Neural Comput & Applic
DOI 10.1007/s00521-016-2286-8

RECENT ADVANCES IN PATTERN RECOGNITION AND ARTIFICIAL INTELLIGENCE

Adaptively stepped SPH for fluid animation based on asynchronous time integration

Xiaojuan Ban[1] · Xiaokun Wang[1] · Liangliang He [1] · Yalan Zhang [1] · Lipeng Wang[1]

Received: 12 November 2015 / Accepted: 25 March 2016
© The Natural Computing Applications Forum 2016

Abstract We present a novel adaptive stepping scheme for SPH fluids, in which particles have their own time steps determined from local conditions, e.g. courant condition. These individual time steps are constrained for global convergence and stability. Fluid particles are then updated asynchronously. The approach naturally allocates computing resources to visually complex regions, e.g. regions with intense collisions, thereby reducing the overall computational time. The experiments show that our approach is more efficient than the standard method and the method with globally adaptive time steps, especially in highly dynamic scenes.

Keywords Fluid simulation · Adaptive SPH · Individual time steps · Asynchronous

1 Introduction

Physics-based fluid simulations are widely used, for example, in movies, virtual realities and even in computer games, despite the difficulty due to the complexity of fluid behavior. Lagrangian methods based on smoothed particle hydrodynamics (SPH), which have been gaining increased interest in graphics community, are able to successfully simulate complex scenes with versatile effects [1]. To produce appealing visual results with small-scale details, SPH fluids demand a high discretization resolution, i.e., large number of particles. The computational expenses of simulating millions of particles are, however, too large for practical use [2]. To cope with the increasing demand for more detailed fluids, adaptive methods, that adapt either the spatial resolution or the sampling in time, have been proposed. They follow the idea to allocate computing resources to regions with complex flow behavior.

Space adaptive methods [3–5] dynamically exchange particle sets. Large particles are divided into smaller ones when a high discretization resolution is needed, and vice versa. In contrast to homogeneous fluids, these adaptively sampled particle fluids use fewer particles to produce the similar details and thus are more efficient. However, difficulties exist in reproducing quantity when refining particles. Furthermore, while neighborhood searching is usually the bottleneck [2], non-uniform smoothing radii make it slower since hierarchical data structures such as kd-trees have to be used [1].

As an alternative to adaptive spatial discretization, time adaptive methods adapt the sampling distance in time. Globally adaptive time-stepping methods [6–8] use a single time step adjusted in each simulation step with respect to the CFL condition [6] for all particles. Since globally adaptive time steps update all particles according to the particle that has the current smallest time step, it is not the most efficient. Locally adaptive time-stepping methods [3, 9, 10] use different time steps for particles. In [3], each particle evaluates forces only when needed, according to its current individual time step determined from individual stability conditions.

In this paper, we adopt the idea of [3] for weakly compressible SPH (WCSPH) [11]. To make the simulation

Electronic supplementary material The online version of this article (doi:10.1007/s00521-016-2286-8) contains supplementary material, which is available to authorized users.

✉ Xiaokun Wang
wang1xiao2kun3@163.com

Xiaojuan Ban
banxj@ustb.edu.cn

[1] School of Computer and Communication Engineering, University Science and Technology Beijing, Beijing, China

Published online: 18 April 2016

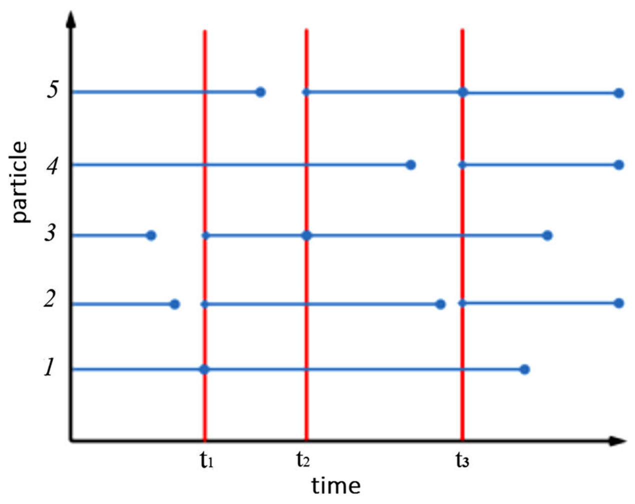

Fig. 1 Asynchronous time integration. The starting points of the line segments are update time, and the length of line segments iterate individual time step

of stiff fluids stable, small time steps are spread to neighbor particles in order to respond to strong shocks. The proposed simulation is very similar to the globally adaptive time-stepping methods, except performing neighborhood searching and forces evaluating for only fraction of all particles in every simulation step. The position, velocity and density of the particle that currently does not need force evaluations are interpolated. We show that the presented simulation algorithm naturally allocates computing resources to visually complex regions with intense collisions (Fig. 1) and thus reduces the computational time.

2 Related work

Since the SPH concept [12] was first introduced to graphics community by the work of Mathieu [6], it has become an active topic. Matthias et al. [7] first applied the SPH method to interactive fluid simulation. Markus et al. [13] use a stiff equation of state (EOS) for weakly compressible SPH (WCSPH). To enforce incompressibility efficiently, iterative solvers are proposed including predictive-corrective incompressible SPH (PCISPH) [12], local poisson SPH (LPSPH) [14] and implicit incompressible SPH (IISPH) [15]. These methods are, however, in the cost of increasing the complexity of programming.

