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

Hybrid Smoothed Particle Hydrodynamics

Karthik Raveendran[1], Chris Wojtan[2]and Greg Turk[1]

[1]Georgia Institute of Technology
[2]IST Austria

Abstract

We present a new algorithm for enforcing incompressibility for Smoothed Particle Hydrodynamics (SPH) by preserving uniform density across the domain. We propose a hybrid method that uses a Poisson solve on a coarse grid to enforce a divergence free velocity field, followed by a local density correction of the particles. This avoids typical grid artifacts and maintains the Lagrangian nature of SPH by directly transferring pressures onto particles. Our method can be easily integrated with existing SPH techniques such as the incompressible PCISPH method as well as weakly compressible SPH by adding an additional force term. We show that this hybrid method accelerates convergence towards uniform density and permits a significantly larger time step compared to earlier approaches while producing similar results. We demonstrate our approach in a variety of scenarios with significant pressure gradients such as splashing liquids.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation.

1. Introduction

One of the challenges in particle based fluid animation is that of enforcing incompressibility. Incompressibility plays an important role in creating realistic animations of fluids such as water, where this property leads to the formation of important visual effects like splashes and dynamic turbulent motions. A popular Lagrangian fluid simulation technique is Smoothed Particle Hydrodynamics (SPH) which was originally designed to model compressible flow. In order to simulate incompressible fluids in SPH, researchers have proposed various techniques that either enforce a divergence free velocity field or preserve a uniform particle density. The most commonly used version of SPH (weakly compressible SPH) attempts to enforce uniform particle density by introducing a stiff equation of state, which imposes a severe timestep restriction on the simulation. As a consequence, the animator is forced to strike a balance between the timestep and the quality of the simulation. Other techniques that solve for a divergence free velocity on each particle usually involve an expensive Poisson solve on an unstructured grid which makes simulation of a large number of particles intractable.

We present a new hybrid particle-grid method (Hybrid SPH) to enforce incompressibility of a SPH fluid. Our method works with both weakly compressible SPH as well as incompressible PCISPH (where strict upper bounds are imposed on the density measured at each particle). We begin by sampling particle velocities on a coarse uniform grid and solve for pressure values that enforce a divergence free velocity

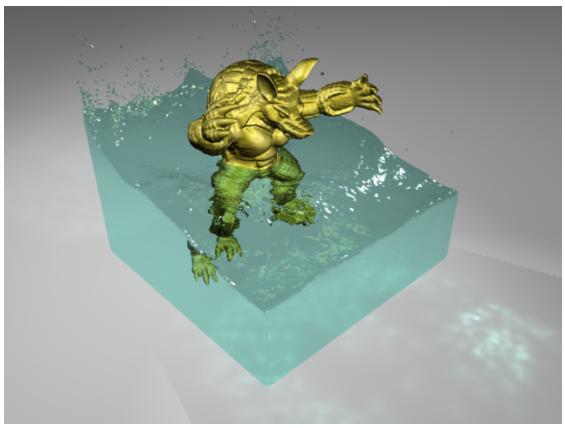

Figure 1: *Dam break around an armadillo, 200k particles. Simulated using our hybrid method together with PCISPH.*

field on this grid. However, unlike Eulerian simulations, we do not use these pressure values to compute forces on the grid faces but instead transfer these pressures directly onto the particles. This preserves the Lagrangian nature of the simulation and avoids the numerical dissipation associated with a coarse grid. Next, we compute forces on each particle using SPH formulations to obtain sub-grid detail. We finally ensure that each advected particle meets the density requirement by performing a local per-particle density correction with SPH pressure forces, like those from PCISPH or weakly compressible SPH.

Our method preserves high frequency detail while using a coarse grid, thus avoiding the high cost associated with a per-particle Poisson solve. We maintain the desired level of incompressibility by focusing the bulk of computational effort on only resolving compressions locally within a small subset of the entire domain. A key benefit of this method is that it is significantly faster than existing SPH techniques. It allows the animator to take much larger timesteps for simulation and relax the stiffness constant of the SPH pressure update equations without introducing compression artifacts. Further, the grid resolutions required for high quality simulations remain relatively low. For instance, a 40^3 grid with around 300 particles per cell can be used to simulate close to 2 million particles. Our algorithm is easy to implement, does not introduce any additional CFL restrictions, and can be integrated with existing versions of SPH simply by adding a low cost force calculation prior to the pressure force computation.

Our paper is structured as follows: in Section 2, we discuss related work and other approaches to incompressible fluid simulation. In Section 3, we give a brief overview of the mathematical equations used in SPH. In Section 4 we describe the details of algorithm, including how it is used together with either WCSPH or PCISPH. In Section 5, we compare hybrid SPH with existing methods and demonstrate that it not only outperforms them in common test cases but that it remains very similar in terms of the visual quality. We then discuss our method in Section 6 and give conclusions and future work in Section 7.

