EUROGRAPHICS 2016 / J. Jorge and M. Lin
(Guest Editors)

Volume 35 (2016), Number 2

Narrow Band FLIP for Liquid Simulations

Florian Ferstl[1], Ryoichi Ando[2], Chris Wojtan[2], Rüdiger Westermann[1], and Nils Thuerey[1]

[1]Technische Universität München [2]IST Austria

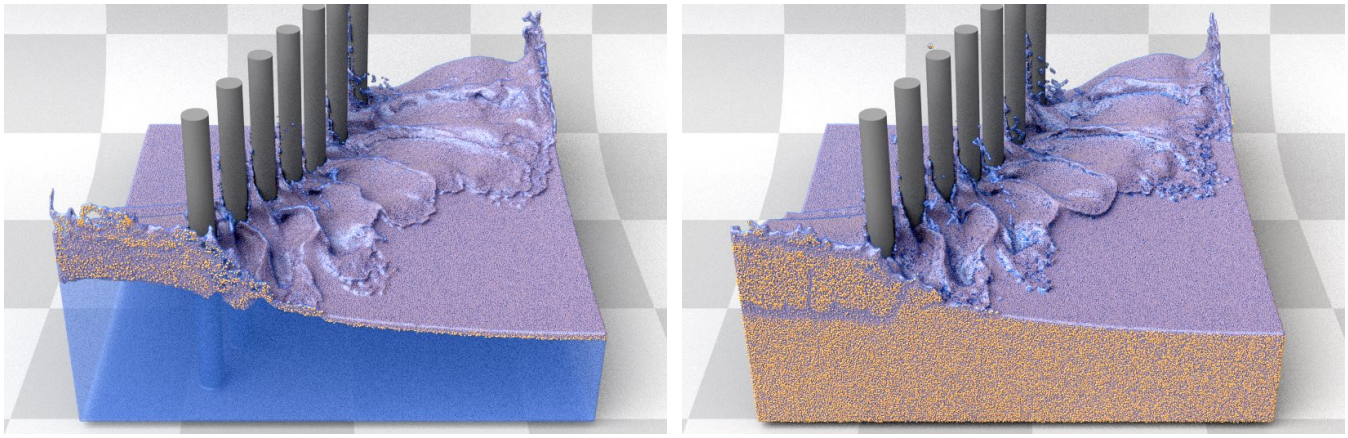


Figure 1: *Two simulations with a (background-)grid resolution of* 200^3*: Our method (left) is nearly indistinguishable from regular FLIP (right), but uses only one million particles instead of 24 million. Computations outside of the pressure solver are sped up by a factor of more than 6, the total computation time for 250 frames is reduced from 4.4h to 2h, and the average memory footprint is reduced by a factor of 2.*

Abstract

The Fluid Implicit Particle method (FLIP) for liquid simulations uses particles to reduce numerical dissipation and provide important visual cues for events like complex splashes and small-scale features near the liquid surface. Unfortunately, FLIP simulations can be computationally expensive, because they require a dense sampling of particles to fill the entire liquid volume. Furthermore, the vast majority of these FLIP particles contribute nothing to the fluid's visual appearance, especially for larger volumes of liquid. We present a method that only uses FLIP particles within a narrow band of the liquid surface, while efficiently representing the remaining inner volume on a regular grid. We show that a naïve realization of this idea introduces unstable and uncontrollable energy fluctuations, and we propose a novel coupling scheme between FLIP particles and regular grid which overcomes this problem. Our method drastically reduces the particle count and simulation times while yielding results that are nearly indistinguishable from regular FLIP simulations. Our approach is easy to integrate into any existing FLIP implementation.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

The Fluid Implicit Particle method (FLIP) is arguably the most widely used simulation algorithm for high-quality liquid special effects [BLMB13, BBAW15]. Impressive large-scale effects ranging from characters splashing in the ocean to flooding cities have been simulated with this method. Its main strengths result from leveraging non-diffusive Lagrangian transport in combination with accurate Eulerian discretizations to ensure divergence-freeness. The particles in a FLIP simulation also provide important visual cues for events like complex splashes and small-scale features near the liquid surface.

Unfortunately, FLIP simulations can be computationally expensive, as they require a dense sampling of the fluid domain with particles. Furthermore, the vast majority of these FLIP particles contribute nothing to the fluid's visual appearance, especially for larger volumes of liquid.

We present a method that only uses FLIP particles within a nar-

row band of the liquid surface, while efficiently representing the remaining inner body of the fluid on a regular grid. Although the main idea may seem straightforward, its implementation is non-trivial, and the naïve approach introduces unstable and uncontrollable energy fluctuations. Our method drastically reduces the particle count and simulation times, while still exhibiting the desired bounded energy behavior and yielding results that are nearly indistinguishable from regular FLIP simulations.

The contributions of our method are:

- A method to efficiently reduce the number of particles in a FLIP simulation, leading to speed-ups up to a factor of 5 in our examples (factor 2 in average), and
- Insights into instabilities caused by naïve couplings between particle and grid velocities, leading to a stable narrow band approach.

In this way, we achieve significant performance improvements with an approach that can be easily integrated into any existing FLIP implementation.

1.1. Related Work

Researchers in computer graphics use a variety of methods for simulating fluid dynamics. Eulerian discretizations are popular because they avoid complicated re-meshing of Lagrangian meshes and neighbor-finding between Lagrangian particles, and they can take advantage of fast methods for indexing and caching data in regular grids [FM96, Sta99, FF01]. However, Eulerian methods tend to either suffer from CFL conditions or numerical diffusion from continual re-sampling operations. On the other hand, Lagrangian fluid simulation techniques, such as smoothed particle hydrodynamics (SPH) [MCG03, IOS*14] avoid continual re-sampling operations at the expense of stability restrictions, neighborhood searches, and numerical smoothing over the radius of a pre-defined kernel. We refer the reader to [Bri08] for a more thorough look at fluid dynamics research in computer graphics.

