A Point-based Method for Animating Incompressible Flow

Funshing Sin
Carnegie Mellon University
Pixar Animation Studios

Adam W. Bargteil
University of Utah

Jessica K. Hodgins
Carnegie Mellon University
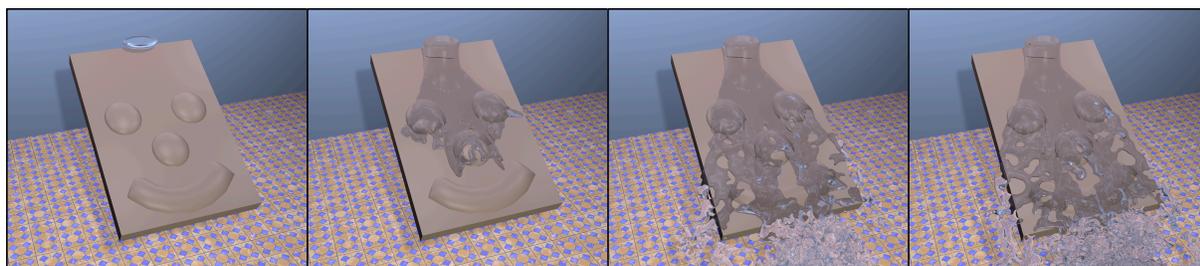
Figure 1: *Frames from a sequence where water is poured on a lumpy board.*

Abstract

In this paper, we present a point-based method for animating incompressible flow. The advection term is handled by moving the sample points through the flow in a Lagrangian fashion. However, unlike most previous approaches, the pressure term is handled by performing a projection onto a divergence-free field. To perform the pressure projection, we compute a Voronoi diagram with the sample points as input. Borrowing from Finite Volume Methods, we then invoke the divergence theorem and ensure that each Voronoi cell is divergence free. To handle complex boundary conditions, Voronoi cells are clipped against obstacle boundaries and free surfaces. The method is stable, flexible and combines many of the desirable features of point-based and grid-based methods. We demonstrate our approach on several examples of splashing and streaming liquid and swirling smoke.

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.
[Fluid simulation, incompressible flow, Voronoi diagram, point-based simulation, natural phenomena, physically based animation, computational fluid dynamics.]

1. Introduction

Over the last decade, fluid simulation has emerged as an important tool in computer animation. The seminal work of Foster and Metaxas [FM96] introduced Eulerian grid-based techniques for solving the Navier-Stokes equations that govern fluid flow. While grid-based techniques have remained popular, Lagrangian point-based techniques have been receiving increasing attention. With a few notable exceptions these techniques have been based on the Smooth Particle Hydrodynamics (SPH) methods originally developed for astrophysics applications [GM77]. SPH was sug-

gested for computer graphics applications by Desbrun and Cani [DC96] and Müller and colleagues [MCG03] demonstrated the method's utility for interactive fluid simulation.

One of the drawbacks of the SPH methods commonly used in graphics is that they model compressible flows. This limitation requires users to find the right balance between timestep, pressure force stiffness, and acceptable "bounciness."

In this paper, we present a point-based method for incompressible flow. While maintaining the Lagrangian solution to
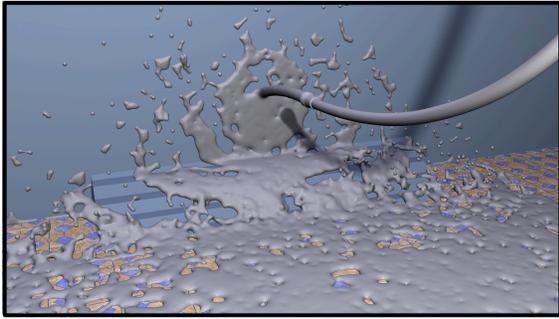
Figure 2: *Fluid is sprayed against a wall and falls down some stairs.*

the advection term, we include a Poisson solve to ensure that the velocity field is divergence free. The Poisson solve is carried out using a Voronoi diagram that is built every timestep using the points as the Voronoi sites. This Voronoi diagram is then clipped against both obstacle and free-surface boundaries, allowing for accurate handling of arbitrary boundaries. Additionally, moving least-squares methods are used to interpolate simulation variables stored at the sample points. Our method strikes a balance between grid-based and SPH methods allowing for incompressible animations of liquids while being able to represent small scale details and features. Moreover, our method avoids the numerical damping associated with semi-Lagrangian advection and handles hard, sharp boundary conditions in a unified way, without having to rely on boundary particles or other ad hoc solutions. Thus, our method is most competitive for animations of splashing and streaming liquids where grid-based methods are unable to resolve thin features and suffer from excessive damping and SPH methods can require small timesteps to avoid oscillations and large pressure forces. Our method is also able to animate swirling smoke with millions of trace particles without the numerical damping of semi-Lagrangian or Eulerian advection schemes or the oscillating velocity fields defined by SPH smoothing kernels.

