Fast Grid-Based Fluid Dynamics Simulation with Conservation of Momentum and Kinetic Energy on GPU

Ka-Hou Chan(✉) and Sio-Kei Im

MPI-QMUL Information Systems Research Centre,
Macao Polytechnic Institute, Macao, China
chankahou@ipm.edu.mo

Abstract. Since the computation of fluid animation is often too heavy to run in real-time simulation, we propose a fast grid-based method with parallel acceleration. In order to reduce the cost of computation keeping a balance between fluid stability and diversity, we consider the Navier-Stokes equation on the grid structure with momentum conservation, and introduce the kinetic energy for collision handling and boundary condition. Our algorithm avoids the mass loss during the energy transfer, and can be applied to the two-way coupling with a solid body. Importantly, we propose to use the forward-tracing-based motion and design for parallel computing on Graphics Processing Unit (GPU). In particular, these experiments illustrate the benefits of our method, both in conserving fluid density and momentum. They show that our method is suitable to solve the energy transfer when object interaction is considered during fluid simulation.

Keywords: Computational Fluid Dynamics
Momentum conservation · Kinetic energy

1 Introduction

Fluid effects play an important role in the computer games industry: these are the complex interplay of various phenomena, such as convection, diffusion, turbulence and surface tension. Considerable research has taken place to improve the behaviour and performance and with the ever-increasing performance of hardware, simulations of the underlying physics can better approximate natural dynamics for representing water, waves, fire and gas. Hence, currently researches propose to simulate the interaction of multiple materials (fluid and solid objects). Physically, the Navier-Stokes Equation describes the state of the fluid, and there are many methods for solving incompressible flow by this equation.

1.1 Physical Equation

It is well known in Computational Fluid Dynamics (CFD) that the Navier-Stokes Equation that precisely describes the fluid's acceleration by governs the behaviour of the fluid

$$\nabla \cdot \boldsymbol{v} = 0 \tag{1}$$

$$\frac{\partial \boldsymbol{v}}{\partial t} = -\left(\boldsymbol{v} \cdot \nabla\right)\boldsymbol{v} - \frac{1}{\rho}\nabla p + \mu \nabla^2 \boldsymbol{v} + \boldsymbol{f} \tag{2}$$

where $\boldsymbol{v}$ is the fluid's velocity at a grid's centre, p is pressure, ρ is density, μ is the coefficient of viscosity and $\boldsymbol{f}$ includes any other external forces such as gravity or boundary confinement. These equations give a precise description of the evolution of a velocity field over time, and tell exactly how the velocity will change over an infinitesimal time-step as a function of the four terms (advection, pressure, viscosity and external-forces) in Eq. 2, but these equations do not consider the mass evolving in the computational physics transferring.

1.2 Related Work

The Eulerian grid-based fluid implementations have been popular for real time solutions as they provide a better description of the fluid properties. However, they do have a major disadvantage in that the grid must be fixed in the space, so making it difficult to track and depict the detailed behaviours (such as mist, foam and spray). The Navier-Stokes Equation was first used in [7] to animate gases and it produced good results on relatively coarse grids. In order to increase the details with lower memory consumption for grid-based simulation, [17] used a dynamic grid that had a low memory footprint when representing a high-resolution Level Set method. In [12], all quantities at cell centres were stored for an iterative solver, although a staggered MAC grid is robust and can make it easier to define boundary conditions [11,20]. In order to reduce the effect of numerical dissipation caused by the use of an implicit semi-Lagrangian integration scheme [6], physical based approaches have considered to use momentum conservation in incompressible flow and the work in [14] can conserve the energy with the semi-Lagrangian method used in the simulation.

For the coupling between fluid and solid objects, there exist alternative concepts to incorporate boundary conditions for Eulerian fluids. [2] improved the FLIP method of [21] for two-way solid-fluid interaction. In that approach, the intra-pressure is formulated as a kinetic energy for the coupling problem. [10] presented a two-way coupling for deformable and solid thin objects, the algorithm using ray-casting to avoid fluid leaking through thin solids represented by mesh. [15] presented a GPU approach for the semi-Lagrangian scheme of [19] with arbitrary boundary conditions for the fluid simulation being generated and traced directly in Real-time.

