Adaptively Sampled Particle Fluids

Bart Adams
Stanford University / KU Leuven

Mark Pauly
ETH Zürich

Richard Keiser
LiberoVision Inc. / ETH Zürich

Leonidas J. Guibas
Stanford University

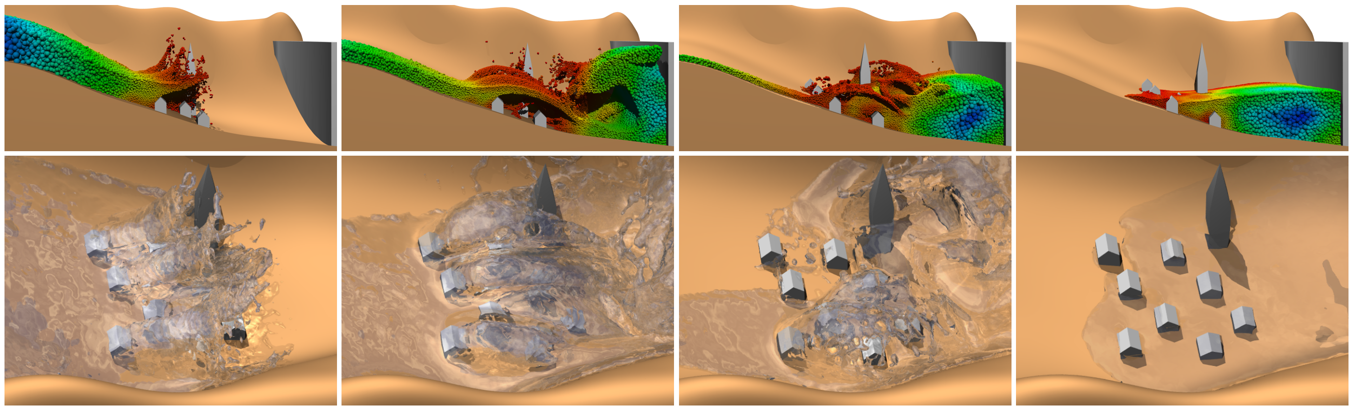


Figure 1: *Flooding a valley. The particle density is dynamically adapted based on geometric complexity and visual importance (as color coded on the particles). The bottom row shows the actual camera views. The top row shows cross sections of the whole simulation domain.*

Abstract

We present novel adaptive sampling algorithms for particle-based fluid simulation. We introduce a sampling condition based on geometric local feature size that allows focusing computational resources in geometrically complex regions, while reducing the number of particles deep inside the fluid or near thick flat surfaces. Further performance gains are achieved by varying the sampling density according to visual importance. In addition, we propose a novel fluid surface definition based on approximate particle–to–surface distances that are carried along with the particles and updated appropriately. The resulting surface reconstruction method has several advantages over existing methods, including stability under particle resampling and suitability for representing smooth flat surfaces. We demonstrate how our adaptive sampling and distance-based surface reconstruction algorithms lead to significant improvements in time and memory as compared to single resolution particle simulations, without significantly affecting the fluid flow behavior.

1 Introduction

Physics-based fluid simulation is used extensively in feature films and is starting to appear in real-time applications such as computer games. State-of-the-art mesh-based methods allow for complex fluid animations including interactions with rigid [Carlson et al. 2004; Klingner et al. 2006] and deformable [Guendelman et al. 2005; Feldman et al. 2005] objects. To cope with the increasing demand for more complex animations, adaptive methods have been proposed that reduce the computational complexity by allocating computing resources to regions with interesting fluid flow behavior. Examples include the use of octrees [Losasso et al. 2004], coupled 2D/3D simulations [Irving et al. 2006] and dynamic unstructured tetrahedral meshes [Klingner et al. 2006]. As an alternative to Eulerian grids, meshfree methods sample the fluid volume with particles to solve the governing equations in a Lagrangian way [Desbrun and Cani 1996; Müller et al. 2003; Kipfer and Westermann 2006]. The particles themselves move with the fluid from one time step to the next, leading to temporally coherent fluid discretizations. Computational resources are naturally allocated to the region of interest, avoiding memory or CPU overheads for simulating empty space. As such, Lagrangian particle methods are becoming a viable alternative to grid-based methods, in particular for animations with many small droplets or very large or unbounded simulation domains.

Although various adaptive algorithms and data structures have been proposed for Eulerian methods, few researchers have addressed this issue for particle-based fluid animations (a notable example in the graphics community is [Desbrun and Cani 1999]). We propose new algorithms for adaptively sampled particle fluids that significantly reduce the computational cost and thus facilitate more complex and visually interesting simulations. Since graphics applications focus mostly on visual quality, we introduce novel sampling conditions that allow for fewer particles in regions of low geometric complexity (e.g., near a thick flat surface or deep inside the fluid volume) and in regions of low visual interest (e.g., outside the viewing frustum or far away from the viewpoint). We show how our adaptive sampling leads to substantially fewer particles and thus lower simulation times without compromising the visual quality of the animations. A crucial ingredient in our approach is a new surface model that avoids visually disturbing temporal discontinuities in the reconstructed surface. Classical surface reconstruction methods for particle volumes are sensitive to particle resampling near the fluid interface (e.g., introducing a large particle near the interface typically creates a surface bump). We avoid this problem with a novel surface definition based on approximate particle–to–surface distances that are carried along with the fluid flow. The main contributions of this paper are:

- A feature size based volume sampling condition, extending traditional surface sampling conditions used for surface reconstruction [Amenta et al. 1998]. This condition reduces the number of particles in regions with large local feature size, i.e., regions deep inside the fluid or near thick flat surfaces, allowing us to focus computational resources in geometrically complex regions.

- A Lagrangian distance-based surface model that defines the fluid surface as the zero set of an implicit function derived from approximate particle–to–surface distances. We show how this surface model outperforms existing methods, especially for handling flat surfaces and particle resampling near the fluid interface, and present an efficient algorithm for computing the distance field.