2. Related Work

One of the earliest works in fluid simulation for computer animation was the Eulerian grid-based method of Foster and Metaxas [FM96], which was later advanced by the introduction of semi-Lagrangian advection by Jos Stam [Sta99]. The Lagrangian fluid simulation technique of Smoothed Particle Hydrodynamics (SPH) was developed for use in astrophysics by [Luc77, GM77]. SPH was applied to graphics for the first time by Desbrun and Cani [DG96]. Müller [MCG03] demonstrated that SPH could be used to create fluid simulations with higher particle counts at interactive rates and laid the foundations for particle based approaches to animate various phenomena such as fluid-fluid interaction [MSKG05,SP08], solid-fluid coupling [MST*04, KAG*05,SSP07,BTT09], weakly compressible flow [BT07] and porous flow [LAD08]. Adams and colleagues introduced adaptive sampling in SPH simulations [APKG07]. SPH simulations have also been run at interactive rates on GPUs [HCM06, HKK07].

Much of the work mentioned in the last paragraph uses a compressible formulation of SPH in which a stiff equation of state is used to map densities to pressures. In order to prevent any violations of the CFL condition, the timestep must be decreased as the desired extent of incompressibility is increased. This restriction makes it difficult to run high resolution simulations within reasonable time limits. Further, the stiffness needs to be tuned by hand for each simulation, making it unfriendly for use by animators. Our hybrid approach (presented in Section 4) permits a lower stiffness constant at a larger timestep while preserving plausible fluid motion.

One way that is used to enforce incompressibility is to create a divergence-free vector field by solving a Poisson equation. For particle-based simulations, a Poisson equation can be formulated on a per-particle basis, leading to a linear solve involving an N $\times$ N matrix (where N is the number of particles in the simulation). Some techniques from computational physics literature take this approach, including [CR99, SL03, HA07]. In graphics, Idelsohn and colleagues [IODP04] solve for incompressibility on Delaunay triangulations, and Sin and colleagues [SBH09] use Voronoi regions. One drawback of this general approach is that the cost of an $N \times N$ linear solve is quite high. Also, such methods can still result in uneven particle densities, as can be seen in Figure 5(ii) of [CR99]. While Cummins and Rudman use a multigrid scheme based on regular grids to accelerate this Poisson solve on particles to compute a zero divergence field, we use a simple Poisson solve on a coarse grid to account for global effects followed by traditional SPH for local density corrections.

Other approaches towards incompressibility like PCISPH [SP09] and Incompressible SPH [ESE07] eschew the global Poisson solve and instead enforce a strict upper bound on the maximum level of compression. Both of these approaches are global iterative methods on particles for converging to rest density. The PCISPH approach allows for a signficantly larger timestep relative to compressible forms of SPH but at a higher cost per timestep. The beauty of the PCISPH method is that the predictor-corrector iterations propagate density information rapidly, and the final result is a simulation that maintains even particle spacing. For most fluid configurations, the speed of this approach depends on the maximum pressure inside the liquid, which is typically related to the depth of the liquid volume. Recent work by Bodin and colleagues [BLS11] also tackles the particle density problem by solving a mixed LCP and allows for larger time steps.

There have been numerous papers based on FLIP [ZB05, HHK08, LTKF08] that use an auxiliary grid to determine

particle motions. These methods prevent the particles from directly affecting each other like they do in SPH, and instead advect the particles after transferring velocities to and from the background grid. These particles only track sub-grid velocities and do not have a volume associated with them. The main strength of FLIP is that it is quite fast and still achieves stable non-dissipative advection. One drawback is that the particles can clump or have voids, making it difficult to create a surface for rendering. Similar to our own work, Losasso et al. [LTKF08] use a hybrid particle and grid-based simulator. Their approach uses particle level sets to represent dense liquid volumes and SPH for diffuse regions. In contrast, our approach is primarily based on SPH and just uses a low resolution grid to provide good initial values for particle pressures. We aim towards a uniform density while their method alters the Poisson solve to allow for multiple density targets across the domain.Their technique uses a combination of FLIP and PIC to transfer changes in velocity to the particles; these changes in velocity can be interpreted as *impulses* due to the pressure gradient integrated over one time step. In contrast, our method transfers the *actual pressure* to each particle.

The notion of treating hair as a continuum was introduced by Hadap et al. [HMT01]. A combination of Eulerian and Lagrangian approaches has also been used by McAdams et al. [MSW*09] in hair simulation in order to deal with bulk volume preservation and self-collisions efficiently.

Although the work of Lentine et al. [LZF10] is purely Eulerian, their work is related to ours in that they also make use of a coarse pressure solve to accelerate their simulator. They then refine this initial pressure field to gain back details at high resolutions.

3. Background

Smoothed Particle Hydrodynamics is an interpolation technique for particle systems that was first introduced in [Luc77, GM77] for simulating astrophysical problems. To apply this interpolation to a fluid, SPH divides the fluid into a set of small mass elements. The density at a point is then defined as follows:

$$\rho_i = m\sum_j W(r_{ij}) \tag{1}$$

where, W is a symmetric monotonic decreasing kernel that has a support (or smoothing radius) of h (which is usually around two times the default particle spacing) and r_{ij} is the distance between particle i and particle j.

