Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2011)
A. Bargteil and M. van de Panne (Editors)

# A Particle-based Method for Preserving Fluid Sheets

Ryoichi Ando[1] and Reiji Tsuruno[2]

[1]Graduate School of Design, Kyushu University, Japan
[2]Faculty of Design, Kyushu University, Japan

**Abstract**

*We present a new particle-based method that explicitly preserves thin fluid sheets for animating liquids. Our primary contribution is a meshless particle-based framework that splits at thin points and collapses at dense points to prevent the breakup of liquid. In contrast to existing surface tracking methods, the proposed framework does not suffer from numerical diffusion or tangles, and robustly handles topology changes by the meshless representation. As the underlying fluid model, we use Fluid-Implicit-Particle (FLIP) with weak spring forces to generate smooth particle-based liquid animation that maintains an even spatial particle distribution in the presence of eddying or inertial motions. The thin features are detected by examining stretches of distributions of neighboring particles by performing Principle Component Analysis (PCA), which is used to reconstruct thin surfaces with anisotropic kernels. Our algorithm is intuitively implemented, easy to parallelize and capable of producing visually complex thin liquid animations.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

## 1. Introduction

Over the past decades physically based simulations of fluid have attracted much attention in the graphics community [Bri08]. State-of-the-art fluid animation has proven effective for creating impressive scenes of various fluid phenomena, such as smoke, fire, and water, and is widely used in the film industry.

In recent years, the interest in simulating thin liquid surfaces has been growing [MÖ09, WTGT09, WTGT10, BBB10]. The thin fluid features are very difficult to track precisely with traditional methods. An Eulerian representation such as the level set method [OS88] filters the features due to the numerical diffusion. The particle level set method [ELF05] breaks up thin sheets at the reinitialization phase. Present day researchers favor the use of a mesh-based surface tracking method, also known as the front tracking method, under an Eulerian fluid flow. The surface tracking method typically represents surfaces explicitly with vertices connected in a triangle that is advected through the underlying fluid motion. Unlike implicit approaches, the surface tracking method can keep details smaller than a grid cell and is free from numerical smear; however, the topology changes are complex and so the algorithm also becomes complex.

Particle-based methods have become popular since the introduction of the SPH method in computer graphics by Müller et al. [MCG03] due to their ease of implementation and interactive applications [GSSP10, HKK07, YWH*09]. Conventional particle-based methods are feasible for animations of splashes, although the animations suffer from oscillations due to the large spring forces, which make it difficult to track thin fluid features. Many Eulerian-Lagrangian hybrid approaches have been proposed [SBH09, ZB05, LTKF08], but these methods can exhibit splashy particle behaviors for tracking liquid in our applications.

In this paper, we introduce a new particle-based framework that preserves thin fluid features like those in Eulerian fluid. Note that our method is not a family of surface tracking methods in which surfaces are discretized only around the surfaces. In our method, particles are placed both on the surfaces and inside the liquid domain similarly to the SPH method. To achieve our goal, we use a Particle-In-Cell/Fluid-Implicit-Particle (PIC/FLIP) [ZB05] method as an underly-

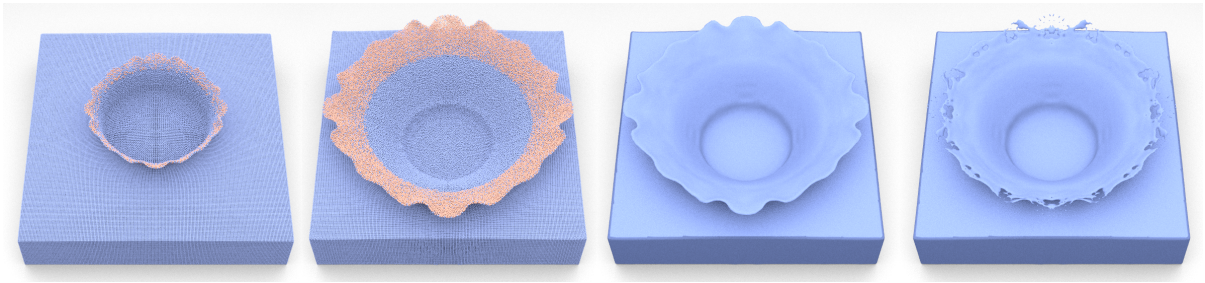


**Figure 1:** ***Water splash by our method.*** *Left two columns: A visualized splash with particles. The pink points indicate newly inserted particles. Third column: A thin surface generated by anisotropic kernels. Fourth column: Our method without particle insertion at the same timing.*

ing fluid solver with weak SPH-like spring forces to make the overall distribution less uneven.

Our main contribution is a purely meshless particle-based method that keeps thin fluid sheets. The thin sheets are preserved by inserting new particles at sparse thin points in the sheets. These particles are then quickly removed as they dive into the deep water. Our work is greatly inspired by the anisotropic kernel method proposed by Yu and Turk [YT10]. This method performs the PCA over the neighboring particles to compute the stretch and orientation of a particle for reconstructing sharp smooth surfaces. We use the stretches as criteria to judge whether to insert particles (see Figure 1).

## 2. Previous work

Our work draws upon a broad range of backgrounds in fluid animation, including particle-based liquid animations and surface tracking methods applied to fluids. In this section, we briefly review the research closely related to our work.

State-of-the-art surface tracking for liquid can be categorized into two approaches: the implicit method and the explicit method. The level set method proposed by Osher and Sethian [OS88] is one of the most popular implicit methods. In the level set method, each grid node is assigned a distance function from the closest surface position, and the surface is implicitly located where the function is zero. The method has been refined and revised in many ways. Enright et al. [EFFM02], Wang et al. [WYS09], and Mihalef et al. [MMS07] placed Lagrangian particles to reduce numerical dissipation. Bargteil et al. [BGOS06] updated the signed distance field from a reconstructed surface mesh to increase the accuracy. Heo and Ko [HK10] preserved the surface detail with a spectrally refined level set (SRL) and a high-order re-initialization method. These implicit methods can be utilized to increase the overall detail with a high resolution grid. However, to the best of our knowledge, such methods cannot completely prevent the rupture of thin surfaces. In addition, some researchers used seed particles to produce splashes [FF01, GSLF05, KCC*06] or to preserve sheets [CFL*07]. We only use particles so that the data structure is more consistent.

