EUROGRAPHICS 2018 / D. Gutierrez and A. Sheffer
(Guest Editors)

Volume 37 (2018), Number 2

Extended Narrow Band FLIP for Liquid Simulations

T. Sato[1], C. Wojtan[2], N. Thuerey[3], T. Igarashi[1], and R. Ando[4]

[1]The University of Tokyo [2]IST Austria [3]Technical University of Munich [4]National Institute of Informatics

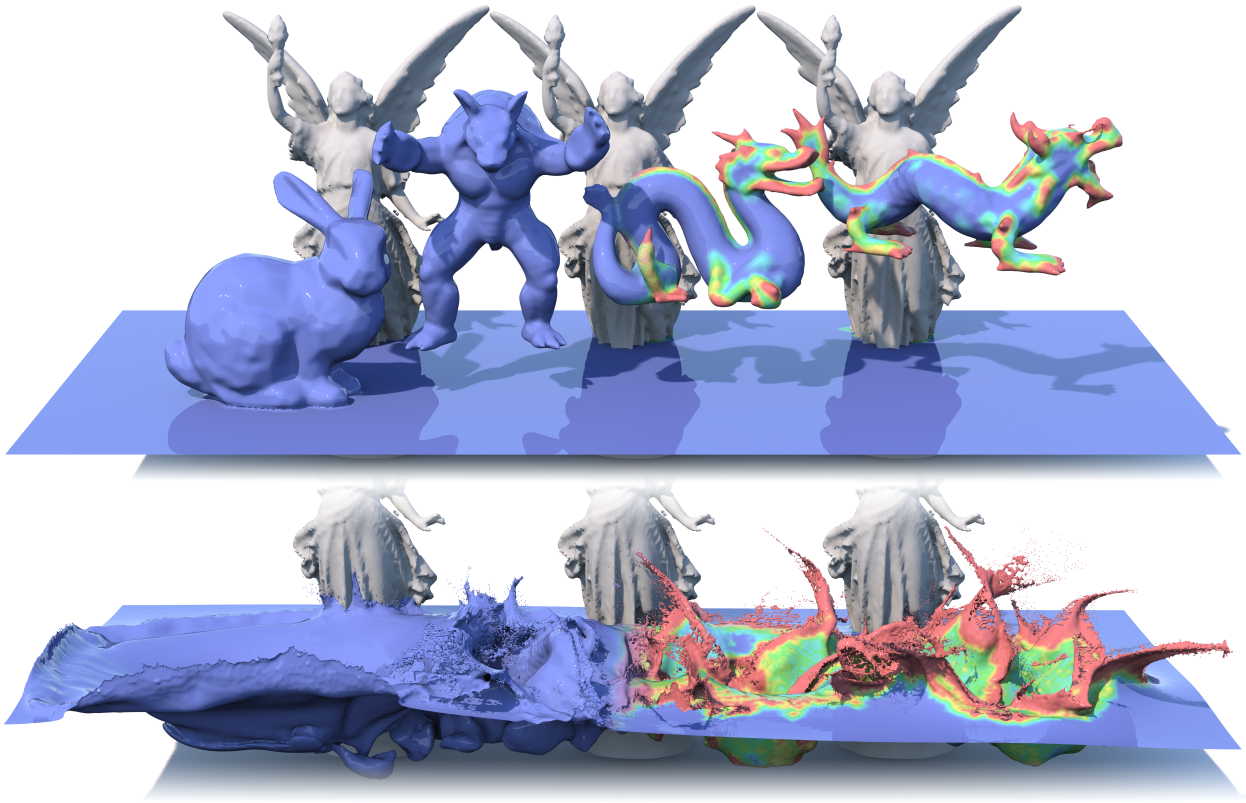

Figure 1: *Right: Visualized heat map of our extended narrow band FLIP. The red color indicates the region where the NB-FLIP method is used to track small scale details, while the blue region uses the Eulerian level set method to represent smooth surfaces. Left: Liquid surface only. Notice that no seams between the two surface representations are visible.* $384 \times 192 \times 192$ *resolution,* 17 *seconds per time step and* 39.0 *seconds per video frame.*

Abstract

The Fluid Implicit Particle method (FLIP) reduces numerical dissipation by combining particles with grids. To improve performance, the subsequent narrow band FLIP method (NB-FLIP) uses a FLIP-based fluid simulation only near the liquid surface and a traditional grid-based fluid simulation away from the surface. This spatially-limited FLIP simulation significantly reduces the number of particles and alleviates a computational bottleneck. In this paper, we extend the NB-FLIP idea even further, by allowing a simulation to transition between a FLIP-like fluid simulation and a grid-based simulation in arbitrary locations, not just near the surface. This approach leads to even more savings in memory and computation, because we can concentrate the particles only in areas where they are needed. More importantly, this new method allows us to seamlessly transition to smooth implicit surface geometry wherever the particle-based simulation is unnecessary. Consequently, our method leads to a practical algorithm for avoiding the noisy surface artifacts associated with particle-based liquid simulations, while simultaneously maintaining the benefits of a FLIP simulation in regions of dynamic motion.

CCS Concepts

•***Computing methodologies*** → ***Physical simulation;***

1. Introduction

The surface tracking step of a liquid simulation strongly influences its visual appearance. The level set method [OS88] is an established tool to represent dynamic surfaces, and was demonstrated to be particularly well suited for smooth surfaces. On the other hand, particle-based methods like the Fluid Implicit Particle method (FLIP) [ZB05] and SPH [MCG03] can capture small-scale features like droplets and splashes very well. All of these methods also handle topology changes easily, as they rely on implicit representations of the surface. While these methods are powerful and can be used in wide range of situations, they exhibit certain inherent limitations: Simulations based on the level set method tend to aggressively smooth and delete small-scale surface features, while particle-based simulations can lead to high-frequency noise at the surface due to irregular particle distributions.

