. Our method is able to handle very complex high-resolution regions that dynamically change over time, thus any other criterion to define H can be incorporated as well.

In order to keep the computation cost low, as many operations as possible are executed in L. Therefore, the high-resolution region is detected in L and then transferred onto H. Each particle in H stores a parent particle, which is the closest particle in L, and is classified according to the region of its parent. In the following, we refer to the particles that are inside the high-resolution region as *active*.

4.2 Boundary Region

We detect the boundary region analogously to H using the flood fill method. Boundary particles are advected by the flow of L, the velocity is interpolated from L onto H (interpolated quantities q from L are indicated with $\hat{q}$ in the following) by

$$\hat{\mathbf{v}}_{i \in H} = \sum_{k \in L} \mathbf{v}_k W, \tag{2}$$

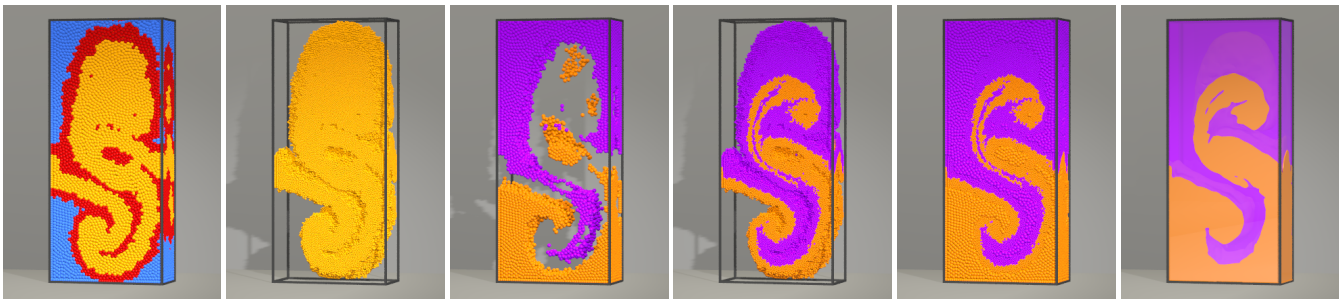


Figure 4: *Our method allows the definition of dynamically changing, complex high-resolution regions, like for example the interface between two fluids with a density ratio of 10. From left to right: In L, the interface is detected and regions are determined. The interface region (yellow) is then computed with doubled resolution. The particle subset of L and the particles of H are combined and rendered.*

where W is the box kernel to avoid expensive distance computations. More accurate kernels, for example the SPH density kernel, could be used as well. $k \in L$ refers to a subset of the neighbors of the parent of i. Since in SPH each particle has 30-40 neighbors on average, we define the subset to include all neighbors within a radius of $h/2$ to avoid excessive smoothing.

The computational overhead of boundary particles is small since no physics or neighborhoods have to be computed for them. Nevertheless, they are included in the neighborhood of close, active particles and hence contribute to the density and exert viscous and pressure forces. To compute the pair-wise forces, a boundary particle additionally stores the interpolated density and pressure values computed analogously to Equation 2. We have to guarantee that the neighborhood of an active particle close to the boundary region is completely filled with particles to avoid imbalanced pressure forces. The minimal size of the boundary region is therefore defined by the kernel size h as illustrated in Figure 5 b).

4.3 Dynamic Boundary Particle Generation

When a particle in L enters the boundary region from outside (blue to red) we dynamically create high-resolution boundary particles in H. In case of doubled resolution, 8 particles with mass $m_H = m_L/8$ are initialized on a cube around the parent position as illustrated in 2D in Figure 6(a), 1). The particle spacing in H is given by $d_H = d_L/r$, where d_L is the particle spacing in L and r is the resolution difference factor; in case of doubled resolution the spacing is halved. Our cubical initialization results in a regular particle distribution and is compared to a 2-particles ([Adams et al. 2007]) and spherical initialization using 7 particles ([Desbrun and Cani 1999]) in Figure 6(a), 2) and 3). Particles created outside the domain (Figure 6(b), i) should not be deleted because this would lead to volume loss. In such situations, we set the particles onto the boundary and slightly shift positions to avoid excessive clustering (Figure 6(b), ii). Since particles undergo a relaxation process to slowly rearrange as soon as they enter the active region (see Section 4.4), the initial particle configuration is not too critical regarding particle disorder. A boundary particle is deleted as soon as its parent does neither belong to the active nor the boundary region.