Algorithms to handle the fluid–rigid coupling problems were proposed [16, 17]. In order to obtain a satisfied surface, [18–21] made efforts on surface reconstruction. In additional, [22, 23] revised tension model to get the details of surface. Since there are almost no data dependencies, SPH methods generally map well to parallel architectures. [2] presented a parallel SPH implementation on multi-core CPUs, and [24, 25] implemented the SPH on GPUs. [26, 27] used the CPU/GPU asynchronous computing to improve the efficiency of the method. [28, 29] employed motion blur, particle blending texture mapping, and other computer graphics techniques to achieve real-time effect. [30] presented a sleepy algorithm to improve efficiency by ignoring particles appearing to be at rest.

For high-resolution simulations, many adaptive methods have been presented reducing the computational cost. Space adaptive methods, e.g. adaptively sampling methods [3, 4], multi-resolution methods [31] or multi-scale method [5, 32], change local resolution, i.e., particle size, dynamically or couple multiple resolution levels to reduce the number of particles. To reduce the error introduced when changing particle configurations, temporal blending approach [33] smoothly changes quantity fields along with simulation time.

As an alternative to reducing the number of particles, many techniques try to optimize the time steps, either globally or locally. Since the convergence of SPH simulations is bound to CFL condition, globally adaptive time-stepping methods use a single optimal time step for each simulation step. [8] proposed an adaptive time-stepping method for PCISPH, in which the time steps are changed smoothly to overcome stability problems. The idea of using individual time step with each particle was first proposed in [3] for the simulation of deformable objects. In [10], the movement of inactive particles is temporarily restricted. [9] proposed a block-based regional time-stepping method using different time steps for different regions. Compared to [9], our approach uses individual time steps for each particle which is similar to [3] with careful selected stability conditions.

A good introduction into SPH fluid simulation is given in the state-of-the-art report of Markus et al. [1].

3 Methodology

3.1 Basic SPH

In SPH, a fluid is consisted by a set of particles. Each particle *i* has several attributes, namely

m_i	Mass
ρ_i	Density
p_i	Pressure
$\mathbf{x}_i$	Position
$\mathbf{v}_i$	Velocity

When the particles flow with the fluid, the relations between these attributes are governed by governing equations. For WCSPH [11, 12], the governing equations are

$$\rho_i = \sum_j m_j W_{ij} \tag{1}$$

$$p_i = \frac{\rho_0 c_S^2}{\gamma}\left(\left(\frac{\rho_i}{\rho_0}\right)^{\gamma} - 1\right) \tag{2}$$

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} = \mathbf{v}_i \tag{3}$$

$$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\sum_j m_j\left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij} + \mathbf{g} \tag{4}$$

where $W_{ij} = W(\mathbf{x}_i - \mathbf{x}_j, h)$ is the kernel function with h the smooth radii, j iterates all the neighbors, $\rho_0 = 1000, c_S, \gamma = 7$ are constants, ∇ is the gradient operator and $\mathbf{g}$ is the external force. For the other force term, e.g. viscous force and surface tension force, we can add them on the right side of Eq. (4). In our implementation, we use the viscous force and kernels presented in [6] and rigid–fluid coupling methods of [16]. The Basic SPH method uses the semi-implicit Euler numerical integration scheme, which is illustrated in Algorithm 1.

Algorithm 1 Basic SPH

```
1: while animating do
2:     for all particle $i$ do
3:         find neighbors $j$
4:     for all particle $i$ do
5:         compute $\rho_i, p_i$ (e.g. Eq. (1), (2))
6:     for all particle $i$ do
7:         compute forces (e.g. Eq. (4))
8:     for all particle $i$ do
9:         $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \Delta t \frac{d\mathbf{v}_i}{dt}(t)$
10:        $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t+\Delta t)$
11:    $t = t + \Delta t$
```

The density derivative is given by

$$\frac{\mathrm{d}\rho_\mathrm{i}}{\mathrm{d}t} = \sum_j m_j \mathbf{v}_{ij} \nabla W_{ij} \tag{5}$$

where $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$. We use density derivative to interpolate density.

3.2 Globally adaptive time stepping

The time step of SPH fluids must be constrained for numerical stability and convergence. The Courant–Friedrich–Levy (CFL) condition

$$\Delta t_{\mathrm{CFL}} \le \lambda_\mathrm{v}\left(\frac{h}{v_{\max}}\right) \tag{6}$$

ensure that numerical propagation speed is higher than physical propagation speed, where $v_{\max} = \max_i(||\mathbf{v}_i||)$ is the maximum of the particle velocities, coefficient $\lambda_\mathrm{v} < 1$. In addition, particle forces have to be considered

$$\Delta t_\mathrm{f} \le \lambda_\mathrm{f}\left(\sqrt{\frac{h}{f_{\max}}}\right) \tag{7}$$

where $f_{max} = \max_i\left(\left|\left|\frac{\mathrm{d}\mathbf{v}_\mathrm{i}}{\mathrm{d}t}\right|\right|\right)$ denotes the maximum force per unit mass of particles, $\lambda_\mathrm{f} < 1$. In [8], $\lambda_\mathrm{v} = 0.4, \lambda_\mathrm{f} = 0.25$ are used for PCISPH. Since WCSPH uses a stiff EOS which restricts the time step [2, 34], we use $\lambda_\mathrm{v} = 0.1, \lambda_\mathrm{f} = 0.05$ for globally adaptive time-stepping method. Instead of using a constant time step in Algorithm 1, we can now adjust it dynamically by

$$\Delta t = \min(\Delta t_{\mathrm{CFL}}, \Delta t_\mathrm{f}). \tag{8}$$

4 Individual time stepping

In SPH fluids, each particle only interacts with its neighbors. The convergence of the simulation of each particle is determined locally. In our method, the individual time stepping of each particle is determined by the local condition in its smooth radii. Allowing particles to have different time steps is more efficient than using a globally restricted time step for all particles. Regarding the calculation speed, the algorithm is two times larger compared to globally adaptive time-stepping algorithm and six times larger compared to WCSPH.