SPH can be used to simulate weakly compressible fluids (WCSPH) by using a stiff equation of state:

$$P_i = \frac{k\rho_0}{\gamma}((\frac{\rho_i}{\rho_0})^\gamma - 1) \tag{2}$$

where P_i is the pressure at particle i, ρ_0 is a reference density and γ is a constant that is either set to 1 (gas equation) or 7 (Tait's equation). Typically, implementations choose a value for the constant k that results in a mean compression of 1 to 3 percent for a fixed simulation timestep (determined by the CFL condition).

The pressure force (in its symmetric form) on each particle can then be computed as follows:

$$F_{pressure} = -m^2\sum_j(\frac{P_i}{{\rho_i}^2} + \frac{P_j}{{\rho_j}^2})\nabla W(r_{ij}) \tag{3}$$

Weakly compressible implementations usually require hand-tuning of various parameters (such as the stiffness, the coefficient of viscosity, and the timestep) to obtain a well behaved fluid that does not appear to be bouncy. In contrast, if the stiff equation of state is replaced by a pressure solve that guarantees incompressibility, many of the described problems can be avoided, albeit at a higher computational cost per timestep. This typically involves either computing pressures for each particle such that the divergence of the resulting velocity is zero or using a method that ensures that the particles preserve uniform density.

In PCISPH [SP09], pressures are determined through a predictor-corrector approach so that the maximum density does not exceed a defined threshold. First, particle positions and velocities are saved at the start of each timestep. Then, at each iteration, those particles that have a density greater than the rest density have their pressures increased by performing following update:

$$P_i = P_i + \frac{\rho_0 - \rho_i}{\beta(-\sum_j \nabla W_{ij}\cdot\sum_j \nabla W_{ij} - \sum_j(\nabla W_{ij}\cdot\nabla W_{ij}))} \tag{4}$$

$$\beta = 2(\frac{m\Delta t}{\rho_0})^2 \tag{5}$$

These pressures are then used to compute pressure forces and predict the new positions and velocities of the particles. The iterations continue until each particle has achieved the desired density. An outline of this density correction for a single particle is illustrated in Procedure 1.

4. Hybrid SPH

In this section, we present the intuition behind our hybrid approach and give the details of its integration with two popular SPH simulation techniques. The well known pressure projection equation from fluid dynamics is an elliptic PDE,

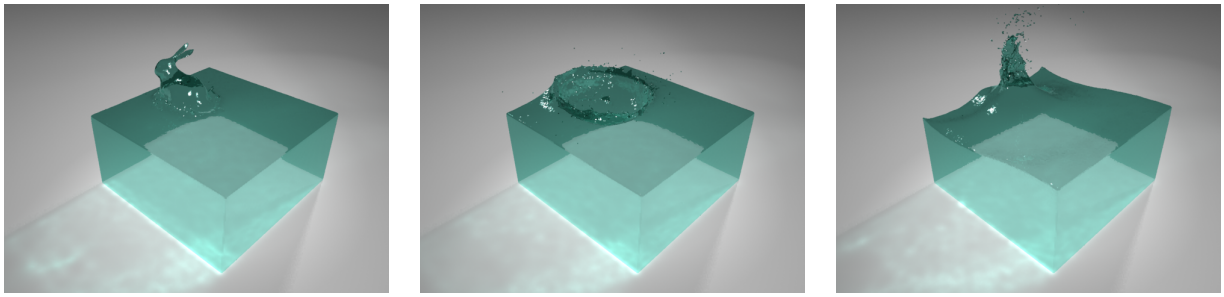

Figure 2: *A water bunny splashing into a pool, 350k particles. Simulated using our hybrid method together with PCISPH.*

Procedure 1 performDensityCorrection(i)

1: **if** $\rho_i > restDensity$ **then**
2: restore velocity $v(t)$ and position $x(t)$ of particle i
3: $p_i += correction$ (from Equation 4)
4: compute $F_{pressure}$
5: integrate F_{total} to obtain $\hat{v}(t+1), \hat{x}(t+1)$
6: predict new density ρ_i at particle i
7: **end if**

which implies that any change in boundary conditions must affect the entire domain instantaneously. A common property of both weakly compressible SPH and incompressible SPH is that a certain degree of incompressibility is enforced by updating pressure values and moving particles based on the force due to these assigned pressures. However, since each particle only affects a limited neighbourhood of other particles within a timestep, sharp changes in pressure do not get passed onto the entire volume immediately and instead create acoustic waves (where bands of particles get compressed in successive iterations or timesteps). The most noticeable artifact of this phenomenon is the rise and fall in the level of a standing pool of water (where the only external force on the system is due to gravity). In order to resolve this discrepancy, SPH simulations for graphical purposes usually increase the stiffness constant and lower the timestep until such waves are no longer apparent to the human eye. In the case of PCISPH, the iterative approach allows for a larger timestep than the weakly compressible version but it suffers from the same problem when a sufficiently large timestep is used (since the iterative pressure update is neither unconditionally stable nor is it guaranteed to converge).

Our hybrid approach removes this problem by transferring information to all particles instantaneously using a coarse grid. We solve for pressures that result in a divergence free velocity field on a low resolution grid (the sparse linear system can be solved quickly using conjugate gradient). This captures the effect of large scale forces acting on the simulation (for instance, gravity) and provides a good initial guess for each particle's position. Naturally, this coarse global solve is not aware of density fluctuations that have occurred inside a grid cell, but we can now use existing techniques mentioned in the previous paragraph to efficiently deal with local compressions. It is important to note that our hybrid simulations still contain sub-grid details in terms of particle velocities. After the solve, we only transfer pressures onto particles and compute the pressure force exerted by each particle on its neighbors by using the standard SPH formulation.