2. Related Work

Foster and Metaxas [FM96] first demonstrated the ability of Eulerian grid-based methods to generate very compelling animations of fluids. Semi-Lagrangian advection [Sta99] notwithstanding, this Eulerian fixed-grid approach dominated computer graphics fluid simulation for nearly a decade. Although Desbrun and Cani [DC96] introduced SPH techniques for computer animation and Stora and colleagues [SAC*99] used SPH to animate lava flows, it was the work of Müller and colleagues [MCG03] and Premoze and colleagues [PTB*03] that demonstrated the viability of point-based Lagrangian methods for general fluid simulations. In particular, Müller and col-

leagues [MCG03] showed that SPH could produce compelling fluid simulations at interactive rates which led to a large body of follow-on work. Researchers have used SPH to model such phenomena as viscoelasticity [CBP05], solid-fluid coupling [MST*04, KAG*05, SSP07], fluid-fluid interaction [MSKG05, SP08], fluid control [TKPR06], rivers [KW06], weakly compressible flow [BT07], bubbling and frothing liquids [CPPK07], and porous flow [LAD08]. Adams and colleagues [APKG07] demonstrated adaptive sampling in SPH simulations and Hegeman and colleagues [HCM06] and Harada and colleagues [HKK07] demonstrated SPH simulations on GPUs. More recently, Zhang and colleagues [ZSP08] performed adaptive SPH simulations on the GPU.

While much of the work on point-based methods have focused on modeling compressible flows, there has been some work that has targeted incompressible fluids. Inspired by the work of Koshizuka and colleagues [KTO96], Premoze and colleagues [PTB*03] used the moving-particle semi-implicit method to simulate incompressible flow. Furthermore, in the computational physics literature, researchers have developed SPH methods that incorporate pressure projections [CR99, Sha05, CEL06]. As in our approach, these methods include a global pressure projection to ensure incompressibility, but they differ substantially in how the Poisson solve is carried out. While we use a Voronoi diagram to determine the interactions between particles, these methods perform the solve on the particles themselves and handle obstacles with special obstacle particles. In our experience, it is difficult to enforce hard, sharp boundary conditions with obstacle particles. Enforcing such sharp boundary conditions is our primary motivation for using the background Voronoi diagram. Hu and Adams [HA07] describe an approach that solves the Poisson equation on the particles, but handles the boundary conditions in the pressure projection. However, in their approach, standard von Neumann boundary conditions led to small fluctuations in particle pressure and, consequently, velocity. Researchers have used finite volume methods with Voronoi diagrams [TAK91, dVM04]. However, these approaches used fixed grids and different differential operators than those presented here. Yet another approach to incompressible SPH was suggested by Ellero and colleagues [ESE07] and relies on Lagrange multipliers. Most recently, Solenthaler and Pajarola [SP09] describe a method that uses a relaxation technique to achieve a target density, greatly reducing pressure oscillations in SPH simulations. While our approach is able to achieve a greater degree of incompressibility, their approach, by maintaining the number of particles, exactly preserves mass over time.

Until recently, one of the greatest advantages of point-based simulation techniques was their ability to handle complicated boundaries without the voxelization artifacts commonly found in grid-based techniques. However, the recent work on *cut cells* by Roble and colleagues [RbZF05] and the approach of Batty and colleagues [BBB07] seem to have
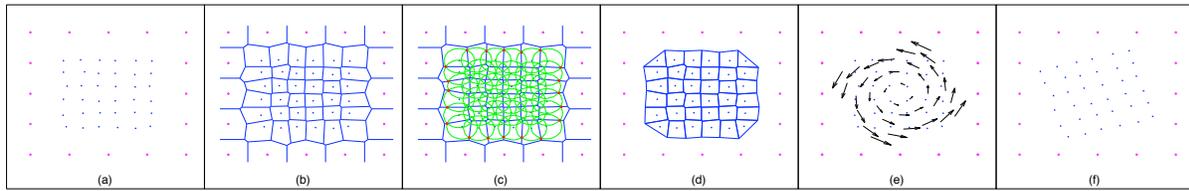
Figure 3: *An overview of our method: (a) We begin with a set of sample points that store velocity field samples. (b) We build a Voronoi diagram (the purple sites are added to ensure closed cells around sample points). (c) We define a free surface as the union of spheres around the sample points and (d) clip the Voronoi diagram against this surface. We then perform a pressure projection to generate (e) a divergence-free velocity field. Finally, we advect the sample points through the velocity field (f).*

improved the ability of grid-based techniques to handle complicated boundaries. Like our approach, cut cells leverage the divergence theorem to deal with arbitrary boundaries. Johansen and Colella [JC98] similarly used finite volume methods to handle irregular domains. However, their approach arrives at a non-symmetric system that is solved with multi-grid techniques. Losasso and colleagues [LGF04] also invoked the divergence theorem when developing the divergence operator for their octree simulation method.

To address several of the drawbacks of SPH, researchers have enhanced the method with moving least-squares techniques [Dil99]. Brownlee and colleagues [BHLR07] used both moving least-squares and radial basis functions. As pointed out by Belytschko and colleagues [BKO*96], SPH methods are not consistent. That is, SPH methods do not converge to the continuous solution as the timestep goes to zero and the number of particles goes to infinity. Moving least-squares methods are able to overcome this limitation. Initially, the goal of our project was to adapt such moving least-squares methods for computer animation. After our attempts to use boundary particles and incorporating boundary conditions into the moving least-squares solve [KT01] failed to handle sharp boundary conditions, we settled on using Voronoi diagrams and the divergence theorem. This approach has the additional advantage of producing a symmetric linear system, while moving least-squares based approaches do not.