We propose a novel asynchronous time integration algorithm based on particle's individual time stepping. The algorithm has the same skeleton as the globally adaptive time-stepping methods do. The main difference is that, at every time step, only the *active* particles will be updated. For those inactive particles, their physical quantities are obtained by linear interpolation.

4.1 Time steps

Our individual time-stepping method computes time step for each particle i according to

$$\Delta t_i = \min_j\left(\lambda_v \frac{h}{||\mathbf{v}_j||}, \lambda_f\sqrt{\frac{h}{||\mathrm{d}\mathbf{v}_j/\mathrm{d}t||}}\right) \tag{9}$$

where j iterates over all neighbors. In contrast to [3], we take the neighbors into account, which make the algorithm stable for stiff fluids. The proposed method, however, requires small coefficients due to the asynchrony. We use $\lambda_\mathrm{v} = 0.05, \lambda_\mathrm{f} = 0.025$, i.e., half of that of globally adaptive time-stepping method. Fortunately, this restriction will be counteracted by the fact that the proposed method updates only a small fraction of all particles in every time step.

4.2 Asynchronous update

As each particle has its own time step, we need to carry out asynchronous time integration for the algorithm. To avoid the computing cost, the time step is determined by current minimize individual time step

$$\Delta t = \min_i (\Delta t_i) \tag{10}$$

where Δt_i is computed by Eq. (9).

The particle i will be updated when it meets the condition

$$t_i^{last} + \Delta t_i < t_{\mathrm{sim}} \tag{11}$$

where t_i^{last} iterates the last time particle i updated, t_{sim} iterates the system time. According to Eq. (9), when the system time is larger than the individual time step, the particle i will be set as active particle and be updated.

SPH method algorithm usually adopts the semi-implicit Euler numerical integration (see Algorithm 1 line 9–10). To suit for asynchronous update, the semi-implicit Euler numerical integrations have the following form:

$$v_i\left(t_i^{last} + \Delta t\right) = v_i\left(t_i^{last}\right) + \Delta t\ a_i\left(t_i^{last}\right) \tag{12}$$

$$x_i\left(t_i^{last} + \Delta t\right) = x_i\left(t_i^{last}\right) + \Delta t\ v_i\left(t_i^{last} + \Delta t\right) \tag{13}$$

For inactive particles, the operation is equal to a interpolating, while for active particles, the operation is equal Alg. 1, a semi-implicit Euler numerical integration.

Asynchronous time integration is shown in Fig. 1. At t_1, the particles 1, 2 and 3 are activated and updated. The particle 3 has the shortest individual time step, which determines global time step. After this time step, the particles 3 and 5 are activated. The time step of particle 5 is shortest and set as global time step. At t_3, the particles 2, 4 and 5 are activated. From here we can see that, in every global time step, only active particles are updated.

In order to synchronize the animation every $\Delta t_{\mathrm{frame}}$, [3] restricted the time step Δt_i to $\Delta t_{\mathrm{frame}}/2^q$ (q is a integer), which decreases the Δt_i. For example $\Delta t_{\mathrm{frame}} = 32$, $\Delta t_i = 7$, finally, Δt_i will be clamped to 4. We found that increasing the synchronize times can improve the stability of the simulation, as shown in Fig. 7. However, the small time steps cost more computational time. So we propose a simple partial synchronization mechanism. The particles that have the same value of $\lfloor \Delta t_i / \Delta t_{\mathrm{sim}} \rfloor$ will be synchronized, which depends on an assumption that the Δt_{sim} is a constant in a short time window (i.e., Δt_i). Experiments show that this mechanism does not add constraints on the Δt and improves the efficiency, as shown in Table 1.

Table 1 Comparison of efficiency on whether using the partial synchronize mechanism

Partial sync	Total comp. time (min)	Avg. Δt_{sim} (avg. active pct.)
No	31	0.21 ms (28 %)
Yes	27	0.23 ms (31 %)

4.3 Algorithm

Our individual time-stepping algorithm is illustrated in Algorithm 2. In our algorithm, each particle i maintains a few additional variables

$\frac{d\rho_i}{dt}$	Density derivative
Δt_i	Time step
Δt_i^{raw}	Raw time step
t_i^{last}	Last updated time

Algorithm 2 Individual time stepping for SPH

```
1: while animating do
2:     select active
3:     for all active particle i do
4:         find neighbors j
5:     for all particle i do
6:         if active then
7:             compute $\rho_i, p_i$ (e.g. Eq. (1), (2))
8:         else
9:             interpolate $\rho_i, p_i$ using $\frac{d\rho_i}{dt}$
10:    for all active particle i do
11:        compute forces (e.g. Eq. (4))
12:        compute $\frac{d\rho_i}{dt}$ (e.g. Eq. (5))
13:        $\Delta t_i^{raw} = \min\left(\lambda_v \frac{h}{\|\mathbf{v}_i\|}, \lambda_f \sqrt{\frac{h}{\|d\mathbf{v}_i/dt\|}}\right)$
14:        $t_i^{last} = t_{sim}$
15:    for all particle i do
16:        $\Delta t_i = \min_j \left(\Delta t_j^{raw}\right)$
17:    $\Delta t_{sim} = \min_i (\Delta t_i)$
18:    for all particle i do
19:        $\Delta t = t_{sim} + \Delta t_{sim} - t_i^{last}$
20:        $\mathbf{v}_i\left(t_i^{last} + \Delta t\right) = \mathbf{v}_i\left(t_i^{last}\right) + \Delta t \frac{d\mathbf{v}_i}{dt}\left(t_i^{last}\right)$
21:        $\mathbf{x}_i\left(t_i^{last} + \Delta t\right) = \mathbf{x}_i\left(t_i^{last}\right) + \Delta t \mathbf{v}_i\left(t_i^{last} + \Delta t\right)$
22:    $t_{sim} = t_{sim} + \Delta t_{sim}$
```

