Vis Comput
DOI 10.1007/s00371-015-1080-4

ORIGINAL ARTICLE

Water simulation using a responsive surface tracking for flow-type changes

Jae-Gwang Lim[1] · **Bong-Jun Kim**[1] · **Jeong-Mo Hong**[1]

Abstract The realistic simulation of fluids largely depends on a temporally coherent surface tracking method that can deal effectively with transitions between different types of flows. We model these transitions by constructing a very smooth fluid surface and a much rougher, splashy surface separately, and then blending them together in proportions that depend on the flow speed. This allows creative control of the behavior of the fluids as well as the visual results of the simulation. We overcome the well-known difficulty of obtaining smooth surfaces from Lagrangian particles by allowing them to carry normal vectors as well as signed distances from the level set surface and by introducing a new surface construction algorithm inspired by the moving least-squares method. We also implemented an adaptive form of the fluid-implicit-particle method that only places particles near visually interesting regions, which improves performance. Additionally, we introduce a novel subgrid solver based on the material point method to increase the amount of detail produced by the FLIP method. We present several examples that show visually convincing water flows.

Keywords Fluid modeling · Water simulation · Fluid-implicit-particle method · Surface tracking · Material point method

1 Introduction

Water appears as a metaphor for change and spontaneity in many artworks, and its diverse shapes and movement have inspired computer graphics artists to produce attractive and spectacular imagery. Many techniques have been developed to simulate free-surface water by solving the incompressible Navier–Stokes equations. The fluid-implicit-particle (FLIP) method [7], introduced to computer graphics by Zhu and Bridson [36], has become popular because it produces good results quickly. It reduces numerical dissipation by the use of Lagrangian particles, and an efficient Eulerian incompressibility solver permits large time steps. However, there is a difficulty that comes with the particle representation of water: obtaining a desirable surface for the placement of boundary conditions and for rendering, as discussed by many authors (see Sect. 2). The slow laminar flow of water is dominated by surface tension and viscosity and produces smooth surfaces. Faster transitional flow has both laminar and turbulent regions. Fully turbulent flow creates complicated motions with rough surfaces, and very fast-moving highly turbulent water often becomes atomized. Simulating all these characteristics in a unified manner largely depends on having surface tracking that is temporally coherent while accommodating transitions between different types of flow.

In this paper, we propose a new method of simulating water in which an interfacial surface tracking technique is coupled to flow types in an efficient and controllable way. The key idea is to construct both a smooth and a rough water surface independently and then to blend them using flow velocity as a scalar weight. A major benefit of doing so is that it gives artists the ability to control the behavior of the fluid as well as the appearance of the simulation results by adjusting intuitive parameters and, hence, design scenes in which smooth laminar flows, turbulences, and splashing are all present. This technique also overcomes the well-known difficulty of obtaining smooth surfaces from FLIP particles by allowing them to carry the normal vectors as well as the signed distance values of the level set surface [22]. We have developed a new reconstruction algorithm, inspired by the

✉ Jeong-Mo Hong
jmhong@dongguk.edu

[1] Dongguk University, Seoul, Republic of Korea

Published online: 17 April 2015

moving least squares (MLS) method of Shen et al. [24], to take advantage of the availability of normal constraints. In Sect. 4, we introduce our surface tracking method in detail.

Our surface tracking method allows us to implement an adaptive form of the FLIP method that only places particles near visually interesting regions, to improve both performance and image quality. This approach saves more than enough memory to compensate for the additional storage needed for the normals attached to particles; it also alleviates some practical issues, e.g., the 'void artifacts' that can become apparent in regions where particles are sparse, such as the scarcity of particles in thin films. We introduce details of the adaptive FLIP method in Sect. 5.

We make an additional contribution in the form of a novel subgrid solver based on the material point method (MPM) [30], which increases the amount of dynamic detail of the FLIP method. Although the FLIP method largely overcomes mass and momentum dissipation, it does not follow any clear theory of dynamic interaction among particles, even though these interactions influence the accuracy of a simulation at the subgrid level as well as the particle distribution and volume conservation. MPM addresses this limitation of the FLIP method by using the Eulerian grid as a scratch pad, so that interparticle operations can be performed without the need for neighbor search algorithms. This allows us to implement MPM with only a minor overhead for updating within a FLIP solver. In Sect. 3, we introduce the incompressible MPM, which is MPM and a general FLIP hybrid method.

The contributions of this work can be summarized as follows:

- Efficient and robust smooth-surface construction from FLIP particles.
- Novel surface tracking method, which blends rough and smooth surfaces to be responsive to flow velocity.
- Adaptive particle placement to enhance the performance of the FLIP method.
- MPM as a subgrid solver to improve dynamic detail.

2 Related work

Although level set-based water simulation methods are very effective at tracking smooth water surfaces, many computer graphics researchers have struggled against their intrinsic tendency toward excessive smoothing and numerical dissipation. A noticeable breakthrough was introduced with the incompressible FLIP method of Zhu and Bridson [36], in which Lagrangian particles suppress numerical dissipation, while the Eulerian grid enforces incompressibility on particle velocities. It is not easy to obtain a good water surface from FLIP particles, and this issue has been discussed by many authors. One approach [36] is to use a weighted average of particle positions near each sampling point, and we take this approach in fast splashing regions. In concave regions, the use of weighted averages can be improved [25] by Eigen analysis of particle velocities, and we believe this can replace the approach [36] in our method. The representation of flat surfaces is improved [1] by using the particle-to-surface distances from the previous time step. We deal with flat or very smooth surfaces by using the normals from the previous step.

The level set skinning method [5] repeats biharmonic smoothing operations on a level set iteratively until it finds spatially and temporally coherent surfaces that correspond to a volume of fluid that lies between a predefined minimum and maximum. Our method makes use of these forms of coherence without the need for iteration. Anisotropic reconstruction [33] achieves impressive smoothness, but the computational load is too expensive to be considered in some practical applications. The need for computational efficiency and a novel way to control the behavior of fluids motivated us to develop a tracking technique, so that we do not have to depend solely on the current particle distribution.

By placing particles near the interface of level set intensively, a hybrid particle level set method [8] alleviates the dissipation caused by the advection. Inspired by this method, Ianniello and Di Mascio proposed a new tracking interface method [15] that carries information of the level set on particles for improved accuracy and fast reconstruction of the interface. Like these methods, our surface tracking method also places particles on the level set.

Several adaptive techniques have been developed to improve the performance of the FLIP method, including an octree structure with variable particle size [11], an adaptive tetrahedral mesh [4], and adaptively sampled particles with anisotropic kernels [2] and [3], which are good at representing thin sheets. We think that some of these approaches might be usefully combined with our adaptive particle placement technique in future works.

