INTERNATIONAL JOURNAL OF NUMERICAL MODELLING: ELECTRONIC NETWORKS, DEVICES AND FIELDS
Int. J. Numer. Model. 2015; **28**:254–274
Published online 25 May 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/jnm.2003

Lagrangian particle-based simulation of fluid–solid coupling on graphics processing units

Xuqiang Shao, Zhong Zhou*,†, Jinsong Zhang and Wei Wu

State Key Laboratory of Virtual Reality Technology and System, Beihang University, Beijing 100191, China

SUMMARY

Lagrangian particle method has been widely used in computer physics and graphics; however, numerically solving the partial differential physical equation on a great number of particles is a computationally complex task. In this paper, a unified particle method on graphics processing units is proposed to simulate fluid–solid interaction with large density ratio interactively. Motivated by microscopic molecular dynamics, we consider the solid object as a particular fluid limited to solid motions; therefore, fluid–solid interaction as well as solid–solid interaction could be solved directly using multiphase weakly compressible smoothed particle hydrodynamics solvers. And then, we present a momentum-conserving particle collision handling scheme to prevent fluid penetrating into solid objects. In the simulation, a measure of particle densities is used to handle density discontinuities at fluid–solid interfaces, and consequently, new formulations for density-weighted inter-particle pressure and viscous forces are derived. Moreover, to realistically simulate various small-scale interaction phenomena such as water droplets flowing on solids' surfaces, a surface tension model that uses density-weighted color gradient and can obtain a stable and accurate surface curvature is employed to capture the interfacial fluid–solid tensions. Because all of the computation is carried out on graphics processing unit and no CPU processing is needed, the proposed algorithm can exploit the massive computational power of graphics processing unit for interactive simulation with a higher particle resolution. The experiment results show that our method can simulate realistic fluid–solid couplings at interactive frame-rates even for up to 126 k particles. Copyright © 2014 John Wiley & Sons, Ltd.

Received 25 September 2013; Revised 20 February 2014; Accepted 10 April 2014

KEY WORDS: graphics processing units; physics-based modeling; fluid–solid coupling; Lagrangian particle method; smoothed particle hydrodynamics

1. INTRODUCTION

Nowadays, physics-based modeling of natural phenomena by solving complex partial differential equation has been studied extensively in manufacture and entertainment industry, and various applications could be found in computational physics, computer games, and special effects. Among many physical phenomena, fluid–solid coupling with high density ratio is common in our daily life, for example, a piece of iron sinks into a pool of water and a rubber duck floats on the water surface.

The coupling processes between fluid and solid are physically complex and difficult to simulate. Currently, the coupled models that usually couple Eulerian grid-based fluid and Lagrangian solid together [1–3] have been widely employed. But in this kind of approach, various simulated effects and materials are usually constrained by interfaces between different models. In addition, the low computational efficiency of coupled models is unsuitable for interactive applications. It is highly desirable to have a unified framework that can handle different types of materials and eliminate the need to define an interface for coupling different solid and fluid models. Recently, to simplify the fluid–solid coupling, the fully particle-based approaches [4, 5] have been put forward to simulate fluids, solids, and phase

*Correspondence to: Zhong Zhou, 6863 Box, Xueyuan Road, Haidian District, 100191, Beijing, China.
†E-mail: zhouzhong2011@gmail.com

transitions in a unified way. However, the density discontinuities and surface tension at interfaces are not handled in particle-based fluid–solid coupling.

On the basis of the work of [5], we propose an interactive particle-based method to model the phenomena arising from fluid–solid coupling with high density ratio. In the method, we think of the solid object as a particular fluid limited to solid motions; therefore, mutual interactions could be calculated through solving the Navier–Stokes equations based on any multiphase smoothed particle hydrodynamics (SPH) solver. And a momentum-conserving particle collision handling scheme is proposed to prevent fluid leaking into solid. To handle density discontinuities at fluid–solid interfaces, we employ a measure of particle densities, which was used in the work of [5] to handle discontinuities in density contrast fluid–fluid interaction. In addition, surface tension is an important factor in small-scale fluid simulations such as water droplets flowing on the surface of solid. In our approach, a surface tension model that can obtain a stable and accurate surface curvature is employed to capture small-scale details in fluid–solid coupling. The method uses a density-weighted color gradient formulation to reflect an asymmetrically distributed surface tension force.

However, because of the large computational demands that arise from numerically solving the complex partial differential equation, the fast simulation of particle-based fluid–solid coupling presents a great challenge. In a practical application, producing high quality fluid–solid coupling requires hundreds of thousands of particles and takes several hours or days to compute a single frame. Recent advancement of parallel computing environments with graphics processing units (GPUs) has significantly promoted progress in scientific computation performance. Through the recent development of tools such as Compute Unified Device Architecture (CUDA) and Open Computing Language (OpenCL), it has become possible to fully utilize the bandwidth and computational power they contain in many fields, such as computational electromagnetics [6–9] and computational fluid mechanics [10]. In this paper, enabling interactive simulation with a higher particle resolution, we fully execute all steps of our unified particle method on the GPU by using CUDA to avoid any CPU–GPU transfer overhead. The performance data show that our GPU implementation is many times faster than the CPU implementation and is able to simulate 126 k particles at interactive frame rates.

The main contributions of our paper can be summarized as follows:

(i) A particle-based framework considering solid as a type of particular fluid limited to solid motions and directly calculating mutual interactions through solving the Navier–Stokes equations of fluid with the multiphase SPH method.
(ii) A momentum-conserving particle collision handling scheme is put forward to prevent fluid leaking into the solid objects.
(iii) A surface tension model capturing the interfacial liquid–solid tensions is employed to realistically simulate a variety of small-scale water–solid coupling phenomena such as water droplets.
(iv) A CUDA-based parallel algorithm for the entire simulation pipeline is designed for time performance improvements.