The algorithm has the same skeleton as the globally adaptive time-stepping methods do. The key difference is that for only the *active* particles neighborhood searching and forces evaluating are performed. Particle i is *active* if $t_i^{last} + \Delta t_i < t_{\mathrm{sim}}$. In line 16, small time steps are spread to neighbors, which lets the simulation respond to shocks. In SPH method, neighbor particles' density, position and velocity are necessary for calculating the density and forces, and therefore these attributes of inactive particles need to be interpolated for the active particles. The density is interpolated by the density derivative (e.g. linear interpolation) in line 9. Lines 20–21 are a tricky step. For inactive particles, the operation is equal to a interpolating, while for active particles, the operation is equal Algorithm 1, a semi-implicit Euler numerical integration.

We make some observations. First, line 9 and line 16 are needed to be performed only for neighbors of active particles. Second, for inactive particles, reusing the last neighbor lists (e.g. line 16) leads to small errors which are negligible. Third, since the Algorithm 2 updates particles asynchronously, the momentum of the system is not conserved (see Fig. 6).

5 Results and discussion

5.1 Parallel speedup

We set a sense to understand the distribution of computation of WCSPH algorithm:

The scale of simulation domain	6 m × 6 m × 6 m
The number of fluid particles	22,491
The number of boundary particles	9752
The smooth radii	0.2 m
The radii of fluid particles	0.1 m

The distribution of computation time is shown in Fig. 2. “Search” is the “find neighbors” step. “Pressure” is the “compute density and pressure” step. “Force” is the “compute forces” step. As can be seen from the figure, the neighborhood query adds roughly constant costs per simulation step and costs 60 % of the total time. The time of “compute forces” step account for 31 % of the total time.

The parallel speedup of WCSPH is shown in Fig. 3. The speedup of mainly computation parts, neighbor query and force computing, are linear scaling of the thread numbers. When the thread number adds to 4, which is equal to the numbers of CPU cores, the speedup is 3.71.

The above results indicate that the optimization of the computational efficiency of the SPH algorithm should be focused on the search of the neighboring particles and the calculation of the force. The particle properties of the SPH algorithm make it particularly suitable for parallelization. If we use data parallel architecture on OpenMP, CPU will get the speedup closely to the number of cores in the case of very small changes to the program.

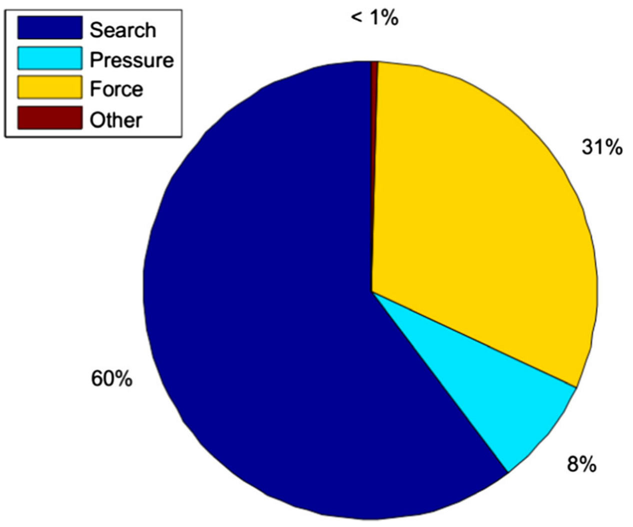

Fig. 2 The distribution of computation time of WCSPH

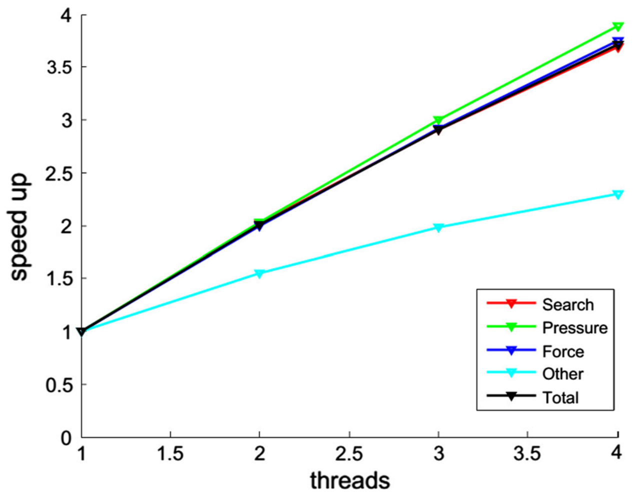

Fig. 3 The speedup of each step of WCSPH for different threads after paralleled

5.2 Performance comparison

The proposed simulation algorithm is compared with basic WCSPH (i.e., with a constant time step) method and also the global time-stepping method. All timings are given for an Intel 3.50 GHz CPU with four cores. The simulation software is parallelized with OpenMP. We reconstruct fluid surface using anisotropic kernels [18], which used PCA [35] to modify isotropic kernels. Images were rendered with Blender. For WCSPH, the density fluctuation is set to 0.01.