Among the explicit approaches, Hirt and Nichols [HN81] proposed a *volume-of-fluid* method that uses a proportion of the interface for the entire cell. This method is seldom employed due to the difficulties of handling a discontinuous interface. In explicit methods, Lagrangian approaches are preferred. Over the past two decades, a number of mesh-based surface tracking methods have been proposed through a variety of research fields, such as medical image analysis and fluid dynamics [KWT88, GGL*98, TBE*01, MÖ9, BB09,WTGT09,WTGT10]. Typically, a mesh-based surface tracker advects explicit surface elements by the underlying motion; however, this method suffers from self-intersection or complex topology changes. These issues may be resolved by trapping complex meshes with uniform cubic cells, or searching adjacent points and resampling them. This strategy tends to make the algorithm complex because it needs to tolerate numerous complex situations. We must emphasize that our method is far from the family of surface tracking methods. In comparison, our particle-based approach does not hold any mesh information. Thus, our method does not suffer from such complexities.

Particle methods such as the Smoothed-Particle Hydrodynamics (SPH) method [MCG03, BT07, SP09], the Moving Particle Semi-implicit (MPS) method [KTO96, PTB*03], and the meshless physical simulations [PKA*05, KAG*05, GBB09] are usually used not only to simulate entire physical motion, but also to reconstruct surfaces. Blinn [Bli82], Zhu and Bridson [ZB05], Adams et al. [APKG07], and Yu and Turk [YT10] proposed a novel surface reconstruction algorithm from a point cloud. In their method, surfaces are implicitly defined with respect to the distance from particles. However, these methods exhibit poor, blobby ruptures where the particles are sparse. In our particle-based method, we employ the method of Yu and Turk [YT10] to reconstruct surfaces after filling ruptures with extra particles so that the sheets are preserved.

Our splittable particle-based approach is considered to be

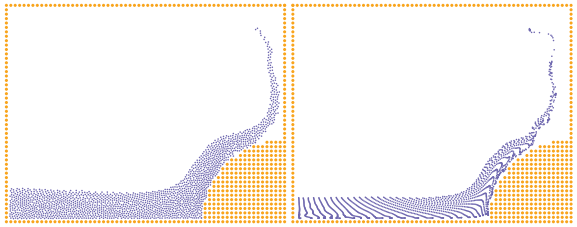


**Figure 2:** ***Two-dimensional dam break.*** *Left: Our liquid simulator. Right: The PIC/FLIP method. Notice that the spatial particle distribution is almost uniform in our image while the PIC/FLIP image is uneven.*

a variation of the adaptive particle-based method [APKG07, HHK08, YWH*09]. The biggest difference between the methods is that ours is specifically designed to keep the continuous fluid sheets whereas the other adaptively resamples particles with ones of a different size to reduce the computational cost while retaining the visual detail.

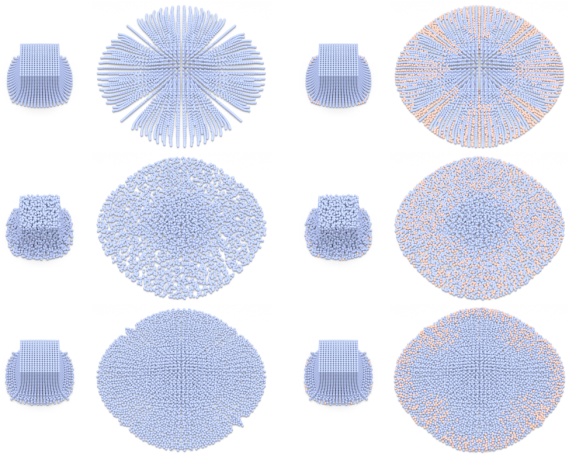


**Figure 3:** ***Effect of our weak spring force.*** *From top to bottom rows: uniformly placed PIC/FLIP, jittered placed PIC/FLIP, uniformly placed PIC/FLIP with the spring force. Left two columns: without particle split. Right two columns: with particle split.*

## 3. Underlying Fluid

This section describes an underlying liquid solver which we used to generate smooth liquid motion for preserving thin surfaces. Our liquid solver is actually a slightly modified version of the Particle-in-Cell/Fluid-Implicit-Particle (PIC/FLIP) method [ZB05], which is an Eulerian-Lagrangian hybrid approach in which a liquid domain is discretized with a collection of particles. Given a particle distribution, the fluid velocity $\boldsymbol{u}$ and the particle density $\rho$ at location $\boldsymbol{x}$ are computed as

$$\boldsymbol{u}(\boldsymbol{x}) = \frac{\sum_i m_i \boldsymbol{u}_i W_{sharp}(\boldsymbol{p}_i - \boldsymbol{x}, \alpha_1 d_0)}{\sum_i m_i W_{sharp}(\boldsymbol{p}_i - \boldsymbol{x}, \alpha_1 d_0)} \quad (1)$$

$$\rho(\boldsymbol{x}) = \sum_i m_i W_{smooth}(\boldsymbol{p}_i - \boldsymbol{x}, \alpha_2 d_0), \quad (2)$$

where $d_0$, $\boldsymbol{p}_i$, $\boldsymbol{u}_i$ and $m_i$ denote the initial space between particles, the position, the velocity and the mass of a particle $i$, respectively, and $\alpha_1$ and $\alpha_2$ are the scaling constants for radius $d_0$. We used $\alpha_1 = 1.0$ and $\alpha_2 = 4.0$. For weighting kernels we used two simple kernels:

$$W_{sharp}(\boldsymbol{r}, h) = \begin{cases} h^2/||\boldsymbol{r}||^2 - 1 & 0 \le ||\boldsymbol{r}|| \le h \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

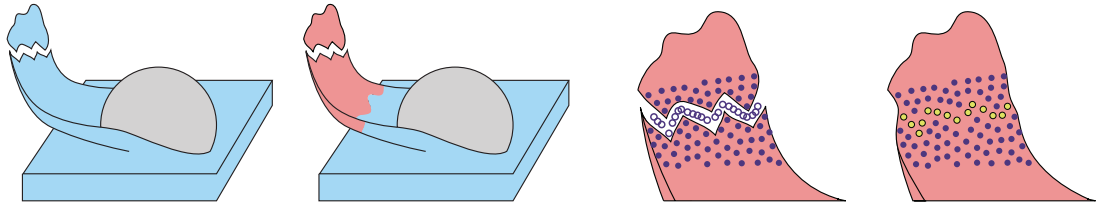
$$W_{smooth}(\boldsymbol{r}, h) = \begin{cases} 1 - ||\boldsymbol{r}||^2/h^2 & 0 \le ||\boldsymbol{r}|| \le h \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