Our work is also closely related to other approaches that use both particles and a background grid. Zhu and Bridson [ZB05] combined the hybrid fluid-implicit-particle (FLIP) [BR86] and particle-in-cell (PIC) [HW65] methods to animate sand as well as fluids. Losasso and colleagues [LTKF08] coupled a particle level-set grid-based method to a particle method that closely resembles FLIP and Hong and colleagues [HHK08] developed an adaptive FLIP method. In all these methods, a large number of particles are use to simulate the fluid. Unlike SPH, these particles do not interact directly, rather, they interact indirectly through a background grid. In this way, such methods leverage the excellent ability of Lagrangian methods to solve the advection term in the Navier-Stokes method, while us-

ing the background grid to enforce incompressibility. In its pure form, FLIP is largely unable to remove noise from the velocity field, while PIC is excessively damped. Thus, the two approaches are often combined, resulting in a balanced method. While our approach also solves the advection term by moving the particles and takes advantage of a background grid to enforce incompressibility, because we only have a single particle per background grid-cell, our particles have more direct interaction with each other. As in SPH, our pressure forces act on neighboring particles directly.

Our approach is also closely related to the work of Feldman and colleagues [FOK05, FOKG05] Klingner and colleagues [KFCO06], and Chentanez and colleagues [CFL*07]. These authors have explored the use of tetrahedral methods for fluid simulation and have allowed both deforming meshes and the complete construction of new meshes to adapt to changing boundary conditions. Our work also allows for moving points and requires the construction of a new Voronoi diagram every timestep. We also borrow some of their approximations, such as their approach to calculating gradients on faces in the mesh. Compared to methods that use regular grids, these tetrahedral approaches accept reduced accuracy in exchange for flexibility in mesh structure. Our approach makes a similar tradeoff. A primary advantage of our work over these approaches is that we are able to avoid the numerical damping caused by repeated interpolation during semi-Lagrangian advection. We also note that Elcott and colleagues [ETK*07] performed incompressible fluid simulations on unstructured grids. However, their approach did not involve a pressure projection. Also, related is the particle finite element method developed by Oñate and colleagues [OIPA04] to solve fluid-structure interaction problems. Like ours, their approach is particle based and builds a mesh every timestep to solve the fluid equations. However, they use the Delaunay triangulation and the finite element method while we use the Voronoi diagram and a finite volume approach.
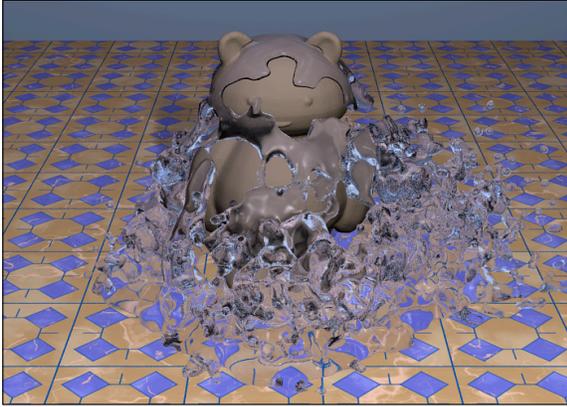
Figure 4: *Liquid is dropped on a model of a bear.*

3. Method

The Navier-Stokes equations model incompressible flow. Ignoring viscosity, we arrive at the Euler equations,

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla p \qquad (1)$$

$$\nabla \cdot \mathbf{u} = 0, \qquad (2)$$

where $\mathbf{u}$ is velocity, p is pressure, and ∇ denotes the gradient $(\partial/\partial x, \partial/\partial y, \partial/\partial z)^T$. As is commonly done in graphics we solve for the advection and pressure terms separately. Figure 3 gives an overview of our method. Given our sample points, we first build a Voronoi diagram. Second, we clip the Voronoi diagram against obstacles and the liquid surface. Third, we solve a Poisson equation to generate a divergence-free velocity field. Fourth, we advect the sample points through this velocity field. We now describe each of these steps in detail.

3.1. Computing the Voronoi Diagram

The Voronoi diagram is a well-studied decomposition of space. Given a set of *sites*, the Voronoi diagram decomposes space into a set of cells. All points inside a cell are nearer to the site in their cell than to any other site. In three dimensions, cell boundaries are composed of convex polygons usually termed *facets*. Facets meet at edges and edges meet at points.

We use Voronoi diagrams to divide space into regions belonging to a particular sample point. The Voronoi diagram is a natural choice because each point in a Voronoi cell will be nearer the sample point for its cell than any other sample point. Our system uses the qhull package [BDH96] to compute Voronoi diagrams. The input includes all of our sample points. To deal with our experience that some infinite facets may not be returned by qhull, we also include a number of points far away from the fluid domain to ensure that all Voronoi cells corresponding to sample points are closed (and the facets finite).

3.2. Clipping the Voronoi Diagram

One of the advantages of using the Voronoi diagram is that it allows us to handle complex boundaries by clipping the Voronoi cells against obstacle boundaries and free surfaces. The resulting decomposition is sometimes referred to as the restricted Voronoi diagram. To ensure that facets in the restricted diagram align with obstacle boundaries, cells are clipped against obstacles before being clipped against free surfaces.