Our method naturally allocates computing resources to complex regions with intense collisions. This is

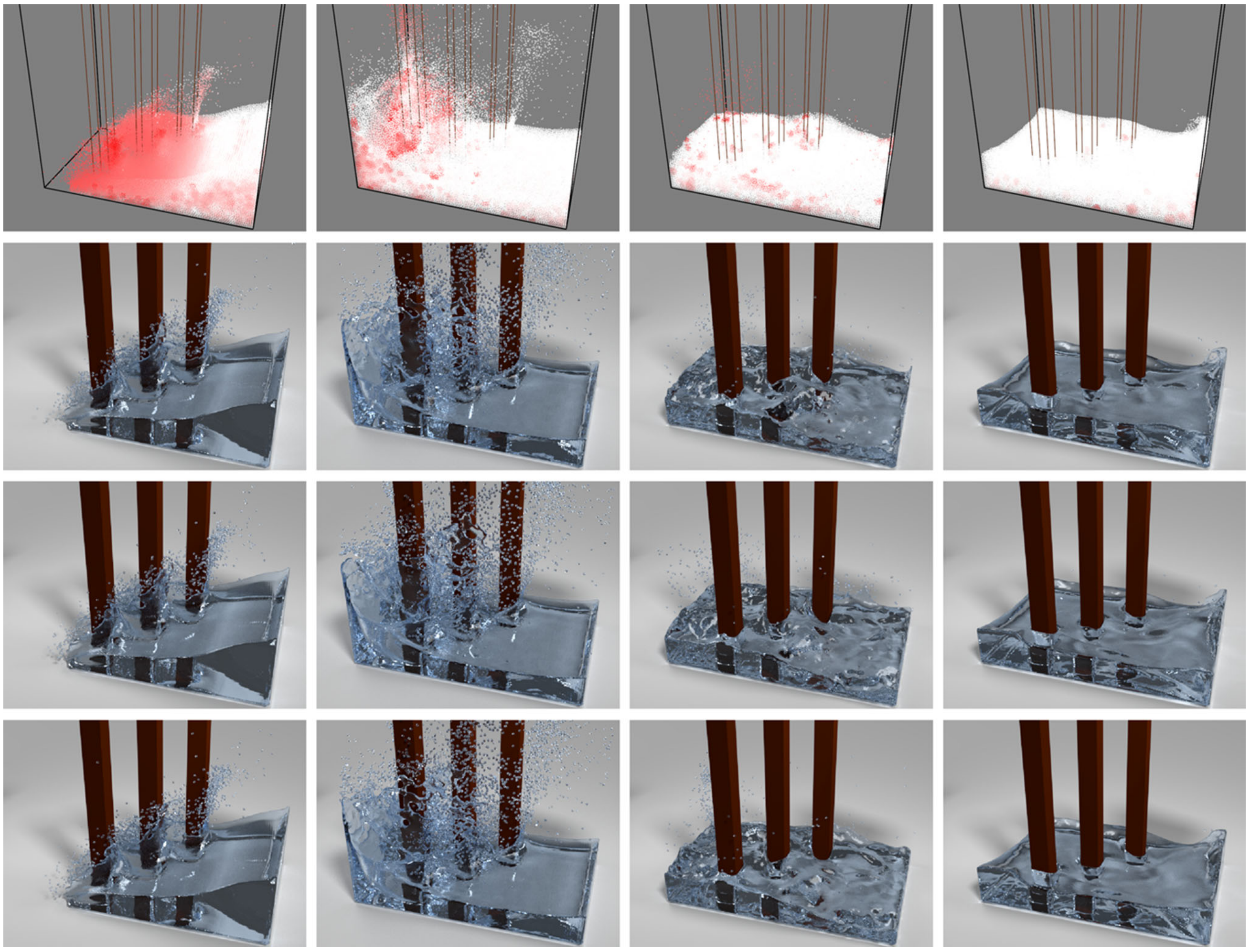

Fig. 4 Breaking dam with obstacles simulated using presented method ($t = 1.2$ s, 2.0 s, 4.0 s and 4.84 s). *Top* Particles colored according to individual step (*red* for small dts and *white* for large dts). *Second row* corresponding frames. *Third row* corresponding frames simulated using globally adaptive stepping method. *Bottom* corresponding frames simulated using basic WCSPH

shown in Fig. 4. The breaking dam scene involves 153K particles. The corresponding video shows that the particles collided with the obstacles intensively in the previous 4 s. The particles in the first row of Fig. 4 are color-coded. Red color means a less time step. As expected, the particles colliding with obstacles or other particles (e.g. drops) have a less time step. From Fig. 5, we can see that when the individual time step is short, the percent of the active particles is also a small value and has the same trend with the time step. These are expected, since less active particles means less computational time.

The performance measurements and simulation data of Figs. 1 and 8 are summarized in Table 2. With our method, the more complex scene can obtain more speedup. Comparing to globally adaptive stepping SPH, our method gets

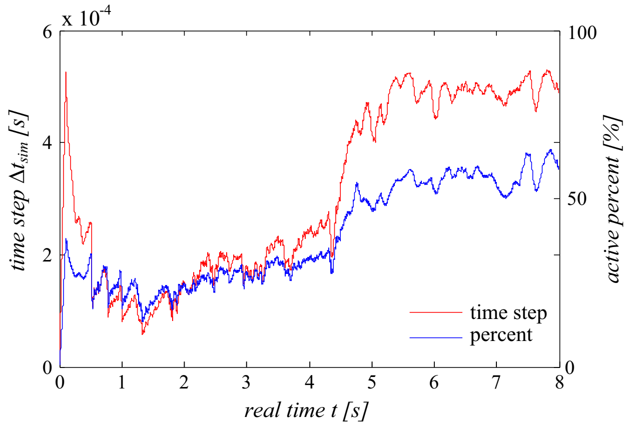

Fig. 5 Time step Δt_{sim} and active percent evolution for breaking dam scene

Table 2 Comparison of individual stepping method and standard SPH methods

Scene	Method	Total comp. time	avg. Δt_{sim} (avg. active pct.)	speedup
Breaking dam with 153K #p	Constant steps	175 min	0.11 ms	–
	Globally adaptive	41 min	0.46 ms	–
	Individual stepping	27 min	0.23 ms (31 %)	1.5 (6.4)
Stirring pool with 1.2M #p	Globally adaptive	32.1 h	0.12 ms	–
	Individual stepping	14.8 h	0.064 ms (29 %)	2.1