Our method is based on an Eulerian-Lagrangian hybrid method called the fluid-implicit particle method (FLIP), which was introduced to the graphics community by Zhu and Bridson [ZB05]. The FLIP method has also been used for fluid control [PHT*13] and two-phase flow [BB12, ATW15]. Note that our narrow band approach is actually used by Ando et al. [ATW15]. In addition, previous researchers have improved upon the FLIP method by correcting particle positions [UBH14] or adapting grid and particle sizes [ATW13]. Ando et al. [ATT12] use directed particle insertion in combination with particle merging and splitting to simulate thin liquid sheets, effectively using a higher FLIP particle density near the liquid surface than in the liquids interior. Adaptive particle sizes have also been used in purely Lagrangian approaches, e.g., by Solenthaler and Gross [SG11].

Several other approaches have taken advantage of hybrid methods in order to leverage the strengths of both Eulerian and Lagrangian fluid discretizations. Several authors [SS06, LTKF08, RWT11] were able to successfully combine SPH simulations with regular grid-based simulations, while Cornelis et al. [CIPT14] use SPH particles for the pressure projection step of a FLIP simulation. The material point method (MPM) combines a grid-based Eulerian simulation with Lagrangian marker particles in order to model a wider range of elastoplastic materials such as snow [SSC*13] or viscoelastic fluids and foams [RGJ*15, YSB*15]. Golas et al. [GNS*12] use Lagrangian particles as a vortex basis in the interior of a grid-based Eulerian simulation. For the related problem of surface tracking, Enright et al. [ELF05] augment the Eulerian level set method with Lagrangian marker particles placed on both sides of a tracked surface, in order to better preserve Lagrangian details while letting the grid handle topological changes. Their method is similar to our tracking of the liquids surface, yet we only require Lagrangian particles on one side of the surface.

Recently, Chentanez et al. [CMK14] coupled a 3D Eulerian method, a 2D Eulerian shallow water solver, and a Lagrangian SPH discretization within the same simulation. Their approach of computing an SPH simulation near the surface of an Eulerian solver is similar in spirit to ours, but their approach focuses solely on SPH, not FLIP. Additionally, in our experience, even a non-trivial adoption of their method to pure FLIP simulations leads to inherently unstable simulations. The analysis and solution of this problem is a central contribution of our paper.

2. FLIP

Our liquid simulation solves the inviscid Navier-Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u}\cdot\nabla\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f},$$

which are subject to the incompressibility constraint $\nabla\cdot\mathbf{u} = 0$. Here, $\mathbf{u}$ denotes the liquid's velocity field, p the internal pressure, ρ the density and $\mathbf{f}$ represents acceleration due to external forces acting on the liquid. In the following, normal and bold variables represent parameters and variables stored on a regular grid, while variables with the subscript $_i$ refer to quantities stored on particles. Variables with the superscript ^{p} refer to particle quantities mapped to the grid.

We will briefly review a typical FLIP approach, before detailing our narrow band algorithm. In a time step of the regular FLIP algorithm, the particles are first passively advected using the grid velocity $\mathbf{u}$. Then, the velocity stored on the particles is mapped to the grid, yielding a velocity field $\mathbf{u}^p$ which covers the whole fluid domain and which is stored as an intermediate velocity field $\mathbf{u}^*$. In a similar manner, a particle region is reconstructed on the grid in the form of a scalar field ϕ^p, whose zero-contour encloses all particles. This ϕ^p is used in the remaining steps as level set function ϕ to mark the location of the liquid surface. The external forces are added, $\mathbf{u}^*$ is projected and the velocity of every particle i is updated according to the formula

$$\mathbf{u}_i \leftarrow (1-\alpha)\,\mathbf{u}_i^{\mathrm{PIC}} + \alpha\,\mathbf{u}_i^{\mathrm{FLIP}}. \tag{1}$$

Here, $\mathbf{u}_i^{\mathrm{PIC}}$ is the current grid velocity, and $\mathbf{u}_i^{\mathrm{FLIP}}$ is the difference between the current velocity and the velocity right after mapping the particles to the grid, both interpolated at the position of particle i. The parameter $\alpha \in [0;1]$ controls the amount of velocity diffusion in the simulation, and can be related to the fluid viscosity. It is advisable to use some form of particle re-sampling to avoid extremely densely and/or sparsely sampled regions. In its simplest form, this re-sampling adds and removes particles where necessary to keep the per-cell particle count in a chosen range. We do not perform

this re-sampling in cells at the liquid surface to prevent visual artifacts. In our implementation we use $\alpha = 0.95$, in combination with tri-linear weights (i.e., a tri-linear kernel function) to map particle data to the grid, and vice versa. Fig. 2 (with statements colored red) summarizes the regular FLIP algorithm described above.

In principle, FLIP particles are relatively cheap: In contrast to particles in, e.g., SPH, there is no communication between the particles themselves, and particle solely communicate via the underlying grid. Also, FLIP particles typically do not carry a mass and are passively advected with the flow. Hence, they are very robust against variations in sampling density, and can be removed and added without much overhead. However, in practice, the sheer amount of particles that is required to run a full FLIP simulation leads to a huge computational workload—typically, eight particles are used per cell in three dimensions [Bri08, BB12], but fewer [BB08] or even more [GB13] particles have also been used successfully. In all our examples (see Section 4), the computations involving FLIP particles take up more than 60% of the computation time in each simulation step when using a standard, preconditioned conjugate gradients solver. This ratio increases further for advanced variations of FLIP, e.g., when a position correction step is performed to ensure a more uniform particle distribution [ATT12, UBH14].

3. Narrow band FLIP

The central idea of our approach is to use FLIP particles only in a narrow band of fixed width R inside of the liquid surface, and to use a standard, grid-based simulation with semi-Lagrangian advection in the interior of the domain. Hence, we refer to our method as narrow band FLIP (NB-FLIP). Our narrow band method consists of three main components:

1. The coupling of grid simulation and FLIP simulation.
2. A surface tracking method for correctly handling the liquids interior.
3. The re-sampling of particles to maintain a narrow band of fixed width.