Any kernels would fit our application if they draw similar curves. In our kernels, $W_{sharp}(\boldsymbol{r}, h)$ is designed to resample a quantity of the particle such that $\boldsymbol{u}(\boldsymbol{p}_i) = \boldsymbol{u}_i$, and $W_{smooth}(\boldsymbol{r}, h)$ is used to compute the average of quantities of the nearby particles. To compute $\boldsymbol{u}(\boldsymbol{x})$ and $\rho(\boldsymbol{x})$, we sorted every particle beforehand into the grid cells and searched nearby particles at $\boldsymbol{x}$ from nearby grid cells. For special cases, Equation 1 picks the nearest particle velocity if the denominators are close to zero. The liquid cell is determined by evaluating whether the cell satisfies

$$\sum_i \rho(\boldsymbol{p}_i) \ge \alpha_3 N_0 \rho_0, \quad (5)$$

where $i$, $\rho_0$, $\alpha_3$, and $N_0$ denote the particle index within a cell, the initial maximum density at the beginning of the simulation, the constant rate that tolerates sparse fluid, and the initial number of particles placed in a cell, respectively. We consistently used $\alpha_3 = 0.2$ and $N_0 = 2^2$ for 2D and $N_0 = 2^3$ for 3D in our implementation. In contrast to the original PIC/FLIP approach, this determination skips liquid flags around the area where particles are sparse, such as a liquid sheet or gaps in the liquid domain. We found this clipping prevents liquid particles from floating on the surface of the liquid, and avoids a volume increase of the liquid domain.

In the first step of our simulation, the advected dirty particle velocity is mapped onto a Marker-And-Cell (MAC) grid. The incompressible velocity on the grid is solved in a conventional manner. In PIC [Har64], the projected grid velocity is mapped to the particles to yield an incompressible particle flow. In our implementation, we used Equation 1 to map the velocity from the particles to the grid, and performed a trilinear interpolation from the grid to the particles. PIC naturally carries the momentum by particles instead of grid-based advection. Notwithstanding, PIC is known to involve successive diffusion due to the back-and-forth remapping of velocities. Later, FLIP [BR86] was proposed to improve the diffusion problem. In FLIP, the changes of the grid velocity from a previous time step is mapped to the particles, and the changes are used to increment particle ve-



(a) Input Simulation (b) Thin Sheet Area Detection (c) Candidate Position Detection (d) Particle Insertion

**Figure 4:** ***Algorithm Overview.*** *(a) Given an input collection of particles, (b) we clip out thin particle clouds. To fill in breaking sheets, (c) we compute the candidate particle positions for insertion and (d) we insert the particles with moderate spacing.*

locities to obtain the new velocity. These particles are then moved through the grid velocity field. Unlike PIC, FLIP does not suffer from numerical dissipation; however, FLIP suffers from noisy behavior. In the PIC/FLIP approach [ZB05], such noise is slightly flattened by linearly blending PIC and FLIP by some scalar $k$, as follows:

$$\boldsymbol{u} = \text{PIC/FLIP}(\boldsymbol{u}^*) = k\ \text{FLIP}(\boldsymbol{u}^*) + (1-k)\text{PIC}(\boldsymbol{u}^*). \quad (6)$$

where $\boldsymbol{u}^*$ and $\boldsymbol{u}$ denote the dirty velocity of particles after the advection and the new particle velocity, respectively. Typically, $k$ is close to 1. In our implementation we set $k = \alpha_4$, where $\alpha_4 = 0.75 \sim 0.85$. For the time step size, we set $\Delta t = 0.6 \cdot 10^{-2}$. A PIC/FLIP linear interpolation could possibly be used to adjust the visual viscosity of a liquid, but both PIC and FLIP deform the spatial particle distribution unevenly. To alleviate this problem, we displace particles by SPH-like artificial weak spring forces, as follows:

$$\boldsymbol{f}_i = -\alpha_5 d_0 \sum_j \frac{\boldsymbol{p}_j - \boldsymbol{p}_i}{||\boldsymbol{p}_j - \boldsymbol{p}_i||} W_{smooth}(\boldsymbol{p}_j - \boldsymbol{p}_i, d_0), \quad (7)$$

where $\alpha_5$ denotes the stiffness of the spring. We set $\alpha_5 = 50$. After the particle positions are slightly moved by $\boldsymbol{p}_{\text{new}} = \boldsymbol{p} + \Delta t \boldsymbol{f}$, we resample a new particle velocity from $\boldsymbol{u}$ such that $\boldsymbol{u}_{\text{new}} = \boldsymbol{u}(\boldsymbol{p}_{\text{new}})$. Note that the mass of a particle is neglected since the spring works as a position corrector.

Because the spring is weak, it does not instantly improve the uneven problem, but, as PIC/FLIP gradually improves noise, our spring also slowly improves the uneven particle distribution. We found this also alleviates the volume loss of the liquid. Another advantage of our simulator is that it allows us to take large time steps, which are not allowed in the traditional SPH method. It should be taken into account that the resampling of a new particle velocity introduces extra diffusion, and, thus, the final visual viscosity must be adjusted together with the PIC/FLIP damper. This may seem to counteract FLIP, but we found viscous behavior is appropriate for generating thin liquid animations. Combining PIC and a weak spring force is also possible, but we found this combination was too viscous for our underlying liquid animation.

For obstacles, in addition to the boundary conditions on a mapped MAC cell, we place obstacle particles at the center of MAC cells underneath the wall surfaces. When fluid particles contact a wall particle, we push them out in the normal direction. The spring force of Equation 7 is computed within wall particles to compute the wall normal vectors. Although mesh obstacles might be directly used to handle collisions, we found this approach is much easier than implementing full collision detection with meshes.

Figure 2 shows a two-dimensional comparison with the PIC/FLIP method alone. In our fluid simulator, we found water keeps spinning for a long time while gaining even spatial particle distributions. In the PIC/FLIP method, we experienced volume loss and uneven spatial particle distributions. Figure 3 shows a three-dimensional comparison with and without the weak spring force. Without the spring, particle distribution can easily develop unevenly. In this scenario, jittered placed particles may alleviate the issue, but this technique works only at the beginning of the simulation; eventually, the distribution will become uneven. In contrast, our method is capable of correcting this issue. It may still be possible to apply the sheet preserving algorithm, which is introduced in the next section, to PIC/FLIP alone, as can be seen in the right column of the figure. But without the correction, our sheet preserving algorithm can be triggered frequently, and the uneven particle distribution can fail to find best position to split particles. In this case, some sheets can unevenly break up.