For the GPU acceleration, NVIDIA's Compute Unified Device Architecture (CUDA) has been applied to a large number of GPU-based fluid dynamics implementations, primarily in the engineering and scientific computing fields [9]. [13] proposed a CPU-GPU multigrid Poisson solver that exploits both the CPU and GPU to improve the performance and accuracy of the advection step. Later, the work in [3] presented a hybrid grid of two kinds of cell composition, and [4] described a novel gas simulation system that dynamically translates the fluid simulation domain to track the object and fluid surface.

1.3 Contribution

In this paper, we propose to maximize the performance of fluid simulation, focusing on a grid-based solution that can be efficient with parallel acceleration. For the Navier-Stokes Equation, we introduce a stable handling for the viscosity-friction term while updating the mass and momentum. The pressure-gradients term can be obtained by the Ideal Gas Law as suggested by [16]. Different from the traditional backwards advection method, we propose to use the forward-tracing-based motion and the evolving of fluid component (A) can be obtained by

$$A' = \frac{\partial A}{\partial t} + (A \cdot \nabla)\, A = -\frac{1}{\rho}\nabla p + \mu \nabla^2 A \tag{3}$$

Consider the velocity evolving in Navier-Stokes Equation, the transmissions of velocities would be replaced by the transporting of momentum and mass (explain in Sect. 2). Further, we propose a fast component collection method for the fluid transport handling. This idea is designed for parallel implementation and the processing time taken is $O\,(n \log n)$ which is suitable for real-time simulations. In addition, the kinetic energy transferring and conservation of momentum can calculate the coupling status, when solid object interaction is considered during fluid simulation.

2 Solution Method

Commonly, the Navier-Stokes Equation in Eq. 2 can describe the velocity field, but it is not enough to describe the density field only by the changing velocity. [19] provided an improved solution to this problem and implemented a gas simulation with the density being described by the Navier-Stokes Equation. We extend this idea into both the momentum and mass fields. We know that the relation between mass and velocity can be considered as momentum, and the law of conservation of momentum describes the energy transfer in nature. Thus, we use mass and momentum instead of the velocity then apply to Eq. 3 as follows,

$$\boldsymbol{E'} = -\frac{1}{\rho}\nabla p + \mu \nabla^2 \boldsymbol{E} \tag{4}$$

$$m' = -\frac{1}{\rho}\nabla p + \mu \nabla^2 m \tag{5}$$

where $\boldsymbol{E} = m\boldsymbol{v}$ is momentum. In these equations, we focus on the internal (viscosity, pressure) status and external coupling forces as described in Sect. 2.3. After solving, the new velocity can be obtained by the new momentum and mass as

$$\boldsymbol{v}^{+\Delta t} = \frac{\boldsymbol{E}^{+\Delta t}}{m^{+\Delta t}} \tag{6}$$

2.1 Viscosity-Friction

The viscosity is an internal friction force that describes a fluid's internal resistance to flow. This resistance results in diffusion of the momentum (also velocity, density, *etc.*). It causes the fluid's components to move towards the neighbourhood balance. To solve the mass and momentum fields we use:

$$\boldsymbol{E}^{+\Delta t} = \boldsymbol{E} + \int \boldsymbol{E'} dt \tag{7}$$

$$m^{+\Delta t} = m + \int m' dt \tag{8}$$

where $\boldsymbol{E'} = \partial \boldsymbol{E}/\partial t$ and $m' = \partial m/\partial t$. However, this method is unstable when the viscosity is large, so we refer to an efficient method in [19] for a discussion on the Gauss-Seidel Relaxation iterative technology, given by,