In contrast to the commonly used FLIP algorithm, the Narrow Band FLIP (NB-FLIP) method [FAW*16] only samples particles within a *narrow band* of the liquid surface. This approach drastically reduces the number of particles necessary for simulation, while effectively retaining the non-dissipative surface tracking behavior of FLIP. Despite being more efficient, this NB-FLIP approach inherits all of the visual artifacts of a standard FLIP simulation. In this work, we present a method that aims to leverage the benefits of both FLIP-based and level-set-based fluid surface trackers. We do this by selectively and seamlessly switching between a FLIP-based surface tracker and a level-set-based surface tracker. Our method can be seen as a generalization of the Narrow Band FLIP approach, in that it can transition from a standard grid-based fluid simulation to a FLIP-based fluid simulation *anywhere*, and not just inside of the bulk volume of the fluid. In particular, our method can seamlessly transition between detailed Lagrangian and Eulerian representations right at the liquid surface itself. Our contributions are as follows:

- We devise a new *transition function* that aims to characterize the state of a fluid surface based on geometry and velocity information.
- We propose a surface tracking method that can represent both detailed liquid splashes and smooth calm surfaces at the same time.
- Our method can reduce the number of particles compared to the NB-FLIP method, leading to extra memory savings and faster runtimes.

We combine these contributions to gain a significant improvement in the state of the art of liquid surface tracking.

2. Related Work

Level Sets: Our method employs the level set method, which was first introduced by Osher and Sethian [OS88]. This method represents a surface as the zero-contour of a higher-dimensional implicit function called the level set function. The main advantages are that this method can naturally deal with topology changes such as merging and separation of the surfaces. This level set function is advected based on a velocity field, which represents a movement of the surface. Unfortunately, this level set advection suffers from numerical dissipation [Bri15]; this dissipation inadvertently filters out high-frequency surface details like sharp edges, and causes the surface to noticeably lose volume. To circumvent this issue, [FF01] introduced the particle level set method. Here, particles are placed near the surface of liquid interior and advected in a Lagrangian manner. When a particle leaves the main volume, it is used to correct the level set of the liquid region, or converted to a ballistic particle. In [EMF02] particles are placed on both sides of the surface to reduce surface tracking errors near the surface. Losasso et al. [LTKF08] proposed to couple SPH particles with an Eulerian simulator near the liquid surfaces. Beyond level sets, triangle meshes were also used for tracking the surface of simulated liquids [WMFB11]. While these methods can yield small scale details, this detail comes at the expense of additional computations for maintaining an explicitly discretized surface.

FLIP: The Fluid Implicit Particle (FLIP) method was introduced to the graphics community by Zhu and Bridson [ZB05]. The method reduces numerical dissipation by using particles to calculate the advection term of the Navier-Stokes equations, and using grid-based techniques to calculate the remaining terms. Various improvements have been applied to this FLIP method. Batty et al. proposed the approach to coupling with solids that have an irregular boundary [BBB07a]. Several researchers correct particle positions to reduce artifacts [ATT12, UBH14]. Jiang et al. [JSS*15] introduced the Affine PIC method to successfully reduce the loss of information when transferring between particles and grids. Fu et al. [FGG*17] further improved the accuracy in terms of information transfer via the use of extended polynomial bases. In spite of these improvements, each of these methods suffers from visualization artifacts tied to their use of particles for surface reconstruction.

Narrow Band FLIP: Our work is based on the NB-FLIP approach proposed by Ferstl et al. [FAW*16]. This method uses particles only within a narrow band of the liquid surface, based on the assumption that interior particles only make a small contribution to the visual appearance of the fluid. NB-FLIP samples the particles in a way that is indistinguishable from FLIP near the surface, but uses a grid-based fluid simulation elsewhere. Our method can be seen as a generalization of this NB-FLIP approach; instead of transitioning between simulation methodologies only near the surface, our approach allows the simulation to transition between methods *anywhere* within the flow. This flexibility allows for novel noise filtering and computational speedups.

Surface Reconstruction: Previous researchers have attempted to solve these problems with surface reconstruction. Bhatacharya et al. [BGB11] rephrase the issue as constrained optimization and solve it using a level set approach. Yu and Turk reconstruct a surface using a stretched, anisotropic smoothing kernel [YT13]. These approaches tend to produce a smoother surface from a collection of particles. Yu et al. achieve more accurate surface tracking by combining a conventional implicit method and new explicit mesh-based method [YWTY12]. Instead of directly smoothing out noise from a surface extracted from particles, our approach leverages the inherent smoothness of level set surfaces to help solve this problem. Wherever necessary, our method smoothly blends between a detailed-but-noisy particle surface and an ideally smooth level set