## 4. Preserving Sheets

The main highlight of our method is an algorithm to preserve thin sheet by inserting extra particles at breaking fluid holes. Figure 4 illustrates an overview of our procedure. First, we extract thin fluid sheet particles. Within those particles, new candidate particle positions are suggested and carefully inserted to avoid collisions. We describe the step details in the following subsections.

### 4.1. Thin Particle Extraction

We exploit the anisotropic kernel method [YT10] to detect thin sheets. To do this, we first compute the weighted aver-

age covariance of particles $C$, as follows:

$$C_i = \frac{\sum_j (\boldsymbol{p}_j - \overline{\boldsymbol{p}_i})(\boldsymbol{p}_j - \overline{\boldsymbol{p}_i})^T W_{smooth}(\boldsymbol{p}_j - \overline{\boldsymbol{p}_i}, \alpha_2 d_0)}{W_{smooth}(\boldsymbol{p}_j - \overline{\boldsymbol{p}_i}, \alpha_2 d_0)}, \quad (8)$$

where

$$\overline{\boldsymbol{p}_i} = \frac{\sum_j \boldsymbol{p}_j W_{smooth}(\boldsymbol{p}_j - \boldsymbol{p}_i, \alpha_2 d_0)}{W_{smooth}(\boldsymbol{p}_j - \boldsymbol{p}_i, \alpha_2 d_0)}. \quad (9)$$

The Singular Value Decomposition (SVD) of the associated $C_i$ yields the direction and the stretch of neighboring particle positions as eigenvectors and eigenvalues, as follows:

$$C_i = \begin{bmatrix} \boldsymbol{e}_1^T \\ \boldsymbol{e}_2^T \\ \boldsymbol{e}_3^T \end{bmatrix}^T \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_1^T \\ \boldsymbol{e}_2^T \\ \boldsymbol{e}_3^T \end{bmatrix} \quad (10)$$

where $\boldsymbol{e}_n$ and $\sigma_n$ denote the principle axes ordered by the variance and the degree of stretch, respectively. Note that the SVD is applied only where $\alpha_6\rho_0 < \rho(\boldsymbol{p}_i) < \alpha_7\rho_0$; otherwise, $\mathrm{diag}(\sigma_1, \sigma_2, \sigma_3) = \boldsymbol{I}$ is used. We used $\alpha_6 = 0.05, \alpha_7 = 0.7$. Since the SVD is applied only around the surfaces, it does not monopolize the computation in terms of the number of particles. The thin particles (Figure 4(b)) are extracted by examining

$$\sigma_3 \leq \alpha_8 \sigma_1, \quad (11)$$

where $\alpha_8$ denotes a threshold that determines the thin degree. In our implementation we used $\alpha_8 = 0.2$. These thin particles are used to split particles in the next step.

### 4.2. Finding the Candidate Position

Within the extracted thin particles, we search for pairs $(\boldsymbol{p}_i, \boldsymbol{p}_j)$ that bridge the breaking holes. A midpoint of the pair $(\boldsymbol{p}_i + \boldsymbol{p}_j)/2$ is recorded as a candidate position to split (Figure 4(c)). In our method we choose pairs $(\boldsymbol{p}_i, \boldsymbol{p}_j)$ that satisfy all of the following conditions:

$$\text{all} \begin{cases} \alpha_9 d_0 \leq ||\boldsymbol{p}_j - \boldsymbol{p}_i|| \leq \alpha_{10} d_0 \\ \sum_k W_{smooth}((\boldsymbol{p}_i + \boldsymbol{p}_j)/2 - \boldsymbol{p}_k, \alpha_9 d_0) = 0 \\ (\boldsymbol{p}_j - \boldsymbol{p}_i) \cdot (\boldsymbol{u}_j - \boldsymbol{u}_i) > 0 \end{cases} \quad (12)$$

where $\alpha_9$ and $\alpha_{10}$ denote constants that control the minimum and maximum space of candidate positions, respectively. We use $\alpha_9 = 0.8$ and $\alpha_{10} = 3.5$. In the first row of Equation 12, we check whether the two particles lie at a moderate distance from each other. In the second row, we check whether the candidate midpoint is sparse enough to contain a new particle at radius $\alpha_9 d_0$. Note that at this time, scarcity is only checked among existing particles. The third row checks whether the distance is increasing between the pair of particles with the passing of time. Once these pairs are found, we put the midpoints into list $S$ without duplication. We illustrate the example of candidate positions with and without thin particle detection in Figure 5.

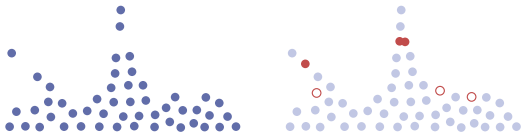
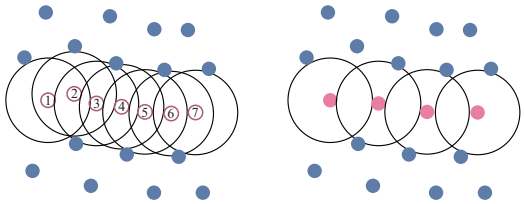


**Figure 5: *2D thin particle clipping.*** *Left: Simulation input. Right: Both the solid and the open red particles satisfy Equation 12. Only the solid red particles satisfy Equation 11.*




(a) Candidate Particles (b) Inserted Particles

**Figure 6: *3D particle insertion (sheet viewed from the top).*** *(a): Suggested candidate particles. (b): Actually inserted particles and insertion conducted in serial order by the numbers shown. A black contour indicates the particle radius.*

### 4.3. Particle Insertion and Collapse

The candidate particles cannot be inserted naively because these candidates are usually too crowded. To cope with this problem, we describe a simple algorithm to insert candidates at a moderate sparseness. Let $I$ be a list of finally inserted candidates indices. The first particle $I_1$ is the most sparse candidate within $S$, as follows:

$$I_1 = \arg\min_{j \in S} \rho(\boldsymbol{p}_j). \quad (13)$$

Once $I_1$ is found, we pop out every nearby candidate around $\boldsymbol{p}_{I_1}$ from $S$ that exists within radius $\alpha_9 d_0$. In the second step we search for nearby particles $N_{I_1}$ in $S$ around $\boldsymbol{p}_{I_1}$ that exists within radius $\alpha_{11} d_0$, where we set $\alpha_{11} = 2.0$. If found, we insert the closest candidate within $N_{I_1}$ as

$$I_2 = \arg\min_{j \in N_{I_1}} ||\boldsymbol{p}_j - \boldsymbol{p}_{I_1}||. \quad (14)$$

If we formulate the search as $I_{n+1} = \mathrm{search}(I_n)$, which means $I_2$ can be found from $I_1$, we can repeat this task to find $I_3, I_4, \cdots$ serially. If the next search fails, Equation 13 is used instead. Figure 6 illustrates an example of our insertion procedure. In this example, crowded candidates are carefully inserted without collisions.