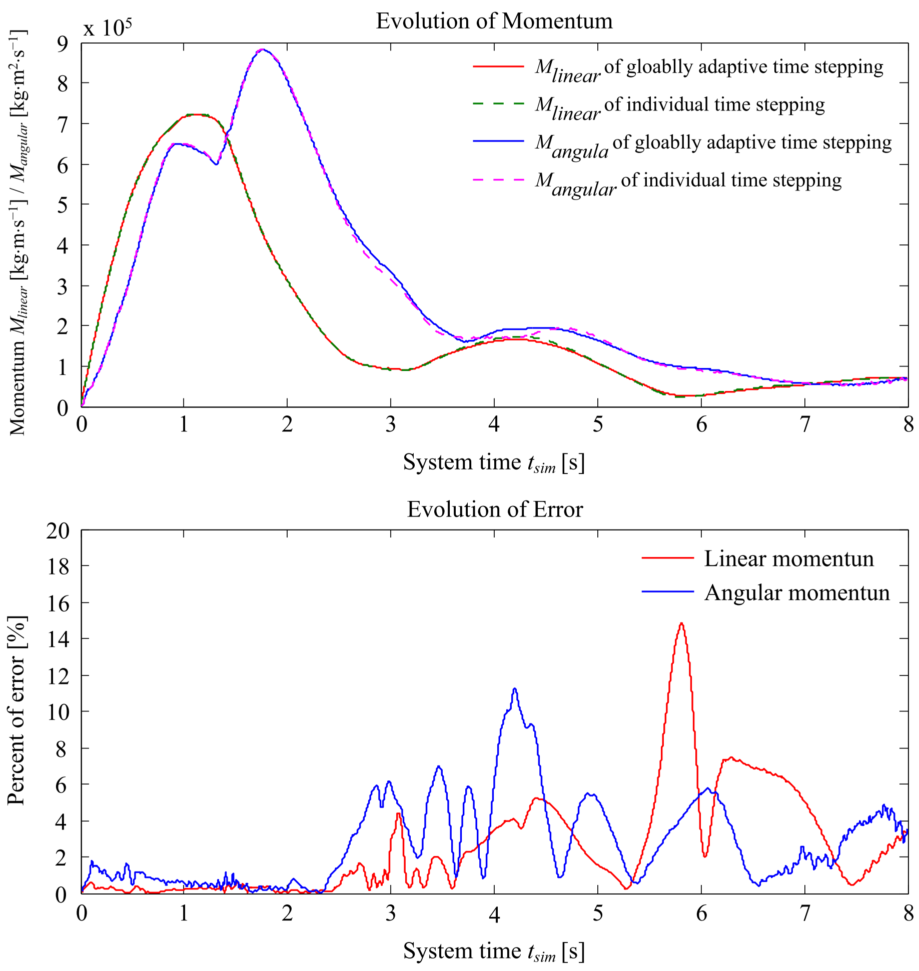

Fig. 6 Evolution of momentum and error for breaking dam scene

a 2.1 times speedup in the scene of stirring pool and gets a 1.5 times speedup in the scene of breaking dam. Comparing to the basic WCSHP, a 6.4 times speedup is gained in the breaking dam scene. For the avg. Δt_{sim}, the basic WCSHP has the smallest value. This is because that its constant time step has to be small enough to handle the collision part. Comparing to the globally adaptive time-stepping method, our method's avg. Δt_{sim} is cut in half, since we chose the λ_{v} and λ_{f} is half of the former due to the asynchrony. In spite of small avg Δt_{sim}, we still obtain a speedup.

$$\mathbf{M}_{linear} = \sum_i m_i \mathbf{v}_i \tag{14}$$

$$\mathbf{M}_{angular} = \sum_i (\mathbf{x}_i - \mathbf{x}^{cm}) \times m_i \mathbf{v}_i \tag{15}$$

The main drawback of our method is non-conservation of momentum. The momentum results for the breaking dam of Fig. 1 are depicted in Fig. 6, and the momentum was calculated by Eqs. 14 and 15, where $\mathbf{x}^{cm} = \sum_i m_i \mathbf{x}_i / \sum_i m_i$ is center of mass. We compare the momentum result with that of the globally adaptive time-stepping method, as shown in

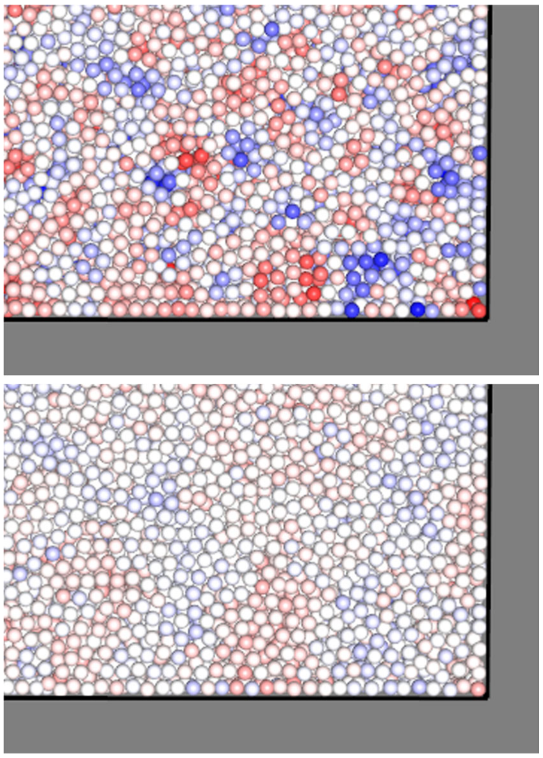

Fig. 7 Comparison of stability on using the partial synchronize mechanism ($t = 0.75$ s of breaking dam scene). *Top* is without the partial sync mechanism, while *bottom* is with the partial sync mechanism. *Red particles* $\rho_i > \rho_0$. *Blue particles* $\rho_i < \rho_0$

Fig. 6. The evolution of momentum shows that the two methods are almost consistent with each other. The evolution of errors shows that our method has tiny errors in the previous 4 s, exactly the time that particles collided with obstacles fiercely. So our method introduces little error when the scene is highly dynamic.