The basic idea behind our clipping algorithm is to find intersections between obstacles or free surfaces and edges in the Voronoi diagrams. These intersection points are then connected to create a new closed cell. Therefore, we do not exactly represent obstacle geometry, but approximate it in a way similar to marching cubes [LC87]. While it should be possible to clip the Voronoi diagram against geometry of arbitrary complexity, this piecewise linear approximation is at the same scale as the simulation and has worked well in practice. After constructing the Voronoi diagram, each Voronoi vertex is examined to see whether it is inside an obstacle and accordingly marked. Then, all Voronoi cells are examined and any cell that has vertices inside an obstacle is clipped. The intersection with the obstacle is found along each edge that has one vertex inside the obstacle and one outside. These new vertices are then joined together. First vertices that share a facet are joined together in pairs. Then all vertices are joined to a new *average* vertex located at the average location of the new vertices. Occasionally, a boundary will pass through a Voronoi cell twice (i.e cutting off opposite corners). In this case, we introduce multiple average vertices at the average locations of new vertices in different connected components.

During simulation, the surface is represented as a union of spheres, as suggested by Batty [Bat08]. To simplify finding intersections of the surface with Voronoi edges, we use an implicit representation—a function that finds the distance to the nearest sample point and subtracts off a user-specified radius. While this function is not a signed-distance function (it may under estimate values inside the surface), it does give the correct sign and it's zero-set lies on the surface. While this simple representation is unacceptable for rendering, it is surprisingly effective at representing the surface during simulation and for clipping the Voronoi cells. It also avoids many of the problems that more advanced surface tracking methods encounter. Clipping against this surface follows the same procedure outlined above. A boundary case occurs when all the Voronoi vertices for a particular cell are outside the liquid. In this case, we let the sample point follow a ballistic trajectory with its velocity initialized by a constant moving least-squares fit.

3.3. Pressure Projection

Following the standard approach for incompressible fluids, we couple the pressure term to the divergence in Equation (2) and solve the resulting Poisson equation

$$\nabla^2 p = \nabla \cdot \mathbf{u}^*, \qquad (3)$$

where $\mathbf{u}^*$ is the velocity field after advection and applying external forces. Because our cells vary in volume and we wish to give cells with larger volume greater weight we multiply Equation (3) by the cell's volume, $V^{\dagger}$:

$$V\nabla^2 p = V\nabla \cdot \mathbf{u}^*. \qquad (4)$$

Solving Equation (4) requires that we define both Laplacian and divergence operators.

The Laplacian of the pressure is computed in a two step process. First pressure gradients are computed on all Voronoi facets. Following Feldman and colleagues [FOK05], only the pressure gradient component that is normal to the facet is computed. Thus for a facet between points i and j we have

$$(\hat{\mathbf{n}} \cdot \nabla p) = \frac{p_j - p_i}{\|\mathbf{x}_j - \mathbf{x}_i\| + \varepsilon}, \qquad (5)$$

where $\hat{\mathbf{n}}$ is the unit normal of the facet and p_i and p_j are the pressures at points i and j, respectively. The ε in the denominator avoids numerical problems when sample points may be near the Voronoi facet. Of course, it is possible that the line between sample points i and j does not pass through the facet between the associated cells, but we have not found this to be a problem in practice.

We then assume that the normal component of the pressure is constant over the facet and apply the divergence theorem to get

$$V\nabla^2 p = \sum_{f \in \text{facets}} a_f (\hat{\mathbf{n}} \cdot \nabla p), \qquad (6)$$

where a_f is the area of the facet. Conveniently, this formulation leads to a sparse, symmetric linear system that is solved to a user-specified tolerance using a conjugate gradient method without ever explicitly constructing the matrix. Our current implementation uses a Jacobi preconditioner for its simplicity, though a better preconditioner or solution method would likely lead to improved performance. We ensure that our system stays well conditioned by removing particles that get closer than a user-specified threshold and by disallowing interactions between particles that are very far apart (in terms of the radius used to define the free surface).

To compute the divergence of the velocity field, first a constant moving least-squares fit of the velocity field (see Equation (12)) is performed at the centroid of every Voronoi facet. We then assume that this velocity is constant

over the facet and again invoke the divergence theorem to get the volume-weighted divergence over the Voronoi cell.

$$V\nabla \cdot \mathbf{u} = \sum_{f \in \text{facets}} a_f (\hat{\mathbf{n}} \cdot \mathbf{u}). \qquad (7)$$

After solving for the pressure field, we need to compute its gradient to update our intermediate velocities, $\mathbf{u}^*$. The pressure field, like the velocity, is stored at the sample points. However, in this case we need the entire gradient not just the divergence so we cannot simply interpolate to facet centroids and apply the divergence theorem. Instead a linear moving least-squares fit to the pressure field is performed at each sample point and the gradient is given by the linear terms. More specifically, for sample point $\mathbf{x}_i$, we let,

$$\mathbf{X}_i = \begin{bmatrix} w_1 & w_1 x_1 & w_1 y_1 & w_1 z_1 \\ w_2 & w_2 x_2 & w_2 y_2 & w_2 z_2 \\ & & \vdots & \\ w_k & w_k x_k & w_k y_k & w_k z_k \end{bmatrix}, \qquad (8)$$

where $w_j = w(\mathbf{x}_j - \mathbf{x}_i)$ and $\mathbf{x}_j = (x_j, y_j, z_j)^T$. $\mathbf{X}_i$ is a matrix of neighbor's positions weighted by an appropriate kernel (see Equation (13)). We then solve for $(p_0, p_x, p_y, p_z)^T$ in the normal equations,

$$\mathbf{X}_i^T \mathbf{X}_i \begin{bmatrix} p_0 \\ p_x \\ p_y \\ p_z \end{bmatrix} = \mathbf{X}_i^T \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_k \end{bmatrix}, \qquad (9)$$