2. RELATED WORK

In computer physics and graphics, physically based animation of fluids such as smoke and water has received considerable attention in recent years. Numerous techniques have been proposed by using both Eulerian and Lagrangian particle approaches. Two-way coupling between solid and fluid is a typical issue in fluid simulation, and the current solving approaches are mainly coupling models. Genevaux *et al*. [2] proposed a method to simulate the interaction between mass-spring solids and a Eulerian grid fluid, with a communication interface between the two phases. But the nodes of a mass-spring network are not quite well suited for the application of coupling forces. Carlson *et al*. [11] simulated coupled fluid and rigid bodies with distributed Lagrange multipliers. Their method considers rigid bodies as fluid on a grid and projects velocity in those regions back to the rigid motion with careful additions to the body force to account for the density ratio between solids and fluids. However, the method cannot stably handle light solids and fails in some cases, allowing fluid to erroneously leak into the solid objects. To simulate the two-way coupling between the Eulerian fluid and finite element or finite

DOI: 10.1002/jnm

difference elastic solid, Chentanez *et al*. [1] enforced coupling constraints by combining both the pressure projection and implicit integration steps into one set of simultaneous equations. Mosher *et al*. [3] proposed a novel solid–fluid coupling method that utilizes the standard Eulerian grid for the fluid and Lagrangian mesh for the solid. Their method treats the coupled system in a fully implicit manner, making it stable for arbitrary time steps, large density ratios, and so on. But it takes from a few minutes to several hours per frame. Müller *et al*. [12] developed a method in which SPH fluid particles interact with Lagrangian meshes by adding boundary particles to the surface of the mesh. Because the density of generated particles varies among polygons, which do not have a uniform surface area, it does not guarantee the constant particle density near the wall boundary. In conclusion, for these coupling models, the interfaces between different objects always limited the variety of the handled effects and materials. Large density ratios even lead to severe numerical instabilities.

Nowadays, several particle-based frameworks that combine simulation methods of fluid, rigid object, deformable object, and fluid–solid coupling are presented. Müller *et al*. [13] put forward a complete particle-based approach to simulate elastic, plastic, and melting solids, in which a moving least squares method is employed to compute interparticle forces. Keiser *et al*. [4] improved the aforementioned method and merged the fluid governing equations with the equations of deformable solids to compute solid deformation, fluid flow, and phase transition. Solenthaler *et al*. [5] can handle coplanar and coarsely sampled particle configurations by adopting SPH method to evaluate the deformation field's Jacobian. However, the aforementioned methods only concentrate on phase transitions. On the basis of the work of [5], Toon *et al*. [14] presented the two-way coupling of a fluid to thin deformable shells in a unified particle model using explicit collision handling to avoid leaks. Furthermore, by altering the local reference shape definition, their method is able to perform SPH cloth simulations. But the density discontinuities and surface tension at interfaces are not handled. Iwasaki *et al*. [15] presented a particle-based approach on GPU to simulate the freezing and melting phenomena and ice-water interactions, in which a novel interfacial tension model is put forward to handle water flowing on the ice surface and the formation of water droplets. Müller *et al*. [16] introduced an interface force to model fluid–fluid interaction based on standard SPH method. Compared with standard SPH, Solenthaler *et al*. [17] presented a new formulation to handle density discontinuities at interfaces between multiple fluids correctly without increasing the computational costs.

The particle-based Lagrangian approaches are playing a more and more important role in recent researches on physical simulation. Among these methods, SPH is one of the most promising methods. Desbrun *et al*. [18] introduced SPH to the graphics community for the simulation of highly deformable solids. Stora *et al*. [19] modeled the flow of lava using SPH method through coupling viscosity with temperature. Müller *et al*. [20] showed that SPH could produce compelling fluid simulations at interactive rates, which led to a large body of follow-on work. Up to now, researchers have used SPH to model such phenomena as viscoelasticity [21], viscoplasticity [22], incompressible flow [23–25], solid–fluid coupling [4, 5], and fluid–fluid interaction [16, 17]. Adams *et al*. [26] demonstrated adaptive sampling in SPH simulations. Along with the great advance of GPU, SPH model can be integrated with general-purpose computation on GPU techniques easily [10, 27]. In addition, Zhang *et al*. [28] performed adaptive SPH simulations on the GPU.

3. PARTICLE-BASED FRAMEWORK

In this paper, our framework solves motion equations of fluids and solids using the SPH numerical method [20] proposed by Müller *et al*. in computer graphics.

3.1. SPH

SPH is an interpolation method for particle systems. With SPH, field quantities that are only defined at discrete particle locations can be evaluated anywhere in space. For this purpose, SPH distributes quantities in a local neighborhood of each particle using radial symmetrical smoothing kernels. According to SPH, a scalar quantity $A(\mathbf{x_i})$ is interpolated at location $\mathbf{x_i}$ by a weighted sum of contributions from all particles:

DOI: 10.1002/jnm

$$A(\mathbf{x}_i) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{x}_{ij}, h), \tag{1}$$

where j iterates over all particles, $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$, m_j is the mass of neighboring particle j, and ρ_j its density. $W(\mathbf{x}, h)$ is a smoothing function, which is a smoothed, symmetric, and normalized function with finite support, that is, $\int W(\mathbf{r}, h) d\mathbf{r} = 1$ and $W(\mathbf{r}, h) = 0$ for $|\mathbf{r}| > h$.

In fluid and solid governing equations, derivatives of quantity field need to be calculated. Using the SPH interpolation method, the derivatives will only influence the smoothing kernel function. Thus, the first-order gradient and second-order Laplacian of the smoothing quantity field $A(\mathbf{x})$ are

$$\nabla A(\mathbf{x}_i) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(\mathbf{x}_{ij}, h), \tag{2}$$

$$\nabla^2 A(\mathbf{x}_i) = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(\mathbf{x}_{ij}, h). \tag{3}$$

3.2. Fluid simulation

Our particle-based fluid framework is based on the researches in [16, 20, 23]. Fluid motions are usually governed by the famous Navier–Stokes equations

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \mu \nabla^2 \mathbf{v} + \rho \mathbf{g}, \tag{4}$$

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{v}, \tag{5}$$

where p is fluid pressure, $\mathbf{v}$ is velocity, $\mathbf{g}$ the gravity acceleration, and μ fluid viscosity. Equation (4) formulates the momentum conservation law, and Equation (5) assures conservation of mass.

The method applies Equation (1) to the summation density of a particle p_i located at $\mathbf{x}_i$, and obtains the so-called mass density

$$\rho_i = \rho(\mathbf{x}_i) = \sum_j m_j W(\mathbf{x}_{ij}, h). \tag{6}$$

Then, substitute Equations (2) and (3) into the first formula of Navier–Stokes equations and symmetrize; we can obtain two forces exerted on a particle p_i

$$\mathbf{F}_i^{pressure} = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W(\mathbf{x}_{ij}, h), \tag{7}$$

$$\mathbf{F}_i^{viscosity} = \mu \sum_j m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \nabla^2 W(\mathbf{x}_{ij}, h). \tag{8}$$

where $\mathbf{F}_i^{pressure}$ and $\mathbf{F}_i^{viscosity}$ are forces per unit volume, and we use the kernels $W(\mathbf{r}, h)$ adopted in [18]. The fluid pressure for p_i is evaluated by Tait's equation used in [23]

$$p_i = \frac{\kappa \rho_0}{\gamma} \left(\left(\frac{\rho_i}{\rho_0} \right)^{\gamma} - 1 \right), \tag{9}$$

where ρ_0 is the rest density, k represents the fluid stiffness, and $\gamma = 7$ enforces weakly compressible fluid.