The green statements in Fig. 2 highlight that few changes are necessary to turn a standard FLIP simulation into a NB-FLIP simulation.

Throughout our algorithm, we make use of a level set function ϕ. In contrast to standard FLIP, we need information about the distance of particles to the liquid surface. However, because ϕ is not our main visual representation, it does not need to be overly accurate, and we initialize a Manhattan distance with several quick passes over the grid. As a consequence, the reduced amount of particles in NB-FLIP easily compensates for the additional cost of the level set, as well as for other additional steps that our method introduces.

Our choice to restrict particles to the surface region can be further motivated by comparing FLIP velocity fields with their counter parts in pure grid-based simulations. We observe that differences between the velocity fields—advected via FLIP particles on the one hand, and advected using a semi-Lagrangian scheme on the other hand—are predominantly noticeable near the surface. The impact of the FLIP particles is strongest right at the surface and quickly decreases away from it, which is illustrated in Fig. 3. This effect can

1: Advect particles
2: [NB-FLIP] Advect $\mathbf{u}$ and ϕ on grid $\Rightarrow \mathbf{u}'$ and ϕ'
3: Map particles to grid $\Rightarrow \phi^p$ and $\mathbf{u}^p$
4: Update level set and velocity on grid
  [FLIP] $\phi \leftarrow \phi^p$
  $\mathbf{u}^* \leftarrow \mathbf{u}^p$
  [NB-FLIP] $\phi \leftarrow \min(\phi' + h, \phi^p)$ (Eq. 4)
  Re-initialize ϕ
  $\mathbf{u}^* \leftarrow \text{nbcombine}(\mathbf{u}', \mathbf{u}^p, \phi)$ (Eq. 3)
5: Add external forces $\mathbf{f}$
6: Project $\mathbf{u}^* \Rightarrow \mathbf{u}$
7: Update particle velocities (Eq. 1)
8: [FLIP] Resample particles (optional)
  [NB-FLIP] Resample particles in narrow-band

Figure 2: *Comparison of the time step algorithms for FLIP and NB-FLIP. Steps tagged with* [FLIP] *and* [NB-FLIP] *are unique to FLIP and NB-FLIP, respectively. Untagged steps are common to both methods.*

be observed independently of the order of the employed advection scheme. Rather, it is a result of the inherent differences between the backward semi-Lagrangian advection and the forward FLIP advection. Close to the surface, these differences are amplified through high frequencies in the underlying velocity fields and through the use of approximate, extrapolated velocities outside of the liquid's surface.

A comparison of the memory requirements for a regular FLIP simulation versus a pure Eulerian simulation indicates the memory savings that can potentially be achieved with our method: Each FLIP particle has to store at least position and velocity (3 floats each). Assuming 8 particles per cell, this leads to a memory requirement of 48 floating point numbers per cell. In contrast, when using a purely Eulerian representation, the required information per cell is a single velocity and a level-set value (4 floats in total). Thus, in all areas where our method can switch to the grid based representation, memory is reduced by a factor of 12.

3.1. Grid-particle coupling

The FLIP simulation in the narrow band has to be properly coupled with the grid simulation in the liquids interior: First, we advect the grid velocity $\mathbf{u}$ on the whole liquid domain by performing a semi-Lagrangian advection step, which yields an advected grid velocity $\mathbf{u}'$. Similarly, we advect all FLIP particles and map them to the grid, which in turn yields an advected particle velocity $\mathbf{u}^p$. Finally, at all grid vertices where a particle velocity is available (i.e., where $\mathbf{u}^p$ is valid), we combine the values of $\mathbf{u}'$ and $\mathbf{u}^p$ into a single velocity $\mathbf{u}^*$. At all remaining grid vertices, we set $\mathbf{u}^*$ equal to $\mathbf{u}'$. The combination of $\mathbf{u}^p$ and $\mathbf{u}'$ should ensure a full FLIP simulation near the liquid surface, i.e., values of $\mathbf{u}^p$ should completely overwrite values of $\mathbf{u}'$ in the interior of the narrow band as well as at the surface itself. In addition to that, the transition from grid to narrow band is crucial, and therefore has to be handled properly. To demonstrate that a naive transition can quickly lead to unphysical gains in momentum, we will **first** present an **erroneous** combination rule that is based on ideas from existing methods, and, **afterwards**, we will present our "**fixed**" combination rule.

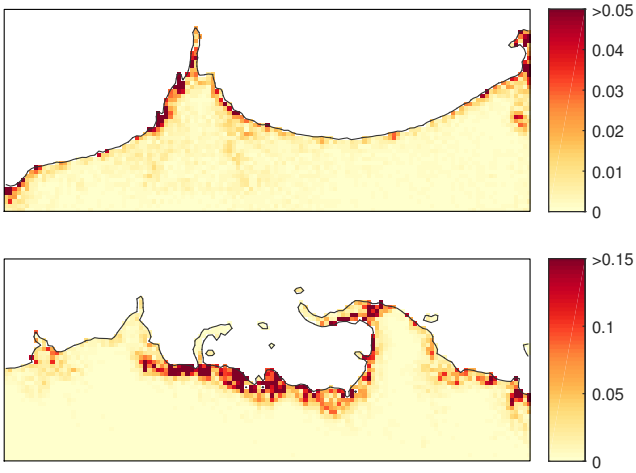


Figure 3: *Comparison of semi-Lagrangian and grid-based advection for two timesteps of a 2D simulation. Shown is a color coding of* $||\mathbf{u}' - \mathbf{u}^p||/||\mathbf{u}||$*: the difference between* $\mathbf{u}'$ *and* $\mathbf{u}^p$ *(velocity advected with a semi-Lagrangian scheme and FLIP particles, respectively), relative to the magnitude of the velocity* $\mathbf{u}$ *(see Fig. 2).*

