Pacific Graphics (2018)
H. Fu, A. Ghosh, and J. Kopf (Editors)

Short Paper

Robust and Efficient SPH Simulation for High-speed Fluids with the Dynamic Particle Partitioning Method

Zhong Zheng[1], Yang Gao[1†], Shuai Li[1,2], Hong Qin[3] and Aimin Hao[1]

[1]State Key Laboratory of Virtual Reality Technology and Systems, Beihang University
[2]Beihang University Qingdao Research Institute,Beihang University
[3] Department of Computer Science, Stony Brook University

Abstract

In this paper, our research efforts are devoted to the efficiency issue of the SPH simulation when the ratio of velocities among fluid particles is large. Specifically, we introduce a k-means clustering method into the SPH framework to dynamically partition fluid particles into two disjoint groups based on their velocities, we then use a two-scale time step scheme for these two types of particles. The smaller time steps are for particles with higher speed in order to preserve temporal details and guarantee the numerical stability. In contrast, the larger time steps are used for particles with smaller speeds to reduce the computational expense, and both types of particles are tightly coupled in the simulation. We conduct various experiments which have manifested the advantages of our methods over the conventional SPH technique and its new variants in terms of efficiency and stability.

CCS Concepts

•***Computing methodologies*** → *Animation; Physical simulation;*

1. Introduction and Motivation

Most of the current SPH models are either using a unified constant time step, or adopting an adaptive one which is directly derived from the CFL condition by using the maximum velocity of all particles. To address the efficiency issues in turbulent fluid simulation, we propose a dynamic particle partitioning SPH method (PPSPH) that improves the simulation's efficiency through changing the time step size for different fluid particles. We divide each simulation time step into certain numbers of small sub-steps, and in each sub-step we simplify the computational procedure when computing the particles with low velocities, and then promise a correct final result at the end of each major step. Specifically, the salient contributions of this paper include:

- We introduce a *k*-means clustering scheme into SPH to partition particles based on their velocities.
- We propose a dynamic two-scale time step model for particles with large difference in velocities.
- Our method is easy to be performed on most of the existing SPH frameworks and can be implemented on GPU for high-efficiency applications.

Fig. 1 illustrates the main procedure of our simulation steps. Compare to the tradition SPH framework, our simulation step consists of three main components: (1) Partitioning particles into two disjoint groups by *k*-means clustering; (2) Uncoupling high-speed particles from fluids and simulating faster particles with refined time-step size; and (3) Integrating all fluid particles including faster ones and slower ones.

2. Related Works

In recent years, a lot of visual-effect-dominant fluid simulation methods have been introduced for visually-appealing animation. Fluid implicit particle (FLIP) method [ZB05] is a popular particle-grid coupling method to simulate multiphase fluid [BB12, GLY*17], splashing water [GB13, YLHQ15], fluid-solid coupling [GLQH17], etc. Also, data-driven methods [RWTT14, LJS*15] and position based fluid [MM13, MMCK14] all achieve good achievements. However, among the fluid simulation methods, SPH may be the most widely used and most studied method.

The SPH method is commonly used in computer graphics for fluid simulation [Mon92], and is extended for free surface flow [Mon94]. Müller et al. [MCG03] derived inter-particle forces from Navier-Stokes equations for interactive water simulation, but it was hard to model strong incompressible scenes. The fluid simulation method of Müller et al. [MCG03] was based on equation of state to weakly enforce incompressibility. Later, WCSPH [BT07] was proposed to enforce a weakly compressible form of SPH for fluid flow based on the Tait equation. Implicit pressure solvers

[†] Z.Z. and Y.G. contributed equally and should be regarded as co-first authors.