Solving Equation (9) only requires inverting a 4x4 matrix. The gradient is then extracted,

$$\nabla p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}. \qquad (10)$$

For the boundary facets, we apply the usual free-slip boundary conditions. Obstacle facets have $(\hat{\mathbf{n}} \cdot \nabla p) = 0$. This condition is enforced by setting $(\hat{\mathbf{n}} \cdot \nabla p) = 0$ on obstacle facets when computing $\nabla^2 p$. Additionally, obstacle facets are assigned the velocity of the obstacle instead of the usual moving least-squares estimate when computing divergence. To enforce $p = p_a$ on the air side of free surface facets, $(\hat{\mathbf{n}} \cdot \nabla p)$ is computed on air facets as,

$$(\hat{\mathbf{n}} \cdot \nabla p) = \frac{p_i - p_a}{2(\mathbf{x}_i - \mathbf{x}_f) \cdot \hat{\mathbf{n}}}, \qquad (11)$$

where p_i is the pressure of the sample point on the liquid side of the facet, p_a is atmospheric pressure, and $\mathbf{x}_f$ is the position of the facet centroid. Finally, the condition that $\nabla \mathbf{u} \cdot \hat{\mathbf{n}} = 0$ on air facets is enforced automatically because the moving least-squares estimate only uses points inside the liquid.

$\dagger$ While scaling a single row of the linear system would have no effect if we were performing an exact solve, it does give the row greater weight in our iterative solution.
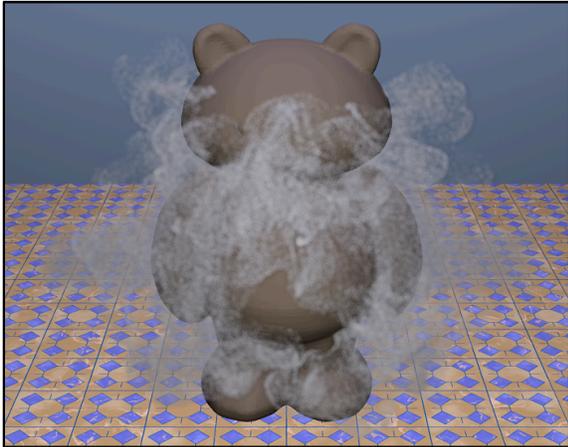
Figure 5: *Smoke being blown on a bear model.*

Figure 7: *A frame from a splash sequence.*

3.4. Advection

The advection term in Equation (1) is handled in our system by moving the sample points through the flow. Although the velocity field is sampled at these points, unlike most SPH implementations, we do not move them with these velocities. Instead, we perform a constant moving least-squares fit to the velocities sampled at nearby points. We take this approach so that the particles move through a velocity field that is continuously defined over all of space, without any unnecessary fluctuations. We can use this velocity field to perform second-order advection or to trace marker particles for animating smoke. While this approach does introduce some damping compared to moving the particles with their sampled velocity, it is important to note that the particle's stored velocity is not affected. Thus unlike semi-Lagrangian techniques, the smoothing does not build up over time. We also note that many other methods use smoothed velocities for advection (e.g. FLIP). More specifically, the velocity we use to advect the point, $\hat{\mathbf{u}}$, is computed as

$$\hat{\mathbf{u}} = \frac{\sum_j \mathbf{u}_j \cdot w(\mathbf{x}_i - \mathbf{x}_j)}{\sum_j w(\mathbf{x}_i - \mathbf{x}_j)}, \qquad (12)$$

where $\mathbf{u}_j$ is the velocity stored at point j, $\mathbf{x}_j$ is the position of point j, and $w(\cdot)$ is a reasonable weighting function with compact support. In our implementation, we use the quartic spline suggested by Belytschko and colleagues [BKO*96]:

$$w(\bar{s}) = \begin{cases} 1 - 6\bar{s}^2 + 8\bar{s}^3 - 3\bar{s}^4 & \text{for } \bar{s} \le 1, \\ 0 & \text{for } \bar{s} > 1. \end{cases} \qquad (13)$$

Our implementation uses a kd-tree to look up the nearest n neighbors, subject to a maximum distance. This approach, which is also employed in the open source SPH code written by Adams and colleagues [APKG07], avoids the dramatically varying numbers of neighbors that occur if a constant lookup radius is used. Such variations require large amounts of computation for some particles to ensure that other particles have enough neighbors to make reasonable estimates. Distances provided to the weighting function are then normalized by the distance to the farthest neighbor. To avoid interactions between disconnected regions of fluid we only consider particles to be neighbors if they are in the same connected component of the Voronoi diagram. Two Voronoi cells are in the same connected component if there is a path between them that only passes through original Voronoi facets (i.e. the path does not pass through a facet inserted when clipping against obstacles or the free surface). This restriction avoids some artifacts when distinct bits of fluid approach each other and is applied not just during advection, but during all neighborhood lookups. Finally, note that particles always have at least one neighbor—themselves. With only one neighbor, the particle will follow a ballistic path. Particles that enter an obstacle are projected onto the obstacle's surface.