DOI: 10.1002/jnm

3.3. Solid simulation

We simulate solid objects based on the work of [5]. The rigid objects are represented by particles, and forces exerted on all particles can be accumulated to a total force and torque to control the motion of rigid bodies.

Then, we have to deal with rotation explicitly for restricting the motion of a solid object to the rigid motion. To do this, we calculate a torque vector τ using

$$\tau_i = (\mathbf{x}_i - \mathbf{x}^{cm}) \times \mathbf{F}_i, \tag{10}$$

where $\mathbf{x}^{cm}$ is the mass center of an object and $\mathbf{F}_i$ is the total force acting on the ith solid particle.

The total force exerted on an object is computed by $\mathbf{F}_{total} = \sum_i \mathbf{F}_i$, and the total torque is evaluated by $\tau_{total} = \sum_i \tau_i$. Then, we perform time integration through iterating over the rigid body to compute the influence of the torque and force, that is, the influence on the object's position and the angular and linear velocity.

For the simulation of deformable solids, at every particle position, the gradient of displacement $\nabla \mathbf{u}$ from the reference shape of the body is used to compute the strain ε, stress σ, and elastic forces $\mathbf{F}^{elastic}$. Because the approximation of the deformation gradient in this method is only zero-order consistent, it is not rotationally invariant. Instead, rotations are misinterpreted as deformations, resulting in forces that prevent a body from rotating. Becker *et al*. [29] presented a novel corotational SPH formulation for elastic solids. The approximation for the deformation gradient is

$$\nabla \mathbf{u}_i = \sum_j \bar{v}_j \nabla W(\mathbf{x}_{ij}, h) \left(\mathbf{R}_i^{-1} (\mathbf{x}_j - \mathbf{x}_i) - \left(\mathbf{x}_j^0 - \mathbf{x}_i^0 \right) \right)^T, \tag{11}$$

where $\bar{v}_j$ is the body volume of particle j and $\mathbf{R}_i$ the individual rotation matrix for a particle i computed using a polar decomposition of transformation matrix.

Basing on the continuum elasticity theory, the elastic force $\mathbf{F}^{elastic}$ can be defined as the negative gradient of strain energy U with respect to displacement. The force that particle i exerts on its jth neighbor is given by

$$\mathbf{F}_{ji}^{elastic} = -\nabla \mathbf{u}_j U_i = -2\bar{v}_j^2 \left(\mathbf{I} + \nabla \mathbf{u}_i^T \right) \sigma_i \nabla W(\mathbf{x}_{ij}, h), \tag{12}$$

where $\mathbf{I}$ is the identity matrix, $U_i = \frac{1}{2}(\varepsilon_i \cdot \sigma_i)$.

4. FLUID–SOLID COUPLING SIMULATION

In this section, we show how the proposed algorithm calculates two-way interaction between fluid and solid with high density ratio avoiding penetration artifacts.

4.1. Interaction forces

By representing fluids and solids by particles carrying various properties, we adopt unified particle-based method based on SPH to model fluids, solids, and interaction between them. In the standard SPH for a single material, attributes are identical for all particles and can be stored globally, for example, the particle mass m, the density ρ, and the viscosity coefficient μ. Similar to the fluid–fluid interaction method in [16], our unified particle-based model makes each particle carry all those attributes individually. The attributes of a particle are summarized in Table I.

DOI: 10.1002/jnm

Table I. Attributes of each particle.

Attribute	Description	Unit
m	Mass	kg
T	Particle type	1
V	Volume	m^3
$\mathbf{x}$	Position	m
$\mathbf{v}$	Velocity	m/s
$\mathbf{f}$	Accumulated forces	N/m^3
ρ	Density	kg/m^3
C	Color attribute for surface tension	1

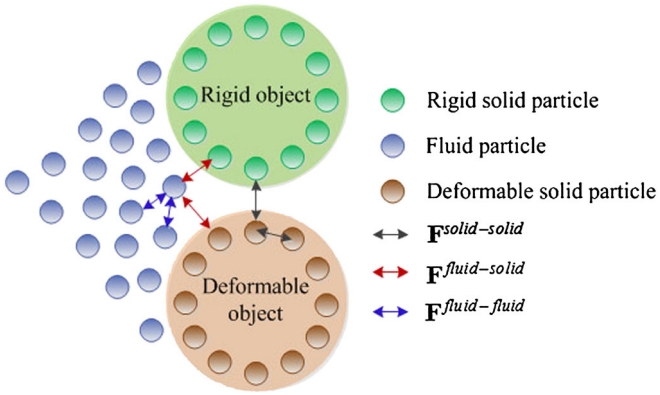

Figure 1. Forces between particles in fluid–solid coupling.

As Figure 1 shows, during fluid–solid two-way coupling simulation, we need to handle three different forms of interparticle forces: the force $\mathbf{F}^{fluid-fluid}$ between fluid particles , the force $\mathbf{F}^{solid-solid}$ between solid particles, and the interactive force $\mathbf{F}^{fluid-solid}$ between fluid particles and solid particles. In the proposed unified model, we consider the solid object as a particular fluid limited to solid motions; therefore, fluid–solid interaction as well as solid–solid interaction could be solved directly using multiphase SPH solvers to solve the fluid governing equations. We evaluated $\mathbf{F}^{fluid-solid}$ in the same way that we evaluated $\mathbf{F}^{fluid-fluid}$, which is equal to the summation of $\mathbf{F}^{pressure}$ and $\mathbf{F}^{viscosity}$ formulated by Equations (7) and (8), respectively. $\mathbf{F}^{solid-solid}$ has two forms: forces between neighbor solid particles from different solid objects and forces between neighbor deformable particles from the same solid object. The former forces are used to couple different solid objects and are also calculated in the same way as $\mathbf{F}^{fluid-fluid}$. The latter are elastic inner forces of deformable solid and are solved using Equation (12).

4.2. *Penetration prevention*

In our method, we only consider the impermeable solid objects. When the velocity difference between fluid and solid or the fluid pressure is too large, only depending on the aforementioned interparticle forces is not enough to overcome penetration. In this section, we propose a momentum-conserving particle collision handling scheme to prevent penetration artifacts at the fluid–solid interfaces, which improves the stability of fluid–solid couplings.