2 Background

Fluid simulation for computer animation has become popular due to a series of papers by Foster and Metaxas [1996; 1997], who solved the Navier-Stokes equations using finite differences on a Eulerian grid. Stam [1999] improved on this method by introducing an unconditionally stable semi-Lagrangian technique and implicit solvers, allowing for large time steps. Recent efforts in computer graphics have focused on further improving the efficiency to enable more complex simulations. Prominent examples include the use of octree data structures [Losasso et al. 2004; Shi and Yu 2004; Hong and Kim 2005], coupled 2D and 3D simulations [Irving et al. 2006; Thürey et al. 2006], model reduction techniques [Treuille et al. 2006] and dynamic non-uniform mesh refinement [Klingner et al. 2006].

Particle-based methods for fluid simulation have recently become popular as an alternative to Eulerian schemes (e.g., [Premoze et al. 2003; Müller et al. 2003; Clavet et al. 2005; Kipfer and Westermann 2006]). Two main strategies are used to achieve adaptivity in this setting. One approach adapts the interaction kernels allowing for more accurate simulations with a constant number of particles [Owen et al. 1998; Liu et al. 2006]. However, during the simulation, no particles can be removed to improve efficiency or added to resolve finely detailed geometric features. We follow an alternative approach that adapts the particle distribution itself. Examples include [Desbrun and Cani 1999; Kitsionas and Whitworth 2002; Lastiwka et al. 2005], where typically particles are added or removed according to requirements on physical accuracy. For example, [Lastiwka et al. 2005] achieves a higher particle resolution in regions of high velocity gradient.

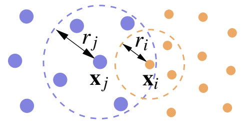
These methods typically do not take into account that the particles are not only used as simulation nodes, but in addition define the fluid surface. In a graphics context, resampling should thus also be guided by geometric complexity. Fine resolution particles should be used to resolve small splashes or thin sheets, while large particles can be used to represent the surface of thick fat fluid volumes. Therefore, we propose to adapt the particle resolution based on a feature size sampling criterion, much alike the one used for manifolds [Amenta et al. 1998; Kolluri et al. 2004]. A similar sizing criterion has also been proposed for adaptive tetrahedral mesh generation [Alliez et al. 2005] which has been used for example in the fluid simulation framework of Klingner et al. [2006].

Our surface definition is based on the model presented by Zhu and Bridson [2005]. Their approach reduces the blobby appearance of particle surfaces compared to classical metaballs [Blinn 1982]. We extend their method by incorporating particle–to–surface distance information to better handle particle resampling near the interface and to improve the smoothness of the surface. Distance information is computed at the particles in the previous time step and carried along to the current time step, similar in spirit to the semi-Lagrangian contouring method proposed by Bargteil et al. [2006]. In this way, our particle volume can be seen as a Lagrangian analog to adaptively sampled distance fields (ADFs) [Frisken et al. 2000]. An alternative surface model for dynamically changing particle volumes was presented in [Desbrun and Cani 1998]. Their method uses a fixed regular grid to track the interface, making it less suitable for large-scale simulations with varying sampling densities.

3 Physics Framework

Our simulation framework is based on Smoothed Particle Hydrodynamics (SPH) (see [Monaghan 2005] for a good overview). In SPH, the fluid volume is discretized with particles that are used as simulation nodes to solve the Navier-Stokes equations. Each particle p_i is defined by its position $\mathbf{x}_i$, support radius r_i, and mass m_i. During the simulation, the particle sampling density is dynamically adapted, introducing particles at different levels $l = 0, 1, 2, 3, \ldots$ The smallest level 0 particles have a mass m and support radius r. The mass and support radius of the other particles can be derived from their level: $m_i = 2^{l_i} m$, $r_i = \sqrt[3]{2^{l_i}}\, r$. The initial and average inter-particle spacing is given by r_i/h. We set $h = 2.5$ in all our examples.

Throughout the paper, a particle p_j is considered a neighbor of particle p_i if $\|\mathbf{x}_i - \mathbf{x}_j\| \leq \max(r_i, r_j)$. The particles p_i and p_j in the



above figure are thus neighbors of each other. These neighborhoods can be computed efficiently using range queries supported by standard spatial data structures such as *k-d* trees.

Since a particle can have other particles of different radius within its neighborhood, we use the shooting-gathering approach of Desbrun and Cani [1999] to avoid asymmetrical force distributions. Applying this approach to the forces defined in [Müller et al. 2003], we obtain the symmetric fluid forces acting from particle p_j on p_i:

$$\mathbf{f}_{ij}^{\text{pressure}} = -V_i V_j (P_i + P_j)(\nabla W(\mathbf{x}_{ij}, r_i) + \nabla W(\mathbf{x}_{ij}, r_j))/2, \quad (1)$$

$$\mathbf{f}_{ij}^{\text{viscosity}} = \mu V_i V_j (\mathbf{v}_j - \mathbf{v}_i)(\nabla^2 W(\mathbf{x}_{ij}, r_i) + \nabla^2 W(\mathbf{x}_{ij}, r_j))/2, \quad (2)$$

with $V_i = m_i/\rho_i$ the particle volume, $\rho_i = \sum_j m_j W(\mathbf{x}_{ij}, r_i)$ the particle density, $\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$, $P_i = k(\rho_i/\rho - 1)$ the pressure (with ρ the physical fluid density and k a user specified constant), μ the viscosity and $\mathbf{v}_i$ the particle velocity. We use the radially symmetric kernel functions $W(\mathbf{x}, r)$ with support r as defined in [Müller et al. 2003]. It is easy to see that the above defined forces obey Newton's third law (action=reaction) even in the case of non-uniformly sized particles.

4 Feature Adaptive Sampling

Our goal is to dynamically allocate the available computational resources to the regions of interest using adaptive particle sampling. To faithfully recover the fluid surface, the refinement criterion should maintain an adequate particle density to resolve thin streams or small droplets. However, for regions deep inside the fluid volume or near a thick flat surface fewer particles are required to accurately reconstruct the surface. Below we will define an extended local feature size criterion that exactly captures this intuition and show how other sampling criteria, such as view-dependent sampling, can be incorporated. We demonstrate that, although our sampling criteria are purely geometrical, they do not significantly affect the fluid flow behavior, especially in the region of interest.