We implement subgrid dynamics with the material point method, which was first used in computer graphics to simulate snow [28]. Stomahkin et al. [28] provided an introduction to MPM and compared it to FLIP. The MPM was designed to extend FLIP to solid mechanics. In contrast to FLIP, the full stress tensor σ [see Eq. (2)] is carried by particles to track deformation of the fluid body. Therefore, with the stress tensor, we can control properties of fluids in several ways. In MPM, information from the particles is transferred to a background grid, where the momentum equation is solved. Then the grid solution is used to update particles. A variety of problems have been addressed using MPM [10,29,34] and we refer the readers who are not familiar with MPM to the introductory code written by Grant Kot.[1]

[1] https://github.com/kotsoft/.

3 Water simulation

The Navier–Stokes equation can be written as

$$\mathbf{u}_t + (\mathbf{u} \times \nabla)\mathbf{u} = \frac{(\nabla \times \sigma)^{\mathrm{T}}}{\rho} + \mathbf{f}, \tag{1}$$

where $\mathbf{u}$ is the velocity, σ the stress tensor, ρ the density, and $\mathbf{f}$ an external force term that includes gravity, buoyancy, and collision forces. The stress tensor σ is defined as

$$\sigma = -p\mathbf{I} + \lambda \mathrm{tr}(\mathbf{D})\mathbf{I} + \nabla(2\mu\mathbf{D}), \tag{2}$$

where p is the pressure, $\mathbf{I}$ is the second-order identity tensor, $\mathbf{D} = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathrm{T}})$ ($\nabla\mathbf{u}$ means Jacobian matrix) a tensor that expresses the rate of deformation, tr($\mathbf{D}$) the trace of $\mathbf{D}$, μ the coefficient of shear viscosity, and $\lambda + 2\mu/3$ the coefficient of bulk viscosity. Further details are available elsewhere [23].

We also define a signed distance function ϕ to separate water from air, following Enright et al. [9]. The advection of ϕ, driven by $\mathbf{u}$, can be described by the level set equation

$$\phi_t + \mathbf{u} \times \nabla\phi = 0. \tag{3}$$

3.1 Explicit step

We use MPM to solve Eq. (1) in the same way as Sulstky et al. [30]. We can obtain the grid mass m_i of a node i from the mass m_a of a particle a by rasterizing the particle data to the grid as follows:

$$m_i = \sum_a w_{ai} m_a, \tag{4}$$

where ω_{pa} is the weight function relating particle a to node i, which is defined by the dyadic products of univariate cubic B-splines [27], following Stomakhin et al. [28]. Note that $\sum_i \omega_{ia} = 1$ for the neighbor nodes in a uniform three-dimensional grid, $w_{ia} = w_{ai}$ and $\nabla w_{ia} = -\nabla w_{ai}$. The grid density ρ_i is m_i / V_i, where V_i is the cell volume. The node velocity $\mathbf{u}_i$ can be interpolated from the velocities of neighboring particles $\mathbf{u}_a$ as follows:

$$\mathbf{u}_i = \sum_a w_{ai} m_a \mathbf{u}_a / m_i. \tag{5}$$

We can calculate the particle density ρ_a by interpolating ρ_i from neighboring nodes as follows:

$$\rho_a = \sum_i w_{ia} \rho_i. \tag{6}$$

Derivatives can be calculated in a similar way by finding the gradients of the weight functions ∇w_{ia} and ∇w_{ai}.

The operations described above allow us to calculate the stress tensor σ_a from $\mathbf{u}_i$. We can then obtain the node force $\mathbf{f}_i = -\nabla \times \sigma + \mathbf{f}$, which can either be added to $\mathbf{f}_i$ directly or distributed across the particles. This is yet another convenient property that comes from the hybrid nature of MPM. We can now update the grid and particle velocities between time steps l and $l + 1$ as follows:

$$\mathbf{u}_i^{l+1} = \mathbf{u}_i^l + \triangle t \mathbf{f}_i \tag{7}$$

$$\mathbf{u}_a^{l+1} = \mathbf{u}_a^l + \triangle t \sum_i w_{ia} \mathbf{f}_i. \tag{8}$$

Equation (8) specifies that the particle velocity is updated by a force determined at each grid node. This is analogous to the way particle velocities are updated from velocity differences in the FLIP method.

Particle positions are updated by grid velocity, as follows:

$$\mathbf{x}_a^{l+1} = \mathbf{x}_a^l + \triangle t \sum_i w_{ia} \mathbf{u}_i. \tag{9}$$

This provides the advection terms for Eqs. (1) and (3). Note that our FLIP particles carry ϕ_a and $\mathbf{n}_a = \nabla\phi_a / |\nabla\phi_a|$ as well as $\mathbf{u}_a$ and m_a, allowing them to be used for the surface tracking developed in Sect. 4. In regions with no particles, the semi-Lagrangian method [26] was used to advect ϕ and $\mathbf{u}$.

Algorithm 1 Incompressible material point method

```
1: Update solid objects
2: for all substeps do
3:     Explicit step (MPM)
4:     Reconstruct level set and velocity
5:     Redistance level set
6:     Update particles
7: end for
8: Sourcing particles
9: Eulerian projection
```

3.2 Projection

First, we compute an intermediate velocity field $\mathbf{u}^*$ over time step $\triangle t$ using explicit terms, but without the pressure term. After this step, we force $\mathbf{u}^*$ to be incompressible. This is equivalent to computing the pressure p. The mass conservation equation

$$\frac{\partial \rho}{\partial t} + \nabla \times (\rho \mathbf{u}) = 0 \tag{10}$$

expresses the relationship between the divergence of the velocity field and the density calculated from the particles. The velocity $\mathbf{u}^{n+1}$ at the next time step from $\mathbf{u}^*$ can now be expressed in terms of pressure:

$$\mathbf{u}^{n+1} - \mathbf{u}^* + \frac{\nabla p}{\rho}\triangle t = 0. \tag{11}$$

After taking the divergence operator in this equation and substituting for $\mathbf{u}^{n+1}$ using Eq. (10), we have

$$\nabla \times \frac{\nabla p}{\rho}\triangle t = \nabla \times \mathbf{u}^* + \frac{1}{\rho^{n+1}}\frac{\rho^{n+1} - \rho^n}{\triangle t}. \tag{12}$$

Since this system of equations is designed to make ρ converge to a constant rest density ρ_0, we set $\rho = \rho_0$ on the left-hand side and $\rho^{n+1} = \rho_0$ on the right-hand side. Finally, we have a Poisson equation for pressure as follows:

$$\nabla \times \frac{\nabla p}{\rho_0}\triangle t = \nabla \times \mathbf{u}^* - \frac{k}{\triangle t}\left(\frac{\rho^n}{\rho_0} - 1\right). \tag{13}$$

In practice, the constant density in the left-hand side helps the convergence and stability of the pressure solver. We also introduce a control parameter k to achieve a similar effect to the volume control method [17] and the targeting particle number density [18]. In solving this Poisson equation, we set up boundary conditions in terms of ϕ [9]. We use a multigrid Poisson solver, which uses a GPU to speed up computation at the coarsest level solver [16].