As shown in Figure 2, the influence range of a solid particle is divided into two regions. When the distance between a solid particle i and a fluid particle j is smaller than the summation of their radius $|\mathbf{x}_{ij}| < r_i + r_j$ (white region in Figure 2), we correct the position and velocity of the fluid particle by ensuring the conservation of momentum. By translating the fluid particle j along the direction of vector $\mathbf{x}_{ji}$, we firstly correct its position $\mathbf{x}_j$ as

DOI: 10.1002/jnm

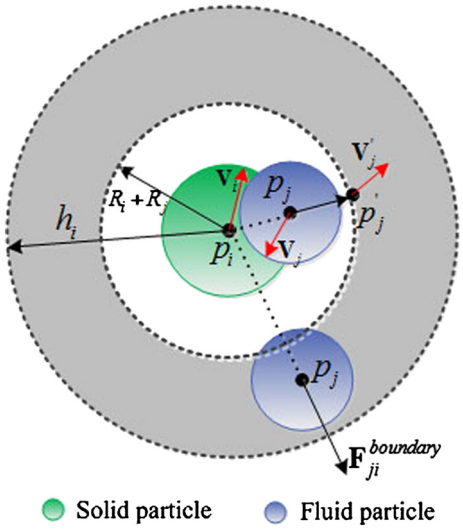

Figure 2. The momentum-conserving particle collision handling scheme.

$$\mathbf{x}_j' = \mathbf{x}_i + (r_i + r_j)\frac{\mathbf{x}_{ji}}{|\mathbf{x}_{ji}|}, \tag{13}$$

where $r = \sqrt[3]{\frac{3V}{4\pi}}$ is the particle radius and $\mathbf{x}_j'$ the displaced position of fluid particle j.

Then, the velocity of j is corrected according to boundary material and the law of momentum conservation. We project the velocities of i and j to the direction and the tangential direction of $\mathbf{x}_{ji}$, then obtain $\mathbf{v}_i^n, \mathbf{v}_i^t, \mathbf{v}_j^n$, and $\mathbf{v}_j^t$. Considering momentum conservation along these two directions, we formulate the following equations

$$m_i\mathbf{v}_i^n + m_j\mathbf{v}_j^n = m_i\tilde{\mathbf{v}}_i^n + m_j\tilde{\mathbf{v}}_j^n, \tag{14}$$

$$m_i\mathbf{v}_i^t + m_j\mathbf{v}_j^t = m_i\tilde{\mathbf{v}}_i^t + m_j\tilde{\mathbf{v}}_j^t, \tag{15}$$

where $\tilde{\mathbf{v}}_i^n$ and $\tilde{\mathbf{v}}_j^n$ denote the unknown velocities after the collision.

To enforce the nonpenetration constraint at the fluid–solid interfaces, we ensure that the corrected velocity components in the direction of $\mathbf{x}_{ji}$ are equal, that is, $\tilde{\mathbf{v}}_i^n = \tilde{\mathbf{v}}_j^n$. Substitute it into Equation (14) and obtain

$$\tilde{\mathbf{v}}_i^n = \tilde{\mathbf{v}}_j^n = \frac{m_i\mathbf{v}_i^n + m_j\mathbf{v}_j^n}{m_i + m_j}. \tag{16}$$

As for the velocity correction in the tangential direction of $\mathbf{x}_{ji}$, we define a variable ∂ to control the different slip conditions.

$$\partial = \frac{\tilde{\mathbf{v}}_i^t - \tilde{\mathbf{v}}_j^t}{\mathbf{v}_i^t - \mathbf{v}_j^t}, \tag{17}$$

where $\partial = 0$ means no-slip in the collision, while $\partial = 1$ states that the collision is free to slip.

DOI: 10.1002/jnm

Combining Equations (15) and (17), we obtain

$$\begin{aligned}\tilde{\mathbf{v}}_i^t &= \frac{(m_i + m_j\partial)\mathbf{v}_i^t + m_j(1-\partial)\mathbf{v}_j^t}{m_i + m_j},\\ \tilde{\mathbf{v}}_j^t &= \frac{(m_j + m_i\partial)\mathbf{v}_j^t + m_i(1-\partial)\mathbf{v}_i^t}{m_i + m_j}.\end{aligned} \tag{18}$$

When the distance between i and j is larger than the summation of their radius $|\mathbf{x}_{ij}| >= r_i + r_j$ (gray region in Figure 2), following [23, 30], we apply a boundary force $\mathbf{F}^{boundary}$ between these two particles

$$\mathbf{F}_{ji}^{boundary} = -\mathbf{F}_{ij}^{boundary} = k_b \frac{m_j}{m_i + m_j} W(\mathbf{x}_{ji}, h_i) \frac{\mathbf{x}_{ji}}{|\mathbf{x}_{ji}|} \tag{19}$$

where k_b is a user-defined coefficient whose unit is 1, the kernel function W is defined as

$$W(\mathbf{x}_{ij}, h) = \frac{1}{|\mathbf{x}_{ij}|} \begin{Bmatrix} \frac{2}{3}, & 0 < q < \frac{2}{3} \\ 2q - \frac{3}{2}q^2, & \frac{2}{3} \le q < 1 \\ \frac{1}{2}(2-q)^2, & 1 \le q < 2 \\ 0, & \text{otherwise}, \end{Bmatrix} \tag{20}$$

where $q = \frac{|\mathbf{x}_{ij}|}{h}$. Therefore, fluid particles approaching solid boundary particles are now first slowed down. Only when there is enough pressure they will actually collide with the solid particles. We notice fluid particles are now able to slide off the solid surface instead of being repelled.

4.3. *Handling discontinuities at interfaces*

The mass density summation Equation (6) for multiphase fluid would become infeasible if the neighbor particles carry different rest densities [17]. For particles close to the interface, the computed density is underestimated if they belong to the material with higher rest density, and overestimated otherwise. This happens because the kernel function smoothes the density and cannot accurately represent sharp

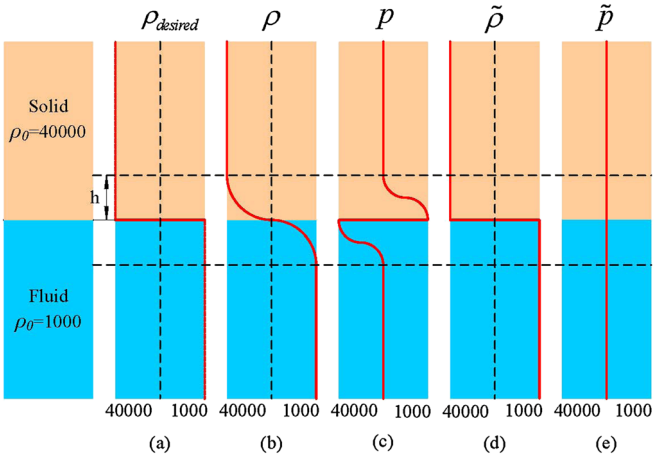

Figure 3. Handling discontinuities at interfaces. (a) Actual desired density discontinuities. (b) Standard SPH smoothes the density at the interface. (c) Erroneous pressures are present near the interface. Our SPH equations using the particle density (d), resulting in densities (e).