4.1 Extended Local Feature Size

Our geometric sampling condition is based on a local feature size descriptor. The local feature size of a point $\mathbf{y}$ on a (closed) manifold

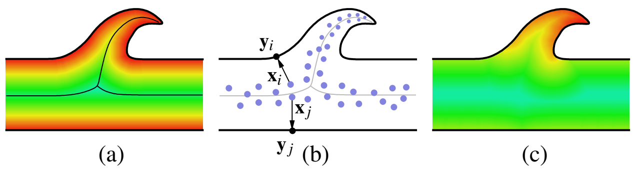


Figure 2: *Extended local feature size for part of a fluid volume. (a) Distance field and medial axis. (b) Particle approximation of the medial axis. (c) Local feature size defined in the volume. Red regions have a low feature size and should be sampled densely, blue regions have a large feature size and can be sampled coarsely.*

$\mathscr{M}$, is defined as the distance of $\mathbf{y}$ to the closest point on the manifold's medial axis [Amenta et al. 1998]. We denote this feature size as $\mathrm{lfs}(\mathbf{y})$. This definition can be extended to the volume $\mathscr{V}$ enclosed by $\mathscr{M}$ (similar to [Kolluri et al. 2004] and [Alliez et al. 2005]):

$$\mathrm{elfs}(\mathbf{x}) = \min_{\mathbf{y}\in\mathscr{M}} \{\|\mathbf{x}-\mathbf{y}\| + \mathrm{lfs}(\mathbf{y})\}, \quad \mathbf{x}\in\mathscr{V}. \tag{3}$$

Thus, the extended local feature size of $\mathbf{x}$ is defined as the minimal sum of the distance to the point $\mathbf{y}$ on the manifold, plus the (classical) local feature size defined at this point. It is easy to show that this function is 1-Lipschitz continuous [Alliez et al. 2005], which guarantees a smooth variation of particle sizes necessary to maintain a stable physical simulation. Figure 2 illustrates the local feature size definition for part of a closed 2D manifold.

We now show how we can efficiently evaluate Equation 3 starting from the assumption that we know for each particle p_i its closest point, or footpoint, $\mathbf{y}_i$, on the surface. We will discuss in Section 5.2 how footpoints can be computed efficiently. We estimate the extended local feature size by first computing a particle approximation of the medial axis of the fluid surface, which is used to calculate $\mathrm{elfs}(\mathbf{x}_i)$ for particles near the surface. Using a fast marching method [Sethian 1999], the local feature size is then propagated to the interior particles.

Approximate Medial Axis Construction We compute an approximate medial axis using the closest point information similar to [Foskey et al. 2003], but adapted to the particle setting. Given two neighboring particles p_i and p_j, positioned at $\mathbf{x}_i$ and $\mathbf{x}_j$, and their respective footpoints $\mathbf{y}_i$ and $\mathbf{y}_j$, we decide that the particles lie near the medial axis, if the following two conditions hold (see also Figures 2 and 3):

$$\mathrm{acos}\left(\frac{\mathbf{y}_i-\mathbf{x}_i}{\|\mathbf{y}_i-\mathbf{x}_i\|}\cdot\frac{\mathbf{y}_j-\mathbf{x}_j}{\|\mathbf{y}_j-\mathbf{x}_j\|}\right) > \gamma, \quad \|\mathbf{y}_i-\mathbf{y}_j\| > \|\mathbf{x}_i-\mathbf{x}_j\|, \tag{4}$$

with a threshold angle γ (we take $\gamma = 60$ degrees in all our simulations). The first condition states that the two particles should be on opposite sides of the medial axis, while the second condition is necessary to prune spurious branches, which can especially appear close to the surface. The resulting set of medial axis particles is further reduced by only considering those particles which have at least 50% of their neighbors near the medial axis. Finally, an isolated particle without neighbors is also considered a medial axis particle.

The (classical) local feature size $\mathrm{lfs}(\mathbf{y}_i)$ of a footpoint $\mathbf{y}_i$ is now defined as its minimal distance to the medial axis particles. Starting from the local feature size for the footpoints, we can easily compute the extended local feature size at the particle positions.

Extended Local Feature Size Propagation The extended local feature size defined in Equation 3 can be computed using a fast marching algorithm. The algorithm is initialized by setting $\mathrm{elfs}(\mathbf{x}_i) = \|\mathbf{x}_i-\mathbf{y}_i\| + \mathrm{lfs}(\mathbf{y}_i)$ for particles p_i near the surface (i.e., when $\|\mathbf{x}_i-\mathbf{y}_i\| < r_i$). These particles are then inserted into a priority queue that is sorted in order of increasing extended local feature size. Additionally, we store $\mathbf{y}'_i \leftarrow \mathbf{y}_i$, the surface point used to compute the local feature size at particle p_i (note that in general $\mathbf{y}'_i$ might differ from $\mathbf{y}_i$, especially for particles far away from the surface). The extended local feature size of all other particles is initialized to infinity. The algorithm then propagates the local feature size in a greedy fashion to the interior particles by iteratively taking the particle at the top of the queue, i.e., the particle with the smallest local feature size. The surface point $\mathbf{y}'_i$ of this particle is used to compute an updated local feature size for each neighbor $\mathbf{x}_j$, i.e., $\mathrm{elfs}(\mathbf{x}_j)$ is set to $\|\mathbf{x}_j-\mathbf{y}'_i\| + \mathrm{lfs}(\mathbf{y}'_i)$, if this value is smaller than its previously computed $\mathrm{elfs}(\mathbf{x}_j)$. Next, p_j is added to the queue and $\mathbf{y}'_j$ is set to $\mathbf{y}'_i$. After updating all neighbors, the algorithm continues by taking the next particle from the queue.

