Comparing SPH method and FLIP method for Fluid Simulation

JiongTian Guo
University of Toronto
Toronto, Canada
jiongtian.guo@mail.utoronto.ca

Haoda Li
University of Toronto
Toronto, Canada
haoda.li@mail.utoronto.ca

ABSTRACT

This paper will compare Smoothed-Particle Hydrodynamics (SPH) method [Müller et al. 2003] and Fluid-Implicit-Particle (FLIP) method [Brackbill et al. 1988] for real-time fluid simulation. Both methods are popular particle-based algorithms, while provide different approaches to evaluate incompressible Navier-Stokes equations. SPH method numerically integrates quantities from neighboring particles with a smoothing kernel to update particles' velocities, while FLIP method would use a volume-based method to calculate pressure within each cell of a predetermined pressure-grid using projection method, and then use that pressure to update velocities of particles.

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**.

KEYWORDS

Fluids, Physically Based Animation

ACM Reference Format:
JiongTian Guo and Haoda Li. 2018. Comparing SPH method and FLIP method for Fluid Simulation. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Fluid, such as gas and water, plays an important role in animations. Although its flows look simple, simulation of fluid is much more complex than that of solid. The motion of fluid is governed by the incompressible Navier-Stokes equations, and various methods have been developed to simplify and solve the system.

Two approaches are often taken to measure the mothion of fluid. Lagragian approach treats the fluid as a set of particles, while Eulerian approach treats the fluid space as a grid field and measures quantities at each fixed point.

In this report, we will present the particle based SPH method and particle-grid hybrid FLIP method. In SPH method, we view each particle as a blob of fluid with constant mass and varying volume. Similar to the finite element approach, for each particle position, we accumulate its density and forces by a weighted sum of particles within a radius, where the weights comes from smoothing kernels. We would then use the accumulated force to update the velocity of the particle. In FLIP method, we view each particle as a blob of fluid with constant mass and volume. We would evenly divide the space into fixed-size volumes using a staggered-grid system, and project particles' velocities onto velocity-grids of that staggered-grid system. We will then use the projected velocities of particles to calculate the pressure within each cell of the pressure-grid, and use that pressure to update each particle's own velocities. In both methods, we would use updated velocities of particles to update particles' positions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Woodstock '18, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

2 RELATED WORK

Due to the complexity of fluid simulation, many works aim to build an efficient, stable, and visually appealing system for fluid simulation. For example, Stam's Stable Fluids [Stam 1999] is one of the pioneer work for real-time fluid simulation. In addition, many works focus on specific phenomenons, such as ocean waves [Hinsinger et al. 2002] and lava drops [Stora et al. 1999].

3 PROBLEM SETUP

For an arbitrary fluid, the incompressible Navier-Stokes equation describes the motion [Bridson and Müller-Fischer 2007] as

$$\rho\frac{\partial \mathbf{v}}{\partial t} + \rho\nabla\mathbf{v}\cdot\mathbf{v} = -\nabla p + \rho\mathbf{g} + \mu\nabla^2\mathbf{v} \tag{1}$$

$$\textbf{subjected to } \nabla\cdot\mathbf{v} = 0 \tag{2}$$

where ρ is the density, $\mathbf{v}$ is the velocity, $\mathbf{g}$ is an external force, p is the pressure, and μ is the viscosity coefficient of the fluid.

For a system of fluid, we represent it with a finite number of particles $\mathcal{P} = \{P_1, ..., P_n\}$. For each particle P_i, we store its quantities such as velocity $\mathbf{v}_i$ and position $\mathbf{p}_i$.

4 SPH METHOD

Instead of the incompressible condition as equation (2), SPH method assures conservation of mass

$$\frac{\partial \rho}{\partial t} + \nabla\dot{(}\rho t) = 0 \tag{3}$$

by assigning constant mass m to each particle. Then, we can split the forces in equation (1) into pressure ($-\nabla p$), external forces ($\rho\mathbf{g}$), and viscosity ($\mu\nabla^2\mathbf{v}$). Finally, as particles are moving with the fluid, we can update the particle's velocity via self-advection. For each particle, we have

$$\frac{D\mathbf{v}_i}{Dt} = \frac{1}{\rho_i}(-\nabla p_i + \rho_i\mathbf{g} + \mu\nabla^2\mathbf{v}_i) \tag{4}$$

To update density of each particle and forces acting on each particle, we use Smoothed-Particle Hydrodynamics, which interpolates the

quantities of A at some location $\mathbf{p}$ as a weighted sum of all particles

$$A(\mathbf{p}) = \sum_{j=1}^{n} m_j \frac{A_j}{\rho_j} W(\mathbf{p} - \mathbf{p}_j) = m \sum_{j=1}^{n} \frac{A_j}{\rho_j} W(\mathbf{p} - \mathbf{p}_j) \tag{5}$$

where W is called the smoothing kernel, and the desired properties for W are being even ($W(\mathbf{r}) = W(-\mathbf{r})$) and normalized ($\int W(\mathbf{r})d\mathbf{r} = 1$). An appealing property for SPH is that the quantities are functions of the position, hence taking derivative is simply taking derivative on the smoothing kernel. In addition, to reduce the computation, we may only weigh the neighborhood of the target location, which means $W(\mathbf{p} - \mathbf{p}_j) = 0$ if $\|\mathbf{p} - \mathbf{p}_j\| > h$ for some support radius h. In this paper, 3 smoothing kernels are proposed for density, pressure, and viscosity.