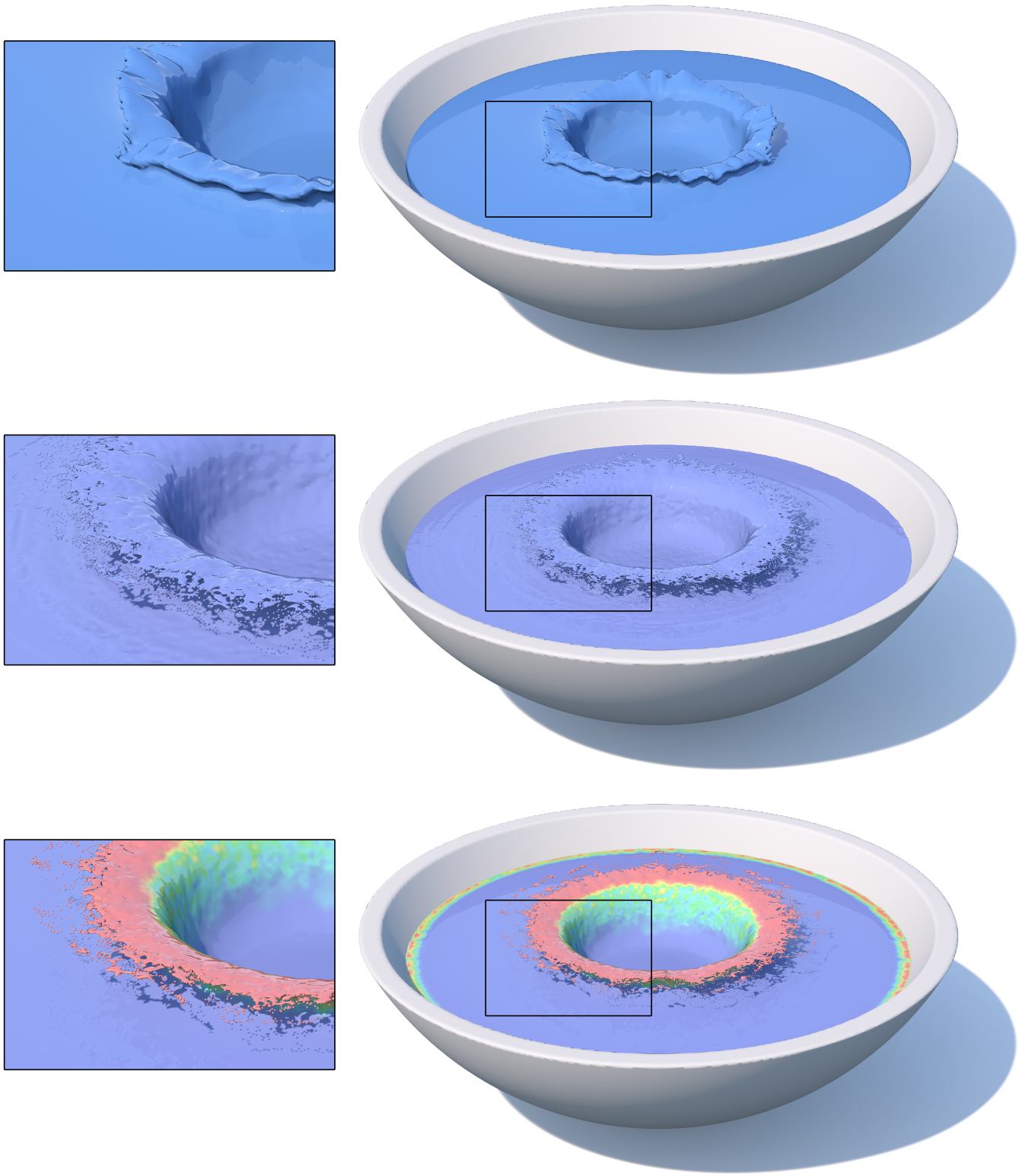

Figure 2: *Drop falling into a pool creating crown splashes. Top: Level set method. Middle: The NB-FLIP method. Bottom: Our method. Notice that levelset surface provides smooth surfaces for calm areas while it tends to smear sharp details. The NB-FLIP method on the other hand can express complex splashes but the calm surfaces tend to be noisy. Our method successfully preserves both desirable representations for both calm and dynamic areas without visual seams.* $256 \times 128 \times 256$ *resolution,* 6.1 *seconds per time step and* 12.9 *seconds per video frame.*

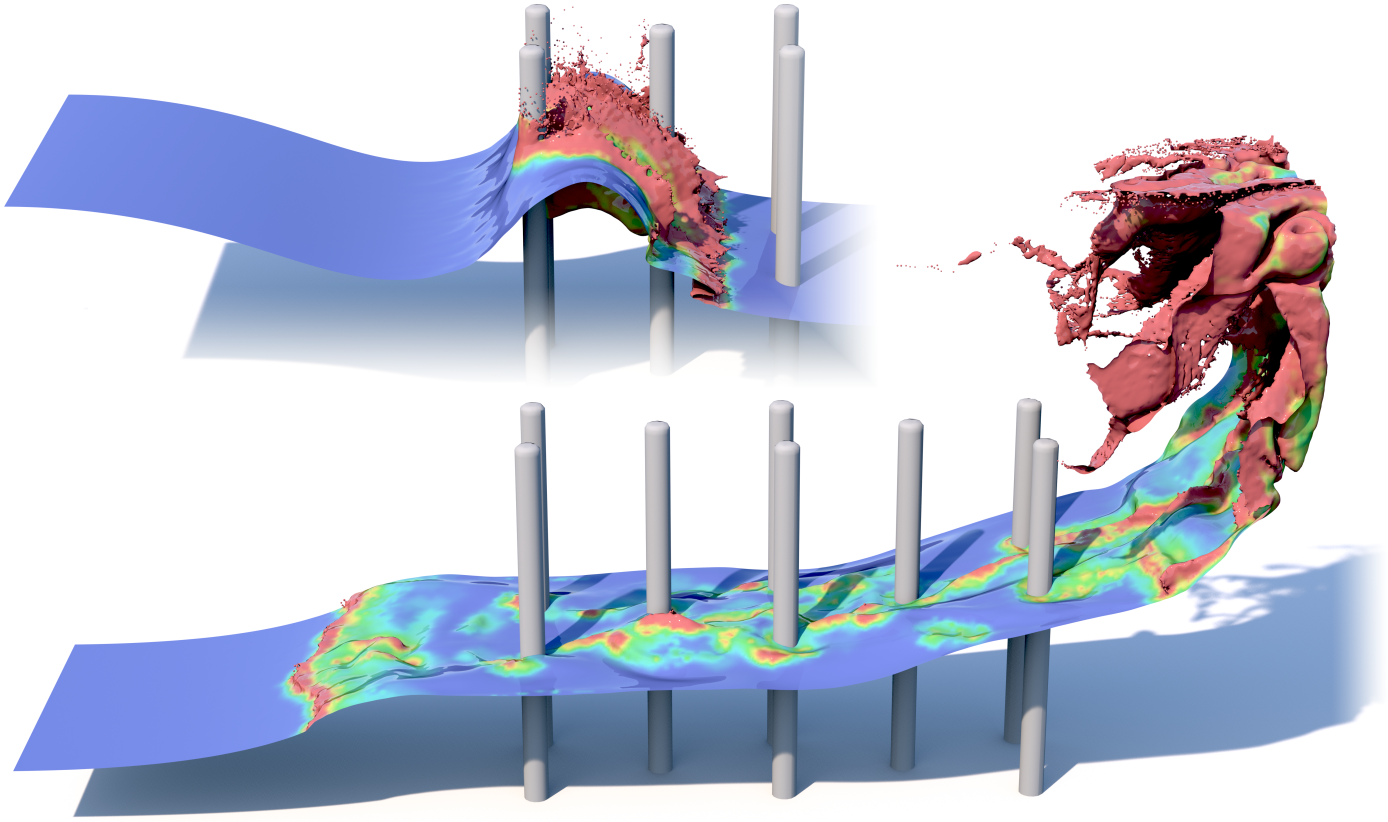

Figure 3: *Breaking dam creating splahes:* $256 \times 128 \times 64$ *resolution,* 3.2 *seconds per time step and* 7.0 *seconds per video frame. Notice that the NB-FLIP particles are generated only around the cylinders and continue to hit the wall resulting in vibrant splahes.*

surface. This approach allows us to strategically simulate both detailed splashes and sprays as well as calm, noise-free waves.