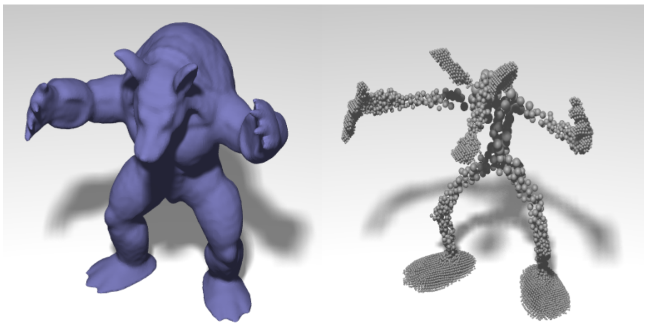


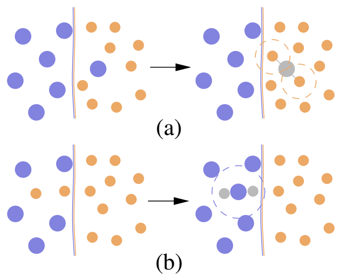
Figure 3: *Particle approximation of the armadillo's medial axis computed for one of the first frames of the animation of Figure 9.*

4.2 Adaptive Sampling

A particle p_i is split, if $\mathrm{elfs}(\mathbf{x}_i) < \alpha r_i$, and merged, if $\mathrm{elfs}(\mathbf{x}_i) > \beta r_i$. To prevent oscillatory down- and up-sampling, α should be smaller than β (we use $\alpha = 2$ and $\beta = 3$ in all our simulations). Splitting and merging are performed using very basic operators (see also Figure 4) that are special cases of the resampling operators presented in [Desbrun and Cani 1999]. However, to improve stability and maintain a uniform particle distribution, we optimize the spatial locations of newly created particles as discussed below.

Splitting A particle p_i at level l_i is split in two particles p_j and p_k at level $l_j = l_k = l_i - 1$. The two new particles p_j and p_k are positioned symmetrically around p_i at a distance $d = r_j/(2h) = r_k/(2h)$ to p_i. The exact positions are chosen so that p_j and p_k are within the fluid volume and not closer to any other particle than d to avoid the introduction of large pressure forces. The regions satisfying this last condition can be obtained as follows (see also Figure 4, (c)). All particles p within a distance $2d$ to p_i define invalid regions on the sphere with radius d and center $\mathbf{x}_i$, i.e., the points on this sphere which are within a distance d to p. These regions are denoted as red arcs on Figure 4, (c). The remaining regions become valid sampling positions if the two particles p_j and p_k can be placed symmetrically around particle p_i outside the invalid regions. The valid regions are denoted by green arcs in Figure 4, (c). This algorithm can be efficiently implemented using sphere-sphere Boolean operations in spherical coordinates. Note that there are possibly many valid sampling positions, from which we pick a random one. In the rare event that there is no valid region, we iteratively lower the minimal distance d until a valid region is found. It can be easily seen that at a certain point all positions on the inner sphere become valid (i.e., the red arcs in the figure vanish). This means that a new particle will be placed not closer to an existing particle than half the local particle spacing. Finally, the new particles' velocities are inherited from p_i.

Merging A particle p_i that is tagged to merge looks for another particle p_j of the same level in its neighborhood that should also be merged. A new particle p_k at level $l_k = l_i + 1 = l_j + 1$ is created at



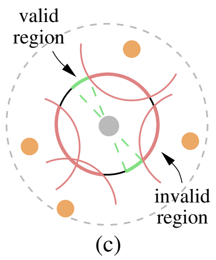


Figure 4: *Particle resampling. (a) Splitting of a level l particle into two level $l-1$ particles. The two new particles are placed symmetrically around the (deleted) level l particle so that no other particle is too close to the new particles. (b) Merging of two level $l-1$ particles into a level l particle. The merging only proceeds if the merged particle is sufficiently far away from the other particles. (c) Determining valid positions for the split particles. The red regions on the inner sphere denote positions which are too close to existing particles. The green regions denote valid positions to place the opposing split particles.*

position $\mathbf{x}_k = (\mathbf{x}_i + \mathbf{x}_j)/2$ if this position is within the fluid volume and there is no other particle within a distance $r_k/(2h)$ (in order to prevent high pressure forces). If these conditions are violated, particle p_i looks for another neighbor to initiate the merging. If no such neighbor is found, the merging is canceled and possibly postponed to the next time step. The new particle's velocity is set to the average of the old particles.

Other particle resampling algorithms can be used as well, e.g., [Desbrun and Cani 1999; Pauly et al. 2005]. The advantages of the above scheme are efficient evaluation, stable transitions from one time step to the next, and a smooth variation of particle radii that only differ by a factor $\sqrt[3]{2} \approx 1.26$ from one level to the next. Note that the resampling operators preserve mass and linear momentum.

4.3 Additional Sampling Conditions

It is straightforward to incorporate further sampling criteria. Figure 1 illustrates how the particle density can be additionally decreased in regions of low visual interest. In this example, we change the splitting and merging criteria so that distance to the view frustum $d_i(\mathrm{vf})$ is taken into account (i.e., a particle is split if $\mathrm{elfs}(\mathbf{x}_i) + d_i(\mathrm{vf}) < \alpha r_i$ and merged if $\mathrm{elfs}(\mathbf{x}_i) + d_i(\mathrm{vf}) > \beta r_i$). Such an additional sampling condition is especially interesting for simulations involving large or unbounded simulation domains.

Although we only consider geometric sampling conditions in this paper, physics-based sampling conditions can also be easily integrated. One might for example want to maintain a high sampling rate in very turbulent regions or reduce the sampling density in static or non-rotational regions, or vary the particle resolution purely based on accuracy of the SPH approximations [Desbrun and Cani 1999]. However, we noted from our experiments that there is often a high correlation between physical and geometric complexity. For example, highly turbulent regions often introduce geometrically complex fluid surfaces, but intricate fluid surfaces can also appear for stationary fluids (e.g., due to complex boundaries).