When splitting particles, every attribute is linearly interpolated except for the mass. In the case of splitting a pair $(\boldsymbol{p}_i, \boldsymbol{p}_j)$, a new candidate mass is given by $m = (m_i + m_j)/3$. After the split, new masses of particles $i$ and $j$ are reduced to $m_i^{new} = \frac{2}{3} m_i, m_j^{new} = \frac{2}{3} m_j$. The change of mass affects the functions of Equations 1 and 2.

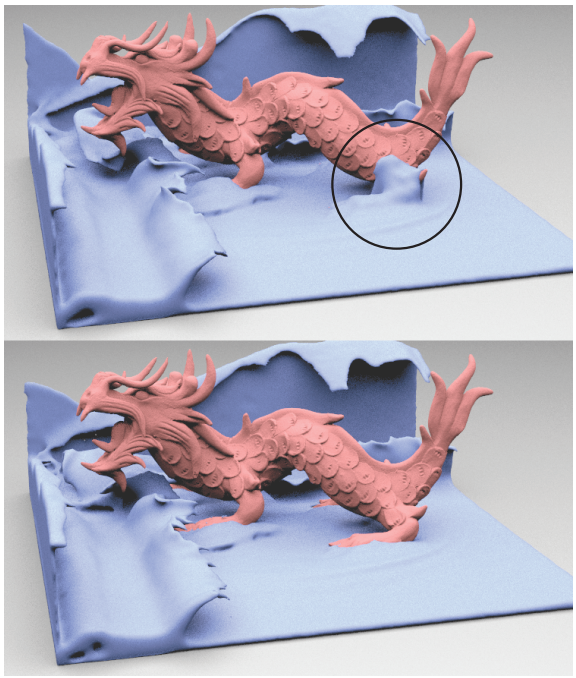


**Figure 7:** ***Liquid stuck to an obstacle.*** *Top: The liquid sheet remains stuck to the dragon model. Bottom: The sheet was removed by repositioning the stuck particles into the deep water.*

In contrast, an inserted particle $\boldsymbol{p}_i$ collapses if it satisfies any of the two following conditions

$$\text{any} \begin{cases} \rho(\boldsymbol{p}_i) > \alpha_{12}\rho_0 \quad \text{and} \quad \sigma_3 \geq \alpha_8\sigma_1 \\ \text{for any particle } j \quad ||\boldsymbol{p}_i - \boldsymbol{p}_j|| < \alpha_{13} d_0 \end{cases} \tag{15}$$

where $\alpha_{12}$ and $\alpha_{13}$ denote the maximum density coefficient and the minimum space, respectively. We use $\alpha_{12} = 0.2, \alpha_{13} = 0.2$. Nonetheless, this collapse can introduce a flickering effect in terms of the cycle of the split and the collapse. In our approach we reduce the flicker by waiting some random time steps before the removal. We only remove the particle if it continues to satisfy Equation 15. When removing particles, the mass is simply returned to the source of its split particles. Note that this redistribution of mass can lead to sudden non-local changes in momentum near both the disappearing split particle and its owner particle. But the effect is only temporary and is almost unnoticeable. When splitting particles more than once, we track the hierarchy of these parent particles, and the mass is returned to the top of these parents.

The aforementioned split can trap liquid sheets on obstacles. As a remedy, we detect particles that are stuck on walls by watching whether the cells up or down from the particles are a wall cell and whether the particle density is lower than some threshold $\rho(\boldsymbol{p}_i) < \alpha_{14}\rho_0$. If the particle is a split particle, we remove it from the simulation. If not, we reposition it to some random place in the liquid so that the global volume is maintained. The effect of the stuck sheet removal is shown in Figure 7. In this example, we used $\alpha_{14} = 0.2$.

**Algorithm 1** PRESERVE THIN FLUID SHEET

```
C ← Compute Covariance Around p by Eq. 8
RΣR^T ← SVD(C)
P ← Extract Thin Sheet Particles(Σ) by Eq. 11
for all (i, j) ∈ P do
    if a pair (p_i, p_j) satisfies Eq. 12 then
        S ← Insert New Candidate (p_i + p_j)/2
    end if
end for
I_1 ← by Eq. 13 in S    // Find First Candidate to Insert
for n = 2, 3, 4, ··· do
    I_n ← search(I_{n−1}) in S as Eq. 14
end for
Insert New Particles I
Collapse Particles that Satisfy Eq. 15
Reposition Particles Stuck on Walls
```

## 5. Surface Reconstruction

The SVD information of Equation 10 is reused by anisotropic kernels [YT10] to reconstruct the thin surfaces. For brevity, we employed a simple implicit function:

$$\phi(\boldsymbol{x}) = \min_i(||G_i(\boldsymbol{p}_i - \boldsymbol{x})||), \tag{16}$$

where $G_i$ denotes a deformation matrix in terms of particle $i$

$$G_i = \frac{1}{k_s} \begin{bmatrix} \boldsymbol{e}_1^T \\ \boldsymbol{e}_2^T \\ \boldsymbol{e}_3^T \end{bmatrix}^T \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{e}_1^T \\ \boldsymbol{e}_2^T \\ \boldsymbol{e}_3^T \end{bmatrix} \tag{17}$$

where $k_s$ denotes a scaling constant such that $||k_s C_i|| \approx 1$.To prevent a significant stretch, $\sigma_n$ is constrained within some range. Because we meshed with the Marching Cube algorithm [LC87], we set the minimum stretch to be larger than half of a grid width to make sure the thin sheets were captured. In addition, we applied straightforward mesh-based smoothing to reduce the bumps.

## 6. Implementation

The pseudo code of the thin fluid sheet preservation is illustrated in Algorithm 1. Every liquid solver step, such as mapping between the grid and a particle, solving for pressure and applying the weak spring correction, is done in a parallel manner. The thin feature preserving algorithm is also parallelized, except for particle insertion and collapse. In our implementation we gained significant acceleration by OpenMP directives.

