PBF Method for Fluid Simulation

Ruiqi Wang
University of Toronto
Toronto, ON, Canada
ruiqi.wang@mail.utoronto.ca

Hao Yu
University of Toronto
Toronto, ON, Canada
hpeiyu.yu@mail.utoronto.ca

Abstract

This paper will present the Position Based Fluids (PBF) method for real-time fluid simulation. This method enforces incompressibility by formulating and solving a system of positional constraints under the Position Based Dynamics framework (PBD). By incorporating an artificial pressure term, this method mitigates particle clustering and creates surface tension. This method also reduces additional damping by using vorticity confinement to replace lost energy.

CCS Concepts: • **Computer Graphics → Computational Geometry and Object Modeling**; • **Three-Dimensional Graphics and Realism Animation**;

Keywords: fluid simulation, SPH, constraint fluids, position based dynamics

ACM Reference Format:
Ruiqi Wang and Hao Yu. 2021. PBF Method for Fluid Simulation. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With their numerous phenomena, such as diffusion, turbulence, and surface tension, fluids have long been an intriguing and challenging subject for simulation. A variety of techniques have become available for animating fluid motion.

One pioneer work is Smoothed Particle Hydrodynamics (SPH) [Monaghan 1992]. It adopts the Lagrangian (particle based) approach, which discretizes a fluid into particles, keeps track of particles as they move, and monitor changes in particle properties. The method interpolates particle properties with smoothing kernels to generate smooth, continuous quantities of a fluid over space. The method preserves several properties of fluids such as conservation of momentum, mass, and energy by solving the famous Navier-Stokes Equations.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Particle-based approaches like SPH are susceptible to density variation caused by the lack of particles at fluid surface. Meanwhile, enforcing incompressibility incurs large computational costs and impedes high quality, real-time water animations. In this paper, we present Position Based Fluids (PBF) [Macklin and Müller 2013], which proposes a stable solver for density constraints, allowing accurate simulation with large time steps.

2 RELATED WORK

Predictive-corrective incompressible SPH (PCISPH) [Solenthaler and Pajarola 2009] is one of the previous works to address the problem with incompressibility. It proposes an implicit pressure solver to maintain fluid density. The method formulates density changes between two time steps as a function of pressure. It then solves for pressure values that conserve densities at the new time step using the Jacobi method, which iteratively updates pressure and applies forces on particles until convergence.

Traditional approaches to simulating objects work with forces. Accelerations are computed to update velocities and positions hereafter. Position Based Dynamics (PBD) [Müller et al. 2007] provides an approach that bypasses velocities and works immediately on positions.

Similar to PCISPH, the PBF method also solves the constraint problem implicitly with the Jacobi method. In addition, based on the PBD framework, it achieves uniform densities by accumulating particle position correction directly.

3 METHOD

We formulate a fluid as a set of particles and track the state of each particle, including position $\mathbf{p}$, velocity $\mathbf{v}$, acceleration $\mathbf{a}$, and density ρ. At each time step, we update the state of a particle based on its state and the states of its neighbors, which we refer to as $P_1, \ldots, P_n$, and $Pi = (\mathbf{p}_i, \mathbf{v}_i, \mathbf{a}_i, \rho_i)$.

3.1 Density

We calculate the density at a particle's location by the standard SPH density estimator:

$$\rho_i = \sum_j m_j W(\mathbf{p}_i - \mathbf{p}_j, h) \tag{1}$$

We treat all particles as having equal mass and will drop this term in subsequent equations.

3.2 Incompressibility

For an incompressible fluid, the density is constant throughout the fluid volume. To enforce this, we formulate a constraint for each particle. The density constraint on the ith particle is defined as:

$$C_i\left(\mathbf{p}_1, \ldots, \mathbf{p}_n\right) = \frac{\rho_i}{\rho_0} - 1 \tag{2}$$

where ρ_0 is the rest density and ρ_i is density at particle i.

The fluid density is conserved by adding a particle position correction $\Delta\mathbf{P}$ to particle positions $\mathbf{P}$ that satisfies the system of non-linear constraints:

$$\mathbf{C}(\mathbf{P} + \Delta\mathbf{P}) = \mathbf{0} \tag{3}$$

$\Delta\mathbf{P}$ is found by a series of Newton steps along the constraint gradient:

$$\Delta\mathbf{P} \approx \nabla\mathbf{C}(\mathbf{P})\lambda \tag{4}$$

$$\mathbf{C}(\mathbf{P} + \Delta\mathbf{P}) \approx \mathbf{C}(\mathbf{P}) + \nabla\mathbf{C}^T\Delta\mathbf{P} = \mathbf{0} \tag{5}$$