4. Results

Figure 1 shows frames from an animation of fast moving liquid being poured on a lumpy board. Our method is able to resolve the resulting thin sheets and high velocity motion while taking only four timesteps per frame. Figure 2 shows a similar example where a fast moving liquid is sprayed at a wall and then runs down some stairs. Both these examples would be difficult to create with standard fluid simulation techniques. Figure 4 shows liquid interacting with a bear model. This example tests our ability to clip the Voronoi diagram against a complex boundary and maintain a divergence-free velocity field. Figure 5 shows a frame from an animation where smoke is blown on a bear model. Millions of trace particles were advected through the velocty field for rendering, demonstrating the smoothness of the
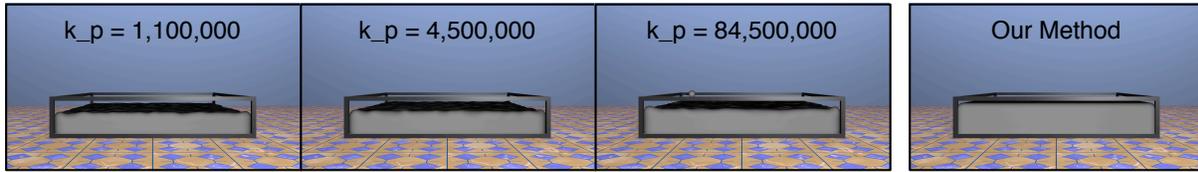
Figure 6: *Comparing our approach to SPH. SPH is sensitive to the choice of pressure constant and must settle into a lattice structure. Our method does not require tuning a pressure constant and is not particularly sensitive to the configuration of the sample points. With SPH, different pressure constants lead to different amounts of resulting volume. In contrast, our approach maintains the initial volume.*

Figure	Timestep(ms)	Particles	Sec/Step
Fig. 1	8.33	34,462	70
Fig. 2	8.33	32,952	30
Fig. 4	4.16	13,078	6
Fig. 5	4.16	35,342	17
Fig. 7	4.16	100,671	70
Fig. 6 (far right)	4.00	12,152	8

Table 1: *Timing results for the examples in this paper.*

MLS velocity field. Figure 7 shows a frame from a splash simulated with our method. In this example enforcing incompressibility keeps the liquid from demonstrating pressure oscillations. Furthermore, due to the low level of numerical damping the liquid remains active over long periods of time. Finally, we compare our approach to the open source SPH code written by Adams and colleagues [APKG07] (see Figure 6). We test three different pressure constants and note that each leads to varying amounts of volume loss and requires time to settle, while our method resists gravity and remains motionless.

Timing results for our examples are summarized in Table 1. Because the computation time varies substantially over the coarse of an animation, we report the time required for a particular, representative step. For example, in the animation of the lumpy board, as particles are added above the board, particles far below the board are deleted. The timing information is for a timestep after the number of particles has stabilized. As an optimization, when particles are far from other particles and unlikely to have significant interactions, they simply follow ballistic trajectories. This optimization explains the difference in speed between Figure 1 and Figure 2. In the comparison with SPH, our method took timesteps twenty times larger than the SPH simulator. However, the SPH simulator's steps were much faster and, overall, it generated frames eight times faster than our simulator.

We performed some rough profiling to determine the most expensive components of our simulator. Computing and clipping the Voronoi diagram was particularly expensive, requiring between 42% and 87% of the computation time, depending on the example. The splash and smoky bear examples spent smaller percentages of their computation time

computing the restricted Voronoi diagram and examples like the board and spray required larger percentages. There was also significant variance in the proportion of time spent constructing the diagram versus clipping the diagram, but construction usually dominated. Not surprisingly, solving the Poisson equation generally took about 10% of the computation time. Perhaps surprisingly, computing divergence often takes more time than solving the Poisson equation. These slow divergence calculations appear to be due to overuse of pointers and a consequent lack of cache coherence.

During simulation, we represent the liquid surfaces as a union of spheres, however this surface is unacceptable for rendering convincing animations. Generating surfaces from particle data is an open and difficult problem and a number of authors have proposed solutions including Blinn [Bli82], Müller and colleagues [MCG03], Premoze and colleagues [PTB*03], Zhu and Bridson [ZB05], and Adams and colleagues [APKG07]. In this work we used an implementation of the method proposed by Williams [Wil08]. While this approach works quite well for fast moving liquids, when the liquid is stationary it tends to introduce high-frequency temporal flickering of the surface (that is exacerbated by reflection maps). For this reason, a level-set based variation of this technique was used for the animations depicted in Figure 6.

5. Discussion

There are many avenues for future work. Many of the enhancements that have been applied to SPH methods, such as adaptive sampling, two-way coupling with rigid and deformable bodies, multiphase flow, and viscoelasticity remain to be explored. Among these, adaptive sampling seems the most useful. Our current implementation performs only very limited adaptive sampling, removing sample points that are very close together to maintain a well-conditioned linear system for the Poisson solve and avoid other numerical problems. Consequently, we probably cannot run arbitrarily long simulations with our current implementation. We believe a method similar to that developed by Adams and colleagues [APKG07] could be used in our system. Another interesting avenue for future work would be to avoid kd-tree queries all together and use the Voronoi diagram to find

nearby neighbors. We would also like to explore kinetic Delaunay/Voronoi methods (e.g. [SMH04]) as these may provide substantial performance improvements. It would also be interesting to consider replacing our moving least squares estimates with barycentric interpolation on the Delaunay triangulation—the dual of our Voronoi diagram. Another interesting area of future work would be enforcing no-slip boundary conditions. One approach would be to introduce ghost particles at the center of boundary facets with zero velocity. Alternatively, we could set the velocity of particles with boundary facets to zero.