DOI: 10.2312/pg.20181268

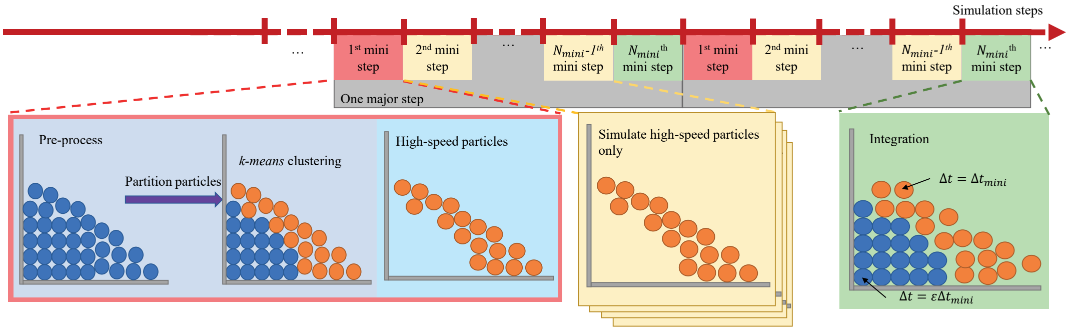

Figure 1: *A brief illustration of simulation steps. There are mainly three parts of a major step, the first one includes a pre-process that partitions particles into two disjoint groups, the last one integrates all particles with a refined time step for both types of particles, and the rest mini steps only simulate high-speed particles.*

was introduced by Solenthaler et al. in PCISPH [SP09]. Ihmsen et al. proposed IISPH [ICS*14]. And Bender et al. [BK17] employed two solvers to enforce a constant density condition and a divergence-free condition. And there have been some asynchronous methods [HF17] [TPS08] [GB14] proposed to boost the efficiency of SPH simulations. Ihmsen et al. [IOS*14] summarized the state-of-the-art research within the graphics community.

3. Particle Classification

The function of k-means clustering is partitioning n points into k clusters. We use predicted particle velocities as sample observations to partition particles into two different clusters. The predicted velocity is computed as

$$\mathbf{v}_i^* = \mathbf{v}_i + \Delta t \frac{\mathbf{F}_i^{ext}}{m_i}. \tag{1}$$

The k-means clustering usually converges to a local optimal value. In order to have a more clear dividing line for the particles, our method usually sets the maximum predicted velocity to the initial mean value of one cluster, and sets the threshold value to another cluster. The threshold value is a user-defined value used to determine whether the k-means clustering procedure will be proceeded. In the end of the clustering, particles will be partitioned into two clusters, one with slower particles, another with faster ones. And please note that, the clustering and the time-step updating procedure are only performed in the first mini step of every major time step.

4. Simulation Steps

4.1. Major Step

A major step consistes of N_{mini} mini steps, but the major time-step size is not a simple multiple of mini time-step size in most cases during the simulation. Particles with high velocity will be processed in each mini time step, and we should guarantee that each particle has a normal motion so we implement the mini and the major time step framework in the simulation. According to the CFL condition

$$\Delta t \leq 0.4 \frac{d}{\|\mathbf{v}_{max}\|_2}, \tag{2}$$

apply the maximum velocity of all particles $\mathbf{v}_{max,all}$ and the maximum velocity of particles in low-speed group $\mathbf{v}_{max,low}$ to Eq. (2) we get the major and the mini time step size

$$\Delta t_{mini} \leq 0.4 \frac{d}{\|\mathbf{v}_{max,all}\|_2}, \Delta t_{major} \leq 0.4 \frac{d}{\|\mathbf{v}_{max,low}\|_2}. \tag{3}$$

Also, we set a user-defined threshold (N_{tres}), when the ratio is smaller than the threshold, the major step will only contains one standard SPH time step. To prevent the ratio from being too large which may cause an unreal simulation result, we set a maximum mini time step amount N_{max}. The amount of mini step in a major step is computed as

$$N_{mini} = \begin{cases} \min(N_{max}, \left\lceil \frac{\Delta t_{major}}{\Delta t_{mini}} \right\rceil) & \text{if } \frac{\Delta t_{major}}{\Delta t_{mini}} \geq N_{tres} \\ 1 & \text{else} \end{cases}, \tag{4}$$

where we set the threshold N_{tres} in Eq. (4) to a number less than 1.5. After determining the amount of mini steps N_{mini}, different strategies will apply on the following mini steps.