The gradient of the constraint function (1) with respect to a particle k is given by:

$$\nabla_{\mathbf{p}_k} C_i = \frac{1}{\rho_0}\begin{cases}\sum_j \nabla_{\mathbf{p}_k} W\left(\mathbf{p}_i - \mathbf{p}_j, h\right) & \text{if } k = i \\ -\nabla_{\mathbf{p}_k} W\left(\mathbf{p}_i - \mathbf{p}_j, h\right) & \text{if } k = j\end{cases} \tag{6}$$

Solving for λ in (4), (5) gives

$$\lambda_i = -\frac{C_i\left(\mathbf{p}_1, \ldots, \mathbf{p}_n\right)}{\sum_k \left|\nabla_{\mathbf{p}_k} C_i\right|^2 + \varepsilon} \tag{7}$$

where ϵ is a constant parameter over the simulation. It is used to prevent instability caused by vanishing gradients when particles are further apart.

The total position update $\Delta\mathbf{p}_i$ is:

$$\Delta\mathbf{p}_i = \frac{1}{\rho_0}\sum_j (\lambda_i + \lambda_j)\nabla W\left(\mathbf{p}_i - \mathbf{p}_j, h\right) \tag{8}$$

3.3 Tensile Instability

When a particle has only a few neighbors, it is unable to satisfy the rest density. This causes particles to be attracted towards each other. We use an artificial pressure term to mitigate particle clustering or clumping:

$$s_{corr} = -k\left(\frac{W\left(\mathbf{p}_i - \mathbf{p}_j, h\right)}{W(\Delta\mathbf{q}, h)}\right)^n \tag{9}$$

where k is a small positive constant. $\Delta\mathbf{q}$ is a point inside the smoothing kernel radius h, such that $|\Delta\mathbf{q}| = 0.1h...0.3h$. In our implementation, we choose $|\Delta\mathbf{q}| = 0.3h$ and $n = 4$.

We include this term in the particle position update:

$$\Delta\mathbf{p}_i = \frac{1}{\rho_0}\sum_j \left(\lambda_i + \lambda_j + s_{corr}\right)\nabla W\left(\mathbf{p}_i - \mathbf{p}_j, h\right) \tag{10}$$

3.4 Vorticity Confinement

Turbulence is an important phenomenon of fluids. Position based methods introduce numerical energy dissipation which negatively impacts the visual liveliness. We use vorticity confinement to add back lost energy.

The vorticity at a particle's location is estimated by [Koschier et al. 2020]:

$$\omega_i = \nabla \times \mathbf{v} = \sum_j \frac{1}{\rho_j}(\mathbf{v}_j - \mathbf{v}_i) \times \nabla_{\mathbf{p}_j} W\left(\mathbf{p}_i - \mathbf{p}_j, h\right) \tag{11}$$

Then we amplify the existing vorticity by applying a corrective force to the particle:

$$\mathbf{f}_i^{\text{vorticity}} = \varepsilon\left(\frac{\eta}{|\eta|} \times \omega_i\right) \tag{12}$$

using the vorticity location vector

$$\eta = \sum_j \frac{1}{\rho_j}|\omega_j|\nabla W\left(\mathbf{p}_i - \mathbf{p}_j, h\right) \tag{13}$$

Alternatively, we can estimate vorticity and the location vector by

$$\omega_i = \nabla \times \mathbf{v} = \sum_j (\mathbf{v}_j - \mathbf{v}_i) \times \nabla_{\mathbf{p}_j} W\left(\mathbf{p}_i - \mathbf{p}_j, h\right) \tag{14}$$

$$\eta = \nabla|\omega_i| \tag{15}$$

3.5 Viscosity

We apply XSPH viscosity, which improves coherency of motion by smoothing each particle's velocity with neighboring particle velocities:

$$\mathbf{v}_i^{\text{new}} = \mathbf{v}_i + c\sum_j \mathbf{v}_{ij} \cdot W\left(\mathbf{p}_i - \mathbf{p}_j, h\right) \tag{16}$$

where $\mathbf{v}_{ij} = \mathbf{v}_j - \mathbf{v}_i$ and the parameter c is chose to be 0.0001 in our simulations.

3.6 External Forces

For external forces such as gravity, we apply these forces directly to the particles. When particles collide with solid objects such as the walls in our examples, we simply push them away from the object.

3.7 Smoothing kernels

In our implementation, we use the Poly6 kernel for kernel density estimation:

$$W_{poly6}(\mathbf{r}, \mathbf{h}) = \frac{315}{64\pi h^9}(h^2 - |\mathbf{r}|^2)^3 \tag{17}$$

and the Spiky kernel for gradient calculation:

$$\nabla_{\mathbf{r}} W_{spiky}(\mathbf{r}, \mathbf{h}) = -\frac{45}{\pi h^6}(h - |\mathbf{r}|)^2\frac{r}{|r|} \tag{18}$$

Both kernels have radius $h = 0.1$.