3.3 Summary

The simulation steps shown in Algorithm 1 are the same as those in many FLIP solvers, except for the use of MPM in the explicit step and update particles step. To synchronize particles and the level set, particles are consistently updated by interpolating ϕ_a, $\mathbf{n}_a$ from the level set right after the redistance level set step.

The general FLIP method does not consider interactions among particles, so it may lose details such as viscosity and pressure caused between particles. For that reason, we couple MPM to the FLIP method, and MPM improves the details of particle movement as subgrid dynamics. There are several practical ways to couple MPM to the existing incompressible FLIP method. One is to use MPM as an explicit subgrid viscosity solver in which the pressure term $(-p\mathbf{I})$ in Eq. (2) can be ignored because the Eulerian FLIP projection takes care of the incompressibility.

In our examples, for memory efficiency we did not configure subgrid as higher resolution than main grid resolution. Instead of high-resolution subgrid for MPM, we chose to proceed with MPM through substeps to improve the physical accuracy and prevent instability caused by high pressure. Then the Eulerian projection step is processed once per larger frame step to reduce computational cost. Figure 1 depicts the interesting small-scale details and each different movement of water caused by MPM in a non-gravity environment.

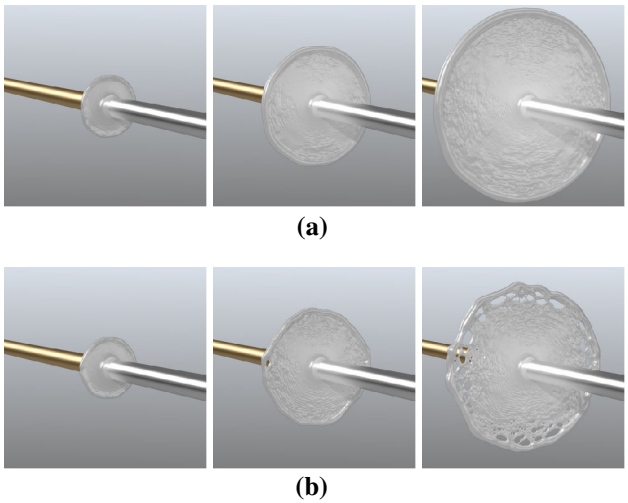


Fig. 1 Our method shows more details of the motion of a thin water film under strong surface tension than the standard FLIP method. **a** FLIP. **b** Our method

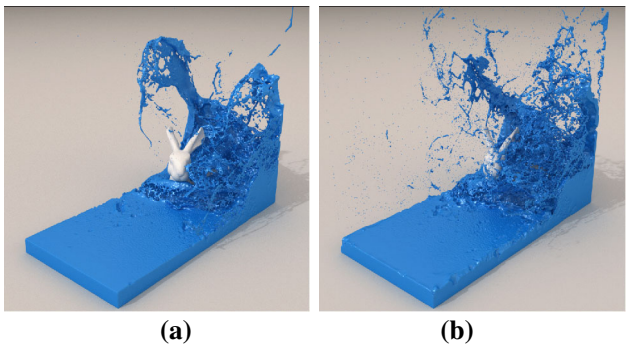


Fig. 2 Splash effects in a dam break scenario resulting from FLIP (*left*) are strengthened by our MPM sub-solver (*right*). **a** FLIP. **b** FLIP+MPM

Another approach is to calculate the MPM pressure to compensate for the density deviation. We can calculate pressure in MPM in the same way that it is calculated in SPH [20]. We use this pressure to control the strength of the MPM step in several examples.

The example in Fig. 2 shows the effect of the MPM step. By adjusting the explicit pressure and viscosity, we can control the strength of the splash patterns, which is good for graphics designers. The additional step in MPM took 5 % of the total computation time in this example.

4 Responsive surface tracking

4.1 Surface construction

Our construction method begins by finding the average position $\bar{\mathbf{x}}_i$ of the particles $\mathbf{x}_a$, which are close to the grid node $\mathbf{x}_i$, as follows:

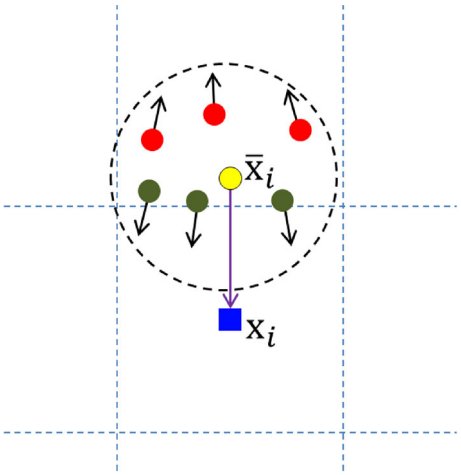


Fig. 3 We reduce the influence of particles facing in opposite directions by reducing their weights in the averaging process

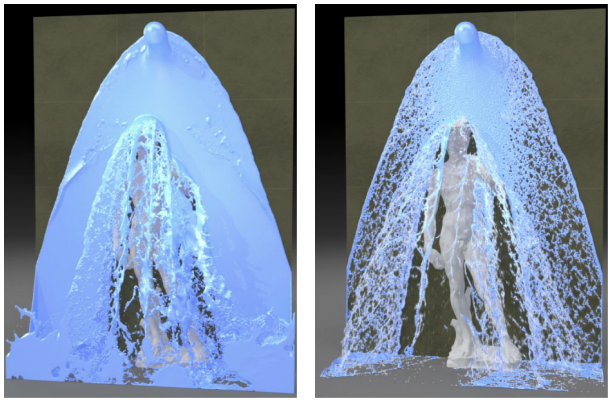


Fig. 4 Our surface tracking can generate (*left*) very smooth surfaces which satisfy temporal coherence. It can also generate rough surfaces (*right*) that more strictly follow the current spatial distribution of particle mass. **a** Smooth surface from Eq. (17). **b** Rough surface from Eq. (15)

$$\bar{\mathbf{x}}_i = \frac{\sum_a \bar{w}_{ia}\mathbf{x}_a}{\sum_a \bar{w}_{ia}}, \tag{14}$$

where $\bar{w}_{ia}$ is a weight function whose finite support is the predefined particle radius r. We use a spline kernel [19] for $\bar{w}_{ip}$. From this averaged position, we can obtain a signed distance for node i:

$$\bar{\phi}_i = |\bar{\mathbf{x}}_i - \mathbf{x}_i| - r, \tag{15}$$

where r is the predefined particle radius. So far, this method follows Zhu and Bridson [36].