DOI: 10.1002/jnm

density changes as it has been desired. This is illustrated in Figure 3(a) and (b). When this density is used to calculate the pressure through the two phase equations of state, the pressure will be very wrong (see Figure 3(c)), leading to a spurious interface tension and a large gap. Even worse, the erroneous pressure forces induce numerical instabilities at the interface, and it is impossible to set up a stable interface in equilibrium.

Our method considers a solid to be a particular fluid restrained to solid motions, so the aforementioned problems also exist in our particle-based model for fluid–solid coupling with high rest density ratio. To handle density discontinuities at fluid–solid interfaces correctly, we replace the mass density formula Equation (6) by a measure of particle density, which was adopted in multiphase fluid solvers with high density ratio in [17]. The idea is to ignore the mass in the computation of the particle density. The continuous and derivable particle density is defined as

$$\delta_i = \sum_{j \in N^{neighbor}} W(\mathbf{x}_{ij}, h). \tag{21}$$

Then, the adapted density $\tilde{\rho}_i$ of a particle is computed by multiplying the particle density by the mass of the particle

$$\tilde{\rho}_i = m_i \delta_i = m_i \sum_{j \in N^{neighbor}} W(\mathbf{x}_{ij}, h) \tag{22}$$

where $N^{neighbor}$ includes neighboring fluid particles and solid particles.

Because neighboring particles contribute to the particle density only by affecting the specific volume of particle i, Equation (22) allows for density discontinuities when there are large density differences between nearby particles. As illustrated in Figure 3(d), when dealing with fluid–solid coupling with different densities, we can achieve a density field reproducing sharp density changes at the interface by using Equation (22). We replace ρ_i by $\tilde{\rho}_i$, yielding the equation for the pressure

$$\tilde{p}_i = \frac{\kappa \rho_0}{\gamma} \left(\left(\frac{\tilde{\rho}_i}{\rho_0} \right)^{\gamma} - 1 \right). \tag{23}$$

Then, we again replace ρ_i by $\tilde{\rho}_i$ and p_i by $\tilde{p}_i$, yielding the final equation for the pressure and viscosity force

$$\mathbf{F}_i^{pressure} = - \sum_{j \in N^{neighbor}} \frac{1}{\delta_j} \frac{\tilde{p}_i + \tilde{p}_j}{2} \nabla W(\mathbf{x}_{ij}, h), \tag{24}$$

$$\mathbf{F}_i^{viscosity} = - \sum_{j \in N^{neighbor}} \frac{\mu_i + \mu_j}{2} \frac{\mathbf{v}_j - \mathbf{v}_i}{\delta_j} \nabla^2 W(\mathbf{x}_{ij}, h). \tag{25}$$

On the basis of the improved mass density, force, and pressure formulations in [17] for multiphase fluid, our method can eliminate all unnatural and spurious interface tension effects when simulating the two-way coupling between fluid and solid with large density ratio (as shown in Figure 3(e)). Our approach is able to deal with rest density ratios of up to 100.

DOI: 10.1002/jnm

4.4. Surface tension model

Surface tension is an important factor for small-scale details in fluid simulations. It is caused by unbalanced molecular cohesive forces in the interfacial region where two phases meet (liquid–air, liquid–solid, or solid–air). If we wish to synthesize a variety of small-scale fluid phenomena including water droplets flowing on the surface of solid objects, surface tension effects become too strong to be neglected. With SPH, there are generally two surface tension models in computational fluid dynamics (CFD): one is the microscopic model based on inter-phase attractive potentials [15, 23] and the other one is the macroscopic model [16, 20]. Although the microscopic model is straightforward, one of the difficulties is that the resulting surface tension does not converge to a fixed value with increasing resolution. On the other hand, the macroscopic surface tension model converges to the exact value with increasing resolution. However, the macroscopic model requires the calculation of curvature, which is difficult to predict, that is, the divergence of the unit interface normal direction. In addition, Wang *et al*. [31] presented a physical approach to enforce contact angles at the intersection of the fluid free surface and the solid, allowing to model many small-scale interaction phenomena including water drops on the object surface in grid-based fluid solvers.

The surface tension model we employed for particle-based two-way fluid–solid interaction with high density ratios is considered to be a variation of the method of [32] for multiphase SPH fluids in CFD, which can obtain a stable and accurate surface curvature. Following the work of [16, 20], the surface tension force can be expressed as a body force

$$\mathbf{F}^{surface} = \alpha\kappa\mathbf{n} = -\alpha\nabla^2 C\frac{\mathbf{n}}{|\mathbf{n}|} \tag{26}$$

where α is surface tension coefficient, C a smoothed color field, $\mathbf{n} = \nabla C/|\nabla C|$ the normal, and $\kappa = -\nabla^2 C/|\mathbf{n}|$ the curvature that is the divergence of the normal.

To calculate surface tension forces between fluid phase and solid phase, we introduce a color function C as

$$C_i^j = \begin{cases} 1, \text{ if } i \text{ and } j \text{ belong to the different phase} \\ 0, \text{ if } i \text{ and } j \text{ belong to the same phase.} \end{cases} \tag{27}$$

Physically, at the fluid–solid interface, the surface tension forces in the phase with high density are much more prominent than those in phase with low density. The interfacial motion is mainly driven by the material with high density. To reflect this behavior, on the basis of Equation (2), the following density-weighted gradient of the color function is used

$$\nabla C_i = \delta_i \sum_j \left(\frac{1}{\delta_i^2} + \frac{1}{\delta_j^2}\right)\frac{\tilde{\rho}_i}{\tilde{\rho}_i + \tilde{\rho}_j} C_i^j \nabla W(\mathbf{x}_{ij}, h) \tag{28}$$

where δ_i is calculated by Equation (21).