4.4 Transition Between Boundary and Active

If the parent of a particle j in H leaves the active region and enters the boundary region, the state of j changes as well, hence it is advected by the velocity field of L in the following. The reversed case where a particle j in H enters the active, high-resolution region is critical regarding stability of the simulation. We therefore take care that the physics quantities of a particle j that becomes active (Figure 5a) as well as the quantities of an active particle i that has j as a neighbor (Figure 5b) do not change from time t to $t+1$ if the particles do not experience any position changes. The latter case is trivially fulfilled; j was already previously included in the computation and thus the density ρ_i and the pressure force $\mathbf{F}_i^p$ do not change abruptly. The former case is more critical because advection introduces errors that could lead to irregular particle sampling. Since the quantities computed with SPH are very sensitive to the number of neighboring particles and their locations, the interpolated quantities at time t do not necessarily correspond to the quantities computed by the physics at $t+1$. The density, for example, can be much larger than the interpolated value when particles are clustered, leading to large pressure forces. To alleviate this problem, we apply a particle relaxation process that allows the particles to settle into a stable configuration. The simplest way to achieve this is by letting the physics slowly pushing the particles into the right location. This is done during the time t_{relax} that we have set to 0.05s in all our examples. During this time, the quantities ρ_j and $\mathbf{F}_j^p$ are computed as follows:

- ρ_j: We slowly adjust the density to get a smooth transition between $\hat{\rho_j}(t)$ and $\rho_j(t+1)$. The density is computed by $\rho_j = \alpha\rho_j + (1-\alpha)\hat{\rho_j}$, where α is set to 0 at time $t+1$ and is increased to 1 in the following time steps during t_{relax}.

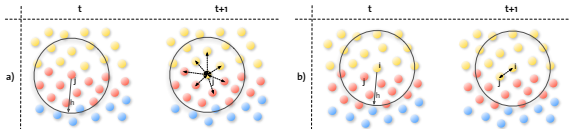
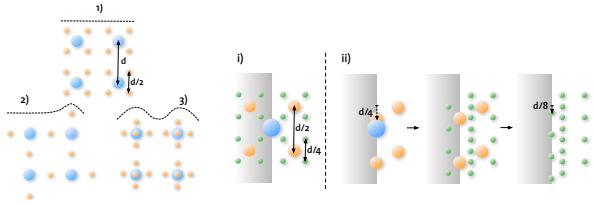


Figure 5: *Abrupt density and force changes are avoided during a region transition. This holds for a boundary particle j entering the active region (a), as well as for a neighboring particle i (b).*



(a) Sampling schemes. (b) Particle creation at walls.

Figure 6: *Our method places boundary particles as shown in 1) to get a regular particle distribution. At domain boundaries, particles are set back onto the wall and are slightly shifted (ii) to avoid particle clustering. In this 2D example, orange indicates doubled resolution, and green quadrupled resolution.*

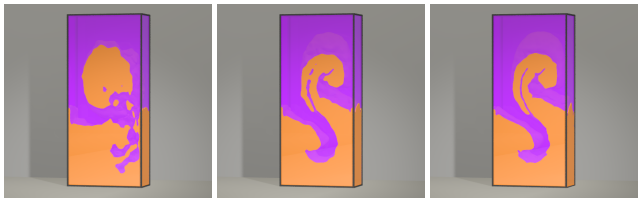


Figure 7: *A feedback force is included to get corresponding flow details in L and H. From left to right: L without feedback, L with feedback from H, H.*

- $\mathbf{F}_j^p$: Pressure forces should slowly push particles away from each other to get j into a relaxed configuration. We do that by restricting the velocity magnitude during t_{relax} to $\tilde{\mathbf{v}}_j = min(\mathbf{v}_j, \beta\frac{h}{\Delta t})$, where Δt is the time step, and β is experimentally determined and set to 0.05 (note the similarity to the Courant condition where β is typically set between [0.2..0.4]). $\tilde{\mathbf{v}}_j$ is then used to update the position $\mathbf{x}_j += \Delta t\tilde{\mathbf{v}}_j$. The velocity is set to the interpolated velocity $\mathbf{v}_j = \hat{\mathbf{v}}_j$ during t_{relax}.

4.5 Feedback

The high-resolution simulation H encounters less damping than L and more small-scale details can be resolved, hence the flows of H and L can diverge. To account for this effect, a feedback force is defined to modify the velocity field of L according to the flow of H. The force is computed for each particle i in L marked to be inside the active region by

$$\mathbf{F}^{feedback}_{i\in L_{active}} = \beta(\mathbf{v}_i - \sum_{j\in H} \mathbf{v}_j W), \tag{3}$$

where β is a constant that defines the influence of the feedback force. In all our examples, β is set to 50 based on experiments. $j \in H$ refers to all active particles in H that have i stored as parent. The effect of the feedback force is shown in Figure 7.