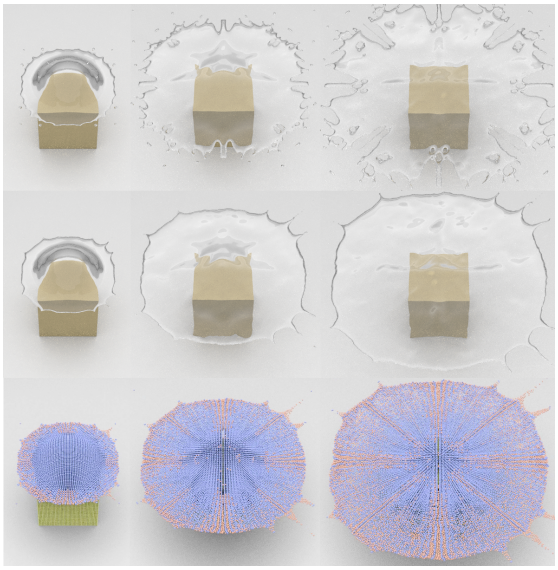


**Figure 8:** ***Water drop on cubic tower.*** *Top row: Our liquid solver only. Middle row: Our liquid solver with the sheet preserving algorithm. Bottom row: Visualization with the particles. The pink points indicate split particles.*

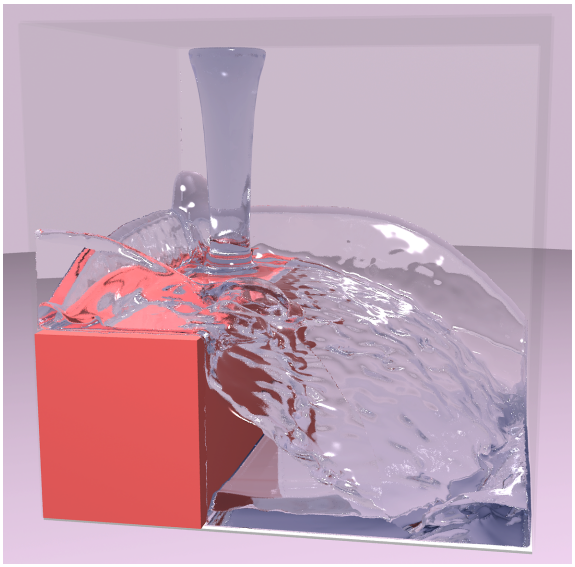


**Figure 9:** ***Water pouring off a box cliff.*** *Poured water jumps off the cliff edge, maintaining a continuous fluid sheet and then landing on the pool.*

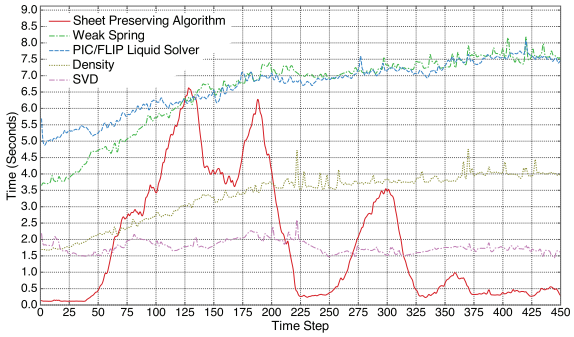



**Figure 10:** ***Timing of dam breaking test of Figure 13.*** *The water dam collapsed at approximately time step 80. This test was simulated with approximately 770k particles at a* $100^3$ *grid size.*

## 7. Parameters

Our algorithm has numerous constant parameters. So far, we have introduced 15 user-adjustable parameters $\alpha_1 \sim \alpha_{14}$ and $\Delta t$, but more are possible in actual implementation. In detail, our fluid simulator has 6 parameters and the sheet preserving algorithm has 9 parameters. Fortunately, we found these parameters are not sensitive in terms of stability. Thus, we believe our constants can tolerate harsh implementation. $\Delta t$ can be heuristically changed but it does not have to be very small for stability.

## 8. Results

We produced several sequences of liquid animation by using our method with a Core i7 Quad Core 2.8 GHz PC running Linux. The meshing took approximately 10 to 30 seconds and the transparent rendering took approximately 1 to 5 minutes per frame. The simulation took approximately 10 to 60 seconds per time step in the examples shown here. Figure 8 shows an example of a water drop splash on a cubic tower. In this example, without the particle split, the water sheet rapidly ruptures as it expands. Nevertheless, we successfully maintained a continuous liquid sheet by inserting new particles in the holes. The inserted particles are shown in pink in the bottom row of the figure.

The entire sheet preserving algorithm took up to 60% (or more) of the simulation time, depending on the animation complexity. The most time-consuming parts were the weak spring correction step and the density computation phase in the liquid solver, which generally took approximately half of the simulation time. Figure 10 shows the detailed timing of the computational time of the dam breaking test of Figure 13. When the dam collapsed in approximately 80 steps, the computational time of the sheet preserving algorithm suddenly grew; however, it gradually settled back to the time of the original steps. In fact, we observed that our sheet preserving algorithm did not exponentially increase the overall simulation time. This simulation ran with approximately 770k particles at a $100^3$ grid size.

Figure 9 shows an example of water pouring off the edge

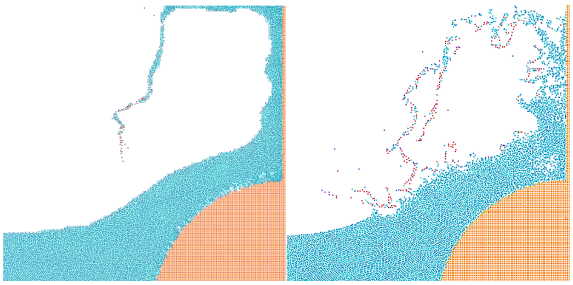


**Figure 11:** ***The SPH method with the particle split.*** *Left: Our liquid simulator with the particle insertion. Right: The SPH method coupled with our particle split method. Notice that the SPH generates numerous strings.*

of a box. Notice that the jumping sheets growing from the cliff edge hit the pool without rupture. Each time step took approximately 5 to 40 seconds. The number of liquid particles was approximately 576*k*. Blue transparent rendering was done with PIXIE[†] and the point rendering was done with pbrt renderer[‡].

Figure 12 shows an Enright deformation test [Enr02]. In this test, a solid sphere filled with particles is deformed according to Enright's artificial flow field. The thin sheet, which is often filtered with the traditional level set method, is well kept due to the extra particle insertion, shown in pink. These particles are then removed as the deformation returns to its original form.