To obtain very smooth surfaces that are temporally coherent, we use the values of ϕ_i and $\mathbf{n}_i$ which are carried by particles. The signed distance value at position $\mathbf{x}$ determined by a particle is

$$\tilde{\phi}_a(\mathbf{x}) = \min(\phi_a, -r) - \mathbf{n}_a \times (\mathbf{x}_a - \mathbf{x}), \tag{16}$$

and we can now find the average of $\tilde{\phi}_a(\mathbf{x})$ of adjacent particles:

$$\tilde{\phi}_i = \frac{\sum_a \tilde{w}_{ia}\tilde{\phi}_a}{\sum_a \tilde{w}_{ia}}, \tag{17}$$

with weights that are defined as

$$\tilde{w}_{ia} = \frac{1}{\epsilon^2 + \left(\frac{|\mathbf{x}_i - \mathbf{x}_a|}{h}\right)^2}\,\hat{w}_{ip}. \tag{18}$$

The parameter ϵ is defined by the users, and h is the distance of influence. We also introduce a normal correction weight to reduce the influence of implausible normals caused by large deformations or topological changes, as shown in Fig. 3.

$$\hat{w}_{ia} = \max\left(\frac{\mathbf{x}_i - \bar{\mathbf{x}}_i}{|\mathbf{x}_i - \bar{\mathbf{x}}_i|} \times \mathbf{n}_a, 0\right). \tag{19}$$

When $\mathbf{x}_i$ is far from $\bar{\mathbf{x}}_i$, these implausible normals would otherwise generate negative signed distances, leading to undesirable increases in volume.

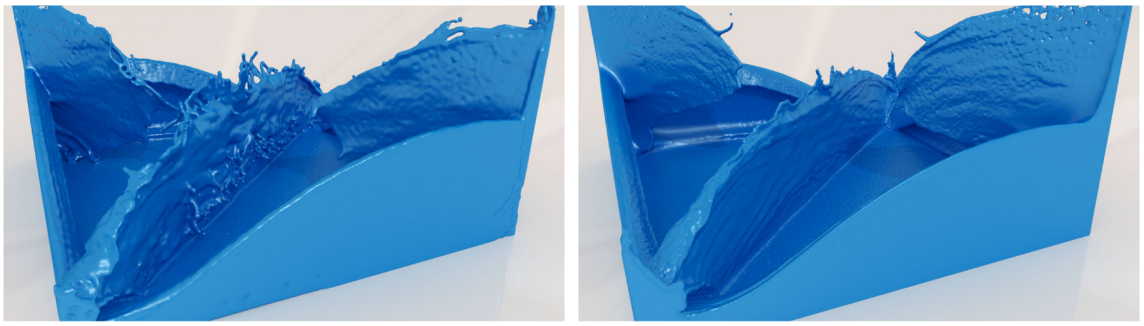
4.2 Velocity-based blending

Now, we have two level set surfaces: the very smooth $\tilde{\phi}$ and the rough $\bar{\phi}$ as shown in Fig. 4. We can blend them by simply taking a weighted average of their signed distances as follows:

$$\phi_i = (1-\alpha)\bar{\phi}_i + \alpha\tilde{\phi}_i. \tag{20}$$

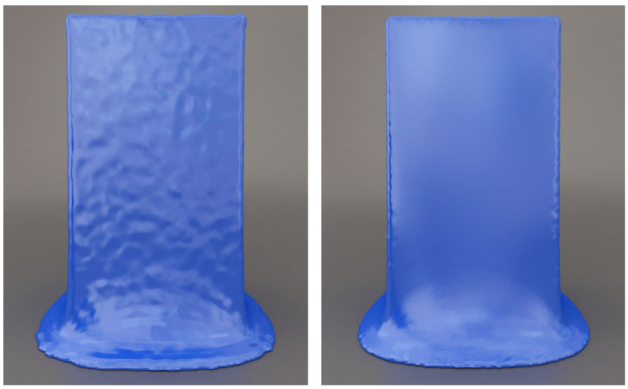
We determine the blending weight α from the magnitude of the velocity $|\mathbf{u}_i|$ and define the minimum and maximum blending velocities. If $|\mathbf{u}_i|$ is smaller than the minimum blending velocity, then α is set to 1 so as to generate smooth surfaces in slow regions. If $|\mathbf{u}_i|$ is larger than the maximum blending velocity, then α is set to 0 so as to generate rough surfaces in fast regions. If $|\mathbf{u}_i|$ is between the minimum and maximum blending velocities, then we make α proportional to $|\mathbf{u}_i|$ or $|\mathbf{u}_i|^2$, depending on various scenarios which include flow speed change. We use the fast sweeping method [35] to reinitialize ϕ_i and tri-linearly interpolate ϕ_a and $\mathbf{n}_a$ for the next time step.

We can also atomize particles based on the particle velocity and minimum and maximum atomization velocities, which control the probability that a particle will be atomized. The chance of atomization is increased as the magnitude of the particle velocity approaches the maximum atomization velocity. Once a particle has been atomized, it is omitted from the surface construction process until it has slowed down suf-



(a) **(b)**

Fig. 5 Comparison between different surface reconstruction approaches on the double dam break animation. Both results were simulated on a 256^3 grid and reconstructed as a surface from particles on a 512^3 grid. Our method (**b**) shows a sufficient smooth surface, which is similar to the surface reconstructed by the method used in (**a**). Example (**b**) was simulated about 5.2 times faster than Example (**a**). **a** Surface reconstruction with anisotropic kernel. **b** Surface reconstruction with our method



(a) **(b)**

Fig. 6 Comparison between the surface reconstruction using an anisotropic kernel and our method, which tracks surfaces with velocity-based blending. **a** Anisotropic kernel. **b** Velocity-based blending

ficiently due to collision with an object or reentry into the body of water.

4.3 Comparison

A surface reconstruction method with anisotropic kernels was proposed by Yu and Turk [33] to track smooth surfaces for particles based on fluid simulation. Their method results in smooth surfaces for fluid simulation, such as SPH, but it is not suitable for FLIP simulation, which has a problem with irregular placement of particles, because this method cannot properly determine a covariance tensor when particles are irregular or are insufficient in number. In that case, it is hard to determine shapes of kernels correctly for smooth surfaces.