Figure 7 shows the comparison of the stability on whether using the partial synchronize mechanism. The color of particles indicates the density of particle. Red means a big value (i.e., $> \rho_0$), and blue a small value (i.e., $< \rho_0$). From the top figure, we can see that the density, which directly determines the value of pressure forces, is oscillating strongly. Since the pressure forces commonly govern the movement of particles, the oscillation will introduce the instability into the simulation. The partial synchronization mechanism can alleviate oscillation obviously, which is shown in the bottom figure. The corresponding statistics are shown in 1. With the partial synchronization mechanism, the avg. Δ_t increases to 0.23 from 0.21 ms. Consequently, we obtain about 10 % speedup.

5.3 Visual result

The physical behavior and visual results of WCSPH and our method are compared in Figs. 4 and 8. As is shown in Fig. 4, the breaking dam scene involves 153K particles.

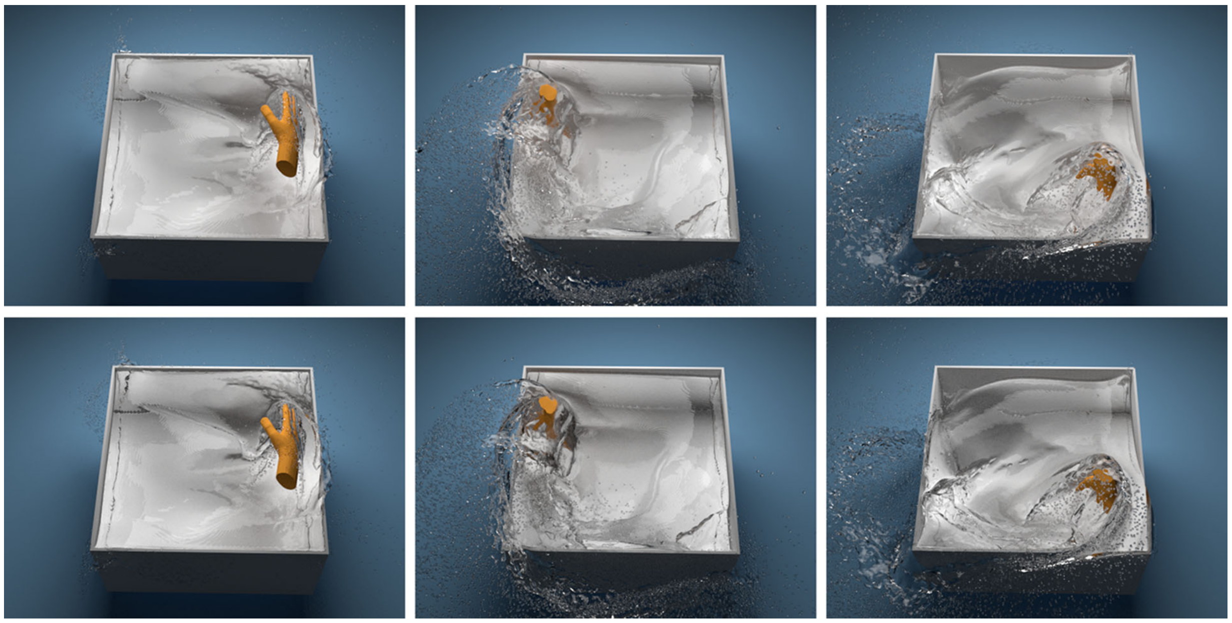

Fig. 8 Stirring pool scene. A hand interacts with a water pool simulated by 1.2M particles ($t = 2.76$, 4.8 and 7.88 s): *top* is the result of individual time-stepping method. *Bottom* is corresponding frames simulated using globally adaptive time-stepping method

The corresponding video shows that the particles collided with the obstacles intensively in the previous 4 s. It can be seen that our methods computations are in full agreement with the WCSPH results with only very minor detail differences. Figure 8 shows a more complex scene with 1.2M particles—a hand stirred the water fiercely. From the side-by-side comparison of the Globally adaptive time-stepping SPH, we can see that the visual results of two methods are in a good agreement (see the videos).

6 Conclusions

We proposed an efficient individual time-stepping method for SPH fluids. Our method updates neighbors and forces of particles only when needed, which naturally allocates computing resources to complex regions. What is more, a partial synchronize mechanism was proposed, which can improve the stability and the efficiency of the simulation. Our method can obtain an obvious speedup when the scene is complex. Despite the method introducing momentum error, it does not alter the visual realism, which is demonstrated in the experiments. Future work would extend the proposed method to PCISPH [34] or IISPH [15].

Acknowledgments This work was supported by National Natural Science Foundation of China (Nos. 61272357, 61300074, 61572075) and Fundamental Research Funds for the Central Universities (FRF-BR-15-058A).