4.1 Density

Using equation (5), the density update is

$$\rho_i = m \sum_j \frac{\rho_j}{\rho_j} W(\mathbf{p}_i - \mathbf{p}_j) = m \sum_j W_{poly6}(\mathbf{p}_i - \mathbf{p}_j) \tag{6}$$

with the density smoothing kernel

$$W_{poly6}(\mathbf{r}) = \frac{315}{64\pi h^9}(h^2 - \|\mathbf{r}\|^2)^3 \tag{7}$$

4.2 Pressure

Instead of directly using equation (5), the pressure update is

$$-\nabla p_i = -m \sum_j \frac{p_i + p_j}{2\rho_j} \nabla W_{spikey}(\mathbf{p}_i - \mathbf{p}_j) \tag{8}$$

we use the mean pressure of two particles to keep it symmetric. Consider two particles interacting with each other, this equation makes sure they get the same amount of pressure. The smoothing kernel is

$$W_{spikey}(\mathbf{r}) = \frac{15}{\pi h^6}(h - \|\mathbf{r}\|)^3 \tag{9}$$

Because the pressure is not carried on each particle, we compute it using modified ideal gas state equation

$$p_i = k(\rho_i - \rho_0) \tag{10}$$

where k is the gas constant, ρ_0 is the rest density for the fluid. Since we only care about the offset of pressure from each direction, removing the constant rest density from all directions have no effect on the simulation and improves numerical stability.

4.3 Viscosity

Since viscosity only depends on velocity differences, the update is

$$\mu\nabla^2\mathbf{v}_i = \mu m \sum_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_i} \nabla^2 W_{visco}(\mathbf{p}_i - \mathbf{p}_j) \tag{11}$$

with smoothing kernel

$$W_{visco}(\mathbf{r}) = \frac{15}{2\pi h^3}\left(-\frac{\|r\|^3}{h^3} + \frac{\|r\|^2}{h^2} + \frac{h}{2\|r\|} - 1\right) \tag{12}$$

4.4 Surface Tension

For simulating liquid, we often model air as void. Then, surface tension, or the attractive forces pulling liquid molecules to each other, creates the imbalance on the liquid's surface. To model such force, we add a smooth color field as

$$c_i = m \sum_j \rho_j^{-1} W(\mathbf{p}_i - \mathbf{p}_j)$$

so that we can define its gradient field $\mathbf{n} = \nabla c$ and curvature as $\kappa = \frac{\nabla^2 c}{\|\mathbf{n}\|}$, and the surface tension is

$$\sigma\kappa\mathbf{n} = \sigma\frac{\nabla^2 c \cdot \nabla c}{\|\nabla c\|} \tag{13}$$

4.5 External forces

For external forces such as gravity and collision forces, we consider each particle as a mass and directly apply the forces onto them. For the collision response, we simply push them away from the object and reflect its velocity around the collision normal.

4.6 Neighboring Search

Note that we only need to sum up the neighboring particles within a certain radius h, a natural optimization will be dividing the space into grid of h, then for each particle, we only need to search for the 27 cubes.

5 FLIP METHOD

We would first simplify the incompressible Navier-Stokes equation by removing the viscosity term, and the simplified equation looks like this:

$$\rho\frac{\partial\mathbf{v}}{\partial t} + \rho\nabla\mathbf{v}\cdot\mathbf{v} = -\nabla p + \rho\mathbf{g} \textbf{ subjected to } \nabla\cdot\mathbf{v} = 0 \tag{14}$$

FLIP method be divided into three parts in general to handle three different effects: advection, external forces, and pressure.

The first two steps of the FLIP method are very simple. At each time step, we would first update each particle's position based on its current velocity, and then we would update each particle's velocity by considering the how external force, such as the gravitational force, influences particles.
The first step takes care of the effect of advection, which states that:

$$\frac{\partial\mathbf{v}}{\partial t} + \nabla\mathbf{v}\cdot\mathbf{v} = 0 \tag{15}$$

The second step takes care of the effect of external forces, which states that:

$$\rho\frac{\partial\mathbf{v}}{\partial t} = \rho\mathbf{g} \tag{16}$$

The third and the most important step is to calculate pressure and use pressure to further update each particle's velocity, and this step takes care of the effect of pressure, which strates that:

$$\rho\frac{\partial\mathbf{v}}{\partial t} = -\nabla p \textbf{ subjected to } \nabla\cdot\mathbf{v} = 0 \tag{17}$$

We will break this step down into multiple parts and explain each part in detail.