## 9. Discussion

The SPH method can be used instead of our liquid solver. However, we felt the SPH was noisy to couple with our particle splitting method, even at high viscosity. Figure 11 shows a comparison of the SPH and our PIC/FLIP-based fluid solver coupled with the sheet preserving method. As can be seen from the figure, the splashy behavior of the SPH makes it hard to detect unique thin fluid sheets. This behavior may seem natural because we did not incorporate surface tension. Perhaps, it may still be possible to couple the SPH with our sheet preserving method.

In some cases the fluid sheet seems to limitlessly expand as it stretches, but it eventually breaks into pieces of water drops because Equation 11 prevents splitting when the density is too low (when the density is too low $\mathrm{diag}(\sigma_1,\sigma_2,\sigma_3) = \boldsymbol{I}$ is used). However, in our example we abandoned the split at some degree. As mentioned in the previous section, the weak spring correction step and the density computing phase were the most time-consuming parts of our simulation, although those steps may be accelerated by some SPH related techniques. In this paper we applied our algorithm only with liquid, but it also would be applicable to other phenomena such as viscoelastic materials or wispy smokes.

We described a quick technique to remove the stuck sheet on walls but in some experiments we found this method did not work sufficiently. For example, we observed that liquid keeps spilling out from objects longer than expected, but we believe such artifacts can be alleviated by incorporating a particle-based surface tension force. It may be possible to place particles only on surfaces or around surfaces, as with the Particle Level Set [EFFM02], but the inner domain must be discretized in some manner to track liquid motion. In our method, particles are directly coupled with a liquid solver, so we believe that filling the liquid with particles is reasonable. One of our promising extensions is to employ adaptive approaches to increase the efficiency [HHK08], which we leave as future work.

## 10. Conclusion

In this paper we described a new particle-based algorithm that preserves thin fluid sheets. As an underlying liquid animation, we employed the PIC/FLIP method with the weak spring force to generate a smooth liquid flow, which we found effective for tracking thin fluid. To merge breaking fluid sheets, we first clipped out an overall thin fluid area by performing the PCA over neighboring particles. Within this area, particles were carefully split to avoid collisions when filling liquid ruptures. As shown in the results, our particle-based method is effective in creating thin liquid animations. In future work we would like to accelerate our algorithm and expand into various physical phenomena.

## 11. Acknowledgements

We would like to thank our anonymous reviewers for their many helpful comments which substantially improved this paper. This work was supported by a research fellowship from the Japan Society for the Promotion of Science.

## References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. Graph. 26* (July 2007).

[BB09] BROCHU T., BRIDSON R.: Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing 31*, 4 (2009), 2472–2493.

[BBB10] BROCHU T., BATTY C., BRIDSON R.: Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. 29* (July 2010), 47:1–47:9.

[BGOS06] BARGTEIL A. W., GOKTEKIN T. G., O'BRIEN J. F., STRAIN J. A.: A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graph. 25* (January 2006), 19–38.

[†] PIXIE: http://www.renderpixie.com/

[‡] pbrt: http://www.pbrt.org/

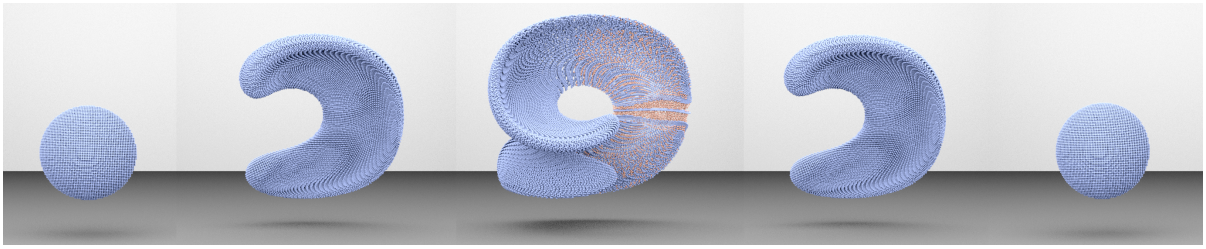


**Figure 12:** ***Enright deformation test.*** *Our particle-based method can tolerate the thin features of Enright's strong deformation by particles that split at the breaking points (shown in pink) and collapse at the dense points.*