Figure 5 depicts a comparison of a surface tracking method using anisotropic kernels with our method that tracks surfaces with velocity-based blending. The example in Fig. 5a shows the result of tracking a water surface with anisotropic kernels in FLIP water simulation. The FLIP simulation causes irregular placement of particles, and

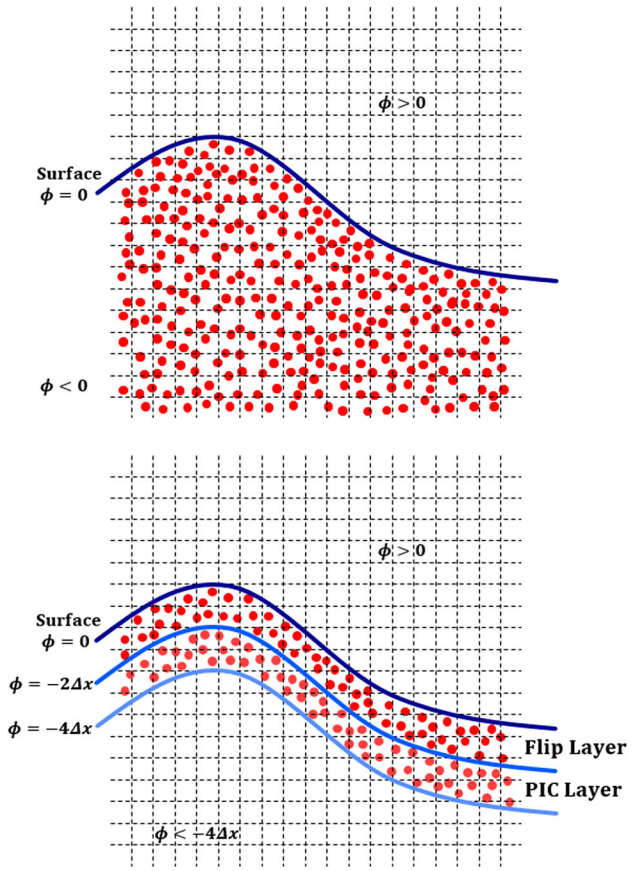
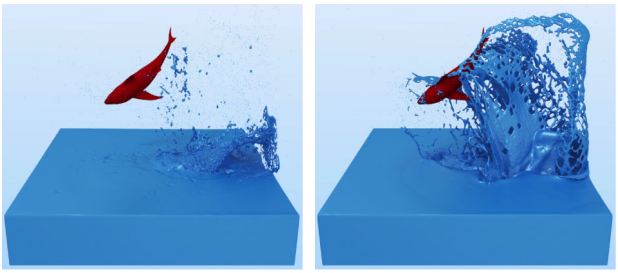


Fig. 7 When a body of water is large, we can place particles adaptively based on ϕ (*bottom*) instead of uniformly (*up*), to achieve better performance

although an anisotropic kernel is used for a smooth reconstruction surface, bumps or indentations can be generated on the surface.

In a reconstruction surface with an anisotropic kernel, the singular value decomposition (SVD) process is required. Unfortunately, the SVD consumes a lot of time while solving



(a) **(b)**

Fig. 8 Our adaptive particle placement improves the visual quality obtained from the FLIP method in visually important regions. Both of these tests used 3 M particles on a 256^3 grid. **a** FLIP. **b** FLIP with adaptive particles

the systems of linear equations, which could create a bottleneck in the simulation step.

It took about 11 h to get the result shown in Fig. 5a, while it took just 35 min to get the result shown in Fig. 5b. The reconstruction step with an anisotropic kernel accounted for 97 % of the whole simulation procedure on a workstation with two Intel Xeon E52697 2.7 GHz CPUs.

As shown in Fig. 5, our method can obtain a similar smooth surface reconstructed with an anisotropic kernel, and the method does not take as long for SVD.

Figure 6 depicts one frame of the result of Fig. 5 that shows the instant when the dam is released. When releasing the water source in slow flow like the result shown in Fig. 6, we would want to reconstruct a very smooth water surface.

In slow flow, to obtain a plausible result of a smooth surface, we had to adjust the parameters repeatedly, as in the method proposed by Yu and Turk [33], as shown in Fig. 6a. In contrast, our reconstruction method requires a simple adjustment for the scalar weight parameter for flow velocity and demonstrates a much smoother water surface, as shown in Fig. 6b.

5 Adaptive particle placement

Algorithm 2 summarizes our adaptive FLIP method. The main difference from the standard FLIP method is the particle reseeding step. This step maintains the particle distribution, as shown in Fig. 7, allowing us to concentrate particles on the important regions near interfaces.

Our method is designed to produce particle motions with the accuracy of the particle-in-cell (PIC) method [36] in the inner PIC layer, and with FLIP accuracy in the FLIP layer (see Fig. 7). Particles in the inner layer are advected along to their velocities. Simultaneously, values on the grid are also advected with the semi-Lagrangian. Then, particles are rasterized to the grid to unify the Lagrangian and Eulerian approaches. Therefore, the level set and the velocities on the grid are reconstructed with particles only in layers where particles exist. To prepare for the Eulerian projection, the boundary conditions are set by classifying the negative region of the level set into water cells. After the Eulerian projection step, we update the particle velocities in the FLIP layers so as to enforce incompressibility as with the FLIP method, but in the PIC region particle velocities are interpolated from grid velocity, allowing us a smooth transition from FLIP accuracy to PIC accuracy. To prevent instability, the reseeded particles are given a PIC velocity. See Fig. 8 for an example. The two examples in Fig. 8 used the same number of particles for simulation. An average of four particles are placed in a cell of the grid, and they can be added or deleted randomly to keep the predefined number of particles. However, these figures show a huge difference in visual quality. The reason is that most particles are placed densely near the water surface as shown in Fig. 8b.

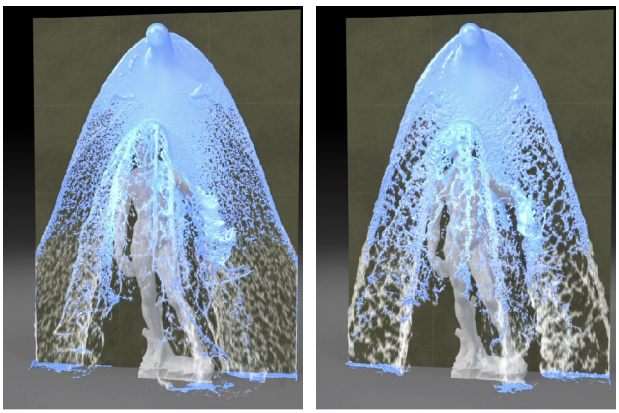
Algorithm 2 Adaptive FLIP method

1: Update solid objects
2: Lagrangian advection
3: Eulerian advection
4: Reconstruct level set and grid velocity partially
5: Sourcing
6: Redistance level set
7: Update particles
8: Eulerian projection
9: Reseed particles

6 Results and discussion

We present a weblink to a high-quality video for results shown in this paper (http://youtu.be/LaruxDVxHVg/).