$$\boldsymbol{E}^{+\Delta t} = \frac{\boldsymbol{E}_i + \mu\left(\Delta x\right)^2 \cdot \Delta t \cdot \sum_J \boldsymbol{E}_j}{1 + \mu\left(\Delta x\right)^2 \cdot \Delta t \cdot |J|} \tag{9}$$

$$m^{+\Delta t} = \frac{m_i + \mu\left(\Delta x\right)^2 \cdot \Delta t \cdot \sum_J m_j}{1 + \mu\left(\Delta x\right)^2 \cdot \Delta t \cdot |J|} \tag{10}$$

where J is the set of neighbouring grids of current i, j is the element index of J, $|J|$ is the number of elements in set J. This method can avoid the density becoming a negative value, and the new velocity can be more stable and realistic than that obtained by only solving the velocity field directly in Eq. 6.

2.2 Pressure-Gradients

Commonly, the pressure is computed by solving an adequately built equations system using a Conjugate Gradient solver [20], which is heavy in computational load and memory consumption. Focusing on the fluid sample taken on a cell, pressure can be sampled on the centre of the cell. Such a scheme is chosen due to it having better stability properties than a scheme where the samples are taken from the same location. Thus, we invoke the Ideal Gas Law to calculate the recent pressure. It can be obtained from the current density as

$$p = k\rho \tag{11}$$

where k is the gas stiffness that depends on the temperature. We know that there is a constant rest density in the material and this state is more obvious in liquid behaviour. Thus, we use a modified Ideal Gas Law suggested in [16], where the fluid internal pressure can be obtained from the current density as

$$p = \begin{cases} k\left(\rho - \rho_{rest}\right), & \rho > \rho_{rest} \\ 0, & otherwise \end{cases} \tag{12}$$

where ρ_{rest} is the rest density in the material. The pressure will be minimized as the density approaches to this value. The condition $\rho > \rho_{rest}$ recognizes that the pressure is an internal repulsion force, and ignores the attraction force between the two nearest grids: this result/effect is more obvious in liquid behaviour. Consequently, the fluid should exhibit some internal cohesion formation, resulting in attraction-repulsion force as

$$\boldsymbol{f}_i = -\frac{1}{\Delta x \cdot \rho_i} \nabla p_i \tag{13}$$

2.3 Energy Transfer

In addition to model the boundary conditions, we must handle the momentum along both the fluid and solid object. Since the conservation of the total momentum demands that, the total momentum before the collision is the same as the total momentum after the collision,

$$\boldsymbol{E}_{fluid}^{+\Delta t} + \boldsymbol{E}_{rigid}^{+\Delta t} = \boldsymbol{E}_{fluid} + \boldsymbol{E}_{rigid} \tag{14}$$

In order to make sure the overall energy is conserved during coupling, we have to find the relation between momentum and energy items. Moreover, the conservation of the total kinetic energy can be expressed by

$$\frac{1}{2}\boldsymbol{E}_{fluid}^{+\Delta t}\boldsymbol{v}_{fluid}^{+\Delta t} + \frac{1}{2}\boldsymbol{E}_{rigid}^{+\Delta t}\boldsymbol{v}_{rigid}^{+\Delta t} = \frac{1}{2}\boldsymbol{E}_{fluid}\boldsymbol{v}_{fluid} + \frac{1}{2}\boldsymbol{E}_{rigid}\boldsymbol{v}_{rigid} \tag{15}$$

Specially, note that gravity should directly add momentum to fluid as potential energy then be converted into kinetic energy. The bottom boundary forces can offset it symmetrically. Thus, we ignore the momentum changes based on gravity in these equations. By solving Eqs. 14 and 15, the evolved fluid velocity $\boldsymbol{v}_{fluid}^{+\Delta t}$ in the new frame of reference can be determined by

$$\boldsymbol{v}_{fluid}^{+\Delta t} = \frac{\boldsymbol{E}_{fluid} + \boldsymbol{E}_{rigid} - C\left(m_{fluid}\boldsymbol{v}_{rigid} - \boldsymbol{E}_{fluid}\right)}{m_{fluid} + m_{rigid}} \tag{16}$$

where C is the coefficient of the restitution and slip belonging to the normal and tangential directions at the collided face. We can demonstrate the scale of coefficient ($C \in [0.0, 1.0]$) for the handling various boundary conditions, as well as complex external constraints such as perfect elastic and free slip effects.