One can think of our hybrid step as a kind of preconditioner for the pressure solve. Technically, this would be a true preconditioner for SPH if the particles were first moved by the forces produced by the hybrid step and if the densities and neighbourhoods were computed at the new positions. However, this adds an additional cost to each step, which in our experience did not produce visually different results. So we avoid the additional density computation and simply add on the hybrid pressure force to the other forces computed using the density at the beginning of the timestep. Our hybrid step can be summarized as follows:

1. Interpolate particle velocities onto grid faces.
2. Classify cells as solid, fluid or air.
3. Construct linear system and solve Poisson equation.
4. Interpolate pressure values back onto particles.
5. Compute the hybrid pressure force between each pair of particles using SPH kernels.

4.1. Construction of the grid

The construction of our grid is similar to other particle-grid methods like FLIP. We use a coarse uniform staggered grid with a typical cell edge length that is 4 times the smoothing radius. This works out to around 512 particles per cell in 3D. Pressure values are stored at cell centers while components of velocity are stored on cell faces. We transfer velocities from particles onto grid faces by first finding the dual cell that the particle lies in and then assign tri-linearly weighted coefficients to each of the 8 corners of the cell based on the generalized barycentric coordinates of the particle within the

dual cell. Note that since there are three components of velocity in 3D, each of these dual grids will be staggered from the original grid (for pressure) by half a cell edge. We add the contribution of gravity to the velocities prior to the Poisson solve.

We use a slightly modified version of the technique presented in [BBB07] to solve the Poisson equation because it elegantly deals with partially filled cells. In order to construct the matrix for the Poisson solve, we need to classify cells as fluid, solid or air. However, unlike purely Eulerian methods or even FLIP, we can use the density information of particles to label cells more accurately and avoid artifacts during topological changes. We label a cell as fluid if it contains a minimum number of fluid particles above a threshold density, so as to avoid cells that largely consist of spray or diffused water particles. We set these parameters conservatively to 50% of the number of particles in a fully filled cell and 80% of the rest density, respectively for all our simulations. A cell is labelled as solid if its centre lies inside a solid. The remaining cells are classified as air.

4.2. Pressure projection and transfer

After classification, we populate the matrix with each row corresponding to a fluid cell or a solid cell that borders a fluid. The right hand side of the equation has the divergence of the cell which is computed by using a centered finite difference of the cell face velocities of that cell. We apply boundary conditions as necessary when the fluid shares a cell face with a solid or air cell. We use the ghost fluid technique [ENGF03] to obtain a second order accurate pressure value at the fluid-air interface. The constructed matrix is a sparse, symmetric positive definite system and can be solved quickly using conjugate gradient. Since this solve is meant to provide a good guess for pressure values, it is sufficient to terminate the iterative solve even when the residual has fairly high divergence (0.01) unlike conventional Eulerian methods. At this point, we have the pressure values at the centers of the cells in our grid.

We now interpolate pressures directly back onto the particles using tri-linear interpolation. This avoids the numerical viscosity associated with Eulerian methods in which velocities are computed on the grid and interpolated onto the particles. We compute a pressure force between each pair of particles using the standard SPH kernels(Equation 3). This retains the Lagrangian nature of our simulation and takes into account the density and the number of neighbours for each particle. Note that this is different from FLIP which interpolates the velocity difference from the grid, and is oblivious of the topology inside a grid cell

We require particles inside solids to acquire a pressure value if there is a fluid particle within their neighborhood. This helps to prevent fluid particles from getting pushed towards solid boundaries. We can easily compute the pressures for solid cells that share a face with a fluid cell using the boundary conditions of the Poisson solve. However, corner cells do not acquire a pressure value and this can cause an unnatural pressure drop-off due to tri-linear interpolation. We found that setting pressures of corner cells to the maximum of pressure values of their adjacent solid cells fixes this problem. This hybrid solve produces a smooth pressure profile that is typically associated with incompressible SPH and the resulting hybrid pressure forces can now be used as external forces in existing SPH implementations. We will now describe how these forces can be integrated with either the WCSPH or the PCISPH simulation method.

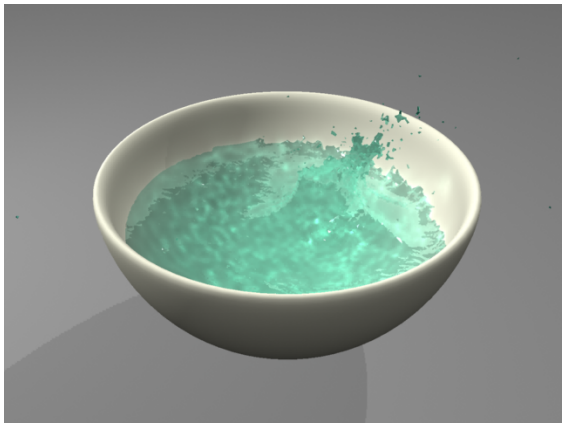

Figure 3: *Water pouring into a bowl, 160k particles. Simulated using our hybrid method together with WCSPH.*