5 Implementation

We have integrated our two-scale method in a standard SPH / WCSPH solver (Algorithm 1 and 2), as well as in the incompressible PCISPH solver (Algorithm 1 and 3). In both cases, we first compute the low-resolution physics L with SPH or PCISPH. Then, the different regions are determined, and boundary particles are dynamically added and deleted in H depending on region changes in L. Physical quantities are interpolated from L onto all boundary particles in H as well as all active particles that just left the boundary region. Next, the high-resolution physics is computed. Several time steps are executed depending on the resolution, given by $nSubsteps = \Delta t_L/\Delta t_H$. The physics (SPH or PCISPH) is computed for all particles inside the active region, and boundary particles are advected. After each particle is forwarded in time, parents are updated for each particle in H. Since particle locations change only little from one time step to the next, we search the new parent in the neighborhood of the current parent. Only in rare cases where the resulting distance to the parent exceeds $h/2$ (approximately the spacing between the particles in L), the parent is recomputed. Simultaneously to the parent update we transfer the velocity information from H back onto L. This is used in the next time step to compute the feedback forces.

The steps executed in SPH (Algorithm 2) correspond to the basic algorithm described in [Müller et al. 2003]. In PCISPH (Algorithm 3), it is important that the predicted values of boundary particles are set correctly, they are simply given by $\mathbf{v}^* = \hat{\mathbf{v}}$, $\rho^* = \hat{\rho}$, $p = \hat{p}$. Predicted values of active particles in H that undergo relaxation get the same restriction as described in Section 4.4. Since these particles cannot move freely, a pressure change does not necessarily result in the expected change of the particle configuration and could slow down the convergence of the PCISPH algorithm. We avoid this problem by allowing larger density errors than 1% for those particles.

Algorithm 1 Two-Scales

```
while animating do
    compute physics L {SPH or PCISPH}
    determine regions in L
    transfer region information from L onto H
    add / delete H_boundary particles
    interpolate quantities from L onto H_boundary,activeRelax
    for nSubsteps do
        compute physics H_active {SPH or PCISPH}
        advect H_boundary
    update parent particle in H
    interpolate feedback infomation from H_active onto L
```

Algorithm 2 Compute Physics SPH (WCSPH) (Particle Set S)

```
for all i ∈ S do
    find neighborhoods N_i
for all i ∈ S do
    compute ρ_i, p_i
for all i ∈ S do
    compute forces F_i^{p,v,g,(feedback)}
for all i ∈ S do
    compute new v_i, x_i
```

Algorithm 3 Compute Physics PCISPH (Particle Set S)

```
for all i ∈ S do
    find neighborhoods N_i
for all i ∈ S do
    compute forces F_i^{v,g,(feedback)}
while (max(ρ_err)* > η) || (iter < minIterations) do
    for all i ∈ S do
        predict v_i, x_i
    for all i ∈ S do
        predict density ρ_i*
        update pressure p_i = f(ρ_i*) according to PCISPH
    for all i ∈ S do
        compute pressure force F_i^p
        determine max(ρ*_{err_Hactive})
for all i ∈ S do
    compute new v_i, x_i
```

In the following, we describe how parameters are set in our implementation. First, a particle spacing d_L is defined, which is the distance between the particles in L. This is used to initialize the particles on a grid. The particle spacing in H, d_H, depends on the resolution difference factor r and is given by $d_H = d_L/r$; $r = 2$ doubles the resolution. The particle mass m_L and m_H are then defined by $m = d^3/\rho_0$, where ρ_0 is set to 1000. The support radius h is set as twice the particle spacing. During the simulation, this results in 30-40 neighbors on average. The time step Δt_L is computed with the Courant condition and depends on h, thus Δt_H is again scaled by r and is given by $\Delta t_H = \Delta t_L/r$.