3 Implementation

In order to do simulation in real-time, we prefer to implement the fluid model in a GPU to accelerate our computation by parallelism.

3.1 Fluid Diffusion and Distribution

In the Eulerian grid-based approach, the fluid behaviour is described in terms of a fixed grid. Fluid components cannot be transported by moving these grids directly. To consider the cell size and diffusing the fluid mass (m) would lose the details in vorticity, so we change to diffuse the density ($\rho = m/\Delta x^d$, d is the dimension) instead. Consider the advection term $-(\boldsymbol{v} \cdot \nabla)\,\boldsymbol{v}$ in Eq. 2, the minus symbol means these fluid components should be found by backward tracing [19]. However, this method may cause some mass loss. If mass (also momentum) is lost and there are few areas of coupling to exchange energy to, changes in energy may cause undesirable noise. To avoid this issue we introduce a forward tracing method that can conserve the total ρ throughout the simulation.

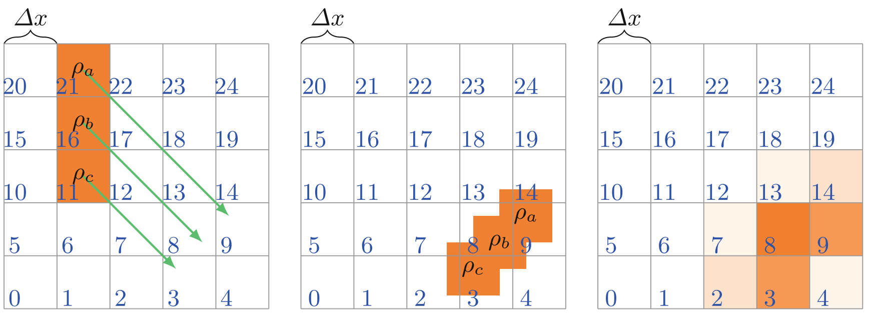


Fig. 1. These figures illustrate the forward tracing steps: after these components have been transported, they should be assigned to their nearest 4 (in 2D, 8 in 3D) grids.

As shown in Fig. 1 in a 2D case, fluid material moves (forward) to a new position. These source components (density, momentum) have been divided by Linear-Interpolation and added to the nearest four grids (*e.g.* ρ_a will distribute to grid 8, 9, 13 and 14). Assume ρ_a is the current density in grid 21, the source ρ_a will divide to ρ_a^8, ρ_a^9, ρ_a^{13} and ρ_a^{14} respectively, with $\rho_a^8 + \rho_a^9 + \rho_a^{13} + \rho_a^{14} = \rho_a$. Additionally, the final density of grid 8 should be $\rho_a^8 + \rho_b^8 + \rho_c^8$. Every grid must repeat this operation to obtain the final state.

3.2 Fluid-Grid Relation Table

In the forward tracing method, all grids must wait for fluid movement to finish, then each grid should evolve to the sum of the fluid components that have arrived, but it is difficult to distribute momentums and density to the grids efficiently. In our proposal, we define a pair of index arrays to represent the *Fluid-Grid* relation between the grid and the arrived fluid after diffusion (see Fig. 2).