Several previous works deal with particle/grid couplings. The one most closely related to ours is the method by Chentanez et al. [CMK14], in which SPH particles at the surface are coupled with a grid-based, semi-Lagrangian simulation. A smooth transition from grid to particles is realized based on the ratio between a grid density and a particle density. Since we do not use a variable density formulation, we adapt this to our case by only computing a single density on the grid, i.e., a non-physical particle density $\rho^p \geq 0$. At each grid vertex, we determine ρ^p by summing up the kernel weights of all contributing, neighboring FLIP particles, and then calculate a combined velocity from $\mathbf{u}^p$ and $\mathbf{u}'$ at every face $F = (i+\frac{1}{2}, j, k)$ of the staggered grid. In line with previous work, an intuitive, yet unfortunately **erroneous** formulation for this combination is given by:

$$u_F^* \leftarrow \text{lerp}\left(u_F', u_F^p, t\right), \quad \text{with } t = \min\left(1, \frac{\rho_F^p}{\rho_{\max}^p}\right). \qquad (2)$$

Here, $\rho_{\max}^p$ is a constant threshold, which we set to 1 in all our experiments, and $\text{lerp}(a,b,t) = a + t \cdot (b-a)$ is the linear interpolation operator. Eq. (2) is analogously applied to the y- and z-components of the velocity, which are stored at the correspondingly oriented faces of the staggered grid.

As we will outline below, the combined velocity calculated with Eq. (2) leads to inherently unstable simulations and cannot be used in practice. Note that in the SPH-coupling approach by Chentanez et al. [CMK14], the FLIP coefficient α (see Eq. (1)) is varied based on the distance to the liquid surface in addition to mixing particle and grid velocities. In our experiments, this variation did not cause significant differences other than increasing the amount of numerical diffusion (because of smaller values for α), and hence we do not use this variant. For the same reason, we found it unnecessary to employ their proposed temporal filtering of transition weights.

The problem with the density-based combination of Eq. (2) occurs at the transition of the narrow band to the inner volume. Grid vertices located just outside of the narrow band —i.e., vertices whose distance to the liquid surface is larger than the narrow band width R—can receive contributions from the narrow-band particles. Effectively, this means that particle velocities are extrapolated into the interior grid region, which results in relatively large differences between $\mathbf{u}^p$ and $\mathbf{u}'$ (compared to the interior of the narrow band). In practice, these differences lead to a consistent over-estimation of the fluid motion. Amplified by the non-dissipative nature of the FLIP simulation in the narrow band, they typically accumulate to large errors over time.

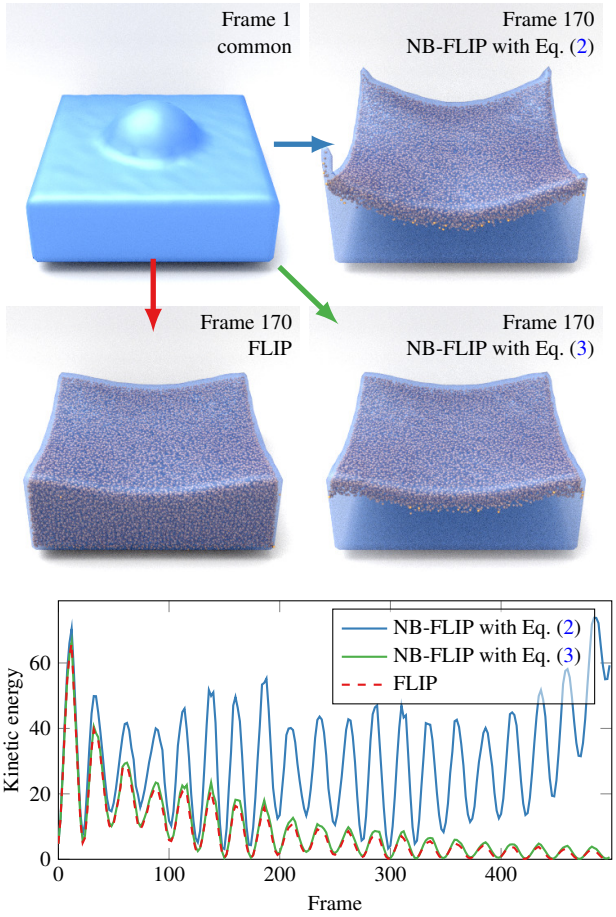
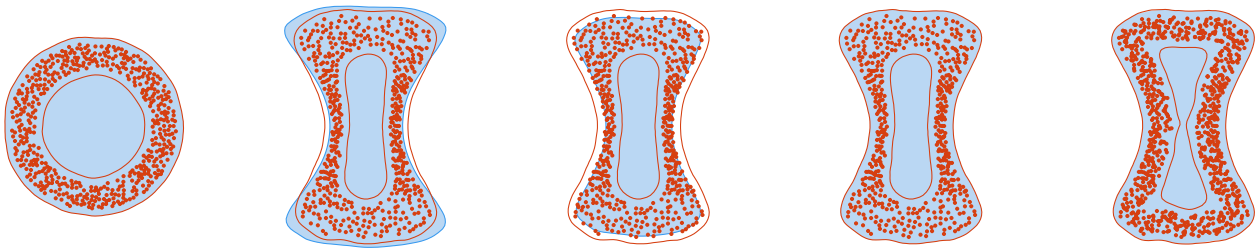


Figure 4: *Comparison of velocity combination variants: (Top) Four snapshots of a* 32^3 *simulation of surfaces waves. (Bottom) Plot of the corresponding kinetic energies. While NB-FLIP with Eq. (2) generates artificial momentum, NB-FLIP with Eq. (3) is almost identical to standard FLIP, and the surface correctly comes to rest.*