5 Distance Based Surface Tracking

Our feature-based resampling scheme allows temporal and spatial adaptation of the fluid discretization during the simulation. Since resampling can also occur near the fluid-air interface (cf. Figure 2), care must be taken to avoid visually disturbing artifacts, such as popping or increased blobbyness, that are often noticeable with existing particle-based surface models. The key idea of our new approach is the use of approximate particle–to–surface distances, which are carried along with the particles and used to define a smooth fluid surface. After reconstructing the surface, these distances are updated using an efficient redistancing algorithm.

5.1 Surface Definition

Given approximate particle–to–surface distances d_i (obtained from the previous time step), our surface is defined as the zero level set of the following level set function:

$$\phi(\mathbf{x}) = d(\mathbf{x}) - \|\mathbf{a}(\mathbf{x}) - \mathbf{x}\|, \tag{5}$$

$$\mathbf{a}(\mathbf{x}) = \sum_i w_i(\mathbf{x})\mathbf{x}_i / \sum_i w_i(\mathbf{x}), \tag{6}$$

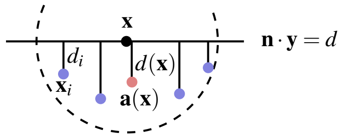
$$d(\mathbf{x}) = \sum_i w_i(\mathbf{x}) d_i / \sum_i w_i(\mathbf{x}), \tag{7}$$

where $w_i(\mathbf{x})$ is a smooth, radially symmetric weight function defined at particle p_i with support r_i. With compactly supported weight functions the summation can be limited to the particles in the neighborhood of the evaluation point $\mathbf{x}$. We use:

$$w_i(\mathbf{x}) = \begin{cases} (1 - (\|\mathbf{x} - \mathbf{x}_i\|/r_i)^2)^3 & \text{if } \|\mathbf{x} - \mathbf{x}_i\| < r_i, \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

which guarantees second order continuity of the level set function. In this surface model a point lies on the surface, if its distance to the weighted average particle position equals the weighted average particle–to–surface distance.

The motivation behind the aforementioned surface definition is that it is particularly well suited to represent flat surfaces, significantly reducing the bumpy appearance compared to traditional particle surface definitions. This can be easily seen as follows. Assume the particles p_i sample a flat surface, i.e., a plane $\{\mathbf{y} \in \mathbb{R}^3 \mid \mathbf{n} \cdot \mathbf{y} = d, \|\mathbf{n}\| = 1\}$. This means that $\mathbf{n} \cdot \mathbf{x}_i + d_i = d$ for all particles p_i. By linearity of the plane equation and the partition of unity of



the weight functions it follows that the average particle position $\mathbf{a}(\mathbf{x})$ and distance $d(\mathbf{x})$ (Equations 6 and 7) also sample the plane: $\mathbf{n} \cdot \mathbf{a}(\mathbf{x}) + d(\mathbf{x}) = d$. Due to the uniform distribution of particles caused by the SPH pressure forces, the average $\mathbf{a}(\mathbf{x})$ is expected to lie perpendicularly underneath the point $\mathbf{x}$ with respect to the plane. Therefore, $\|\mathbf{a}(\mathbf{x}) - \mathbf{x}\| \approx d(\mathbf{x})$ for a point $\mathbf{x}$ on the plane and thus the definition approximates flat surfaces very well. As will be shown in Section 5.3, the surface model also works well for curved surfaces as well as when resampling near the interface.

5.2 Redistancing

The particle–to–surface distances d_i at time step t are carried along with the particles and are used to define the surface at time step $t + \Delta t$ as discussed above. These distances then have to be updated to conform with the new surface. This redistancing can be implemented efficiently by projecting particles near the surface onto the surface and propagating the distance information to the other particles in the interior of the fluid volume.

Particle to Surface Projection Surface points can be computed by projecting particles near the fluid-air interface on the zero level set of ϕ. This can be easily performed by a simple binary search along the ray segment from $\mathbf{x}_i$ to $\mathbf{x}_i + r_i \nabla\phi(\mathbf{x}_i)$, i.e., we evaluate $\phi(\mathbf{x}_i + s\nabla\phi(\mathbf{x}_i))$ for $s = 0$, $s = r_i/2$ and $s = r_i$ and recurse into the

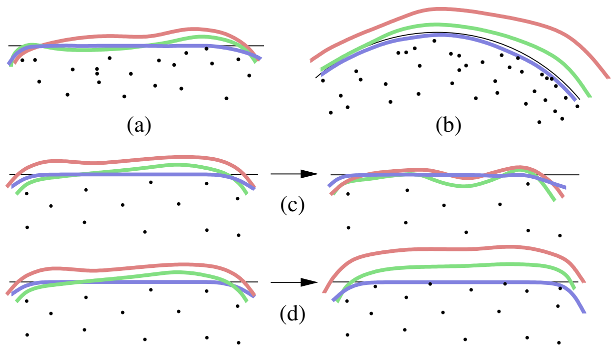


Figure 5: *Comparison between our surface definition (blue curve), the definition of [Zhu and Bridson 2005] (green curve) and blobbies (red curve). The represented surface is the black curve. (a) Representing a straight line. (b) Representing a circular arc. (c),(d) Illustration of the effect of resampling (particle deletion (c) and addition (d)).*

first or second interval depending on where the sign changes. We found that one recursion and linear interpolation (using the level set values) between the last two obtained points gives adequate results. We perform this projection for all particles p_i with a distance–to–surface d_i smaller than their support radius r_i and store the footpoint $\mathbf{y}_i$ for each particle p_i where the projection was successful. Together with the normalized level set gradient $\nabla\phi(\mathbf{y}_i)/\|\nabla\phi(\mathbf{y}_i)\|$, the footpoints describe a piecewise linear approximation of the fluid surface. The result of this projection step is thus a set of oriented disks, positioned at $\mathbf{y}_i$, with orientation $\nabla\phi(\mathbf{y}_i)/\|\nabla\phi(\mathbf{y}_i)\|$ and radius r_i. The obtained distance information can now be propagated to particles in the interior of the fluid using a fast marching method.