As noted in Section 3.3 our pressures and velocities are collocated at the sample points. It is well known that collocation can lead to numerical problems in methods that employ regular grids. The non-regularity of our sample points probably serves to mitigate these problems, but if they were to become noticeable we could switch to an upwinding scheme or introduce an explicit filtering step.

We note that the number of particles is far below the number of particles typically found in FLIP and SPH simulations and even lower than the number of cells in a typical hexahedral grid simulation. Part of the reason for this disparity is computation time. While in FLIP particles do not interact directly and in SPH particles interact only in a local neighborhood, in our simulations a global linear system is solved for all particles, at much greater cost per particle. Second, though our particles are computationally more expensive, in our experience less of them are necessary than in other techniques. For example, high resolution hexahedral grids are often used to avoid numerical damping in the advection step, but our advection step already has very little numerical smoothing. While adding more particles provides more detail, in our experiments with greater numbers of particles, the improvement in visual quality did not seem to justify the larger computation times.

We believe that our method offers some appealing advantages over SPH and grid-based methods. We've already mentioned that our approach can model incompressible fluids, while SPH is limited to compressible flow. Additionally, most SPH implementations in graphics use different smoothing kernels for different terms in the Navier-Stokes equations, implying that the choice of kernel has a significant impact on the method. In contrast, our approach uses only a single kernel for all scattered data interpolation and is not particularly sensitive to this choice. Furthermore, we note that the SPH kernels oscillate between sample points, preventing such kernels from accurately representing constant velocity fields away from the sample points. On the other hand, moving least-squares methods are able to represent arbitrary polynomial vector fields. In comparison to grid-based methods, our approach offers greater flexibility in representing boundary conditions, the ability to track thin features using Lagrangian sample points, and avoids the numerical damping associated with semi-Lagrangian advection.

Acknowledgments

References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. Graph. 26*, 3 (2007), 48.

[Bat08] BATTY C.:. Personal Communication, 2008.

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

[BDH96] BARBER C. B., DOBKIN D. P., HUHDANPAA H.: The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw. 22*, 4 (1996), 469–483.

[BHLR07] BROWNLEE R. A., HOUSTON P., LEVESLEY J., ROSSWOG S.: Enhancing sph using moving least-squares and radial basis functions. *Algorithms for Approximation* (2007), 103—112.

[BKO*96] BELYTSCHKO T., KRONGAUZ Y., ORGAN D., FLEMING M., KRYSL P.: Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering 139*, 1-4 (1996), 3–47.

[Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3 (1982), 235–256.

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

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *The Proceedings of the Symposium on Computer Animation* (2007), pp. 209–217.

[CBP05] CLAVET S., BEAUDOIN P., POULIN P.: Particle-based viscoelastic fluid simulation. In *The Proccedings of the Symposium on Computer Animation* (2005), pp. 219–228.

[CEL06] COLIN F., EGLI R., LIN F. Y.: Computing a null divergence velocity field using smoothed particle hydrodynamics. *J. Comput. Phys. 217*, 2 (2006), 680–692.

[CFL*07] CHENTANEZ N., FELDMAN B. E., LABELLE F., O'BRIEN J. F., SHEWCHUK J. R.: Liquid simulation on lattice-based tetrahedral meshes. In *The Proceedings of the Symposium on Computer Animation* (2007), pp. 219–228.

[CPPK07] CLEARY P. W., PYO S. H., PRAKASH M., KOO B. K.: Bubbling and frothing liquids. *ACM Trans. Graph. 26*, 3 (2007), 97.

[CR99] CUMMINS S. J., RUDMAN M.: An sph projection method. *J. Comput. Phys. 152*, 2 (1999), 584–607.

[DC96] DESBRUN M., CANI M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation 1996* (1996), pp. 61–76.

[Dil99] DILTS G. A.: Moving-least-squares-particle hydrodynamis. i. consistency and stability. *Int. J. Numer. Meth. Engrg. 48*, 10 (1999), 1115–1155.

[dVM04] DE VASCONCELLOS J. F. V., MALISKA C. R.: A finite-volume method based on voronoi discretization for fluid flow problems. *Numerical Heat Transfer Part B: Fundamentals 45*, 4 (2004), 319–342.

[ESE07] ELLERO M., SERRANO M., ESPAÑOL P.: Incompressible smoothed particle hydrodynamics. *J. Comp. Phys. 226*, 2 (2007), 1731–1752.

[ETK*07] ELCOTT S., TONG Y., KANSO E., SCHRÖDER P., DESBRUN M.: Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26*, 1 (2007), 4.

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. In *The Proceedings of Graphics Interface* (1996), pp. 204–212.

[FOK05] FELDMAN B. E., O'BRIEN J. F., KLINGNER B. M.: Animating gases with hybrid meshes. *ACM Trans. Graph. 24*, 3 (2005), 904–909.