5.1 Velocity projection

The first part is to build a three-dimensional staggered-grid system. This staggered-grid system consists of four separate three-dimensional grids. Three velocity-grids of the staggered-grid system are used to handle x, y, and z component of velocity individually, and the remaining pressure-grid is used to handle the pressure. For each of the particles and for each of the three components of the velocity, we would first find the cell of the velocity-grid that contains that particle based on that particle's position, and then project that particle's corresponding velocity component onto eight corners of that cell using trilinear interpolation.

5.2 Solving pressure

We would discretize equation (17) to solve for pressure, and the resulting equation for calculating pressure looks like this:

$$\nabla \cdot \nabla p = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{v}^t \tag{18}$$

Combining with the staggered-grid system and finite difference derivative technique, we could solve for pressure p and obtain the pressure-grid.

5.3 Updating velocity using pressure

After obtaining pressure-grid, we could use this equation to update velocity-grids:

$$\mathbf{v}^{t+1} = \mathbf{v}^t - \frac{\Delta t}{\rho} \nabla p \tag{19}$$

We would project the change in velocity, which is $-\frac{\Delta t}{\rho} \nabla p$ back onto particles.

5.4 Solid boundary conditions

There are two more boundary conditions we need to consider. The first one is the solid boundary condition, which means that the fluid could not flow through the solid wall. It is very simple for this constraint to hold: we just need to the change the velocities on the velocity-grids that are on the solid wall to zero. We would use those changed velocity-grids to calculate the pressure-grid, and use pressure-grid to update velocities on the velocity-grids that are **inside** the solid wall. The velocities on the velocity-grids that are on the solid wall have already been updated to 0, and the changes in velocities on the velocity-grids that are on the solid wall would be the negative projected particle velocity. After implementing the above parts, we could simulate smoke. However, to simulate fluid like water, we still have one more constraint to implement.

5.5 Free surface boundary conditions

One important fact is that the air would not exert pressure on the water. To simulate water, we need to make sure that the pressure is zero on the water surface. To keep track of the water surface, we would mark each cell of the pressure-grid as fluid cell or air cell. We would initialize cells containing particles to −1 and empty cells to +1. Note that this would result in stair-step artifacts in fluid surface, so we then applied three-dimensional Gaussian convolution to smooth that scalar filed containing −1 and +1. After smoothing, the cells containing positive values become air cells, and the cells containing negative values become fluid cells. We would only calculate the pressure inside the fluid cell. When the fluid cell is adjacent to an air cell, we would first find the fluid surface by applying linear interpolation to the value inside the fluid cell and value inside the adjacent air cell to find the position of zero, which represents the fluid surface, we would then modify the pressure inside the fluid cell with ghost pressure to make sure that the pressure on the fluid surface is zero. In that way, the free surface boundary conditions would be satisfied, and we could simulate water.

6 RESULTS

We implemented these methods usign Eigen [Guennebaud et al. 2010] and libigl [Jacobson et al. 2018] on CPU with single threading. We simulated the scene of pouring water into a box with 5625 particles. For both methods, we can animate the scene in real time. The frame rate is quite steady for FLIP method, while in SPH method the frame rate had significant drops when particles are crowded together. In both cases, the scene successfully simulates the water flows and waves.

7 CONCLUSION

In this paper, we introduce two particle-based methods for simulating fluids. SPH method uses a purely particle based approach, and use smoothing kernels to interpolate force components from neighboring particles. FLIP method uses particles to represent fluid blobs, and a staggered-grid system to directly solve pressure. By taking different approaches, the two methods have quite different characteristics. SPH, due to its purely particle based design, is easier to implement, SPH also provides more physical coefficients, hence can model a wide class of fluids. However, SPH method is not incompressible. On the other hand, FLIP simplifies the model by assuming incompressible, inviscid conditions. FLIP has better performance on simulating water. Computational complexity is also differnt for each of the two methods. For SPH method, there would be more computations if the particles are compressed within a small portion of the entire space, since we need to consider more particles to compute forces for each of the particles. However, for FLIP method, there would be more computations if particles are evenly spread across the entire space since in that case, we would have more fluid cells to consider when constructing the pressure grid.

REFERENCES

J.U. Brackbill, D.B. Kothe, and H.M. Ruppel. 1988. Flip: A low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications* 48, 1 (1988), 25 – 38. https://doi.org/10.1016/0010-4655(88)90020-3

Robert Bridson and Matthias Müller-Fischer. 2007. Fluid simulation: SIGGRAPH 2007 course notes Video files associated with this course are available from the citation page. In *ACM SIGGRAPH 2007 courses*. 1–81.

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

Damien Hinsinger, Fabrice Neyret, and Marie-Paule Cani. 2002. Interactive animation of ocean waves. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 161–166.

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

Matthias Müller, David Charypar, and Markus H Gross. 2003. Particle-based fluid simulation for interactive applications.. In *Symposium on Computer animation*. 154–159.

Jos Stam. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 121–128.

Dan Stora, Pierre-Olivier Agliati, Marie-Paule Cani, Fabrice Neyret, and Jean-Dominique Gascuel. 1999. Animating lava flows.