Algorithm 2 Weakly Compressible SPH

1: **for all** i **do**
2: compute nearest neighbours within kernel radius.
3: compute density.
4: compute $F_{viscosity}, F_{gravity}, F_{external}$.
5: ***compute hybrid pressure force*** F_{hybrid}
6: compute $F_{pressure}$ using equation of state
7: Integrate forces to obtain new velocity and position.
8: **end for**

4.3. Weakly compressible SPH

Weakly compressible SPH implementations usually use either the gas equation or Tait's equation to restrict density fluctuations. Our method is compatible with both of these equations of state. Algorithm 2 describes our modified version of weakly compressible SPH. The hybrid step returns a force which we then integrate along with all the other typical forces in an SPH simulation. This simple addition allows for a significantly larger timestep and a smaller stiffness parameter. Further, we discovered that the hybrid approach

allows us to use the more compressible and computationally cheaper gas equation (does not require exponentiation) and yet achieve a plausible look for the animation. Note that the kernels for hybrid pressure forces and the standard SPH pressure force are identical and hence only need to be evaluated once per particle.

4.4. Incompressible SPH

We present further evidence regarding the efficacy of our hybrid method by applying it to the predictive-corrective SPH method introduced in [SP09]. We make two modifications to the PCISPH algorithm. First, we lower the minimum number of global iterations (from three to one) because the hybrid force has already accounted for the global density variation. The second change is that after performing these global iterations, we switch to a more local density correction where only a fraction of the total particles are affected. This can be viewed as a domain decomposition strategy for the pressure solve that identifies regions of particles that require further iterations to achieve convergence. The benefit is that computational power can be used efficiently (even in a serial implementation) to resolve the most compressed regions instead of iterating needlessly over the entire domain. In our implementation, we select particles based on a density cutoff ($\rho_{flagging}$), which is set to half the maximum allowed density variation. We found that this value was sufficient to

Algorithm 3 Hybrid ISPH

```
for all i do
    save current velocity v(t) and position x(t)
    compute neighbors
    compute density
    compute F_viscosity, F_gravity, F_external.
    compute hybrid pressure force F_hybrid
    integrate to obtain v̂(t+1), x̂(t+1)
    compute new density
end for
for j = 1 to minGlobalIterations do
    for all i do
        performDensityCorrection(i)
    end for
end for
for all i do
    if ρ_i > ρ_flagging then
        flag particle and its neighbours
    end if
end for
while ρ_max > maxDensityVariation do
    for all flagged particles i do
        performDensityCorrection(i)
        if ρ_i > maxDensityVariation then
            flag particle and its neighbors
        end if
    end for
end while
for all i do
    set v(t+1) = v̂(t+1)
    set x(t+1) = x̂(t+1)
end for
```

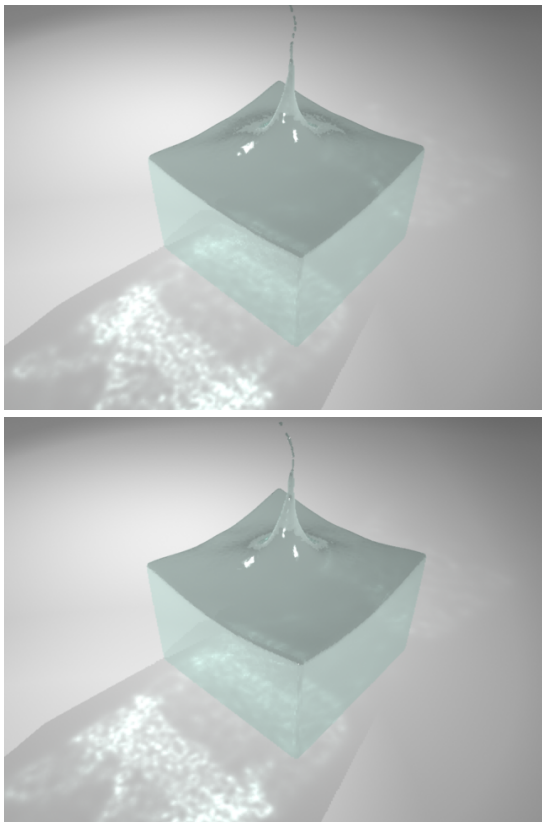

Figure 4: *Dropping a cube of water, 100k particles. Top version simulated using WCSPH; bottom version simulated using our hybrid method with WCSPH.*

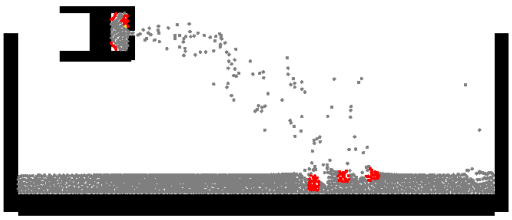

Figure 5: *A PCISPH simulation of water being squirted into a pool. After one global step, only the particles in red are flagged for further local iterations.*

prevent excessive communication between the flagged and unflagged regions for most scenarios. After the initial division of the domain, we iterate on only the flagged particles but allow for the possibility that the compression cannot be contained in the flagged region and instead propagates to the boundary. In such an event, we flag the particle that violates the density constraint as well as its neighbors and continue to perform density corrections until every flagged particle obeys the density limit (see red particles in Figure 5). We outline our modified algorithm in Algorithm 3.