[FOKG05] FELDMAN B. E., O'BRIEN J. F., KLINGNER B. M., GOKTEKIN T. G.: Fluids in deforming meshes. In *The Proceedings of the Symposium on Computer Animation* (2005).

[GM77] GINGOLD R., MONAGHAN J.: Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon. Not. Roy. Astron. Soc. 181* (1977), 375–389.

[HA07] HU X. Y., ADAMS N. A.: An incompressible multiphase sph method. *J. Comput. Phys. 227*, 1 (2007), 264–278.

[HCM06] HEGEMAN K., CARR N. A., , MILLER G. S.: Particle-based fluid simulation on the gpu. In *Computational Science – ICCS* (2006), vol. 3994 of *LNCS*, pp. 228–235.

[HHK08] HONG W., HOUSE D. H., KEYSER J.: Adaptive particles for incompressible fluid simulation. *Vis. Comput. 24*, 7 (2008), 535–543.

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

[HW65] HARLOW F., WELCH J.: Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *The Physics of Fluids 8* (1965), 2182–2189.

[JC98] JOHANSEN H., COLELLA P.: A cartesian grid embedded boundary method for poisson's equation on irregular domains. *J. Comput. Phys. 147*, 1 (1998), 60–85.

[KAG*05] KEISER R., ADAMS B., GASSER D., BAZZI P., DUTRÉ P., GROSS M.: A unified Lagrangian approach to solid-fluid animation. In *The Proceedings of the Symposium on Point-Based Graphics* (2005), pp. 125–133.

[KFCO06] KLINGNER B. M., FELDMAN B. E., CHENTANEZ N., O'BRIEN J. F.: Fluid animation with dynamic meshes. *ACM Trans. Graph. 25*, 3 (2006), 820–825.

[KT01] KUHNERT J., TIWARI S.: Grid free method for solving the poisson equation.

[KTO96] KOSHIZUKA S., TAMAKO H., OKA Y.: A particle method for incompressible viscous flow with fluid fragmentation. *Comp. Fluid Dynamics J. 29*, 4 (1996), 29–46.

[KW06] KIPFER P., WESTERMANN R.: Realistic and interactive simulation of rivers. In *The Proceedings of Graphics Interface* (2006), pp. 41–48.

[LAD08] LENAERTS T., ADAMS B., DUTRÉ P.: Porous flow in particle-based fluid simulations. *ACM Trans. Graph. 27*, 3 (2008).

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), ACM Press, pp. 163–169.

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. In *Proceedings of ACM SIGGRAPH 2004* (2004), ACM Press, pp. 457–462.

[LTKF08] LOSASSO F., TALTON J. O., KWATRA N., FEDKIW R.: Two-way coupled SPH and particle level set fluid simulation. *IEEE Trans. Vis. Comp. Graph. 14*, 4 (2008).

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

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. In *The Proceedings of the Symposium on Computer Animation* (2005), pp. 237–244.

[MST*04] MÜLLER M., SCHIRM S., TESCHNER M., HEIDELBERGER B., GROSS M.: Interaction of fluids with deformable solids: Research articles. *Comput. Animat. Virtual Worlds 15*, 3-4 (2004), 159–171.

[OIPA04] OÑATE E., IDELSOHN S., PIN F. D., AUBRY R.: The particle finite element method. an overview. *Internation Journal of Computational Methods 1*, 2 (2004), 267–307.

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

[RbZF05] ROBLE D., BIN ZAFAR N., FALT H.: Cartesian grid fluid simulation with irregular boundary voxels. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches* (New York, NY, USA, 2005), ACM, p. 138.

[SAC*99] STORA D., AGLIATI P.-O., CANI M.-P., NEYRET F., GASCUEL J.-D.: Animating lava flows. In *The Proceedings of Graphics Interface* (1999), pp. 203–210.

[Sha05] SHAO S.: Incompressible sph simulation of wave breaking and overtopping with turbulence modelling. *International Journal for Numerical Methods in Fluids 50*, 5 (2005), 597–621.

[SMH04] SCHALLER G., MEYER-HERMANN M.: Kinetic and dynamic delaunay tetrahedralizations in three dimensions. *Computer Physics Communications 162* (2004), 9.

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

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible sph. *ACM Trans. Graph. 28*, 3 (2009).

[SSP07] SOLENTHALER B., SCHLÄFLI J., PAJAROLA R.: A unified particle model for fluid-solid interactions. *Journal of Visualization and Computer Animation 18*, 1 (2007), 69–82.

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

[TAK91] TANIGUCHI N., ARAKAWA C., KOBAYASHI T.: Construction of a flow-simulating method with finite volume based on a Voronoi diagram. *JSME International Journal 34* (Feb. 1991), 18–23.

[TKPR06] THÜREY N., KEISER R., PAULY M., RÜDE U.: Detail-preserving fluid control. In *The Proceedings of the Symposium on Computer Animation* (2006), pp. 7–12.

[Wil08] WILLIAMS B.: *Fluid Surface Reconstruction from Particles*. Master's thesis, University of British Columbia, 2008.

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

[ZSP08] ZHANG Y., SOLENTHALER B., PAJAROLA R.: Adaptive sampling and rendering of fluids on the gpu. In *The Proceedings of the Symposium on Point-Based Graphics* (2008), pp. 137–146.