4.2. Mini Step

In our method, during the first $N_{mini} - 1$ mini steps, only particles partitioned into the high-speed group will process pressure solving and update their velocities and positions, because we have noticed that the pressure solving procedure is the most time-consuming procedure in the whole simulation step. And in the last mini step, all particles in the scene will process pressure solving and update their positions.

In the last mini step, we process the complete SPH time step procedure for all particles in the scene, but with some modifications. Since during the first $N_{mini} - 1$ steps, low-speed particles are *waiting* for the high-speed particles, to make up the difference in these two-scale time steps, we propose a solution of having an adjustment on the time step for low-speed particles:

$$\Delta t_i = \begin{cases} \varepsilon\Delta t_{mini} & \text{if particle } i \in \text{low-speed cluster} \\ \Delta t_{mini} & \text{else} \end{cases}, \tag{5}$$

Table 1: *Performance of k-means clustering.*

Number of Particles	350k	550k	1.2m	3.0m
k-means Clustering	29.1ms	40.04ms	64.4ms	138.6ms
Time Step Update (per mini step)	8.48ms	10.5ms	29.1ms	79.0ms

Table 2: *Performance comparison for fluids flushing scene.*

Method	Pressure Solving		Overall Simulation	
	avg	sum	avg	sum
PCISPH	223.3ms	1454s	378.0ms	2462s
PP-PCISPH	114.2ms	918s	237.9ms	1480s
DFSPH	161.0ms	1036s	326.0ms	2099s
PP-DFSPH	71.5ms	535s	205.7ms	1288s

This scene contains 369,000 fluid particles in the fluid block with 100,000 high-speed particles to emit later and 448,000 static boundary particles.

where ε is a user-defined parameter of time compensation of low-speed particles in the last mini step. The parameter ε is usually set to a number larger than 1. The higher value ε is, the more temporal details in the simulation will be produced and more time will be spent. And ε should not set to a number larger than a value the CFL condition indicates. In our implementation, we usually set the number to 1.5 to ensure the equilibrium of detail description and the efficiency when $N_{max} \leq 3$.

5. Experimental Results and Discussion

In this paper, we integrate our method with both PCISPH and DFSPH framework, and employ the cubic spline kernel in simulation and use XSPH variant [SB12] to compute viscosity terms for low viscosity fluids, and employ the method proposed in [AIA*12] to handle the boundary. We fully implement the simulation procedure of our methods on CUDA for efficiency.

Table 1 compares the efficiency of *k*-means clustering procedure in our method (PP-SPH) with the time step updating procedure of standard SPH methods. Note that, in this measurement, *k*-means clustering also includes the time step computing procedure. Although clustering cost some time, we process the clustering procedure only once per N_{mini} steps. It has obvious advantage when simulating high-speed fluid, since the greater the speed difference is, the larger N_{mini} will be, and the better performance we could have than the standard ones.

5.1. Comparisons with Different Parameter Settings

We have two more parameters in our method than the standard SPH methods, the maximum mini steps in a major step N_{max} and the time step adjustment parameter for low-speed particles ε.

Table 3 demonstrates the performance with different parameter settings in double dam break scenes. In our method, choosing a larger value for N_{max} will improve the efficiency of the simulation. In our simulation, we usually choose a number ranging from 2 to 4 for N_{max}. The ε is served as a supplement to the time step for low-speed particles, a larger ε will decrease the efficiency of the simulation as computational cost will increase with the time-step size.