The kernel function W in Equation (28) should have the Dirac delta-function property $\lim_{h\to 0} W(\mathbf{r}, h) = \delta(\mathbf{r})$ and have compact support to allow for numerically efficient approximations of the field quantities and gradients. In this work, we use the quintic spline function presented by Morris *et al*. [33] with a compact support of $3h$.

$$W(s) = \frac{10}{7\pi}\begin{Bmatrix} 1 - \frac{3s^2}{2} + \frac{3s^3}{4}, & 0 \le s < 1 \\ \frac{(2-s)^3}{4}, & 1 \le s < 2 \\ 0, & 2 \le s \end{Bmatrix} \tag{29}$$

where $s = |\mathbf{x}_{ij}| \,/\, h$.

DOI: 10.1002/jnm

To calculate the fluid–solid interface curvature, on the basis of the Taylor series expansion, the continuous vector function $\phi(x) = \nabla C$ is approximated as

$$\phi(\mathbf{x}) = \phi(\mathbf{x}_i) + \nabla\phi(\mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i) + O\left(|\mathbf{x} - \mathbf{x}_i|^2\right) \tag{30}$$

where $\nabla\phi(\mathbf{x}_i) = \partial\phi(\mathbf{x}_i) \,/\, \partial\mathbf{x}_i$.

Neglecting the second and higher-order terms of Equation (30), we can obtain $\nabla\phi(\mathbf{x}_i) = [\phi(\mathbf{x}) - \phi(\mathbf{x}_i)] \otimes (\mathbf{x} - \mathbf{x}_i)^{-1}$. Then, we multiply both sides of the aforementioned equation by the gradient of the kernel function $\nabla W(\mathbf{x} - \mathbf{x}_i)$ and integrate over the entire domain and make discretization; the summation form of the corrected gradient is formulated as

$$\nabla\phi(\mathbf{x}_i) = \left[\sum_j \phi_{ji} \otimes \nabla W(\mathbf{x}_{ji}) \frac{1}{\delta_j}\right] \left[\sum_j \mathbf{x}_{ji} \otimes \nabla W(\mathbf{x}_{ji}) \frac{1}{\delta_j}\right]^{-1} \tag{31}$$

where $\phi_{ji} = \phi(\mathbf{x}_j) - \phi(\mathbf{x}_i)$.

The second term at the right side of Equation (31) is a $d \times d$ matrix, where d is the number of spatial dimensions. This matrix must first be constructed and then inverted. On the basis of the work of [34], we approximate the matrix by

$$\left[\sum_j \mathbf{x}_{ji} \otimes \nabla W(\mathbf{x}_{ji}) \frac{1}{\delta_j}\right] \approx \frac{\mathbf{I}}{d} \left(\sum_j \mathbf{x}_{ji} \cdot \nabla W(\mathbf{x}_{ji}) \frac{1}{\delta_j}\right). \tag{32}$$

where $\mathbf{I}$ is the identity matrix.

Then, plugging Equation (32) into (31) and taking the trace of Equation (31), we obtain the following approximated divergence

$$\nabla \cdot \phi_i = d \frac{\sum_j \phi_{ij} \cdot \frac{\nabla W(\mathbf{x}_{ij})}{\delta_j}}{\sum_j |\mathbf{x}_{ij}| \frac{\partial W}{\partial |\mathbf{x}_{ij}|} \frac{1}{\delta_j}}. \tag{33}$$

Using the aforementioned formulation in which only two simple summations are required, we can obtain a stable and accurate curvature of the interface.

5. GPU IMPLEMENTATION

Our particle-based model is very suitable for GPU implementation, because each particle can be processed in a separate unit, and only their neighbor information is needed. Figure 4 shows the GPU implementation pipeline of each time step.

Initialization To fully exploit the performance of the GPU, all of the physical values of the particles are stored in GPU memory. For updating, the new values will not affect particles not yet processed, our implementation requires four double-buffered CUDA arrays for radix-sort, density and pressure, position, and velocity. Because it could be the case that global memory accesses are not coalesced, we obtain the old attribute values from CUDA texture arrays (i.e., particle positions, densities), while we write updated values into the double-buffered arrays. These two sets of CUDA arrays reverse their role every frame as readable textures and writable global memory. Because force values are not required outside the force kernel, we do not need any global memory for them. Also, we avoid allocating

DOI: 10.1002/jnm

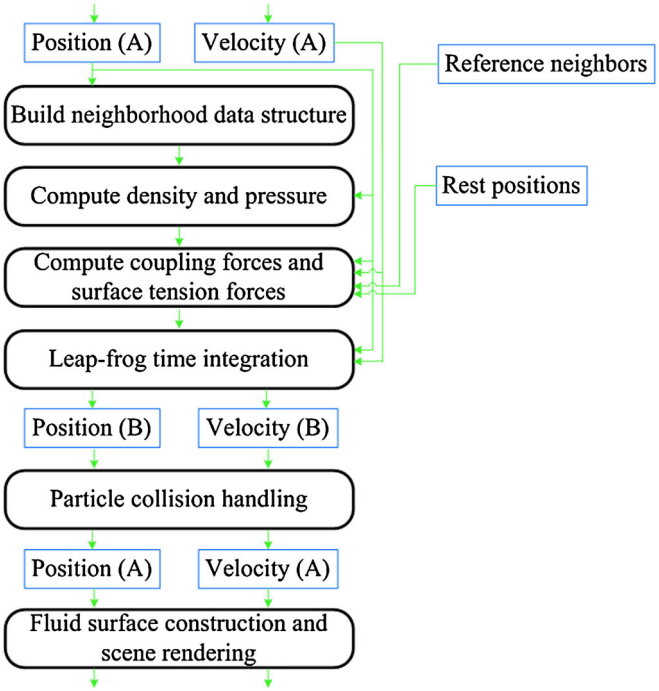

Figure 4. The GPU implementation pipeline of each time step. Blue rectangles represent data, black rounded rectangles represent operations, and green directed line segments represent writing and reading data.

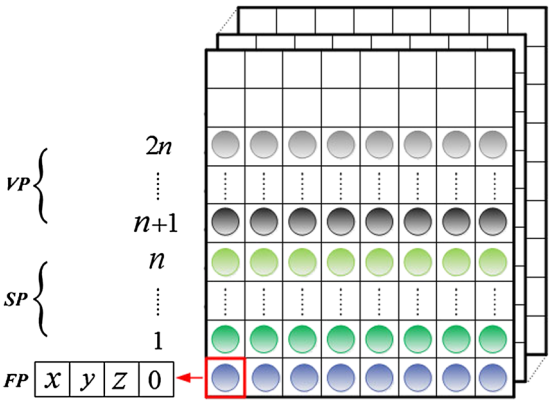

Figure 5. Physical values stored on the GPU. FP, SP, and VP denote fluid particle, solid particle, and mesh vertex, respectively.

separate memory for block computations by doing it in the same CUDA array as we use for radix-sort. Once updated particle positions are copied into the position array, the radix-sort array is free and can be used for block computations. Hence, no extra space is in fact needed for blocks maintenance. To make unified handling, we put forward to store the physical properties of all particles in the same CUDA arrays (see Figure 5) and use an additional flag value T (fluid particle (FP): 0; solid particle (SP): $1, \ldots, n$; mesh vertex (VP): $n+1, \ldots, 2n$; n is the number of solid objects) stored in the fourth element of $float4$ to distinguish to which object and type each particle belongs. For example, if the flag value of a particle i satisfies $T_i > n$ and $T_i \% n == 3$, this particle is a VP of the third solid object.