As shown in Fig. 2, array (a) stores the index of the grid that receives the fluid density and array (b) stores the sorted index of the density relevant to

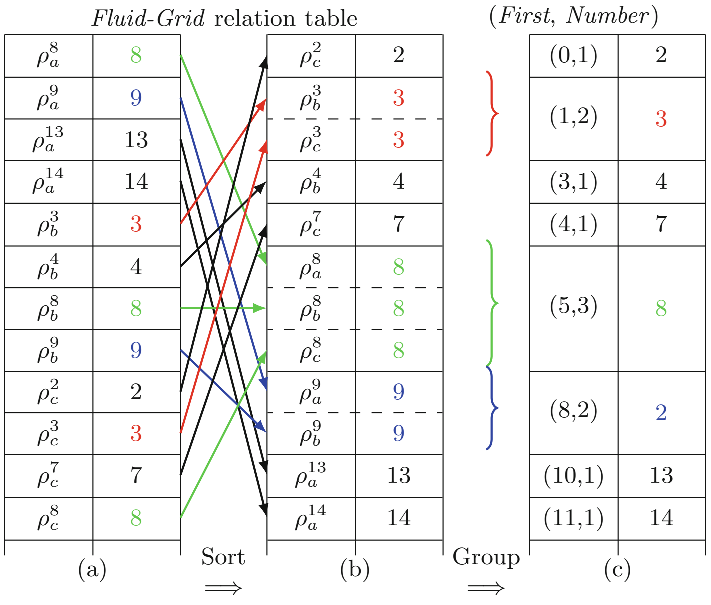


Fig. 2. Flow chart of sorting and grouping the fluid density in the same grid.

the corresponding grids. Sorting is by the parallel radix algorithm in the GPU environment and the performance is $O\,(n \log n)$. According to these two arrays, we must provide the (*First*, *Number*) pair for each cell coordinate to record the table index of the first related density and the number of related densities in the sorted table.

Finally, every cell and its received density are grouped into the same region of the table, thus there is only one loop of the sum of densities and we know exactly how many densities must be queried in our method. This makes it suitable for parallel programming design and implementation. Note that using our conservative advection with diffusing can conserve the total mass throughout the simulation.

4 Experiment Results and Discussion

These experiments has been implemented on a PC with 2 GB memory with video card NVIDIA Quadro 6000 GPU. These simulations are implemented by CUDA 7.5. The scene was rendered by OpenGL 4.5, GLSL 4.5 in 2D and OptiX 3.9 in 3D.

4.1 Results

As shown in Fig. 3, the smoke is running in 2D and the cycle time-step is less than 1/60 s with the Courant-Friedrichs-Lewy (CFL) condition. Note that there is no

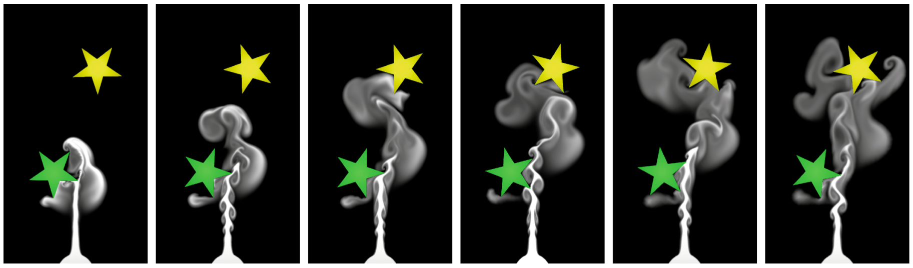


Fig. 3. These figures show the density of smoke at different time-steps. This simulation uses our method with the smoke injected from below. It processes at a resolution of 500×900 grids scale and coupling with two rotating stars.