References

1. Ihmsen M, Orthmann J, Solenthaler B, Kolb A, Teschner M (2014) Sph fluids in computer graphics. In: State-of-the-art report eurographics, pp 21–42
2. Markus I, Nadir A, Markus B, Matthias T (2011) A parallel SPH implementation on multi-core CPUs. Comput Graph Forum 30(1):99–112
3. Desbrun M, Cani M-P (1999) Space-time adaptive simulation of highly deformable substances. Technical Report 3829, INRIA
4. Adams B, Pauly M, Keiser R, Guibas LJ (2007) Adaptively sampled particle fluids. ACM Trans Graph Proc SIGGRAPH 26(3):48
5. Solenthaler B, Gross M (2011) Two-scale particle simulation. ACM Trans Graph Proc SIGGRAPH 30(4):81:1–81:8
6. Monaghan JJ (1992) Smoothed particle hydrodynamics. Ann Rev Astron Astrophys 30:543–574
7. Desbrun M, Cani M-P (1996) Smoothed particles: a new paradigm for animating highly deformable bodies. In: Eurographics workshop on computer animation and simulation (EGCAS), pp 61–67. Springer, Berlin
8. Ihmsen M, Akinci N, Gissler M, Teschner M (2010) Boundary handling and adaptive time-stepping for PCISPH. In: Workshop on virtual reality interaction and physical simulation, pp 79–88. The Eurographics Association
9. Goswami P, Batty C (2014) Regional time stepping for SPH. In: Eurographics, pp 45–48. The Eurographics Association
10. Goswami P, Pajarola R (2011) Time adaptive approximate SPH. In: Proceedings of the 8th workshop on virtual reality interaction and physical simulation, pp 19–28. VRIPHYS
11. Monaghan JJ (1994) Simulating free surface flows with SPH. J Comput Phys 110(2):399–406
12. Becker M, Teschner M (2007) Weakly compressible SPH for free surface flows. In: ACM SIGGRAPH/Eurographics symposium on Computer animation, pp 209–217
13. Muller M, Charypar D, Gross M (2003) Particle-based fluid simulation for interactive applications. In: ACM SIGGRAPH/Eurographics symposium on Computer animation, pp 154–159
14. He X, Liu N, Li S, Wang H, Wang G (2012) Local poisson SPH for viscous incompressible fluids. Comput Graph Forum 31(6):1948–1958
15. Ihmsen M, Cornelis J, Solenthaler B, Horvath C, Teschner M (2014) Implicit incompressible SPH. IEEE Trans Vis Comput Graph 20(3):426–435
16. Akinci N, Ihmsen M, Akinci G et al (2012) Versatile rigid–fluid coupling for incompressible SPH. ACM Trans Graph TOG 31(4):62
17. Becker M, Tessendorf H, Teschner M (2009) Direct forcing for Lagrangian rigid–fluid coupling. IEEE Trans Vis Comput Graph 15(3):493–503
18. Yu J, Turk G (2013) Reconstructing surfaces of particle-based fluids using anisotropic kernels. ACM Trans Graph Proc SIGGRAPH 32(1):5:1–5:12
19. Bhatacharya H, Gao Y, Bargteil A (2011) A level-set method for skinning animated particle data. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation, pp 17–24. ACM
20. Akinci G, Ihmsen M, Akinci N, Teschner M (2012) Parallel surface reconstruction for particle-based fluids. In: Computer graphics forum, vol 31, pp 1797–1809. Wiley Online Library
21. Zhou ZH, Zhao JW, Cao FL (2013) Surface reconstruction based on extreme learning machine. Neural Comput Appl 23(2):283–292
22. Akinci N, Akinci G, Teschner M (2013) Versatile surface tension and adhesion for SPH fluids. ACM Trans Graph Proc SIGGRAPH 32(6):182
23. Yu J, Wojtan C, Turk G, Yap C (2012) Explicit mesh surfaces for particle based fluids. In: Computer graphics forum, vol 31, pp 815–824. Wiley Online Library
24. Goswami P, Schlegel P, Solenthaler B, Pajarola R (2010) Interactive sph simulation and rendering on the GPU. In: Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation, pp 55–64. Eurographics Association
25. Valdez-Balderas D, Domnguez JM, Rogers BD, Crespo AJC (2013) Towards accelerating smoothed particle hydrodynamics simulations for free surface flows on multi-GPU clusters. J Parallel Distrib Comput 73(11):1483–1493
26. Domnguez JM, Crespo AJC, Gesteira MG (2013) Optimization strategies for CPU and GPU implementations of a smoothed particle hydrodynamics method. Comput Phys Commun 184(3):617–627
27. Domnguez JM, Crespo AJC, Valdez-Balderas D, Rogers BD, Gomez-Gesteira M (2013) New multi-GPU implementation for smoothed particle hydrodynamics on heterogeneous clusters. Comput Phys Commun 184(8):1848–1860
28. Chen JX, Fu X, Wegman J (1999) Real-time simulation of dust behavior generated by a fast traveling vehicle. ACM Trans Model Comput Simul 9(2):81–104
29. Chen JX, Lobo NV (1995) Toward interactive-rate simulation of fluids with moving obstacles using Navier–Stokes equations. Graph Models Image Process 57(2):107–116
30. Nie X, Chen L, Xiang T (2014) An efficient sleepy algorithm for particle-based fluids. Int J Comput Games Technol 2014:1–8
31. Keiser R (2006) Multiresolution particle-based fluids. ETH Dep Comput Sci 31(6):17971809

32. Horvath CJ, Solenthaler B (2013) Mass preserving multi-scale SPH. Pixar Technical Memo 13-04, Pixar Animation Studios
33. Orthmann J, Kolb A (2012) Temporal blending for adaptive SPH. Comput Graph Forum 31(8):2436–2449
34. Solenthaler B, Pajarola R (2009) Predictive–corrective incompressible SPH. ACM Trans Graph 28(3):40
35. Rosipal R, Girolami M, Trejo LJ et al (2001) Kernel PCA for feature extraction and de-noising in nonlinear regression. Neural Comput Appl 10(3):231–243