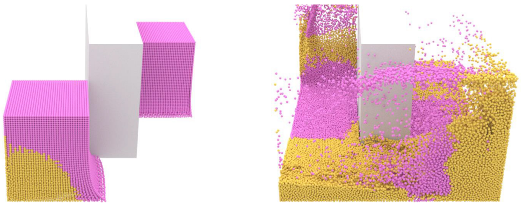

Figure 2: *Double dam break with obstacle (Particle partitioning view.)*

Table 3: *Performance comparison for Double Dam Break (Fig. 2).*

N_{max}	ε	PCISPH		DFSPH	
		pressure	overall	pressure	overall
2	1	159.3s	274.2s	73.2s	197.3s
	1.3	256.9s	370.6s	76.0s	196.1s
	1.5	302.0s	419.2s	79.0s	204.2s
3	1	123.7s	211.9s	61.7s	171.8s
	1.5	236.6s	333.7s	67.4s	181.9s
	2	307.0s	393.1s	78.6s	195.0s
4	1	115.4s	207.2s	73.5s	183.3s
	2	245.8s	332.9s	83.7s	192.8s
	2.5	339.1s	419.2s	98.6s	214.3s
Standard		317.2s	411.7s	123.7s	232.4s

Table 4: *Performance comparison of Flooding Street (Fig. 3).*

Method	Num Part.	Pressure Solving		Overall Simulation	
		avg	sum	avg	sum
PCISPH	2720k	1119.54ms	11006s	1499.8ms	14787s
PP-PCISPH		707.4ms	6780s	1032.2ms	9892s
DFSPH		1202.6ms	8812s	1679.2ms	12305s
PP-DFSPH		303.7ms	3018s	755.7ms	7511s
PCISPH	5330k	2580.9ms	32594s	3871.4ms	48892s
PP-PCISPH		2566.2ms	25573s	3675.2ms	36623s
DFSPH		2956.5ms	27433s	4413.0ms	40948s
PP-DFSPH		841.2ms	8947s	2139.7ms	22757s

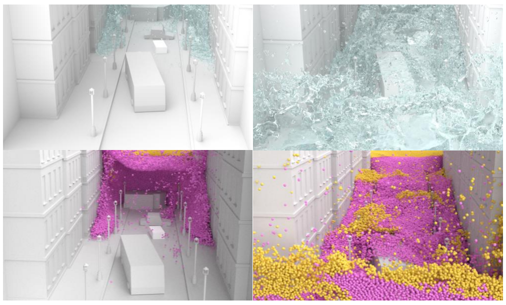

Figure 3: *A fluid block flooding the street. The rendering results are displayed as fluid surfaces and clustered particles.*

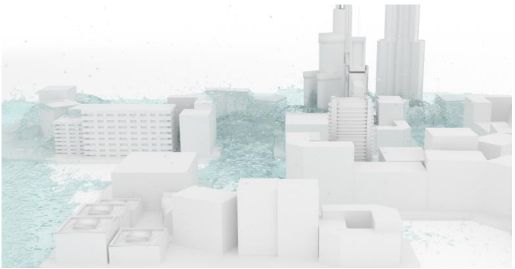

Figure 4: *City blocks with flooding.*

5.2. Large-scale Scenarios

Fig. 3 illustrates a fluid block falling and flushing to the street, hitting the vehicles and street lights. Fluids produce big waves when hitting the vehicles and is divided into two flows when meeting the street lights. Table 4 documents the performance of Fig. 3. Fig. 4 illustrates fluid flooding the city and generating splashes when hitting the buildings and flowing through the roads between buildings. Both scenes have proved that our method is capable of simulating large-scale scenarios with stability and efficiency. We believe that, our method should be able to simulate some larger scenarios, yet due to the limitation on the graphics memory at our lab, we are not able to conduct such simulations currently.

6. Conclusion

In this paper we have detailed a novel SPH framework to boost the efficiency and robustness of fluid simulation. In contrast to the previous SPH methods using a static or adaptive yet unified time step for all fluid particles in the simulation, we introduced a two-scale time-step scheme for SPH, which partitions particles into two groups, and enforced different schedules and different time-step sizes for particles of different groups. Simulating particles with different strategies in fluid simulations has resulted in an improvement on computational efficiency, and can guarantee the numerical stability while still producing realistic details. Our method has apparent advantage over existing SPH methods when simulating large-scale phenomena with high-speed particles in the scenes.