In Fig. 9, water is poured on top of the wall which runs down it. The initial speed of the water is slow enough to produce a smooth, continuous surface of laminar flow, and thus the fluid surface construction is dominated by the smooth



(a) **(b)**

Fig. 9 The *left-hand* image shows how transitional tracking improves the FILP simulation of Fig. 4. The *right-hand* image shows how MPM provides even more details. **a** FLIP. **b** Improved simulation by MPM

surface generator. As it cascades down the wall under gravity, the water speeds up and a rougher region of transitional flow appears in which the surface is constructed by a combination of smooth and rough surface generators. Even lower, there is a turbulent region in which surface construction has been fully switched to the rough surface generator. At this stage, there are many atomized particles that no longer take part in surface construction. When the water and atomized particles are slowed down by striking an object or a mass of slow-moving water, the flow reverts to laminar or transitional type and regains the appropriate surface smoothness. Figure 9b shows how MPM enhances the FLIP simulation of Fig. 9a. More dynamic splash patterns are clearly observable, and atomized particles are successfully reformed into the volume of water.

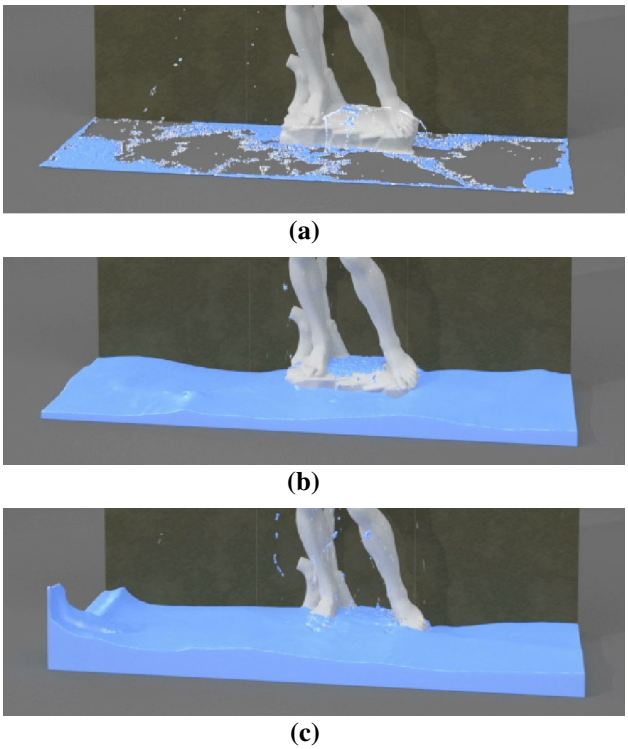


Fig. 10 Comparison of differences of the water volume when the velocity-based blending and MPM are used or these are not used in FLIP water example. **a** FLIP. **b** FLIP + velocity-based blending. **c** FLIP + velocity-based blending + MPM

In Fig. 10, there is a difference in the volume of water according to whether our responsive surface tracking method and MPM are used or not. In the FLIP method, the large time step could lead to dense particle placement, which is the reason for the volume dissipation shown in Fig. 10a, and interaction among particles is not considered, even when particles are intensively gathered to a corner or floor. Instead of using small time steps that cause delay in the simulation process, our reconstruction method solves this problem. Figure 10b shows a larger volume of water than Fig. 10a because our surface tracking method reconstructs the level set from information that particles carry. The level set is used to set the boundary conditions for the Eulerian projection, so that it helps to prevent high cohesions of particles. As stated previously, MPM performs interparticle operations without the need for neighbor search algorithms. This characteristic of MPM improves the accuracy of simulation for the volume conservation, as shown in Fig. 10c.

Figure 11 shows images of a jumping shark simulated by our adaptive FLIP method. A maximum of 100 M particles were used and surface construction and polygonization were performed at twice the simulation resolution. The simulation of 300 frames took 22 h on a workstation with two Intel Xeon E5-2687W 3.1 GHz CPUs, an NVidia GeForce 780 GPU, and Samsung DDR3 PC10600 memory. A flat fluid surface, smooth flow, and sharp splashing regions are all present in the same scene. More than 1000 M particles would have been required for this simulation if we had placed them uniformly. The adaptive approach is more advantageous on larger bodies of water, and the memory size required to store particles is

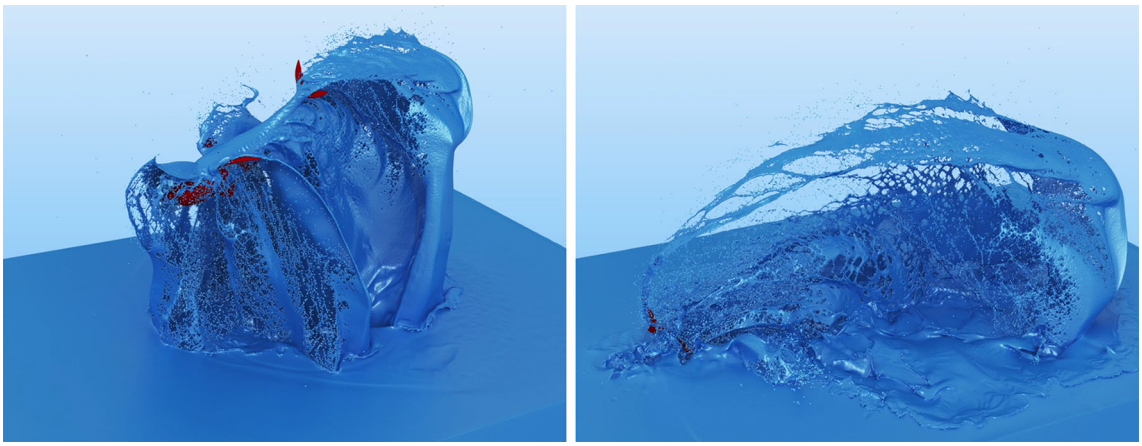


Fig. 11 Selected images of the jumping shark example simulated on a $1024 \times 512 \times 512$ grid