4.5. Implementation Details

To create our animations, we use the cubic SPH weighting kernels from [Mon92] for all computations (density, pressure, viscosity).The support for the kernel is set to twice the particle distance so that each particle has around 27 to 30 neighbors in 3D. At the beginning of every timestep, we compute all neighbors that lie within the support of the kernel centred at each particle using a uniform grid to accelerate the neighborhood search. We treat boundaries by uniformly populating them with solid particles, and these particles also participate in the density correction process by exerting a pressure force on the fluid particles. We also deal with momentum transfer during collisions in a manner similar to [BTT09] instead of using repulsion forces, thereby avoiding additional CFL conditions. To improve the look of streams and splashes, we implemented the surface tension model described in [BT07] with the surface tension coefficient ranging between 0.1 and 0.5 in various examples. We use the surface reconstruction technique described in [YT10] to obtain meshes from our particle data. Our simulation uses adaptive timestepping to handle situations where pressure forces induce velocities that violate the CFL condition. In such cases, we drop the timestep by half and rollback to an earlier snapshot (two timesteps in the past). We used a dual core Intel i5 processor running at 2.52 GHz with hyperthreading enabled for the results presented in this paper. For a 100k particle simulation using our hybrid method with WCSPH, it took on average 12 seconds per frame, while a 200k particle simulation using our hybrid method with PCISPH took around 42 seconds per frame with 1% compression.

5. Results

We demonstrate our hybrid method for several different 3D fluid animations. Figure 1 shows a dam break in which the water flows around a solid armadillo. Figure 2 drops a bunny made of water into a still pool. These first two examples were created using our hybrid method together with PCISPH. Figure 3 shows water being poured into a bowl, and then a liquid armadillo being dropped into the same bowl. Figure 4 is an animation of a cube of water being dropped into a pool. These cube drop and bowl examples were simulated with WCSPH and with our hybrid method together with WCSPH. Note that there is little difference between the visual results of these methods, while our method was typically three or more times faster. Figure 6 shows fluid bunnies splashing against the letters SCA, and then the letters turn into liquid. This animation was using our hybrid version of WCSPH. Please see the accompanying video for the animations of all of these scenes.

In each of our examples, we are consistently able to take a much larger timestep due to the stability afforded by the hybrid solve. Table 1 shows the speedup of our method and the timestep used for each simulation. The pressures obtained from the Poisson solve on the coarse grid capture the large scale motion of the liquid and propagate this information to all particles.

In our accompanying video, we show the case of a 2D standing pool of water. In this example, our interpolated pressure values closely match the analytic solution and hence only local iterations are required to counter numerical errors. In contrast, both WCSPH and PCISPH are almost immediately unstable at the same timestep due to the large uncompensated compressions created due to gravity.

Our video also includes a 2D example of water splashing around a circular obstacle. This sequence demonstrates that our method gracefully handles non grid-aligned obstacles, despite the fact that this leads to cells that are partially filled.

6. Discussion

The serial implementation of our algorithm adds only 18% to the cost of a weakly compressible step and around 10% to the average PCISPH step (with 3 global iterations). Because we can take significantly larger timesteps than these two methods, the net effect is a large speedup. Moreover, all of the components of the hybrid step except for the Poisson solve are embarrassingly parallel. The latter can either be solved on the CPU without much computational effort because it is a small sparse symmetric positive definite matrix or could be run on the GPU if necessary [BFGS03]. We parallelized each of the algorithms using Intel's Thread Building Blocks. On average, the parallelized hybrid step adds around 23% to the cost of a weakly compressible step and around 15% to a PCISPH step. The unparallelized Poisson solve on the coarse grid took less than 7% of the computation time per time step (see Table 2). We observed near linear speedups while increasing the number of cores, which suggest that our hybrid approach is largely parallel. Higher grid resolutions could benefit from a preconditioner and parallelization of matrix-vector products.

It is worth noting the differences between our hybrid approach and other particle-grid techniques such as FLIP.

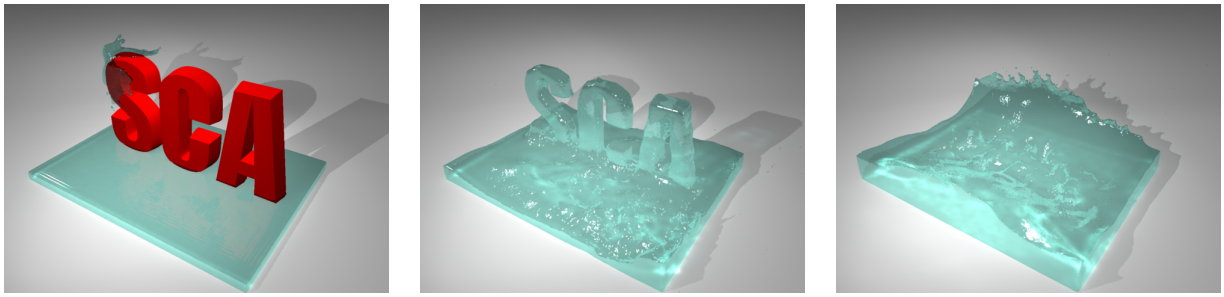

Figure 6: *Water bunnies splashing into the letters SCA, followed by the letters turning into liquid. This example was simulated using our hybrid version of WCSPH.*