To overcome this problem, we make the combination dependent on the distance of the grid vertices to the surface, which ensures that no extrapolated particle velocities are used in the interior of the liquid. While the narrow band has a width of R, we only use the particle velocities $\mathbf{u}^p$ in a thinner band of width $r < R$, which is illustrated in Fig. 6. Because we found, that a smooth transition between grid and particle velocities does not increase the quality of



(a) Initial state (Previous) ϕ and ϕ^p | (b) Advected ϕ' and ϕ^p | (c) Shrunk level set $\phi' + h$ and ϕ^p | (d) Union $\phi := \min(\phi' + h, \phi^p)$ and ϕ^p | (e) Re-sampled particles

Figure 5: *Surface tracking and particle re-sampling in NB-FLIP: FLIP particles (orange dots), zero-contour of the level set ϕ (blue, filled) and zero-contour of the mapped particle region ϕ^p (orange, contour only) during one time step of our algorithm (see Fig. 2). Note that ϕ^p is only shown for illustration purposes in (a+e) and is not actually computed after re-sampling.*

the simulation, we use a sharp transition between $\mathbf{u}^p$ and $\mathbf{u}'$. This results in the following "**fixed**" combination rule, which is referred to as nbcombine$(\mathbf{u}', \mathbf{u}^p, \phi)$ in Fig. 2:

$$u_F^* \leftarrow \begin{cases} u_F^p, & \text{if } \phi_F \geq -r \\ u_F', & \text{else} \end{cases} . \tag{3}$$

The distance r should be chosen depending on the narrow band width R. The choice of R, in turn, is a performance trade-off, with NB-FLIP becoming identical to FLIP if $R \to \infty$. Let h denote the width of one cell. Then, where not stated otherwise, we use a narrow band width of $R = 3h$, and, because we use a tri-linear interpolation kernel, we choose r to be one cell smaller, i.e., $r = 2h$.

Fig. 4 compares both versions and illustrates the importance of the velocity transition. The unphysical behavior produced by Eq. (2) is obvious in the energy plot even for this simplified setup. For more complex scenarios this approach quickly leads to overly violent and unstable liquid motions. In contrast, our transition function using Eq. (3) yields the expected behavior and produces results that are very close to a regular FLIP simulation (Fig. 4 bottom left). Animations of these cases can be found in the accompanying video.

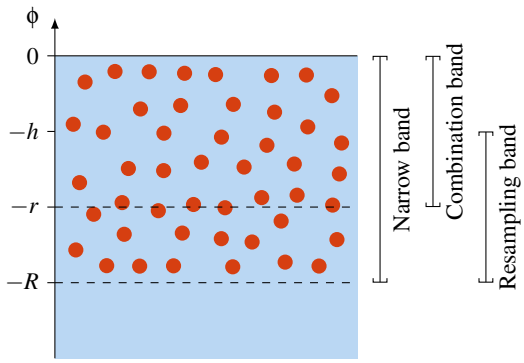


Figure 6: *Breakdown of the narrow band: FLIP particles are maintained in a band of width R inside of the liquid surface, but their velocities only contribute to vertices in a "combination band" of width $r < R$ (see Eq. (3)). Particles are only added/removed in a "resampling band" with minimum distance h to the surface, in order to prevent changes to the liquid surface.*

3.2. Surface tracking

In standard FLIP, the whole liquid domain is covered by particles, which makes surface tracking straightforward. Compared to that, in NB-FLIP, particles are missing in the liquids interior, and the inside and outside of the narrow band cannot be distinguished without additional information. Therefore, to keep track of the particle-free regions that belong to the liquids interior, we use the level set ϕ.

We initialize ϕ and a corresponding band of particles directly from analytical object descriptions or meshes. The substeps of one surface advection step (see Fig. 2) are illustrated in Fig. 5. We start by advecting ϕ, which yields an advected level set ϕ', and map the advected particles to the grid, yielding a level set ϕ^p of the narrow band particles. The zero-contour of ϕ' and the outer zero-contour of ϕ^p (which corresponds to the liquid surface) are similar, but not equal. Since we want to obtain a tracking scheme that is equivalent to standard FLIP, the particle surface should "overrule" the level set surface. Therefore, we shrink the surface of the level set ϕ' by the width of one cell h, and then compute its union with ϕ^p. More precisely, we compute the level set of the next time step as

$$\phi \leftarrow \min(\phi' + h, \phi^p). \tag{4}$$

In principle, this method works if the maximum distance error, i.e., the distance between the surfaces corresponding to ϕ' and ϕ^p, is less than h. Thin liquid features are completely sampled with particles, and are always treated correctly. To ensure a small distance error, we use a 4th order Runge-Kutta scheme to compute the forward- and back-traces required for advection. In practice, our surface tracking works well even for time steps with a CFL number of 5 and higher. Note that, for stability reasons, we do not shrink the level set surface by a distance larger than h. This is because excessive shrinking can lead to spurious holes beneath the narrow band which are hard to fix.

3.3. Particle re-sampling

In order to maintain a narrow band of fixed width R, we re-sample the particles at the end of each time step. Our re-sampling is based on a minimum number n of particles per cell (we use $n = 8$). First, we remove all particles outside of the narrow band, i.e., particles whose distance to the liquid surface is larger than R, and ensure that no more than $2n$ particles are present per cell by removing excess

particles. Likewise, we add new particles in cells where less than n particles are present, whose velocities are initialized with the current grid velocity. Since the liquid surface should not be changed when adding or removing particles, we never add or remove particles at positions that are less than one cell width h away from the surface, i.e. particles are only re-sampled where $-R \leq \phi \leq -h$ (see Fig. 6).

As a consequence of the particle re-sampling in the narrow band in combination with our surface tracking, the volume conservation of our method is equivalent to regular FLIP, i.e., we do not conserve mass exactly. The level set representation in the liquids interior implicitly conserves mass and the particle density in the narrow band is kept uniform. Hence, significant changes in mass can only originate right at the liquid's surface.