Physics-based Surface De-noising: Other recent work has focused on a similar problem of retaining surface details in regions where they are physically plausible, and removing high-frequency surface noise whenever it violates physics. Bojsen-Hansen and Wojtan [BHW13] introduced a surface energy term based on the free surface boundary conditions of the Navier-Stokes equations, and removed the surface noise by minimizing this energy. Goldade et al. [GBW16] proposed a more stable filter to selectively remove the noise from a high-resolution level set. The idea in both of these approaches is to advect an extremely high resolution surface (like a level set or explicit mesh) through the fluid, and then smooth out the surface details by numerically integrating a diffusion-like partial differential equation, where the diffusion rate depends on spatially-dependent physical properties like the magnitude of the pressure gradient. To apply these methods to a particle-based method like FLIP, one would need to advect a highly-detailed set of particles throughout the flow, and use similar smoothing rules to adjust particle data until it matched the low-resolution physics simulation. In contrast to these approaches, we aim to only introduce high-resolution particle samples exactly where they are needed, and nowhere else. This strategy significantly reduces computational burden compared to previous approaches, but it also introduces a significantly different problem of deciding where to introduce particles *before* the interesting flow features arise.

3. Method Overview

Our algorithm inherits many components from the NB-FLIP method, and as such, we will give a brief overview in the next section, before detailing our method.

3.1. Narrow Band FLIP Revisited

In addition to FLIP particles, NB-FLIP also tracks an Eulerian level set ϕ to identify the *narrow band* near the surface. The FLIP particles and the level set are both advected using the corresponding Lagrangian and Eulerian advection schemes [SFK*08]. Next, the level set surface is eroded, e.g., by a half grid cell size. Lastly, the final surface is defined as the concatenation of both surface representations, i.e. the level set of particles surface ϕ_p and grid-based ϕ. Effectively, this step corresponds to computing the union of both implicit surfaces on the grid. In this way, the NB-FLIP method efficiently retains surface details of particles without the need to maintain a dense particle sampling deep inside the liquid. The velocity of the fluid is defined in a similar way. In the NB-FLIP method, particles are sampled within a specified depth from the surfaces e.g.,

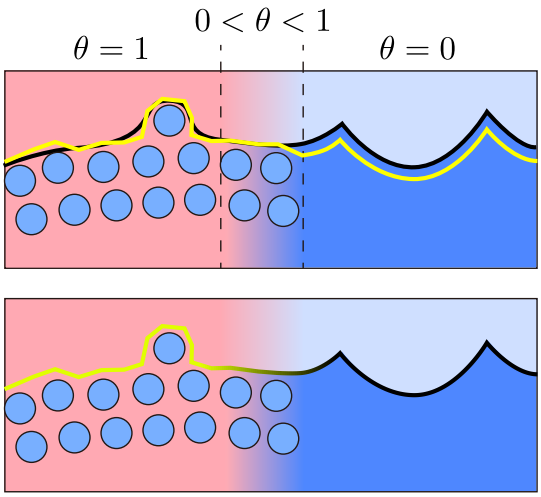

Figure 4: *Blending surfaces overview: Particle-based surfaces of the NB-FLIP method shown on the left (yellow line) and the Eulerian level set surfaces shown on the right (black line) are seamlessly stitched together via a linear combination.*

three cells wide. The grid-based velocities are advected on the grid, and the final velocity field is overwritten by the velocity of FLIP particles near the surface (typically within two cells distance).

3.2. Our Method

At the heart of our method is the algorithm to seamlessly stitch together surfaces of the level set method and the NB-FLIP method without any spatial or temporal artifacts. Before we discuss this algorithm in detail, we will introduce several amendments to the NB-FLIP method. First, we migrate all the FLIP [ZB05] components to the Affine PIC (APIC) method [JSS*15]. We prefer the APIC method because it better preserves shear and rotational motions during the grid-particle transfers.

3.2.1. Overwriting Grid Velocity

Next, when overwriting the grid velocity, we use a slightly different strategy inspired by Jiang et al. [JSS*15]. First, we splat both the momentum p_p and the mass m_p of FLIP particles onto the center of grid faces. We also compute the fraction ρ of fluid on faces using the method of Batty et al. [BBB07b]. The fraction ρ ranges from zero to one near the liquid surfaces, and exactly one for all the interior faces. Once all the variables are ready, we compute the mass m_g and the momentum p_g on grid faces as follows:

$$m_g = \max(0, \rho V - m_p) + m_p, \quad (1)$$

$$p_g = m_g u_g + p_p, \quad (2)$$

where $V = \Delta x^3$ is the volume of a cell, and m_g and u_g are the mass and velocity stored on the grid. Finally, the grid velocity before the pressure solve is given by $u_g^* = p_g / m_g$. More specifically, m_p is defined on a grid face, and is computed through summation: $m_p = \sum_q N_{pq} m_q$ where N_{pq} and m_q denote kernel function and the mass of a particle q. p_p is also defined on a face, and is calculated similarly. Both m_p and p_p are computed exactly the same way as the work of Jiang et al. [JSS*15] Notice that in the absence of particles, the grid-based velocity will not be changed. We will demonstrate later on that this strategy allows us to smoothly blend the grid-based velocity and the FLIP-based velocity without increasing kinetic energy.

3.2.2. Improved Position Correction

Similar to several previous works [ATT12, FAW*16], we also use *position correction* to adjust the position of FLIP particles near the surface in order to gain a more uniform particle distribution as follows: when a pair of particles are closer than the sum of their radii, we compute the displacement: $d_{ij} = \frac{m_j}{m_i+m_j}(r_i + r_j - ||\boldsymbol{p}_i - \boldsymbol{p}_j||)$ where r_i,r_j denote particle radii, $\boldsymbol{p}_i$,$\boldsymbol{p}_j$ particle position vector and m_i,m_j particle mass where we use constant $m_i = 1, m_j = 1$ since we only use constant sized particles. We sum up all the displacements to $\Delta \boldsymbol{p}_i$ as $\Delta \boldsymbol{p}_i = \sum_j d_{ij}(\boldsymbol{p}_i - \boldsymbol{p}_j)/||\boldsymbol{p}_i - \boldsymbol{p}_j||$. Finally, we apply the displacements as $\boldsymbol{p}^{\text{new}} \leftarrow \boldsymbol{p} + \Delta \boldsymbol{p}$. However, because FLIP particles can be seen as spatial samples of the velocity field, repositioning these particles will tend to implicitly alter the velocity field that they represent. This issue can be alleviated in the followig way; before applying the displacement to the particle, we also change the velocity of the particle as: $\boldsymbol{u}^{\text{new}} \leftarrow \boldsymbol{u} + J_{\boldsymbol{u}} \Delta \boldsymbol{p}$ where $J_{\boldsymbol{u}}$ denotes the Jacobian of the grid velocity. When using this improved position correction in our simulations we observed a slightly better kinetic energy preservation.