Simulation	Particles	Grid Res.	Timestep			Simulation time (minutes)			Speedup
			Hybrid SPH	PCISPH	WCSPH	Hybrid SPH	PCISPH	WCSPH	
Bunny Splash	480k	$20 \times 28 \times 20$	0.0024	0.0006	-	326	1336	-	4.09
Armadillo	200k	$17 \times 21 \times 17$	0.0018	0.0006	-	281	952	-	3.39
Cube Splash	100k	$12 \times 14 \times 12$	0.0023	-	0.0005	46	-	140	3.04
Bowl	180k	$20 \times 21 \times 20$	0.0015	-	0.0004	154	-	417	2.71
SCA	200k	$32 \times 20 \times 25$	0.0018	-	-	146	-	-	-

Table 1: *Performance figures for our simulations.*

While FLIP uses particles to remove the numerical dissipation associated with a grid, the particles in our simulation actually represent Lagrangian fluid volumes and have a mass associated with them. Further, we can provide a guarantee on particle density and operate with extremely low resolution grids (with hundreds of particles per grid cell). In contrast, FLIP operates with 8 particles per grid cell in 3D and does not gain additional detail without increasing the grid resolution. Another advantage of using our method is that topological changes are handled accurately without any extra effort since our fundamental representation (SPH particles) is meshless. See Figure 7 for a comparison between our method and FLIP for a falling drop of water. In the FLIP image, the number of particles per grid cell has been increased to highlight the difference in the handling of the topology change. Typical FLIP implementations use 4 particles per grid cell for 2D simulations.

Step	2D (6k particles)	3D (100k particles)
Nearest neighbors	34.91	43.13
Rasterize particles	6.97	6.85
Matrix construction	4.65	0.09
Poisson solve	4.06	6.85
Interpolate pressure	2.90	0.97
Integrate forces	46.51	42.11

Table 2: *Breakdown of a hybrid WCSPH timestep (numbers denote percentage of a timestep)*

In terms of computational performance, FLIP is faster than our method because it does not need to perform a nearest neighbor lookup or calculate forces between pairs of particles, and instead spends the bulk of its effort on a higher resolution Poisson solve. In order to compare both methods, we used serial implementations of each. We ran a 2D simulation of a splashing water drop with 3300 particles. We chose a grid resolution of 64×64 for FLIP with around 4 particles per grid cell and picked the largest timestep (0.004s) that did not create obvious visual artifacts. Our serial hybrid SPH version could use a maximum timestep of 0.0025s. The FLIP version took 0.0068s on average for one frame while our method took 0.0225s on average (3.3 times slower than FLIP). When compared to our unoptimized implementation, FLIP can have 3x more particles for the same computation time. However, our method clearly enforces incompressibility, while FLIP exhibits artifacts when particles bunch together and spread too far apart. We believe that a parallelized version of our method would likely bridge the performance gap. Please see the accompanying video for equal particle count and equal computational cost comparisons between FLIP and our hybrid method.

There are a few limitations of our approach. Our hybrid method is most beneficial when the average depth of the fluid is at least a couple of grid cells. Since we use a larger timestep compared to a typical SPH simulation, high velocity collisions necessitate the use of adaptive timestepping for stable simulation. It might be also be helpful to exclude cells with high variance in particles velocities from the Poisson solve in such situations. Further, there may be occasions

where the topology seen by the coarse grid does not agree with that of the particles and this could lead to incorrect pressure values from the Poisson solve and possibly large forces leading to a drop in timestep. One may consider the use of octrees or finer grid resolutions to overcome this problem. In practice, these occur rarely and do not affect overall simulation time significantly.

7. Conclusion and Future Work

We have presented a new, fast method for achieving incompressible flow in an SPH fluid simulation. Our method uses a coarse Poisson solve on a grid to generate pressure forces that are then passed along to a standard SPH solver. We then handle the fine density corrections with either WCSPH or PCISPH.

There are several directions in which this work could be extended. First, we think that our hybrid solve could augment SPH solvers other than WCSPH and PCISPH. Another possibility is to try using our approach for a GPU-based implementation of SPH. Since solving Poisson equations on GPUs has been demonstrated by a number of researchers, we think this should be a straightforward task.

8. Acknowledgments

This work was funded in part by NSF grants IIS 1130934 and CCF 0811485. We thank our anonymous reviewers for their feedback and suggestions. We would also like to thank Jihun Yu for the valuable discussions on SPH and Jie Tan for his help in creating the video for this paper.