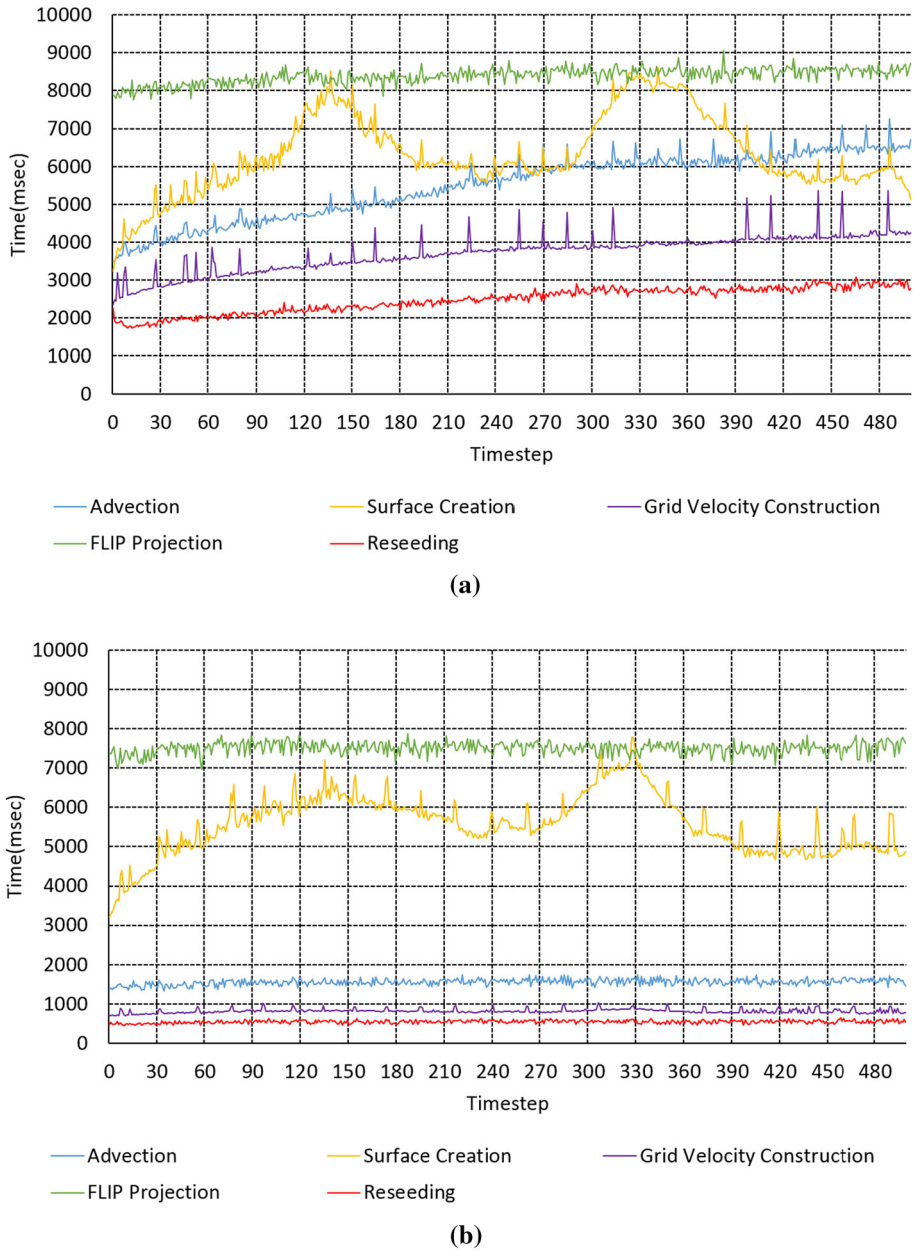


Fig. 12 These *graphs* show the durations for the different parts of the simulation steps. The example in Fig. 11 was repeated on a 512^3 grid with less particle number density: uniformly distributed 260 M particles (**a**), and adaptively placed 15M particles (**b**) were used maximally. **a** FLIP. **b** Our adaptive FLIP

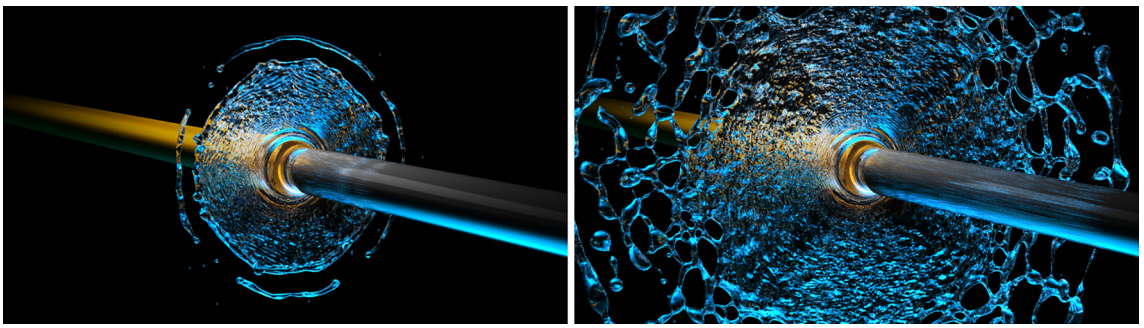


Fig. 13 Selected images from the example of Fig. 1b, but performed on a higher-resolution grid

proportional to the area of the water surface, not its volume. Figure 12 shows the performance increase more specifically. The particle advection step was 3.5 times faster, and the grid velocity construction from advected particles was 4.45 times faster, although a difference in visual quality was not perceived. We believe that larger memory bandwidth for better multi-threading would increase the performance gain.

The example in Fig. 13 shows a small-scale collision of two water jets that produce a spreading film. Surface tension was simulated as in [12] and the explicit step within MPM facilitated the modeling of incompressibility and viscosity, which could not have been done without modeling subgrid particle movements. Surface tension continuously exerts pressure forces on volume, making the subgrid dynamics calculation with MPM very significant. Simulation of 700 frames took 12 h on a $1024 \times 1024 \times 256$ resolution grid with 4 M particles on the same hardware in which Fig. 11 was performed.

The surface tracking method is not designed to achieve complete physical accuracy, but visual flexibility and controllability, and it does not conserve volume directly. This drawback is addressed within MPM, by the particle density, explicit step penalty pressure and volume control in the projection method.

7 Conclusions

We have presented a new surface tracking method that is responsive to changes of flow type. It can produce more natural-looking visual results with existing FLIP-based water simulators. It also supports an adaptive particle placement technique that greatly improves the performance and efficiency of the FLIP method by concentrating particles in visually interesting regions. We have also introduced a MPM subgrid solver to improve the modeling of the details of a fluid motion using the FLIP method.

There are many avenues we could take in extending this work. Pursuing more surface details by coupling to a mesh-based method [31,32] would be interesting. Extension to multiphase fluids [6,13], water spray [21], or fire [14] are also promising.

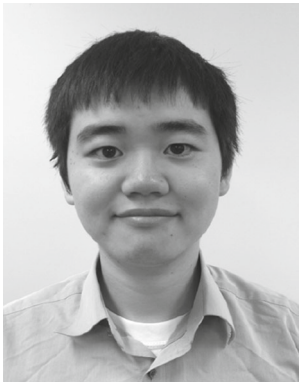
Acknowledgments This work was supported by the research program of Dongguk University, 2015, the National Research Foundation of Korea (NRF-2011-0023134), and the Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2012 (RST201100017).