4 Algorithm

The simulation loop is outlined in Algorithm 1. We denote the quantities of all particles as capital letters.

Algorithm 1 Simulation Step

update velocities $\mathbf{V} \leftarrow \mathbf{V} + \Delta t \mathbf{A}$
predict positions $\mathbf{X}^* \leftarrow \mathbf{X} + \Delta t \mathbf{V}$
find neighboring particles $Neighbor(\mathbf{X}^*)$
while iter < solver Iterations **do**
  calculate $\lambda, \mathbf{S}_{corr}$
  calculate $\Delta \mathbf{P}$
  update predicted positions $\mathbf{X}^* \leftarrow \mathbf{X}^* + \Delta \mathbf{P}$
  collision detection and response on $\mathbf{X}^*$
end while
update velocities $\mathbf{V} \leftarrow \frac{1}{\Delta t}(\mathbf{X}^* - \mathbf{X})$
update positions $\mathbf{X} \leftarrow \mathbf{X}^*$
apply external forces $\mathbf{A} \leftarrow Gravity$
apply vorticity confinement $\mathbf{A} \leftarrow \mathbf{A} + \mathbf{F}^{vorticity}$
apply XSPH viscosity $\mathbf{V} \leftarrow Visco(\mathbf{V})$

4.1 Collision Detection and Response

We detect collisions between particles and objects with signed distances. For a boundary plane, we calculate its signed distance to a particle $\mathbf{p}_i$ by:

$$d = \mathbf{n} \cdot (\mathbf{p}_i - \mathbf{c}) \tag{19}$$

where the plane is represented by $\mathbf{n}$, the surface normal, and $\mathbf{c}$, a point on the plane.

For a watertight mesh, we calculate the signed distance by performing the pseudo-normal test [Baerentzen and Aanaes 2005].

If $d < 0$, we move the particle along the surface normal onto the surface:

$$\mathbf{p}_i = \mathbf{p}_i - d \cdot \mathbf{n} \tag{20}$$

4.2 Neighbor Search

To find neighboring particles efficiently, we bins all particles into a cubic grid. Each cell of the grid has a side length of the kernel radius h. For each particle, we only need to check its distances to particles within a $3 \times 3 \times 3$ grid where the distances can be smaller than h.

5 RESULTS

The algorithm is implemented with Eigen [Guennebaud et al. 2010] and libigl [Jacobson et al. 2018] using a single thread on CPU. We animated a dropping cubic of water with 8000 particles in two scenes, one into an empty box and the other into a box containing the Stanford Bunny. We set the frames with $\Delta t = 0.016s$, one step per frame, and 4 solver iterations per step. The simulation can be performed in real time and successfully displays fluid dynamics and fluid-solid interactions.

We compared two alternate methods for vorticity confinement. Estimating vorticities and location vectors with Equation (11) and (13) generates more energetic particle movements compared to using Equation (14) and (15).

6 CONCLUSIONS

In this paper, we have introduced a particle-based method, PBF, for simulating fluids. It offers a robust and stable algorithm that well balances simulation speed and accuracy. Formulating fluid density constraints under the PBD framework is one of the most important contributions of PBF. It supports simulation with larger time steps in comparison with previous particle-based approaches. Moreover, PBF introduces artificial pressure to prevent particles from clamping without requiring each particle to have a minimum number of neighbours. However, the artificial pressure term is also correlated with surface tension and is hard to tune, so it would be necessary to find techniques that can decouple the two effects and adjust them independently. In our implementation, we have adopted efficient methods for finding neighbours and dealing with collisions. This helps the simulation happen in real-time, but the methods could be further accelerated using parallel computing on GPU in the future.

References

J.A. Baerentzen and H. Aanaes. 2005. Signed distance computation using the angle weighted pseudonormal. *IEEE Trans. Vis. Comput. Graphics* 11, 3 (2005), 243–253. https://doi.org/10.1109/TVCG.2005.49

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. 2020. Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids. *CoRR* abs/2009.06944 (2020). arXiv:2009.06944 https://arxiv.org/abs/2009.06944

Miles Macklin and Matthias Müller. 2013. Position Based Fluids. *ACM Trans. Graph.* 32, 4, Article 104 (jul 2013), 12 pages. https://doi.org/10.1145/2461912.2461984

J. J. Monaghan. 1992. Smoothed particle hydrodynamics. *ARAA* 30 (jan 1992), 543–574. https://doi.org/10.1146/annurev.aa.30.090192.002551

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position Based Dynamics. *J. Vis. Comun. Image Represent.* 18, 2 (apr 2007), 109–118. https://doi.org/10.1016/j.jvcir.2007.01.005

B. Solenthaler and R. Pajarola. 2009. Predictive-Corrective Incompressible SPH. *ACM Trans. Graph.* 28, 3, Article 40 (jul 2009), 6 pages. https://doi.org/10.1145/1531326.1531346