References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L.: Adaptively sampled particle fluids. *ACM Transactions on Graphics 26*, 3 (2007), 48. 2

[BBB07] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. 26*, 3 (2007), 100. 5

[BFGS03] BOLZ J., FARMER I., GRINSPUN E., SCHRÖODER P.: Sparse matrix solvers on the gpu: conjugate gradients and multigrid. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 917–924. 7

[BLS11] BODIN K., LACOURSIERE C., SERVIN M.: Constraint fluids. *IEEE Transactions on Visualization and Computer Graphics 99*, PrePrints (2011). 2

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (2007), p. 217. 2, 7

[BTT09] BECKER M., TESSENDORF H., TESCHNER M.: Direct forcing for Lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics 15*, 3 (2009), 493–503. 2, 7

[CR99] CUMMINS S., RUDMAN M.: An SPH projection method. *Journal of computational physics 152*, 2 (1999), 584–607. 2

[DG96] DESBRUN M., GASCUEL M.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation* (1996), vol. 96, Citeseer, pp. 61–76. 2

[ENGF03] ENRIGHT D., NGUYEN D., GIBOU F., FEDKIW R.: Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. *ASME Conference Proceedings 2003*, 36975 (2003), 337–342. 5

[ESE07] ELLERO M., SERRANO M., ESPAÑOL P.: Incompressible smoothed particle hydrodynamics. *Journal of Computational Physics 226*, 2 (2007), 1731–1752. 2

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graphical models and image processing 58*, 5 (1996), 471–483. 2

[GM77] GINGOLD R., MONAGHAN J.: Smoothed particle hydrodynamics- Theory and application to non-spherical stars. *Royal Astronomical Society, Monthly Notices 181* (1977), 375–389. 2, 3

[HA07] HU X., ADAMS N.: An incompressible multi-phase SPH method. *Journal of Computational Physics 227*, 1 (2007), 264–278. 2

[HCM06] HEGEMAN K., CARR N., MILLER G.: General Pur-

Figure 7: *Water drop comparison between FLIP (top image) and our approach (lower image). Note the unnatural bubble under the water drop in the FLIP version.*

pose Computation on Graphics Hardware (GPGPU): Methods, Algorithms and Applications-Particle-Based Fluid Simulation on the GPU. *Lecture Notes in Computer Science 3994* (2006), 228–235. 2

[HHK08] HONG W., HOUSE D., KEYSER J.: Adaptive particles for incompressible fluid simulation. *The Visual Computer 24*, 7 (2008), 535–543. 2

[HKK07] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics on GPUs. In *Computer Graphics International* (2007), Citeseer, pp. 63–70. 2

[HMT01] HADAP S., MAGNENAT-THALMANN N.: Modeling dynamic hair as a continuum. *Computer Graphics Forum 20*, 3 (2001), 329–338. 3

[IODP04] IDELSOHN S., ONATE E., DEL PIN F.: The particle finite element method: a powerful tool to solve incompressible flows with free-surfaces and breaking waves. *Int J Numer Methods Eng 61*, 7 (2004), 964–989. 2

[KAG*05] KEISER R., ADAMS B., GASSER D., BAZZI P., DUTRÉ P., GROSS M.: A unified Lagrangian approach to solid-fluid animation. In *Proceedings of the Eurographics Symposium on Point-Based Graphics* (2005), Citeseer, pp. 125–134. 2

[LAD08] LENAERTS T., ADAMS B., DUTRÉ P.: Porous flow in particle-based fluid simulations. *ACM Transactions on Graphics (TOG) 27*, 3 (2008), 49. 2

[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* (2008), 797–804. 2, 3

[Luc77] LUCY L.: A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal 82*, 12 (1977), 1013–1024. 2, 3

[LZF10] LENTINE M., ZHENG W., FEDKIW R.: A novel algorithm for incompressible flow using only a coarse grid projection. In *ACM SIGGRAPH 2010 papers* (2010), ACM, pp. 1–9. 3

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (2003), Eurographics Association, p. 159. 2

[Mon92] MONAGHAN J.: Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics 30*, 1 (1992), 543–574. 7

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM, p. 244. 2

[MST*04] MÜLLER M., SCHIRM S., TESCHNER M., HEIDELBERGER B., GROSS M.: Interaction of fluids with deformable solids. *Journal of Visualization and Computer Animation 15*, 3-4 (2004), 159–171. 2

[MSW*09] MCADAMS A., SELLE A., WARD K., SIFAKIS E., TERAN J.: Detail preserving continuum simulation of straight hair. *ACM Transactions on Graphics (TOG) 28*, 3 (2009), 62. 3

[SBH09] SIN F., BARGTEIL A. W., HODGINS J. K.: A point-based method for animating incompressible flow. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aug 2009). 2

[SL03] SHAO S., LO E.: Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Advances in Water Resources 26*, 7 (2003), 787–800. 2

[SP08] SOLENTHALER B., PAJAROLA R.: Density contrast SPH interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), pp. 211–218. 2

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Transactions on Graphics (TOG) 28*, 3 (2009), 40. 2, 3, 6

[SSP07] SOLENTHALER B., SCHLAFLI J., PAJAROLA R.: A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds 18*, 1 (2007), 69. 2

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

[YT10] YU J., TURK G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), Eurographics Association, pp. 217–225. 7

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. In *ACM SIGGRAPH 2005 Papers* (2005), ACM, p. 972. 2