6 Results and Discussion

Our method can stably handle dynamically changing and complex high-resolution regions. This is shown in Figure 4 where the inter-

	Figure 1, 9			Figure 2, 8		Figure 10, left	
	Our method		Single-scale	Our method	Single-scale	Our method	Single-scale
	Fig. 9, left	Fig. 9, right					
#P H*active* #P H*boundary* #P L	93 - 770k 13k - 90k 46k	93 - 411k 13k - 164k 46k	2.8M	28 - 919k 6 - 41k 28k	1.7M	300k 147k 16k	910k
t [s] / time step *(avg)*	2.0	1.07	7.21	1.86	5.59	0.88	2.12
Speed-up	3.6	6.7		3.0		2.4	

Table 1: *Particle number, timings and speed-up factors of our method compared to the single-resolution reference simulation.*

	#P H*active* / H*boundary* / L	t*total*	t*L*	t*H*	t*neighbor, forces*	t*move, advect*	t*region, interpolate*	t*addDel*	t*feedback, parent*
Figure 2, 8 at t=4.4s	461k / 20k / 28k	1.7	0.066	1.64	1.54	0.023	0.003	0.007	0.067

Table 2: *Timings [s / time step] for each step of our algorithm.*

face between two fluids is simulated with doubled resolution (factor 8 smaller particles) to get more interface details in a Rayleigh-Taylor instability. This example shows that the active and the boundary regions can arbitrarily mix without introducing any stability problems.

An example where the area around cylindrical obstacles is computed with quadrupled resolution (factor 64 smaller particles) to get small-scaled splashes and individual droplets at impact locations is shown in Figure 1. We used two different criteria to define H, first, a spatial constraint (Figure 9, left), and second, a combined criterion where spatial, surface and view information is included (Figure 9, right). Compared to the single-scale reference simulation containing 2.8M particles, our method reduces the total particle number and hence the overall computational cost by a factor of 3.6 and 6.7 in these two cases, see Table 1.

The view of a rotating camera is included in a spatial criterion in Figure 2. In this example, the high-resolution region moves according to the camera location in order to allocate computational resources to areas of the fluid that are visible to the user. Again, quadrupled resolution is used. In this example, the computational cost is reduced by a factor of 3 as shown in Table 1. Detailed timings for each step of our algorithm are given in Table 2, indicating that the overhead of our method is comparatively small. The result is visually compared to the single-scale reference simulation in Figure 8, showing that our method produces very similar flow details. The main difference to the reference solution with 1.7M particles is that the fluid movement is slightly damped - this corresponds to the main limitation of our method. This is because the resolution of the base simulation is very coarse in this example (only 28k particles), thus L is suffering from damping that is then transferred onto H. Increasing the base resolution of L or including vorticity confinement forces to adding back energy could alleviate this problem to some extent. With larger sizes of H, the influence of L and thus the damping is reduced, this is shown in Figure 10 (note that L contains only 16k particles in this example). However, regardless of the size of H, similar surface details emerge. Furthermore, the surface resolution is equivalent to the reference simulation, hence similar rendering quality is achieved. The speed-up factor in this example depends on the size of H and is up to a factor of 2.4 (see Table 1).

Another difficulty with our method is the particle creation at the surface. While our cubical initialization results in a regular particle distribution and keeps horizontal surfaces flat, problems arise with curved surfaces. In such situations, staircase artifacts might be visible, at least until the particles have rearranged due to the physics. Again, this problem can be reduced by slightly increasing the resolution of L. However, as future work, we would like to explore

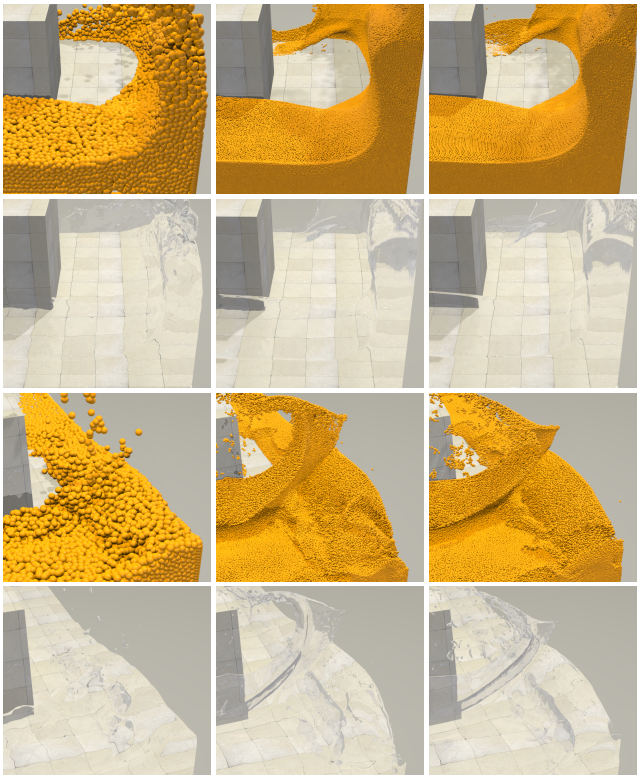


Figure 8: *The result of our method is compared to the single-resolution simulation for the corridor flood scene. Left: L (without feedback), Middle: H with quadrupled resolution computed with our two-scale algorithm, Right: Single-resolution solution.*