3.2.3. Ballistic Particles

The level set surface representation and the particle-based surface representation often deviate from one another due to the mismatch of effective resolutions as also pointed out by Enright et al. [EMF02], and the different advection schemes (Eulerian advection and the Lagrangian advection) employed. In accordance with the work of Guendelman et al. [GSLF05], we prefer the use of escaped particles as ballistic particles to represent "splashing" liquids. The escaped particles are determined by the sign of the level set value interpolated from grids. More specifically, we note that we rely on the same simulation grids to compute the level set value from the particles. This can cause a simulation to overlook sparse particles from the level set as pointed out by Boyd and Bridson [BB12]. We found it convenient to simply treat such particles as ballistic particles.

Ballistic particles are excluded from the simulation until they re-enter the main volume once again, and they instead follow simple parabolic paths dictated by gravity. This modification is in line with previous work [ATW13], and allows us to efficiently omit those individual particles from some of the more expensive fluid simulation steps. However, very splashy scenarios can easily fill the entire scene with such spray particles, which we found undesirable. We mitigate this issue by giving those ballistic particles a finite *life time*. The life time is decreased steadily over time, and the particles are removed when it reaches zero. As an additional visual effect, we reduce the radius of ballistic particles proportional to their life time during rendering. For a reference, the effect without these extensions can be seen in the supplemental material.

3.2.4. Surface Reconstruction

In the simulation, we use the method presented by Ando et al. [ATW13] for reconstructing surfaces of particles at high accuracy. For the final surface generation, which is performed on twice higher grid resolutions, we employ the method by Zhu and Bridson [ZB05] for faster performance.

3.2.5. Blending Liquid Surfaces

We define a spatial *transition function* θ that controls the transition between the particle-based surface for $\theta = 1$, and grid-based level set surfaces for $\theta = 0$. During a time step of the simulation, we advect both particles and the level set field as in the NB-FLIP method. After the advection we remove particles where θ is exactly zero and do not seed any particles in this area, as it is only represented by the Eulerian level set. Finally, we compute our blended surface as the linear combination of the two level set values from the NB-FLIP method and the grid-based level set method. This process is illustrated in Figure 4. One step of our algorithm is illustrated in Algorithm 1.

Algorithm 1: Extended NB-FLIP Steps

1. Advect particles via RK2 (Zhu and Bridson [ZB05])
2. Correct particle positions (Section 3.2.2)
3. Compute transition function on grid $\Rightarrow \theta$ (Algorithm 2)
4. Advect $\boldsymbol{u}$ (MacCormack) and ϕ (semi-Lagrangian) on grid $\Rightarrow$ $\boldsymbol{u}_g$ and ϕ_g
5. Map particles to grid $\Rightarrow \boldsymbol{u}_p$ and m_g (APIC [JSS*15]) and compute level set ϕ_p on grid from particles using the method of Ando et al. [ATW13]
6. Update level set and velocity on grid
   $\phi_{nb} \leftarrow \min(\phi_g + h, \phi_p)$ // Erosion and union
   $\phi \leftarrow \theta\phi_{nb} + (1-\theta)\phi_g$ // Linear blend
   Re-initialize ϕ (Enright et al. [EMF02])
   $\boldsymbol{u}^* \leftarrow$ overwrite($\boldsymbol{u}_g$, m_g, $\boldsymbol{u}_p$, ϕ) // Section 3.2.1
7. Add external forces $\boldsymbol{f}$ and project $\boldsymbol{u}^* \Rightarrow \boldsymbol{u}$ (Bridson [Bri15])
8. Extrapolate velocity field (Bridson [Bri15])
9. Update particle velocities (APIC [JSS*15])
10. Generate high-resolution surfaces from particles (Zhu and Bridson [ZB05])

3.3. Transition Function

The central goal of our transition function is to predict where visually interesting features are likely to emerge, and to represent these regions with the selected NB-FLIP method. Conversely, we aim to represent calm areas with the Eulerian level set method.

3.3.1. Heat Function

To aid us in this goal, we design a spatial *heat function* $\gamma(\boldsymbol{x})$ that takes into account both the motion of the fluid as well as the surface geometry as follows:

$$\boldsymbol{q}(\boldsymbol{x}) = \boldsymbol{u}(\boldsymbol{x}) - \int_\Omega G(\sigma, (\boldsymbol{x}' - \boldsymbol{x})/\Delta x)\boldsymbol{u}(\boldsymbol{x}')dV' \quad (3)$$

$$c(\boldsymbol{x}) = \phi(\boldsymbol{x}) - \int_\Omega G(\sigma, (\boldsymbol{x}' - \boldsymbol{x})/\Delta x)\phi(\boldsymbol{x}')dV' \quad (4)$$

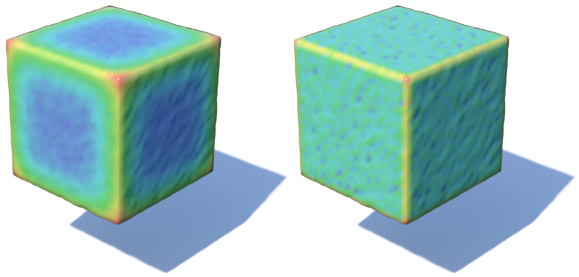

Figure 5: *Visualized curvature measurement applied to a particle-based surface of a cube: Our method on the left with Eq. (4) yields a smoothed curvature approximation, while the curvature values computed with a six-cell stencil on the right strongly respond to the noise on the surface.*

$$\gamma(\boldsymbol{x}) = \max(\max(0, A_u \min(1, ||\boldsymbol{q}(\boldsymbol{x})||) - T_u), \quad (5)$$

$$\max(0, A_g |c(\boldsymbol{x})|/\Delta x - T_g)), \quad (6)$$

where $G(\sigma, \boldsymbol{r})$, A_u, A_g, T_u and T_g denote the Gaussian function, amplification parameters for velocity and geometry, and threshold parameters for velocity and geometry, respectively. With this heat function our aim is to encode the saliency of fluid. E.g., dynamic motions will yield high values. If an insufficient motion is detected, the value is cut off at the given threshold values. In all of our examples we used $\sigma = 1$, $A_u = A_g = 5$, $T_u = 0.2$ and $T_g = 1.0$.