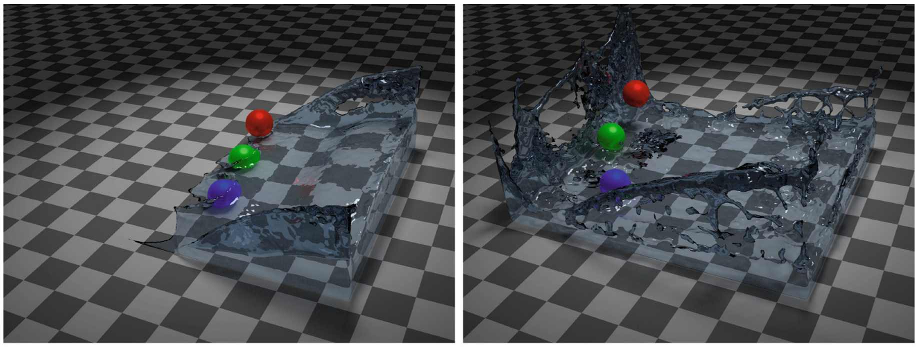
gravitational effect in smoke, thus external forces only include the boundary and coupling forces. We use Eq. 11 for computing the smoke pressure; it illustrates that while the smoke rises by buoyancy, the potential energy will decrease with increasing kinetic energy and that will apply to the energy transference. However, there is no mass loss in our method, but it does cause some areas to contain undesirable dispersion-density/noise that is obvious in the smoke simulation. To alleviate this problem we should adjust neighbouring smoke-grid (For example in Fig. 1, very low density of smoke ρ_c^7 in grid 7 has prorated to its neighbouring smoke-grid 2 and 8).

As shown in Fig. 4, water is running in a 3D scene. We use Eq. 12 for computing the water pressure, and we also add some vorticity confinement [18] to the system for momentum conservation. Note that a particle level set method [5] is used to advect and treat the boundary tracing. For forward advection we need to treat the level set as a solid object, since there is no guarantee that the particle level set method had been also used in both time t and $t^{+\Delta t}$. Furthermore, the water would be affected by gravity and the upper grids energy will increase so that the energy becomes very large at the bottom. This is non-conservative from a global perspective. To solve this we produce a reaction vertical momentum after the gravity is added to these bottom grids.

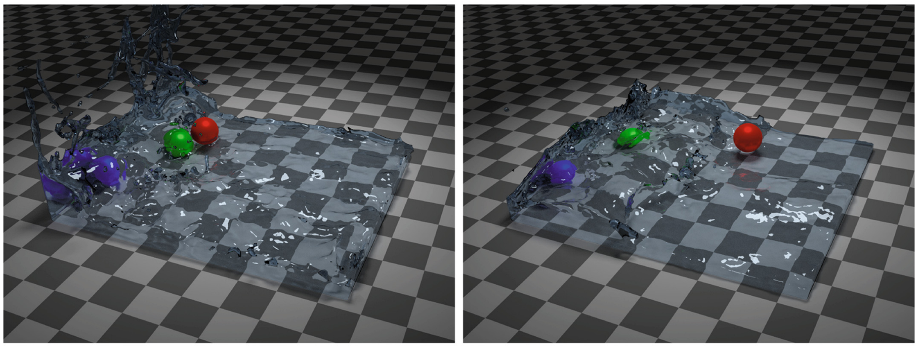
These GPU-based fluid simulations with momentum conservation can be simulated in real time. The results show that our method can be used in gas and liquid with conservation of energy. The time performance of the experimental results is shown in Table 1.

4.2 Comparison

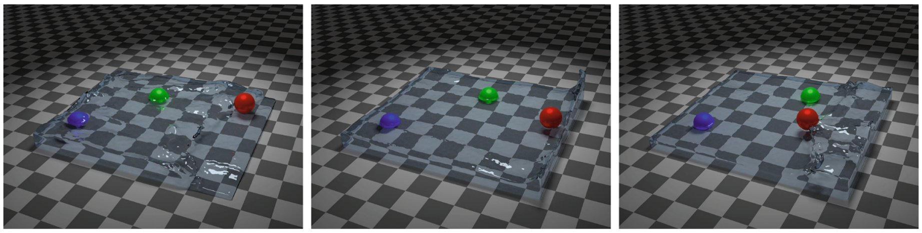
For rendering purposes, we improve the visual quality of the simulation by generating particles that penetrate the level set surface, then applying a local momentum conservation force to slightly alter the level set velocities in fluid surface regions. The comparison between our method, the methods using the GPU-based stable fluid in [1], and the mass and momentum conservation fluid [14] is shown in Fig. 5.