Distance Propagation The algorithm proceeds similarly to the local feature size propagation algorithm of Section 4.1. First, all projected particles are inserted into a priority queue that is sorted with respect to increasing distance–to–surface order. We also assign an infinite distance $d_i \longleftarrow \infty$ to all other particles. Then we recompute the distance to the boundary for all neighbors of the queue's top particle using the particle's footpoint disk. If the distance to this tangent disk is smaller than the previously assigned distance, these particles are updated and added to the queue in turn. The algorithm ends when the queue becomes empty.

This redistancing algorithm assigns to each particle p_i its closest point on the surface $\mathbf{y}_i$ and the corresponding distance $d_i = \|\mathbf{x}_i - \mathbf{y}_i\|$. These values are used for local feature size computation in the current time step and for surface reconstruction in the next time step. Note that while the obtained distance information at the particles is not exact, we have not found this approximation to pose any practical problems.

5.3 Comparison to Alternative Surface Definitions

The most popular surface definition for particle-based fluids are blobbies (or metaballs) [Blinn 1982]. However, as discussed in [Zhu and Bridson 2005], blobbies have several disadvantages (see also Figure 5). The resulting surface is highly dependent on the chosen iso-level and particle support radii, which can lead to unnaturally thick fluid volumes. They are particularly ill-suited to represent flat surfaces, often exhibiting a bumpy appearance. And finally, which is important in our setting, the surface changes drastically when adding or removing particles, leading to visually disturbing temporal discontinuities. The surface model proposed in [Zhu and Bridson 2005] alleviates most of the aforementioned limitations. It is worth noticing that their definition can be obtained from ours by replacing the particle–to–surface distances d_i in Equation 7 by half the particle radii (i.e., set $d_i = r_i/2$). Although the blobbyness is

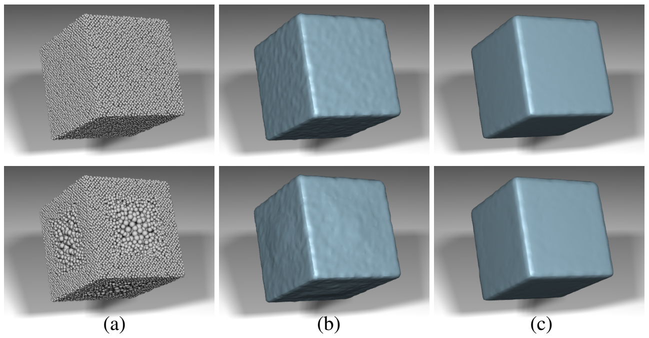


Figure 6: *Comparison between the surface definition of [Zhu and Bridson 2005] (b) and ours (c). The top row shows the surface for a random sampling of a cube with 100k particles. The bottom row shows the surface after adaptive down-sampling to 20k particles. Note how our surface is less sensitive to irregular particle distributions and resampling.*

reduced as compared to metaballs, flat surfaces are still rather difficult to represent and particle resampling still affects the surface appearance (see Figures 5 and 6). As illustrated in the figures, our new distance-based model significantly improves the surface quality for flat surfaces and when resampling near the interface. When a new particle is added, it is first assigned a correct distance d_i to the surface, resulting inherently in a minimal temporal change of the fluid surface.

6 Implementation

For the neighborhood queries we use a *k-d* tree which is rebuilt only once in each time step. During resampling we avoid updating the *k-d* tree by using a separate data structure (basically a linear list) for newly created particles and by flagging deleted particles as such. Obstacles are represented as adaptively sampled distance fields (ADFs) [Frisken et al. 2000]. The resulting animations are rendered using POV-Ray (http://www.povray.org) after extracting a triangle mesh corresponding to the fluid surface using a marching cubes algorithm [Lorensen and Cline 1987].

The actual simulation loop proceeds in each time step as follows. First, the particle neighbors are computed, obstacle collisions are resolved and symmetric particle forces (and hence velocities) are obtained from the Navier-Stokes equations using SPH (Section 3). Next, the particle–to–surface distances are updated (Section 5.2) and the extended local feature size is computed (Section 4.1). Particles are added or removed (Section 4.2) based on the computed feature size criterion. Finally, positions are integrated using an explicit Leap-frog scheme. Currently, we use a fixed integration time step for all particles. However, the time step is dictated by the smallest particles and adaptive time stepping similar to [Desbrun and Cani 1996] should further improve performance. Our dynamic resampling scheme does not incur additional time stepping restrictions.

7 Results & Discussion

All results are obtained on a 3.2 GHz Intel Pentium D CPU with 3.5 GB of memory. We give a detailed overview of the computation time of our algorithm in Table 1 and compare it with the corresponding single resolution simulation. The timings given in the table are for one frame in the final 30fps movie. Because we used a fixed integration time step of 0.001s in all examples, the time required for one simulation step can be derived by dividing by 33.

Figure 1 shows the flooding of a valley. In addition to the proposed feature size based sampling condition, we also varied the sampling density based on visual importance as discussed in Sec-

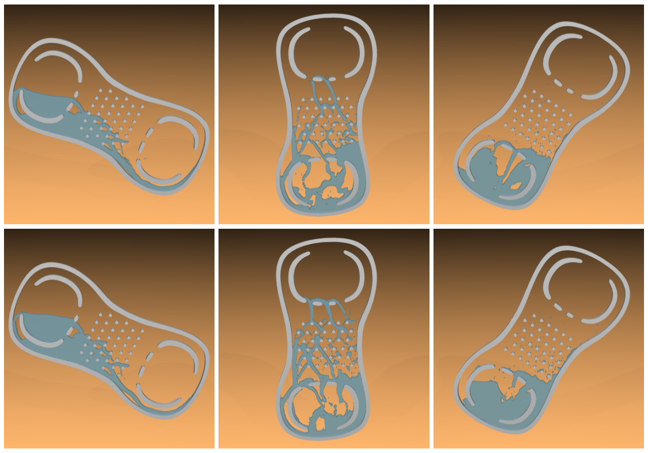


Figure 7: *Rotating desk toy. The top row shows a single resolution simulation using 15k particles. The bottom row shows the corresponding adaptive simulation averaging at 5k particles. As can be seen, our method does not degrade the general fluid flow significantly.*