Limitation. We noticed that some particles with intermediate speeds will be rapidly shifting between high-speed and low-speed clusters. Although it won't affect the stability of our simulation, it might slightly reduce the efficiency of our simulation because more particles are partitioned into the high-speed group, thus more computations need to be handled.

ACKNOWLEDGMENT

This research is supported in part by National Key R&D Program of China, National Natural Science Foundation of China (No. 61672077 and 61532002), Applied Basic Research Program of Qingdao (No. 161013xx), National Science Foundation of USA (No. IIS-0949467, IIS-1047715, and IIS-1049448), and the Academic Excellence Foundation of BUAA for PhD Students.

References

[AIA*12] Akinci N., Ihmsen M., Akinci G., Solenthaler B., Teschner M.: Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 62. 3

[BB12] Boyd L., Bridson R.: MultiFLIP for energetic two-phase fluid simulation. *ACM Transactions on Graphics (TOG) 31*, 2 (2012), 1–12. 1

[BK17] Bender J., Koschier D.: Divergence-free SPH for incompressible and viscous fluids. *IEEE Transactions on Visualization and Computer Graphics 23*, 3 (2017), 1193–1206. 2

[BT07] Becker M., Teschner M.: Weakly compressible SPH for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), Eurographics Association, pp. 209–217. 1

[GB13] Gerszewski D., Bargteil A. W.: Physics-based animation of large-scale splashing liquids. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 1–6. 1

[GB14] Goswami P., Batty C.: Regional time stepping for sph. In *Eurographics 2014* (2014), Eurographics Association, pp. 45–48. 2

[GLQH17] Gao Y., Li S., Qin H., Hao A.: A novel fluid-solid coupling framework integrating FLIP and shape matching methods. In *Proceedings of the Computer Graphics International Conference* (2017), p. 11. 1

[GLY*17] Gao Y., Li S., Yang L., Qin H., Hao A.: An efficient heat-based model for solid-liquid-gas phase transition and dynamic interaction. *Graphical Models 94* (2017), 14–24. 1

[HF17] Hu Y., Fang Y.: An asynchronous material point method. In *ACM SIGGRAPH 2017 Posters* (2017), ACM, p. 60. 2

[ICS*14] Ihmsen M., Cornelis J., Solenthaler B., Horvath C., Teschner M.: Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 426–435. 2

[IOS*14] Ihmsen M., Orthmann J., Solenthaler B., Kolb A., Teschner M.: SPH Fluids in Computer Graphics. In *Eurographics 2014 - State of the Art Reports* (2014), The Eurographics Association. 2

[LJS*15] Ladický L., Jeong S., Solenthaler B., Pollefeys M., Gross M.: Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 1–9. 1

[MCG03] Müller M., Charypar D., Gross M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), pp. 154–159. 1

[MM13] Macklin M., Müller M.: Position based fluids. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 104. 1

[MMCK14] Macklin M., Müller M., Chentanez N., Kim T.-Y.: Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 153. 1

[Mon92] Monaghan J. J.: Smoothed particle hydrodynamics. *Annual review of Astronomy and Astrophysics 30*, 1 (1992), 543–574. 1

[Mon94] Monaghan J. J.: Simulating free surface flows with SPH. *Journal of Computational Physics 110*, 2 (1994), 399–406. 1

[RWTT14] Raveendran K., Wojtan C., Thuerey N., Turk G.: Blending liquids. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 1–10. 1

[SB12] Schechter H., Bridson R.: Ghost SPH for animating water. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 61. 3

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

[TPS08] Thomaszewski B., Pabst S., Strasser W.: Asynchronous cloth simulation. In *Computer Graphics International* (2008), vol. 2. 2

[YLHQ15] Yang L., Li S., Hao A., Qin H.: Hybrid particleâĂŘgrid modeling for multiâĂŘscale droplet/spray simulation. *Computer Graphics Forum 33*, 7 (2015), 199–208. 1

[ZB05] Zhu Y., Bridson R.: Animating sand as a fluid. *ACM Transactions on Graphics (TOG) 24*, 3 (2005), 965–972. 1