(a) Water is poured onto solid balls with different densities.



(b) Red ball is low density; green ball is equal to water density; blue ball has higher density.



(c) The interactive force by fluid does not significantly move the blue ball, and the red ball is floating on the water. The green ball can be pushed by the water but is not floating on the water.

Fig. 4. These figures show the interaction between water and solid balls in different time-steps. This simulation processes at resolution $128 \times 128 \times 128$ grids scale in 3D, and the fluid surface is constructed by the marching cube algorithm [8]. (Color figure online)

Compared with [14], a speedup of around 10% to 20% can be achieved. Although the speed is slower than in [1], a reason is that the fluid surface rendering is also time-consuming, but there is not much difference and our method can avoid the mass loss and converse the energy of interaction.

Table 1. Performance results of GPU-based of fluid simulation with momentum conservation.

Grid Size	Fluid Solver (ms)	Diffusion & Distribution (ms)	Surface Render (ms)	Overall (ms)
$32 \times 32 \times 32$	4.29	0.42	2.31	7.02
$32 \times 64 \times 32$	9.94	0.83	4.57	15.34
$64 \times 32 \times 64$	19.01	0.95	5.23	25.19
$64 \times 64 \times 64$	21.01	1.19	6.55	28.75
$64 \times 128 \times 64$	39.06	1.48	8.14	48.68
$128 \times 64 \times 128$	75.76	2.07	11.39	89.22
$128 \times 128 \times 128$	149.25	3.12	17.16	169.53

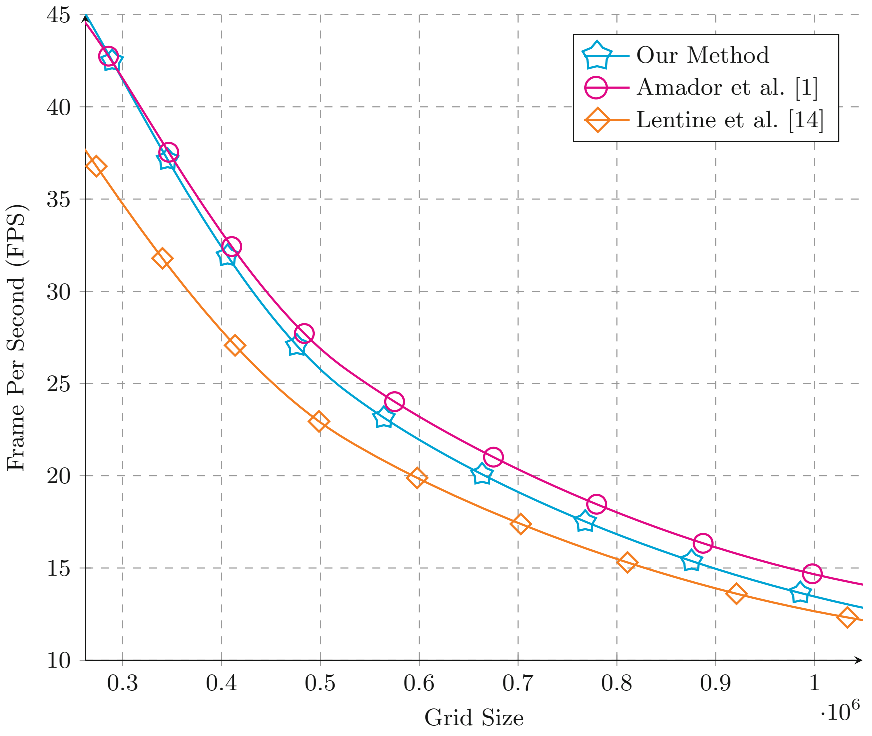


Fig. 5. Comparison of the simulation speed in FPS.