4. Results

We have implemented our method in two FLIP simulation frameworks: a basic, light-weight solver [PT15], and a more complex one with advanced FLIP techniques [ATT12]. Both implementations use eight particles per grid cell, as well as a standard conjugate gradients solver with modified incomplete Cholesky preconditioner for the projection step. Detailed runtimes and particle numbers for the following simulations can be found in Table 1. For the light-weight solver, we additionally specify the average memory footprint of the running simulation in the last column ("Avg. Memory"). We used narrow band widths of $R = 3h$ and $R = 4h$ (with $r = R - 1$), both of which we have found to be a good compromise between performance and memory consumption on the one hand and simulation quality on the other hand. Thinner bands can be problematic because the particles only have an influence in a smaller portion of the full narrow band (see Fig. 6), while slightly thicker bands do not offer significant gains in simulation quality.

First, we present several examples from our light-weight solver using $R = 3h$. In Fig. 7 we simulate an empty glass being filled with liquid. Here the simulation initially behaves like a regular FLIP simulation, while the grid-based data becomes predominant in later stages. On average, this simulation reduces the number of particles by a factor of more than 16. The simulation of Fig. 8 demonstrates how our method handles the complex splashes caused by a series of letter-shaped drops falling into a basin. Here, the regular simulation requires 18 times more particles than our narrow band version.

The simulation of Fig. 9 is our largest scene, with more than 23 million particles for the full FLIP simulation. For this setup, the computations outside the pressure solver were six times faster for our narrow band version. While the time for pressure projection is untouched by our narrow band implementation, techniques such as multi-grid solvers [CMF12] could be used to further reduce the overall runtime.

Overall, our experiments show a very good agreement between NB-FLIP and regular FLIP in terms of large scale liquid motions. Small scale effects like splashes and drops—due to the random nature of the particles and the accumulation of small errors—are not identical, but are very similar in quality. As a consequence, NB-FLIP delivers results that are very hard to distinguish from regular FLIP simulations.

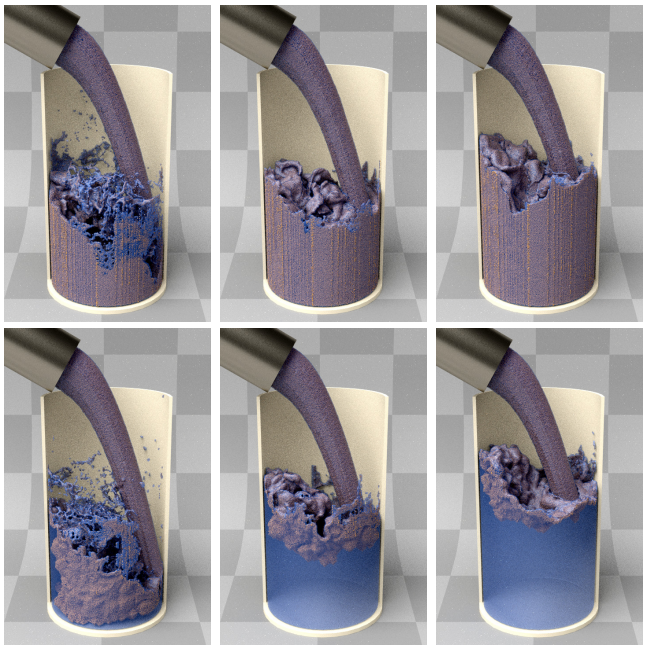


Figure 7: *Filling a glass with liquid (top: regular FLIP, bottom: NB-FLIP).*

To demonstrate that our approach is easily incorporated into existing solvers, we extended a FLIP solver using position correction and an SPH-like velocity interpolation kernel [ATT12] with our narrow band algorithm. Fig. 10 shows a resulting comparison based on this simulation framework. We use a narrow band width of $R = 4h$ for this example. As this solver performs more calculations with the FLIP particles, our method leads to even larger performance speed-ups. In this case, the total simulation time per simulation step is reduced by a factor of 5.6. As many implementations of FLIP in VFX pipelines execute a significant number of calculation passes with the FLIP particles, these numbers are indicative of the performance gains that can be expected from our approach in a practical setting.

Limitations: While we found that our approach works well enough in practice to fully replace regular FLIP simulations, there are limitations that we want to point out. One limitation is that for liquid phenomena that purely consist of thin structures (smaller than $2r$) the grid data will not be used, in which case our algorithm will be slightly slower than a regular FLIP simulation while giving identical results.

Also, the minimum thickness of the narrow-band depends on the maximal error per time step of the Eulerian level-set advection, which is admittedly difficult to quantify in a generic setting. The band should be thick enough to prevent these errors from leading to gaps between the level-set surface and the particles within a single time step. In practice, we have found our preferred choice of band widths (three to four cells) to be safe, even for cases with violent splashes.

Table 1: *Timings of NB-FLIP compared to FLIP for different scenarios and implementations.*

Scene	Grid Resolution	Method	Avg. Time/Timestep (s) Projection	+ Rest	= Total	Avg. #Particles	Avg. Memory (MB)
Fig. 7 / Pour	$128^2 \times 256$	FLIP	4.64	8.66	13.29	9,362,306	518
		NB-FLIP	4.67	1.98	6.64	561,997	303
Fig. 8 / Letters	$256 \times 192 \times 128$	FLIP	5.06	9.85	14.91	16,559,617	857
		NB-FLIP	5.20	2.67	7.88	880,827	456
Fig. 9 / Teaser	200^3	FLIP	9.00	15.78	24.78	23,891,821	1180
		NB-FLIP	8.72	2.43	11.15	1,079,688	578
Fig. 10 / Dam	$256 \times 128 \times 64$	FLIP*	1.57	33.22	34.79	2,889,981	—
		NB-FLIP*	1.49	4.65	6.14	191,625	—

*Advanced FLIP implementation [ATT12]