Eq. (3) can be seen as a high-pass filter function acting on the velocity at a location of interest. Notice that $\boldsymbol{q}(\boldsymbol{x})$ yields larger values near large velocity gradients, and lower values for calm motions. We particularly note that $\boldsymbol{q}(\boldsymbol{x})$ does not respond to translational movements; e.g., free falling liquids.

Eq. (4) can be regarded as a pseudo curvature function inspired by *blob detection* [Lin98], the work of Sussman et al. [SO09] and Thuerey et al. [TWGT10]. This function provides us with a smoothed curvature-like measurement and is robust against noise. We prefer the use of this function over the standard differention-based curvature since the FLIP surfaces are typically noisy. Figure 5 shows a comparison using our method and the six-stenciled curvature evaluation of the level set. We also point out that our method can anticipate the development of interesting dynamics *before* they emerge; this is achieved with the help of the Gaussian convolutions, which have the ability to detect not only local but also longer-range changes, e.g., future liquid collisions. We numerically compute the convolutions on neighborhood cells.

3.3.2. Propagating Heat

We evaluate the heat function only within a one cell neighborhood below the liquid surface, instead of in the full narrow band area. This provides us with an efficient curvature estimate near the surface, but it skips the evaluation in the rest of the narrow band area, which we later need for seeding FLIP particles. We propagate the heat from surfaces towards interior using a differential equation: $\frac{\partial \gamma}{\partial \tau} = ||\nabla \gamma||$, where τ denotes a fictional time. We solve this equation with local maxima fixed as boundary conditions. We use a first-

order upwind differentiation to discretize the gradient operator and the first-order Euler method to discretize fictional time, where we set $\Delta\tau = 0.75\Delta x$. We perform the temporal integration four times per step, but we note that this number is subject to the thickness of the narrow band. Note that unlike extrapolation, our method decays the values at the surface when they are extended into the interior of the liquid.

3.3.3. Transferring Heat to Particles

Once the heat is propagated as $\gamma(\boldsymbol{x}) \rightarrow \gamma^*(\boldsymbol{x})$, we seed FLIP particles in the narrow band area where $\gamma^*(\boldsymbol{x}) > 0$ and assign the interpolated on-grid heat value to the particles as $\gamma_p = \gamma^*(\boldsymbol{x})$. If a seeded particle lies within $0.75\Delta x$ away from the liquid surface (we use the interpolated level set value to check this), we move its position toward the level set normal to exactly touch the surface. If any FLIP particle already exists in the area, we assign the max value of $\gamma^*(\boldsymbol{x})$ and the heat of the particle γ_p. The heat is advected with the FLIP particles, and cooled down over time as: $\gamma_p(t+\Delta t) = \gamma_p(t) - \alpha\Delta t$ where we set $\alpha = 10$. As exception, we do not cool down ballistic particles. We remove FLIP particles when either they leave the narrow band, the particle count per cell exceeds a maximal number, or their heat cools down to zero.

3.3.4. Rasterizing Heat to Grids

To compute the transition function, we first rasterize the heat of the particles onto the grid. For each cell, we loop over all the particles within the cell and pick the one with the highest heat. If the heat exceeds one, we clamp to one. Finally, we use this rasterized heat as our transition function. The whole procedure of our transition function computation is outlined in Algorithm 2.

Algorithm 2: Transition Function Steps

1. Compute heat on grid $\gamma(\boldsymbol{x})$ using Eq. (6)
2. Propagate heat on grid $\gamma(\boldsymbol{x}) \rightarrow \gamma^*(\boldsymbol{x})$ // Section 3.3.2
3. Seed FLIP particles where $\gamma^*(\boldsymbol{x}) > 0$ // Section 3.3.3
4. Cool down particles: $\gamma_p(t+\Delta t) = \gamma_p(t) - \alpha\Delta t$
5. Update heat on FLIP particles: $\gamma_p \leftarrow \max(\gamma_p, \gamma^*(\boldsymbol{x}))$
6. Rasterize heat from particles to grid

4. Results

All the examples (Figure 1, Figure 2 and Figure 3) ran on a Linux machine with Intel Core i9-7920X 2.90GHz. We employed a preconditioned conjugate gradient method [Bri15] for the pressure solve using a target L_∞ norm of 10^{-3} for the relative residual.

The free-falling drop examples of Figure 2 highlight our method's tendency to preserve both dynamic and calm surfaces at the same time. In contrast, the level set method generates smooth surfaces for calm areas but tends to smooth out sharp features at the edge of crown splashes. The NB-FLIP method, on the other hand, creates splashes with the help of particles, but as a result it also creates persistent noise on surfaces even in calm areas. Our method successfully combines the benefits of both surface representations without artifacts. Since we only seed particles around the edges and the tips of splashes, the number of particles in this setup resulted in more than four times fewer particles than the NB-FLIP method on average. The computation of transition function of this example took 492 milliseconds on average. Note that this example contains an invisible ceiling in the scene, which could be noticed in the supplemental video.

The breaking dam example of Figure 3 demonstrates the ability of our algorithm to handle solid obstacles; our transition function automatically detects such collision events and naturally switches to NB-FLIP as desired. As a result, our method exhibits fully dynamic splashes in the wake of the cylinders while retaining smooth surfaces everywhere else. The computation of transition function of this example took 130 milliseconds on average. A high resolution example ($1024 \times 512 \times 256$) of the same setup can be seen in the supplemental material. This example also contains an invisible ceiling.

Figure 1 shows how our method performs when it is applied to a scene with complex obstacles. In this example, the geometric heat function term of Eq. (4) acts to retain sharp corners during the liquid's free-fall. After the impact, the entire scene undergoes active motion but the seams between the NB-FLIP and the level set method are invisible for the entire duration of the simulation. The computation of transition function of this example took 826 milliseconds on average.

Table 1, Figure 6, Figure 7 and Figure 8 illustrate the performance of our method. The kinetic energy plots show that the energy fluctuates, and slowly decreases over time. The particle count plots show that our method consistently requires fewer particles than the NB-FLIP method. Although the reduction factor depends on user parameters, we were able to reduce the number of particles by 6.5 times on average for Figure 2. Although we were able to accelerate the computations associated with particle operations by a factor of up to 3.4 times for Figure 2, the performance gain by our method did not strongly influence the overall simulation time. This is because the cost of pressure solve often dominates the simulation time, as also pointed out by, e.g., Lentine et al. [LZF10].