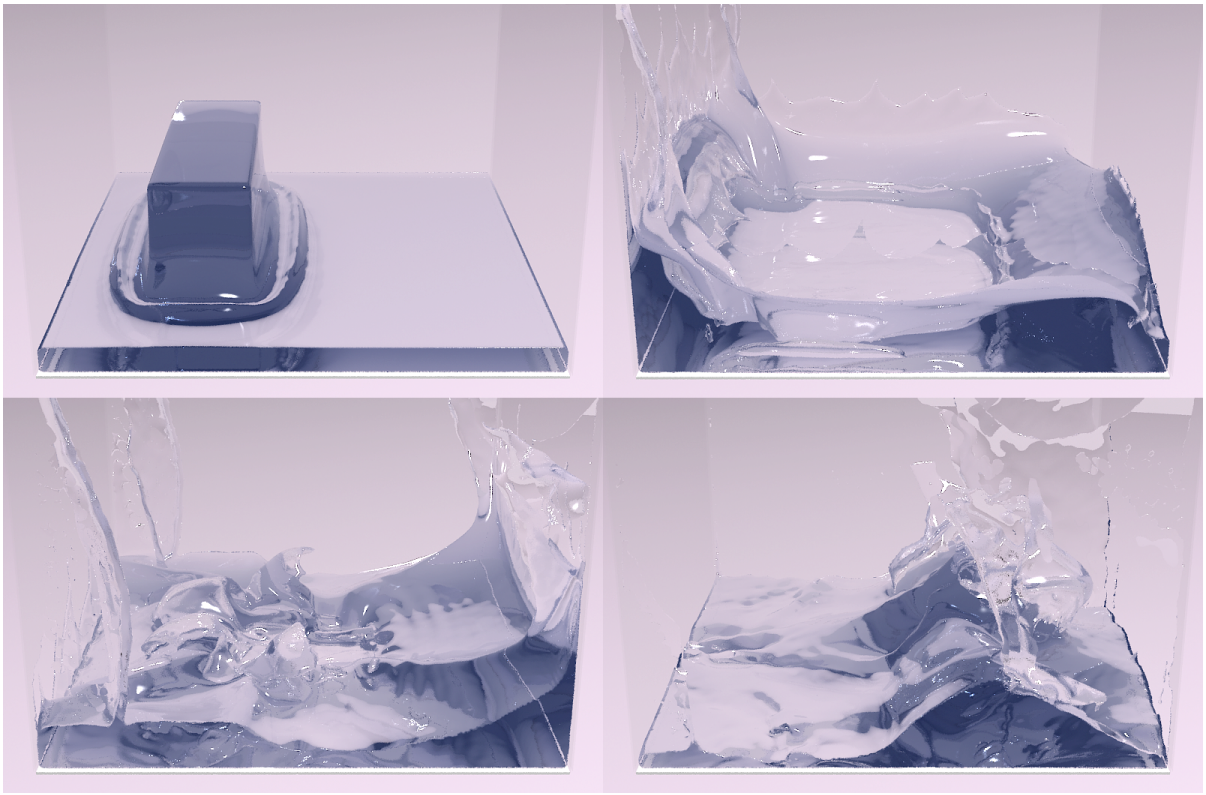


**Figure 13:** ***Dam breaking test with our particle-based thin sheet preserving algorithm.*** *Sequence order 1: Left top 2: Right top 3: Left bottom 4: Right bottom.*

[Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *ACM Trans. Graph. 1* (July 1982), 235–256.

[BR86] BRACKBILL J., RUPPEL H.: Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics 65*, 2 (1986), 314 – 343.

[Bri08] BRIDSON R.: *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press, 9 2008.

[BT07] BECKER M., TESCHNER M.: Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 209–217.

[CFL*07] CHENTANEZ N., FELDMAN B. E., LABELLE F., O'BRIEN J. F., SHEWCHUK J. R.: Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 219–228.

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

[ELF05] ENRIGHT D., LOSASSO F., FEDKIW R.: A fast and ac-

curate semi-lagrangian particle level set method. *Comput. Struct. 83*, 6-7 (2005), 479–490.

[Enr02] ENRIGHT D. P.: *Use of the particle level set method for enhanced resolution of free surface flows*. PhD thesis, Stanford, CA, USA, 2002. AAI3067855.

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 23–30.

[GBB09] GERSZEWSKI D., BHATTACHARYA H., BARGTEIL A. W.: A point-based method for animating elastoplastic solids. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 133–138.

[GGL*98] GLIMM J., GROVE J. W., LI X. L., SHYUE K.-M., ZENG Y., ZHANG Q.: Three-dimensional front tracking. *SIAM J. Sci. Comput. 19*, 3 (1998), 703–727.

[GSLF05] GUENDELMAN E., SELLE A., LOSASSO F., FEDKIW R.: Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. 24* (July 2005), 973–981.

[GSSP10] GOSWAMI P., SCHLEGEL P., SOLENTHALER B., PAJAROLA R.: Interactive sph simulation and rendering on the gpu. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2010), SCA '10, Eurographics Association, pp. 55–64.

[Har64] HARLOW F. H.: The particle-in-cell computing method for fluid dynamics. *Methods Comput. Phys*, 3 (1964), 319–343.

[HHK08] HONG W., HOUSE D. H., KEYSER J.: Adaptive particles for incompressible fluid simulation. *Vis. Comput. 24* (July 2008), 535–543.

[HK10] HEO N., KO H.-S.: Detail-preserving fully-eulerian interface tracking framework. In *ACM SIGGRAPH Asia 2010 papers* (New York, NY, USA, 2010), SIGGRAPH ASIA '10, ACM, pp. 176:1–176:8.

[HKK07] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics on gpus. In *Computer Graphics International* (2007), pp. 63–70.

[HN81] HIRT C. W., NICHOLS B. D.: Volume of fluid vof method for the dynamics of free boundaries. *Journal of Computational Physics 39* (January 1981), 201–225.

[KAG*05] KEISER R., ADAMS B., GASSER D., BAZZI P., DUTRE P., GROSS M.: A unified lagrangian approach to solid-fluid animation. *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics 0* (2005), 125–148.

[KCC*06] KIM J., CHA D., CHANG B., KOO B., IHM I.: Practical animation of turbulent splashing water. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), SCA '06, Eurographics Association, pp. 335–344.

[KTO96] KOSHIZUKA S., TAMAKO H., OKA Y.: A particle method for incompressible viscous flow with fluid fragmentation. *Comput. Fluid Dynamics J. 29(4)* (1996).

[KWT88] KASS M., WITKIN A., TERZOPOULOS D.: Snakes: Active contour models. *International journal of computer vision 1*, 4 (1988), 321–331.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21* (August 1987), 163–169.

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

[MÖ9] MÜLLER M.: Fast and robust tracking of fluid surfaces. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 237–245.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 154–159.

[MMS07] MIHALEF V., METAXAS D., SUSSMAN M.: Textured liquids based on the marker level set. *Computer Graphics Forum 26*, 3 (2007), 457–466.

[OS88] OSHER S., SETHIAN J. A.: Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *J. Comput. Phys. 79*, 1 (1988), 12–49.

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRÉ P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. *ACM Trans. Graph. 24* (July 2005), 957–964.

[PTB*03] PREMŽOE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R. T.: Particle-based simulation of fluids. *Computer Graphics Forum 22*, 3 (2003), 401–410.

[SBH09] SIN F., BARGTEIL A. W., HODGINS J. K.: A point-based method for animating incompressible flow. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 247–255.

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible sph. *ACM Trans. Graph. 28* (July 2009), 40:1–40:6.

[TBE*01] TRYGGVASON G., BUNNER B., ESMAEELI A., JURIC D., AL-RAWAHI N., TAUBER W., HAN J., NAS S., JAN Y.: A front-tracking method for the computations of multiphase flow. *Journal of Computational Physics 169*, 2 (2001), 708–759.

[WTGT09] WOJTAN C., THÜREY N., GROSS M., TURK G.: Deforming meshes that split and merge. In *ACM SIGGRAPH 2009 papers* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 76:1–76:10.

[WTGT10] WOJTAN C., THÜREY N., GROSS M., TURK G.: Physics-inspired topology changes for thin fluid features. *ACM Trans. Graph. 29* (July 2010), 50:1–50:8.

[WYS09] WANG Z., YANG J., STERN F.: An improved particle correction procedure for the particle level set method. *J. Comput. Phys. 228* (September 2009), 5819–5837.

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

[YWH*09] YAN H., WANG Z., HE J., CHEN X., WANG C., PENG Q.: Real-time fluid simulation with adaptive sph. *Computer Animation and Virtual Worlds 20*, 2-3 (2009), 417–426.

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Trans. Graph. 24* (July 2005), 965–972.