References

1. Adams, B., Pauly, M., Keiser, R., Guibas, L.J.: Adaptively sampled particle fluids. ACM Trans. Graph. **26**(3), 48 (2007)
2. Ando, R., Tsuruno, R.: A particle-based method for preserving fluid sheets. In: SCA '11 Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 7–16 (2011). doi:10.1145/2019406.2019408
3. Ando, R., Thürey, N., Tsuruno, R.: Preserving fluid sheets with adaptively sampled anisotropic particles. IEEE Trans. Vis. Comput. Graph. **18**, 1202–1214 (2012). doi:10.1109/TVCG.2012.87
4. Ando, R., Thürey, N., Wojtan, C.: Highly adaptive liquid simulations on tetrahedral meshes. ACM Trans. Graph. **32**(4), 103 (2013)
5. Bhatacharya, H., Gao, Y., Bargteil, A.: A level-set method for skinning animated particle data. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 17–24. ACM (2011)
6. Boyd, L., Bridson, R.: MultiFLIP for energetic two-phase fluid simulation. ACM Trans. Graph. **31**(2), 16 (2012)
7. Brackbill, J.U., Ruppel, H.M.: FLIP: a method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. J. Comput. Phys. **65**(2), 314–343 (1986)
8. Enright, D., Fedkiw, R., Ferziger, J., Mitchell, I.: A hybrid particle level set method for improved interface capturing. J. Comput. Phys. **183**(1), 83–116 (2002)
9. Enright, D., Marschner, S., Fedkiw, R.: Animation and rendering of complex water surfaces. ACM Trans. Graph. **21**(3), 736–744 (2002)
10. Guo, Y.J., Nairn, J.A.: Three-dimensional dynamic fracture analysis using the material point method. Comput. Model. Eng. Sci. **16**(3), 141 (2006)
11. Hong, W., House, D.H., Keyser, J.: An adaptive sampling approach to incompressible particle-based fluid. In: EG UK Theory and Practice of Computer Graphics (2009)
12. Hong, J.M., Kim, C.H.: Discontinuous fluids. ACM Trans. Graph.**24**(3), 915–920 (2005)
13. Hong, J.M., Lee, H.Y., Yoon, J.C., Kim, C.H.: Bubbles alive. ACM Trans. Graph. **27**(3), 48 (2008)
14. Hong, J.M., Shinar, T., Fedkiw, R.: Wrinkled flames and cellular patterns. ACM Trans. Graph. **26**(3), 1–47 (2007)
15. Ianniello, S., Di Mascio, A.: A self-adaptive oriented particles Level-Set method for tracking interfaces. J. Comput. Phys. **229**(4), 1353–1380 (2010)
16. Jung, H.R., Kim, S.T., Noh, J., Hong, J.M.: A heterogeneous CPU–GPU parallel approach to a multigrid poisson solver for incompressible fluid simulation. Comput. Anim. Virtual Worlds **24**(3–4), 185–193 (2013)
17. Kim, B., Liu, Y., Llamas, I., Jiao, X., Rossignac, J.: Simulation of bubbles in foam with the volume control method. In: ACM Transactions on Graphics (TOG), vol. 26, p. 98. ACM (2007)
18. Losasso, F., Talton, J., Kwatra, N., Fedkiw, R.: Two-way coupledSPH and particle level set fluid simulation. IEEE Trans. Vis. Comput. Graph. **14**(4), 797–804 (2008)
19. Monaghan, J.J.: Smoothed particle hydrodynamics. Ann. Rev. Astron. Astrophys. **30**, 543–574 (1992)
20. 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, pp. 154–159. Eurographics Association (2003)
21. Nielsen, M.B., Østerby, O.: A two-continua approach to eulerian simulation of water spray. ACM Trans. Graph. **32**(4), 67 (2013)
22. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces. Springer, Berlin (2002)
23. Shabana, A.A.: Computational Continuum Mechanics. Cambridge University Press, Cambridge (2008)
24. Shen, C., O'Brien, J.F., Shewchuk, J.: Interpolating and approximating implicit surfaces from polygon soup. ACM Trans. Graph. **23**(3), 896–904 (2004)
25. Solenthaler, B., Schläfli, J., Pajarola, R.: A unified particle model for fluid-solid interactions. Comput. Anim. Virtual Worlds **18**(1), 69–82 (2007)

26. Stam, J.: Stable fluids. In: Proceedings. of SIGGRAPH 99, pp. 121–128 (1999)
27. Steffen, M., Kirby, R.M., Berzins, M.: Analysis and reduction of quadrature errors in the material point method (mpm). Int. J. Numer. Methods Eng. **76**(6), 922–948 (2008)
28. Stomakhin, A., Schroeder, C., Chai, L., Teran, J., Selle, A.: A material point method for snow simulation. ACM Trans. Graph. **32**(4), 102 (2013)
29. Sulsky, D., Schreyer, H., Peterson, K., Kwok, R., Coon, M.: Using the material-point method to model sea ice dynamics. J. Geophys. Res. Oceans (1978–2012) **112**(C2), C02S90 (2007). doi:10.1029/2005JC003329
30. Sulsky, D., Zhou, S.J., Schreyer, H.L.: Application of a particle-in-cell method to solid mechanics. Comput. Phys. Commun. **87**, 236–252 (1995)
31. Thürey, N., Wojtan, C., Gross, M., Turk, G.: A multiscale approach to mesh-based surface tension flows. ACM Trans. Graph. **29**(4), 1–48 (2010)
32. Wojtan, C., Thürey, N., Gross, M., Turk, G.: Physics-inspired topology changes for thin fluid features. ACM Trans. Graph. **29**(4), 50 (2010)
33. Yu, J., Turk, G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. ACM Trans. Graph. **32**(1), 5 (2013)
34. Zhang, D.Z., Zou, Q., VanderHeyden, W.B., Ma, X.: Material point method applied to multiphase flows. J. Comput. Phys. **227**(6), 3159–3173 (2008)
35. Zhao, H.: A fast sweeping method for eikonal equations. Math. Comput. **74**(250), 603–627 (2005)
36. Zhu, Y., Bridson, R.: Animating sand as a fluid. ACM Trans. Graph. **24**(3), 965–972 (2005)

Jae-Gwang Lim received his B.S. degree in 2012 and the M.S. degree in 2014 in computer engineering from the Dongguk University, South Korea. He is a researcher at the Visual Simulation Lab and a Ph.D. student at the Dongguk University. His main research interests are GPU parallel programming, fluid, granular material, and rigid body simulation.



Bong-Jun Kim received his B.S. degree in 2012 in computer engineering from the Dongguk University, South Korea. He is a researcher at the Visual Simulation Lab and an M.A. student at the Dongguk University. His main research interests are interactive computer graphics, GPU parallel programming, fluid, deformable, and rigid body simulation.

Jeong-Mo Hong is an associate professor at the Computer Science and Engineering Department. He received his B.S. degree in 2000 and M.S. degree in 2002 in mechanical engineering from the Korea Advanced Institute of Science and Technology (KAIST), South Korea. After obtaining his Ph.D. degree in computer science from the Korea University in 2005, he developed several simulation techniques for visual effects as a research fellow at the Computer Science Department of Stanford University.