5. Discussion and Limitation

We note that Eq. (3) can overlook some physically important features such as vorticity, and it may not completely filter out motions such as rotational rigid movements. Although we did not observe any noticeable artifacts, we would like to explore the use of physics-based measures e.g., the pressure gradient and the vorticity [BHW13] as future work.

We did not fully optimize our code to outperform a NB-FLIP simulation. Although the particle count of our method will generally be lower than that of NB-FLIP, our method spends a bit more computation on maintaining a transition function, and the number of particles simulated by our method depends on the particular thresholds in the transition function. Consequently, our method may not outperform the NB-FLIP method in terms of runtime when the whole scene is actively undergoing chaotic and dynamic motion, but our Extended NB-FLIP method still outperforms the NB-FLIP for the whole duration.

The applications of our method are not tied to a particular surface representation. For example, we expect that our transition function

Scene	Method	Simulation [s]			Particle	Particle operations [ms]						
		Projection	Rest	Total	number	Advect	Grid transfer	Position correct	Surface reconstruct	Reseed	Rest	Total
Figure 3	EXNB-FLIP	1.65	1.58	3.23	173K	21	83	102	246	55	77	583
Figure 3	NB-FLIP	1.61	1.82	3.43	299K	34	135	166	417	84	105	941
Figure 3	Levelset	1.66	0.67	2.33	-	-	-	-	-	-	-	-
Figure 2	EXNB-FLIP	2.80	3.31	6.12	74K	9	42	45	233	33	116	478
Figure 2	NB-FLIP	2.68	3.91	6.60	485K	55	215	249	750	140	231	1639
Figure 2	Levelset	2.85	1.83	4.69	-	-	-	-	-	-	-	-
Figure 1	EXNB-FLIP	9.38	7.74	17.12	533K	73	244	311	896	182	330	2036
Figure 1	NB-FLIP	9.04	8.98	18.02	1304K	177	553	745	1926	409	580	4389
Figure 1	Levelset	9.28	3.83	13.11	-	-	-	-	-	-	-	-

Table 1: *Timing break-downs of our examples (average per step)*

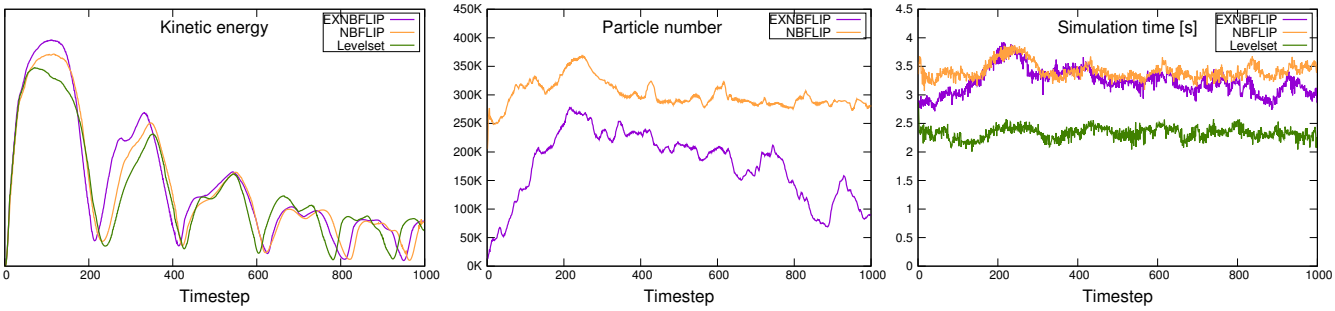

Figure 6: *Kinetic energy, number of particles and simulation time of Figure 3*

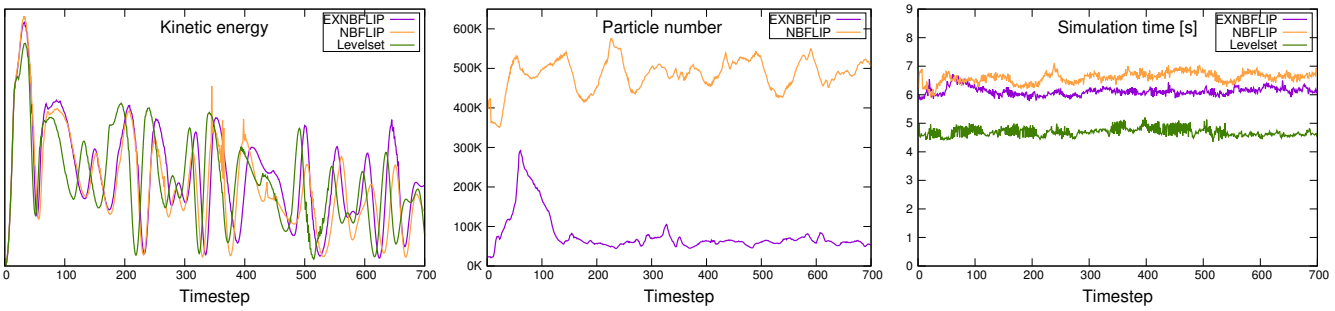

Figure 7: *Kinetic energy, number of particles and simulation time of our method in Figure 2*

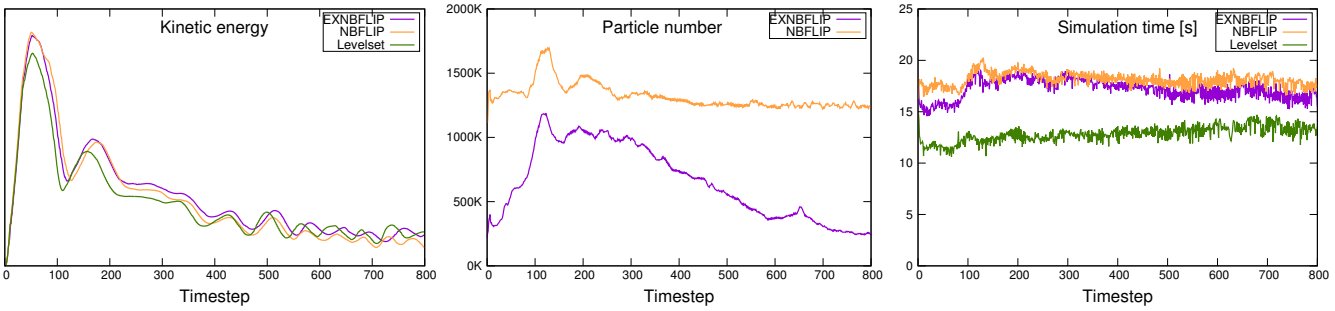

Figure 8: *Kinetic energy, number of particles and simulation time of our method in Figure 1*

might be useful for predicting where to increase resolution in an adaptive physics simulator since all we need is the velocity field and the level set surface to evaluate Eq. (6). In the case of a purely Eulerian simulator, we can instead resort to tracer particles to advect heat. We also expect that the simulation of fluid-fluid interaction (e.g., water/air or water/highly viscous liquid) and two-way coupled rigid bodies will be straightforward, as we compute the physics on grids using a traditional physics solver, and we use our method only to advect/track physics quantities.