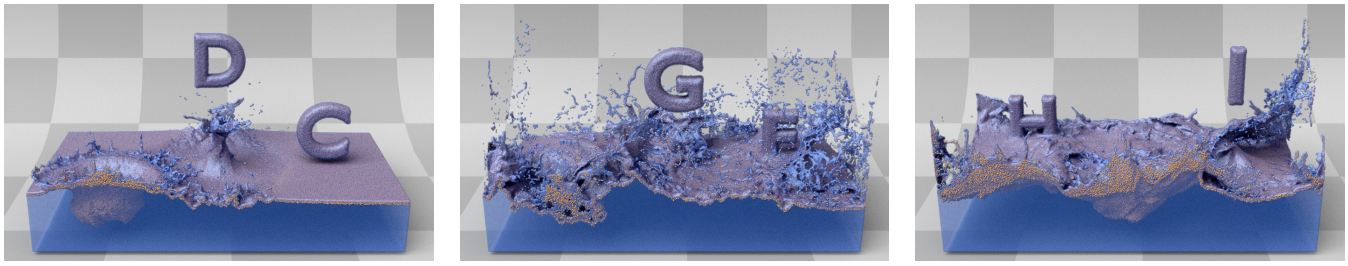


Figure 8: *A series of letter-shaped drops hitting a pool of liquid using narrow-band FLIP requiring* $18\times$ *fewer particles.*

5. Conclusions

We have described a novel approach to significantly reduce the number of FLIP particles. Our algorithm readily integrates into existing FLIP solvers and can result in up to $22\times$ fewer particles. This reduction factor strongly depends on the scenario, but especially for large volumes and advanced FLIP simulation approaches it can lead to speed-up factors larger than 5 without degrading simulation quality.

In addition to performance gains our approach has the potential to significantly reduce the amount of data that needs to be stored on hard disk (up to the aforementioned factor of 12). This is highly interesting for VFX productions, as legal requirements typically dictate that all data of a shot be available long after completion. It will be interesting future work to combine our approach with more aggressive compressions schemes to further reduce storage requirements.

References

[ATT12] ANDO R., THUEREY N., TSURUNO R.: Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE Trans. on Visualization and Computer Graphics 18 (8)* (2012), 1202–1214. 2, 3, 6, 7

[ATW13] ANDO R., THUEREY N., WOJTAN C.: Highly adaptive liquid simulations on tetrahedral meshes. *ACM Transactions on Graphics 32*, 4 (2013), 103:1–103:10. 2

[ATW15] ANDO R., THUEREY N., WOJTAN C.: A stream function solver for liquid simulations. *ACM Transactions on Graphics 34*, 4 (2015), 53:1–53:9. 2

[BB08] BATTY C., BRIDSON R.: Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Symposium on Computer Animation* (2008), pp. 219–228. 3

[BB12] BOYD L., BRIDSON R.: MultiFLIP for energetic two-phase fluid simulation. *ACM Trans. on Graphics 31*, 2 (2012), 16:1–16:12. 2, 3

[BBAW15] BAILEY D., BIDDLE H., AVRAMOUSSIS N., WARNER M.: Distributing liquids using OpenVDB. In *ACM SIGGRAPH 2015 Talks* (2015), ACM, pp. 285:1–285:1. 1

[BLMB13] BUDSBERG J., LOSURE M., MUSETH K., BAER M.: Liquids in "The Croods". In *Digital Prod. Symp. (DigiPro)* (2013). 1

[Bri08] BRIDSON R.: *Fluid Simulation for Computer Graphics*. AK Peters/CRC Press, 2008. 2, 3

[CIPT14] CORNELIS J., IHMSEN M., PEER A., TESCHNER M.: IISPH-FLIP for incompressible fluids. *Computer Graphics Forum 33*, 2 (2014), 255–262. 2

[CMF12] CHENTANEZ N., MUELLER-FISCHER M.: A multigrid fluid pressure solver handling separating solid boundary conditions. *IEEE Trans. Visualization and Comp. Graph. 18*, 8 (2012), 1191–1201. 6

[CMK14] CHENTANEZ N., MÜLLER M., KIM T.: Coupling 3D Eulerian, heightfield and particle methods for interactive simulation of large scale liquid phenomena. In *Symp. Comp. Anim.* (2014), pp. 1–10. 2, 4

[ELF05] ENRIGHT D., LOSASSO F., FEDKIW R.: A fast and accurate semi-Lagrangian particle level set method. *Computers & Structures 83*, 6-7 (2005), 479–490. 2

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *ACM Proc. Comp. Graphics and Interactive Techniques* (2001), pp. 23–30. 2

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graph. Models and Img. Proc. 58*, 5 (1996), 471–483. 2

[GB13] GERSZEWSKI D., BARGTEIL A. W.: Physics-based animation of large-scale splashing liquids. *ACM Transactions on Graphics 32*, 6 (2013), 185:1–185:6. 3