5 Conclusion

In this paper, we have presented a novel momentum conservation method for simulating a wide variety of fluid behaviours. Our algorithm can avoid mass loss during the energy transfer, and the solution, designed for parallel computing, can be implemented in a Graphics Processing Unit (GPU). Moreover, a fast forward-tracing-based motion has been presented to handle the distribution of the fluid components. With the momentum conservation, we also considered the conservation of energy for various boundary conditions, including perfect elastic and free slip effects. The experiment results showed that our method can be around 10% to 20% faster than the previous momentum conservation fluid simulation and this is significant.

References

1. Amador, G., Gomes, A.: A Cuda-based implementation of stable fluids in 3D with internal and moving boundaries. In: 2010 International Conference on Computational Science and Its Applications (ICCSA), pp. 118–128. IEEE (2010)
2. Batty, C., Bertails, F., Bridson, R.: A fast variational framework for accurate solid-fluid coupling. ACM Trans. Graph. (TOG) **26**, 100 (2007). ACM
3. Chentanez, N., Müller, M.: Real-time Eulerian water simulation using a restricted tall cell grid. ACM Trans. Graph. (TOG) **30**(4), 82 (2011)
4. Cohen, J.M., Tariq, S., Green, S.: Interactive fluid-particle simulation using translating eulerian grids. In: Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, pp. 15–22. ACM (2010)
5. Enright, D., Marschner, S., Fedkiw, R.: Animation and rendering of complex water surfaces. ACM Trans. Graph. (TOG) **21**, 736–744 (2002). ACM
6. Fedkiw, R., Stam, J., Jensen, H.W.: Visual simulation of smoke. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 15–22. ACM (2001)
7. Foster, N., Metaxas, D.: Modeling the motion of a hot, turbulent gas. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, pp. 181–188. ACM Press/Addison-Wesley Publishing Co. (1997)
8. Geiss, R.: Generating complex procedural terrains using the GPU. In: GPU Gems 3, pp. 7–37 (2007)
9. Goodnight, N.: Cuda/OpenGL fluid simulation. NVIDIA Corporation (2007)
10. Guendelman, E., Selle, A., Losasso, F., Fedkiw, R.: Coupling water and smoke to thin deformable and rigid shells. ACM Trans. Graph. (TOG) **24**(3), 973–981 (2005)
11. Harlow, F.H., Welch, J.E.: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. Phys. Fluids **8**(12), 2182–2189 (1965)
12. Harris, M.J.: Fast fluid dynamics simulation on the GPU. In: SIGGRAPH Courses, p. 220 (2005)
13. Jung, H.R., Kim, S.T., Noh, J., Hong, J.M.: A heterogeneous CPU-GPU parallel approach to a multigrid Poisson solver for incompressible fluid simulation. Comput. Anim. Virtual Worlds **24**(3–4), 185–193 (2013)
14. Lentine, M., Aanjaneya, M., Fedkiw, R.: Mass and momentum conservation for fluid simulation. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 91–100. ACM (2011)
15. Liu, Y., Liu, X., Wu, E.: Real-time 3D fluid simulation on GPU with complex obstacles. In: 2004 Proceedings of the 12th Pacific Conference on Computer Graphics and Applications, PG 2004, pp. 247–256. IEEE (2004)

16. 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, pp. 154–159. Eurographics Association (2003)
17. Nielsen, M.B., Museth, K.: Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets. J. Sci. Comput. **26**(3), 261–299 (2006)
18. Selle, A., Rasmussen, N., Fedkiw, R.: A vortex particle method for smoke, water and explosions. ACM Trans. Graph. (TOG) **24**, 910–914 (2005). ACM
19. Stam, J.: Stable fluids. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 121–128. ACM Press/Addison-Wesley Publishing Co. (1999)
20. Tome, M.F., McKee, S.: Gensmac: A computational marker and cell method for free surface flows in general domains. J. Comput. Phys. **110**(1), 171–186 (1994)
21. Zhu, Y., Bridson, R.: Animating sand as a fluid. ACM Trans. Graph. (TOG) **24**, 965–972 (2005). ACM