6. Conclusion

This paper introduced a technique to leverage both the NB-FLIP method and the Eulerian level set method to track surfaces that are both dynamic and detailed in certain places, while being calm and smooth in other regions. The key component of our method was to employ the NB-FLIP method only where the motion is dynamic and switch to the Eulerian level set method and the grid-based advection for the rest of the domain.

We showed that the simple linear combination of these two different surface representation yields high-quality surfaces with seamless transitions. In order to compute the transitioning ratio, we developed the *heat function*. This heat function was subsequently mapped onto particles and advected along with them. Finally, the heat was rasterized back to grids to complete the calculation of the *transition function*. We also demonstrated that our method furthermore speeds-up the NB-FLIP method and reduces the amount of data storage. As future work, we would like to explore additional transition functions to more accurately capture physically interesting motions and to further reduce the total particle count.

Acknowledgements

This work is supported by JSPS KAKENHI Grant 17H00752 and the ERC Starting Grants 637014 and 638176.

References

[ATT12] ANDO R., THUREY N., TSURUNO R.: Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (Aug. 2012), 1202–1214. 2, 5

[ATW13] ANDO R., THUEREY N., WOJTAN C.: Highly Adaptive Liquid Simulations on Tetrahedral Meshes. *ACM Transactions on Graphics (SIGGRAPH) 32 (4)* (August 2013), 10. 5, 6

[BB12] BOYD L., BRIDSON R.: Multiflip for energetic two-phase fluid simulation. *ACM Trans. Graph. 31*, 2 (Apr. 2012), 16:1–16:12. 5

[BBB07a] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. 26*, 3 (July 2007). 2

[BBB07b] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. 26*, 3 (2007), 100. 5

[BGB11] 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* (New York, NY, USA, 2011), SCA '11, ACM, pp. 17–24. 2

[BHW13] BOJSEN-HANSEN M., WOJTAN C.: Liquid surface tracking with error compensation. *ACM Transactions on Graphics (SIGGRAPH 2013) 32*, 4 (2013), 79:1–79:10. 4, 7

[Bri15] BRIDSON R.: *Fluid Simulation for Computer Graphics, Second Edition*. Taylor & Francis, 2015. 2, 6, 7

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21*, 3 (July 2002), 736–744. 2, 5, 6

[FAW*16] FERSTL F., ANDO R., WOJTAN C., WESTERMANN R., THUEREY N.: Narrow band flip for liquid simulations. *Comput. Graph. Forum 35*, 2 (May 2016), 225–232. 2, 5

[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. 2

[FGG*17] FU C., GUO Q., GAST T., JIANG C., TERAN J.: A polynomial particle-in-cell method. *ACM Trans. Graph. 36*, 6 (Nov. 2017), 222:1–222:12. 2

[GBW16] GOLDADE R., BATTY C., WOJTAN C.: A practical method for high-resolution embedded liquid surfaces. *Computer Graphics Forum (Proc. Eurographics 2016)* (2016). 4

[GSLF05] GUENDELMAN E., SELLE A., LOSASSO F., FEDKIW R.: Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. 24*, 3 (July 2005), 973–981. 5

[JSS*15] JIANG C., SCHROEDER C., SELLE A., TERAN J., STOMAKHIN A.: The affine particle-in-cell method. *ACM Trans. Graph. 34*, 4 (July 2015), 51:1–51:10. 2, 5, 6

[Lin98] LINDEBERG T.: Feature detection with automatic scale selection. *Int. J. Comput. Vision 30*, 2 (Nov. 1998), 79–116. 6

[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*, 4 (July 2008), 797–804. 2

[LZF10] LENTINE M., ZHENG W., FEDKIW R.: A novel algorithm for incompressible flow using only a coarse grid projection. In *ACM SIGGRAPH 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 114:1–114:9. 7

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based Fluid Simulation for Interactive Applications. In *Symposium on Computer Animation* (2003), pp. 154–159. 2

[OS88] OSHER S., SETHIAN J. A.: Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys. 79*, 1 (Nov. 1988), 12–49. 2

[SFK*08] SELLE A., FEDKIW R., KIM B., LIU Y., ROSSIGNAC J.: An Unconditionally Stable MacCormack Method. *J. Sci. Comput. 35*, 2-3 (June 2008), 350–371. 4

[SO09] SUSSMAN M., OHTA M.: A stable and efficient method for treating surface tension in incompressible two-phase flow. *SIAM J. Sci. Comput. 31*, 4 (June 2009), 2447–2471. 6

[TWGT10] THÜREY N., WOJTAN C., GROSS M., TURK G.: A multiscale approach to mesh-based surface tension flows. *ACM Trans. on Graphics (Proc. SIGGRAPH) 29*, 3 (2010). 6

[UBH14] UM K., BAEK S., HAN J.: Advanced hybrid particle-grid method with sub-grid particle correction. *Comput. Graph. Forum 33*, 7 (Oct. 2014), 209–218. 2

[WMFB11] WOJTAN C., MÜLLER-FISCHER M., BROCHU T.: Liquid simulation with mesh-based surface tracking. In *ACM SIGGRAPH 2011 Courses* (2011), ACM, p. 8. 2

[YT13] YU J., TURK G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Trans. Graph. 32*, 1 (Feb. 2013), 5:1–5:12. 2

[YWTY12] YU J., WOJTAN C., TURK G., YAP C.: Explicit mesh surfaces for particle based fluids. *Comput. Graph. Forum 31*, 2pt4 (May 2012), 815–824. 2

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (July 2005), 965–972. 2, 5, 6