Block dimensions are imposed as 8*8*8, with a total of 512 threads per block, half of the maximum possible but chosen as to allow symmetry alongside the three axes. The grid dimensions are based on the total number of particles and their initial position, dividing the number of particles per length by the block's length, effectively taking advantage of the three-dimensional domain of the problem.

DOI: 10.1002/jnm

Neighborhood search For particle-based simulations, the calculations of the physical values require a search for neighbors of each particle, as shown in Equations (21) and (22). Our method uses CUDA-accelerated uniform spatial hashing algorithm to accelerate the search process. Different from the previous work where all particles have uniform support radius, the support radius of each particle in our model is distinct. We use max support radius $h_{\max}$ of all particles to divide the simulation domain into a uniform grid. This scheme would increase neighbor candidates for the particle with smaller support radius, but it has almost no impact on the efficiency. The particle p_j is considered as a neighbor of p_i if and only if $|\mathbf{x}_i - \mathbf{x}_j| \leq \max(h_i, h_j)$. Particles in the same cell will then lie consecutively in the linear buffer, and finding neighbors is simply a matter of iterating over the correct indices in the buffer. For the sorting we use the fastest radix sort available for the GPU [35] at the time of implementation.

Forces computations The calculations of forces between particles are performed in parallel by launching a CUDA thread for each particle. We first launch a kernel to calculate the forces between particles, which are the summation of $\mathbf{F}^{pressure}$ and $\mathbf{F}^{viscosity}$ formulated as Equation (24) and (25). The calculated forces include $\mathbf{F}^{fluid-fluid}$, interaction force $\mathbf{F}^{fluid-solid}$, and $\mathbf{F}^{solid-solid}$ between particles from different solids. For avoiding undesired calculations of the aforementioned forces between solid particles from the same object, we use particle type T stored by each particle to do it. If a pair of particles all belong to the same solid object, the calculation should be given up in the kernel.

The aforementioned forces exerted on solid particles are treated as external forces when we calculate the motions of solids. In our method, the motions of rigid solids and deformable solids are simulated on GPU in different ways. To simulate the motions of the rigid solids, the total force $\mathbf{F}_{total}$ and the total τ_{total} for each rigid solid are calculated according to its particle type T. During the calculation, it requires the sum of the physical quantities such as positions, forces, and torques over the rigid solid particles. The summation of these physical quantities is computed by using a parallel reduction operation. For the motions of the deformable solids, we calculate the elastic forces between particles from the same deformable solid using Equations (11) and (12) in parallel.

The calculation of surface tension forces can be performed in parallel and is suitable for GPU computations. The surface tension forces are between fluid particles and solid particles. We use the particle type information T in the kernel to identify the particle phase in the kernel. In Equation (26), the term $\mathbf{n}/|\mathbf{n}|$ would become numerical unstable when $|\mathbf{n}| \to 0$. Hence, the surface tension exists only if $|\mathbf{n}| \geq l$, where $l \geq 0$ is some threshold value related to the particle concentration, otherwise it is set to be zero.

Integration and particle collision handling The time integration and particle collision handling scheme are implemented in the same kernel. New particle velocity and position are integrated using an explicit Leap-frog scheme [30]. Then, the kernel implements our collision handling method using the updated velocities and positions. For each solid surface particle, which can be determined by our surface tension model, the kernel calculates its distance to each neighboring fluid particle. If the distance is smaller than the summation of their radius, the velocity and position of fluid particle are corrected. Otherwise, the boundary force is computed and added to the total force in the next time step.

Fluid surface construction and rendering We adopt a GPU-based surface reconstruction method for particle-based fluids using Marching Cubes, in which the distance field is only constructed in the cubes around the free surfaces and fluid–solid interfaces. The surface reconstruction method includes four stages:

First, detecting the fluid surface particles near the free surface and fluid–solid interfaces. We launch a kernel for each fluid particle i, and compute $d_i = \frac{\sum_j \mathbf{x}_{ij} m_j}{\sum_j m_j}$ using the information of neighboring fluid particles. If d_i exceeds a certain threshold or the neighborhood is empty, p_i is considered to be a fluid surface particle. In Figure 6(a), the red colored particles are the detected fluid surface particles.

Second, finding the cubes around the fluid surfaces. For each fluid surface particle j, a kernel is launched to determine which cube contains it. The yellow-colored cubes of Figure 6(a) are the cubes containing fluid surface particles. We compute an axis-aligned bounding box (AABB) that spans

DOI: 10.1002/jnm

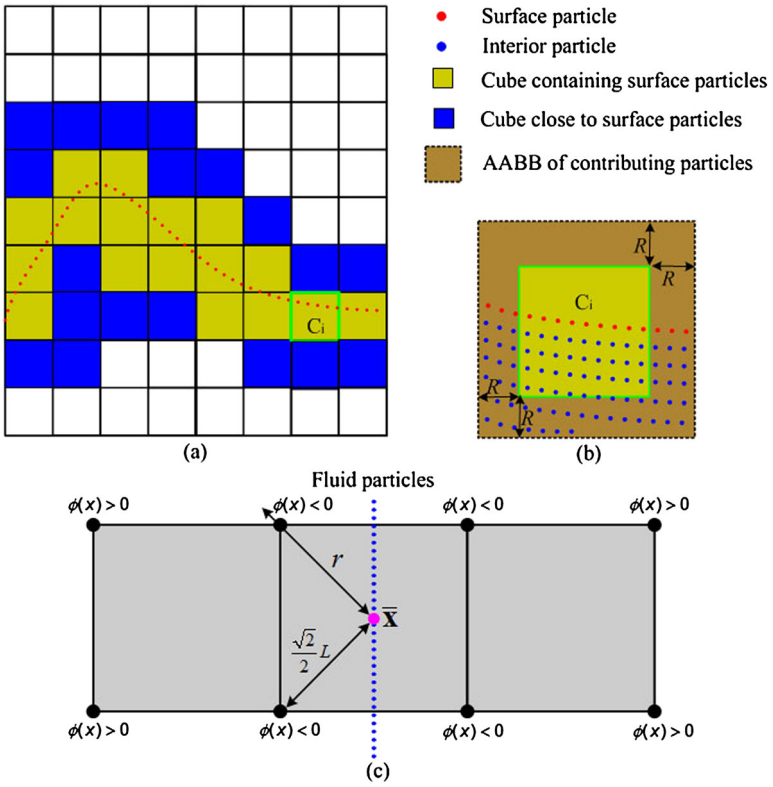

Figure 6. GPU-based fluid surface construction.