[GNS*12] GOLAS A., NARAIN R., SEWALL J., KRAJCEVSKI P.,

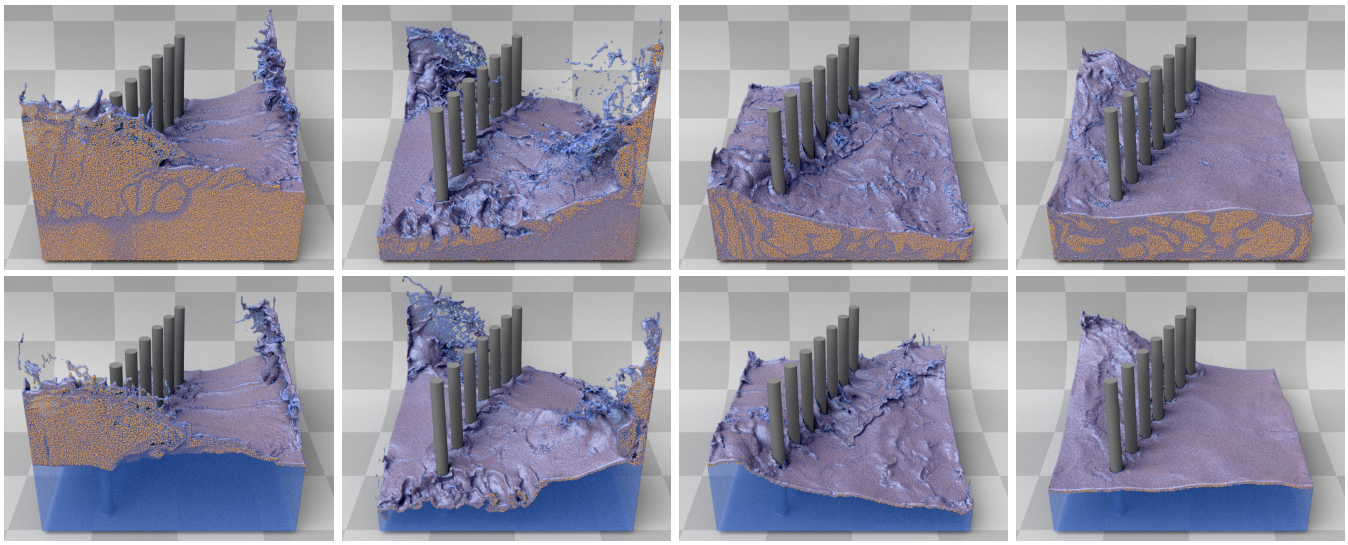


Figure 9: *Several frames of a breaking dam hitting a row of cylinders (top: regular FLIP, bottom: NB-FLIP).*

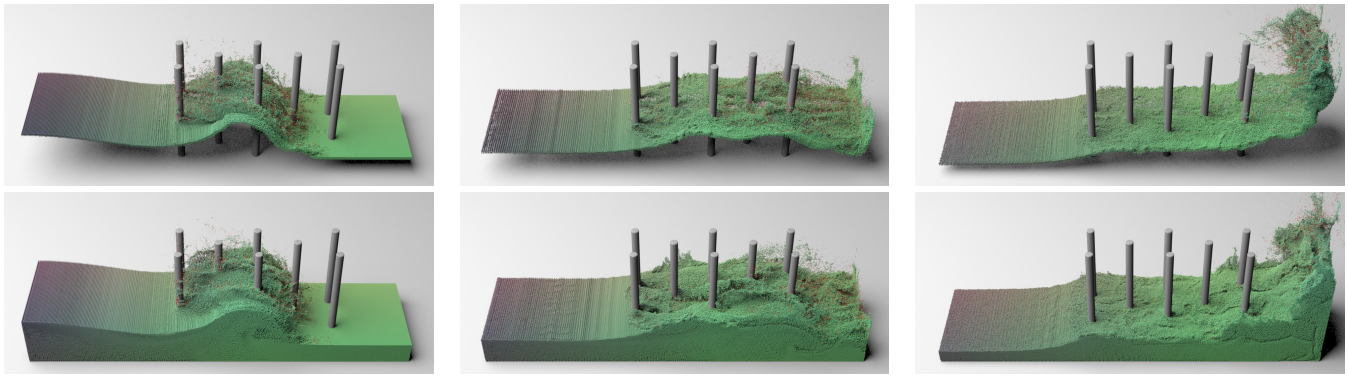


Figure 10: *Stills generated with a different implementation: the underlying FLIP solver performs significantly more calculations with the particles, leading to a speed-up for our narrow band approach of* 5.6×. *The FLIP particles are color coded according to their position.*

DUBEY P., LIN M.: Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Trans. Graph. 31*, 6(2012), 148:1–148:9. 2

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH fluids in computer graphics. In *Eurographics - State of the Art Reports* (2014), pp. 21–42. 2

[LTKF08] LOSASSO F., TALTON J., KWATRA N., FEDKIW R.: Two-way coupled SPH and particle level set fluid simulation. *IEEE Trans. on Visualization and Computer Graphics 14*, 4 (2008), 797–804. 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Symposium on Computer Animation* (2003), pp. 154–159. 2

[PHT*13] PAN Z., HUANG J., TONG Y., ZHENG C., BAO H.: Interactive localized liquid motion editing. *ACM Transactions on Graphics 32*, 6 (2013), 184:1–184:10. 2

[PT15] PFAFF T., THUEREY N.: MantaFlow, 2015. *http://mantaflow.ethz.ch*. 6

[RGJ*15] RAM D., GAST T., JIANG C., SCHROEDER C., STOMAKHIN A., TERAN J., KAVEHPOUR P.: A material point method for viscoelastic fluids, foams and sponges. In *Symposium on Computer Animation* (2015), pp. 157–163. 2

[RWT11] RAVEENDRAN K., WOJTAN C., TURK G.: Hybrid smoothed particle hydrodynamics. In *Symp. on Comp. Anim.* (2011), pp. 33–42. 2

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *ACM Transactions on Graphics 30*, 4 (2011), 81:1–81:8. 2

[SS06] SUÁREZ N., SUSÍN A.: A mesh-particle model for fluid animation. In *Ibero-American Symposium in Computer Graphics* (2006). 2

[SSC*13] STOMAKHIN A., SCHROEDER C., CHAI L., TERAN J., SELLE A.: A material point method for snow simulation. *ACM Transactions on Graphics 32*, 4 (2013), 102. 2

[Sta99] STAM J.: Stable fluids. In *Proceedings ACM SIGGRAPH* (1999), pp. 121–128. 2

[UBH14] UM K., BAEK S., HAN J.: Advanced hybrid particle-grid method with sub-grid particle correction. *Computer Graphics Forum 33*, 7 (Oct. 2014), 209–218. 2, 3

[YSB*15] YUE Y., SMITH B., BATTY C., ZHENG C., GRINSPUN E.: Continuum foam: A material point method for shear-dependent flows. *ACM Transactions on Graphics 34*, 5 (Nov. 2015), 160:1–160:20. 2

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Transactions on Graphics 24*, 3 (2005), 965–972. 2