tion 4.3. As a result, an average of 124k particles is sufficient per animation frame, as compared to almost one million particles for the corresponding single resolution simulation. The adaptive simulation therefore runs almost 7 times faster. This example particularly illustrates the benefits for adaptive sampling based on our local feature size criterion. In the beginning and end of the simulation, the fluid volume is rather fat, allowing large particle sizes, while in the middle, small particles are required to animate the splashing fluid. Figure 7 depicts 3 frames taken from a 2D rotating desk toy animation. While the fluid moves from one side of the toy to the other, our resampling scheme effectively ensures an adequate sampling density to resolve the fine streams and droplets arising from the obstacle collisions. The figure also shows a comparison with the corresponding non-adaptive simulation which uses approximately 3 times more particles. Figure 8 shows fluid being poured in the Utah teapot, averaging at 31k particles, which is almost a factor 8 improvement over the non-adaptively sampled simulation. Finally, Figure 9 shows subsequent frames of an animation involving 4 splashing water armadillos. Using our geometric feature size based sampling condition results in an average of approximately 140k particles per frame. The corresponding full resolution animation would require 4.5 times more particles on average. An interesting observation in this animation is that, although more fluid is added over time, the average number of particles decreases.

As shown in Table 1, the overhead imposed by our algorithms is rather small. Moreover, redistancing and resampling do not have to be performed in every time step. For the results in this paper, we invoked these operators only every fifth simulation time step. Smarter, more adaptive criteria (e.g., based on particle velocities) could possibly further reduce the overhead. The table also summarizes the resulting average memory and speed gains compared to the corresponding single resolution simulations.

	FLOODING	DESK TOY	TEAPOT	ARMADILLOS
NEIGHBORHOOD	28.0s [259.1s]	0.6s [2.2s]	10.4s [102.2s]	33.2s [222.0s]
FORCES	30.1s [203.9s]	1.3s [3.6s]	8.2s [76.6s]	36.1s [159.2s]
PROJECTION	6.3s [17.2s]	0.1s [0.2s]	4.4s [14.6s]	11.9s [25.4s]
REDISTANCING	2.5s [13.1s]	0.05s [0.1s]	0.9s [6.5s]	2.9s [13.9s]
ELFS	2.3s [-]	0.08s [-]	1.2s [-]	2.6s [-]
RESAMPLING	4.4s [-]	0.05s [-]	0.7s [-]	4.5s [-]
GAIN	8.0x/6.6x	3.0x/2.8x	7.9x/7.7x	4.5x/4.3x

Table 1: *Average timing statistics per frame for the various examples shown in this paper. The timings for the corresponding single resolution simulations are shown between brackets. The bottom row shows the memory and speed improvements respectively.*

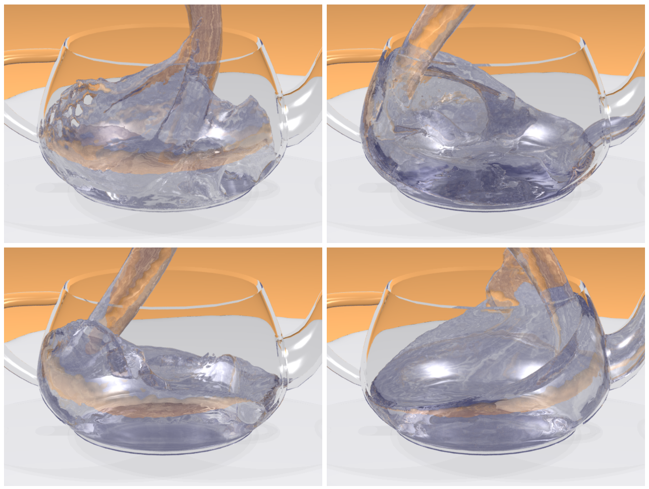


Figure 8: *Pouring water in the Utah teapot.*

To compare our framework to existing work, we set up a number of simulations as in Figure 1 for different particle counts which correspond to results given in the real-time fluid simulation framework of [Kipfer and Westermann 2006]. Here, the authors optimize a similar SPH framework as [Müller et al. 2003] by implementing an efficient search data structure and collision detection scheme. We refer to [Kipfer and Westermann 2006] for details. They report computation times per simulation step of approximately 0.060s for a flooding example involving 20k particles. We obtain simulation times for the same number of particles of approximately 0.21s without adaptive sampling and 0.071s with adaptive sampling. They further report computation times of 0.038s for 8k particles, while we obtain 0.098s without and 0.041s with adaptive sampling. For lower number of particles, our method does not perform as well. Their computation time is reduced to 0.014s when using 3k particles, where we need 0.034s and 0.025s for the non-adaptive and adaptive simulation respectively. The reasons for the lower performance gain are mainly the large overhead in rebuilding the *k-d* tree and the reduced ability for adaptive sampling due to the already coarse particle discretization. However, the multiresolution approach should be superior asymptotically because it needs fewer particles, even if the per particle overhead is higher. Indeed, around 20k particles both methods are comparable and it is expected that our method will perform better for higher particle counts, where more reduction is possible. Nevertheless, it seems promising to adapt and incorporate their optimizations to further increase the performance. Finally, note that our adaptive sampling not only improves computation time, but also results in significant memory gains.

8 Conclusions

We have presented adaptive sampling algorithms for particle fluids and illustrated that, even for very dynamic fluid flow simulations, particle resampling based on extended local feature size can significantly reduce the number of particles and improve the simulation time, while retaining the general fluid flow behavior. Our distance-based surface model leads to smooth and stable fluid surfaces. As future work, we plan to further speed up our multiresolution framework by integrating adaptive time stepping and by optimizing neighborhood searching to improve this major bottleneck.

Acknowledgments The first author is funded as a post-doctoral researcher by the Fund for Scientific Research, Flanders (F.W.O.-Vlaanderen) and by a Fellowship of the Belgian American Educational Foundation. The authors also acknowledge the support of NSF grants ITR 0205671 and CARGO 0310661, ARO grant DAAD19-03-1-0331, NIH grant GM-072970, DARPA grant 32905, and the Max-Planck Center for Visual Computing and Communication. We would also like to thank the anonymous reviewers for their valuable comments.