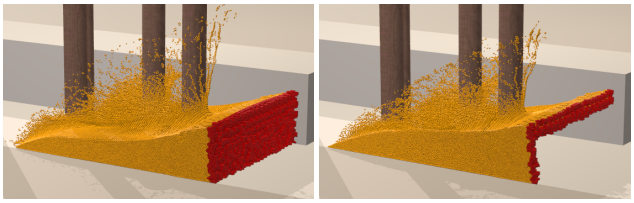


Figure 9: *The spatial sampling condition (left) can be combined with surface and view information (right) to further reduce the particle number and to reach a larger speed-up.*

the inclusion of surface normals in the particle creation process to avoid the dependency on the resolution of L.

In SPH, the computational costs increase linearly with the number of particles. Thus, the optimal speed-up of a multi-scale method is linear to the reduction rate of the particle number, e.g. if the particle number is halved, the frame rate is optimally doubled. Our results show that our method features this characteristics and furthermore indicate that the achieved speed-up factor highly depends on the particular scene set-up and the sampling criterion that is used.

The presented timings are all measured on a 2 2.66 GHz Quad-Core Intel machine. Our code can be optimized by integrating more sophisticated techniques for neighbor search presented in [Ihmsen et al. 2010a] and adaptive time stepping [Ihmsen et al. 2010b]. We used PCISPH to compute the examples in Figure 1 and 10, and SPH for all other simulations. While we focused on two resolution scales only, our method can be extended in a straightforward way to handle multiple scales. However, for most applications, it is sufficient

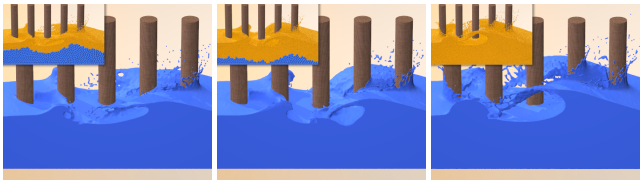


Figure 10: *Similar surface features emerge with different sizes of H. The size of H affects the damping influence of L, the particle resolution at the surface however is equivalent.*

to classify the fluid into regions where small-scale surface details and splashes emerge, and areas of low flow complexity.

7 Conclusion

We have presented a two-scale method for particle-based fluids in order to reduce the overall computational cost while still achieving small-scale surface details comparable to the single-resolution simulation. Our method is based on the idea to simulate distinct particle sizes in individual but coupled simulations. The coupling is done by introducing appropriate boundary conditions as well as feedback forces. Dynamic particle generation and deletion as well as stable transitions between the regions enable changing and complex high-resolution areas. Our method can be easily integrated into an existing particle solver to improve the computational efficiency. Moreover, it allows the allocation of computational resources to those parts of the fluid where a higher resolution is desirable.

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.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic real-time deformations using space and time adaptive sampling. In *Proc. of ACM SIGGRAPH 2001*, 31–36.

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

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 on GPUs. In *Proc. of Computer Graphics International*, 63–70.

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, J., CHA, D., CHANG, B., KOO, B., AND IHM, I. 2006. Practical animation of turbulent splashing water. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 335–344.

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.

LENTINE, M., ZHENG, W., AND FEDKIW, R. 2010. A novel algorithm for incompressible flow using only a coarse grid projection. *ACM Trans. Graph. (SIGGRAPH Proc.) 29*, 4, 1–9.

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. 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.

SOLENTHALER, B., AND PAJAROLA, R. 2008. Density contrast SPH interfaces. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 211–218.

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.

STAM, J. 1999. Stable fluids. In *Proc. of SIGGRAPH 99*, 121–128.

THÜREY, N., RÜDE, U., AND STAMMINGER, M. 2006. Animation of open water phenomena with coupled shallow water and free surface simulation. *Proc. of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 157–166.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Trans. Graph. (SIGGRAPH Proc.) 25*, 826–834.

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular bases for fluid dynamics. *ACM Trans. Graph. (SIGGRAPH Proc.) 28*, 3, 1–8.

ZHANG, Y., SOLENTHALER, B., AND PAJAROLA, R. 2008. Adaptive sampling and rendering of fluids on the GPU. In *Proc. of the Eurographics Symposium on Volume and Point-Based Graphics*, 137–146.