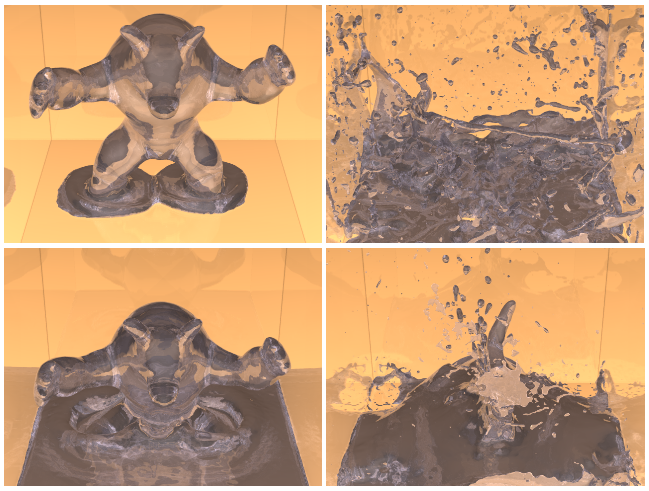


Figure 9: *Splashing armadillos. The number of particles used in these frames are (from left to right and top to bottom): 40k, 232k, 140k, 185k.*

References

ALLIEZ, P., COHEN-STEINER, D., YVINEC, M., AND DESBRUN, M. 2005. Variational tetrahedral meshing. *ACM Trans. Graph. 24*, 3, 617–625.

AMENTA, N., BERN, M., AND KAMVYSSELIS, M. 1998. A new voronoi-based surface reconstruction algorithm. In *SIGGRAPH '98*, 415–421.

BARGTEIL, A. W., GOKTEKIN, T. G., O'BRIEN, J. F., AND STRAIN, J. A. 2006. A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graph. 25*.

BLINN, J. F. 1982. A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3, 235–256.

CARLSON, M., MUCHA, P. J., AND TURK, G. 2004. Rigid fluid: animating the interplay between rigid bodies and fluid. In *SIGGRAPH '04*, 377–384.

CLAVET, S., BEAUDOIN, P., AND POULIN, P. 2005. Particle-based viscoelastic fluid simulation. In *SCA 2005*, 219–228.

DESBRUN, M., AND CANI, M.-P. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *6th Eurographics Workshop on Computer Animation and Simulation '96*, 61–76.

DESBRUN, M., AND CANI, M.-P. 1998. Active implicit surface for animation. In *Graphics Interface*, 143–150.

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

FELDMAN, B. E., O'BRIEN, J. F., KLINGNER, B. M., AND GOKTEKIN, T. G. 2005. Fluids in deforming meshes. In *SCA 2005*.

FOSKEY, M., LIN, M. C., AND MANOCHA, D. 2003. Efficient computation of a simplified medial axis. In *SM 2003*, 96–107.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graph. Mod. and Im. Proc. 58*, 5, 471–483.

FOSTER, N., AND METAXAS, D. 1997. Controlling fluid animation. 178–188.

FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A. P., AND JONES, T. R. 2000. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *SIGGRAPH '00*, 249–254.

GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. 2005. Coupling water and smoke to thin deformable and rigid shells. In *SIGGRAPH '05*, 973–981.

HONG, J.-M., AND KIM, C.-H. 2005. Discontinuous fluids. *ACM Trans. Graph. 24*, 3, 915–920.

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. 25*, 3, 805–811.

KIPFER, P., AND WESTERMANN, R. 2006. Realistic and interactive simulation of rivers. In *Proceedings Graphics Interface 2006*, 41–48.

KITSIONAS, S., AND WHITWORTH, A. P. 2002. Smoothed particle hydrodynamics with particle splitting, applied to self-gravitating collapse. *Monthly Notices of the Royal Astronomical Society 330*.

KLINGNER, B. M., FELDMAN, B. E., CHENTANEZ, N., AND O'BRIEN, J. F. 2006. Fluid animation with dynamic meshes. In *SIGGRAPH '06*.

KOLLURI, R., O'BRIEN, J. F., AND SHEWCHUCK, J. R. 2004. Provably better moving least squares. In *Annual Fall Workshop on Computational Geometry*.

LASTIWKA, M., QUINLAN, N., AND BASA, M. 2005. Adaptive particle distribution for smoothed particle hydrodynamics. *Int. J. Numer. Meth. Fluids 47*.

LIU, M. B., LIU, G. R., AND LAM, K. Y. 2006. Adaptive smoothed particle hydrodynamics for high strain hydrodynamics with material strength. *Shock Waves 15*.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87*, vol. 21, 163–169.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. In *SIGGRAPH '04*, 457–462.

MONAGHAN, J. J. 2005. Smoothed particle hydrodynamics. *Rep. Prog. Phys. 68*, 1703–1758.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *SCA 2003*, 154–159.

OWEN, J. M., VILLUMSEN, J. V., SHAPIRO, P. R., AND MARTEL, H. 1998. Adaptive smoothed particle hydrodynamics: Methodology. II. *The Astroph. J. Sup. Series 116*.

PAULY, M., KEISER, R., ADAMS, B., DUTRÉ;, P., GROSS, M., AND GUIBAS, L. J. 2005. Meshless animation of fracturing solids. *ACM Trans. Graph. 24*, 3, 957–964.

PREMOZE, S., TASDIZEN, T., BIGLER, J., LEFOHN, A. E., AND WHITAKER, R. T. 2003. Particle-based simulation of fluids. *Comput. Graph. Forum 22*, 3, 401–410.

SETHIAN, J. A. 1999. *Level Set Methods and Fast Marching Methods*. Cambridge University Press.

SHI, L., AND YU, Y. 2004. Visual smoke simulation with adaptive octree refinement. In *Computer Graphics and Imaging*.

STAM, J. 1999. Stable fluids. In *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. In *SCA 2006*, 157–166.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. In *SIGGRAPH '06*, 826–834.

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