$R = 2r$ distance on each axis for each yellow colored cube. The cubes that are overlapping this AABB are colored blue in Figure 6(a). Both the yellow and blue cubes are around the free surfaces and fluid–solid interfaces and are called as surface cubes. We obtain a binary array in which surface cubes are marked as 1, and other cubes are marked as -1. Then, we use CUDPP library of NVIDIA to implement parallel compression operation and obtain an index array of surface cubes.

Third, calculating the distance field in the surface cubes. We launch a kernel for each surface cube k, and adopt the method of [5] to compute the distance values for its eight vertices

$$\phi(\mathbf{x}) = |\mathbf{x} - \bar{\mathbf{x}}(\mathbf{x})| - r \tag{34}$$

where $\bar{\mathbf{x}}(\mathbf{x}) = \frac{\sum_j \mathbf{x}_j W(|\mathbf{x}-\mathbf{x}_j|,R)}{\sum_j W(|\mathbf{x}-\mathbf{x}_j|,R)}$. As shown in Figure 6(b), to identify the fluid particles contributing to the final distance field values of cube vertices, we compute AABB that spans R distance on each axis for each surface cube. The fluid particles inside the cubes which are overlapping this AABB are the contributing fluid particles.

Fourth, the triangulation of surface cubes. We launch a kernel for each surface cube to obtain the triangle meshes of the fluid surface based on the 'marching cubes' demo of NVIDIA.

To ensure that the fluid film consisting of one layer particles can be reconstructed, we analyze the relationship between the particle radius r and the cube size L in 2D. As Figure 6(c) shows, if $r < \frac{\sqrt{2}}{2}L$ and $\bar{\mathbf{x}}$ at the center of the cube, the fluid surface cannot be reconstructed. To overcome this problem, we set $r > \frac{\sqrt{2}}{2}L$. In 3D, the relationship between r and L is $r > \frac{\sqrt{3}}{2}L$.

For the fluid rendering, the particle position array is stored as a virtual buffer object, and we map it to CUDA when performing parallel computations and to OpenGL when rendering directly from the virtual buffer object.

DOI: 10.1002/jnm

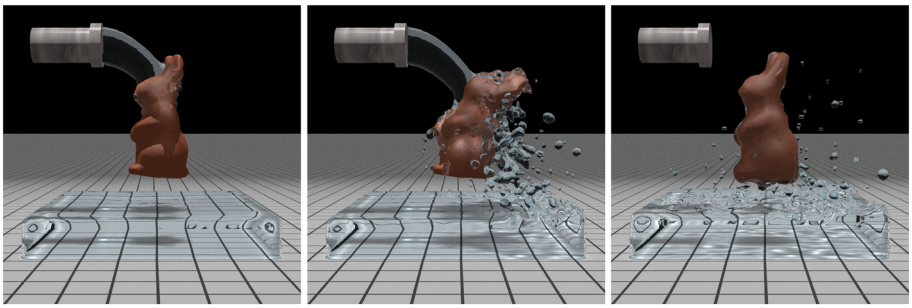

Figure 7. A water stream falls onto a deformable rabbit.

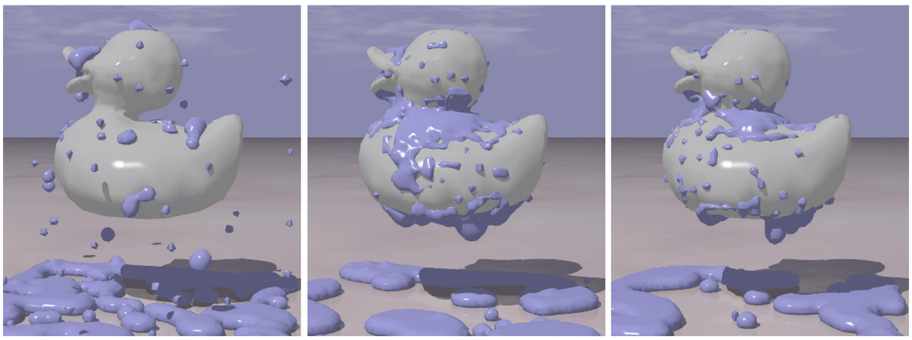

Figure 8. The comparison of surface tension model. The surface tension model [23] (left), the surface tension model [31] (middle), and our surface tension model (right).

6. RESULTS AND DISCUSSION

We have tested the proposed method with several example simulations. We implement the physical simulation of our framework on a Microsoft Windows XP computer with dual Intel Core 2.8 GHz CPUs, 2.0 GB RAM, and NVIDIA GeForce GTX 480 GPU with 1.5 GB VRAM. The programs were written in C++, OpenGL and CUDA. The water rendering results are obtained by employing Pov-Ray engine to render triangle surfaces extracted from a scalar quantity field using the Marching Cube method of [20].

6.1. Coupling results

In Figure 7, the animation describes that a water stream falls onto an elastic rabbit whose bottom is fixed. The rest density of the water particle is 1000 kg/m^3, and the density of the rabbit particle is 40000 kg/m^3. The rabbit causes splashes when it is battered by the water stream, and in the meantime, the water also makes the rabbit deform elastically. The collision handling scheme prevents water penetrating the surface of solid rabbit, and the surface tension method simulates the formation of water droplets and the flow of water on the rabbit surface.

With our surface tension model, a variety of small-scale phenomena including the formation of water droplets and the flow of water on solid surface can be realistically simulated. In Figure 8, we compared our surface tension model with the methods of [23] and [31] when simulating the water dropping onto the surface of a rigid duck. As the figure shows, the surface tension model of [23] is only appropriate for single-phase free-surface flows, while our model can simulate water droplets flowing on the surface of solid as the method of [31].

Figure 9 is a scenario that a fluid stream drops into a thin elastic box ($\rho_0 = 200$). Because of the adoption of the momentum-conserving particle collision handling scheme, our method can handle the stable coupling of SPH fluids and thin deformable structures avoiding penetration artifacts under larger velocity differences.

DOI: 10.1002/jnm

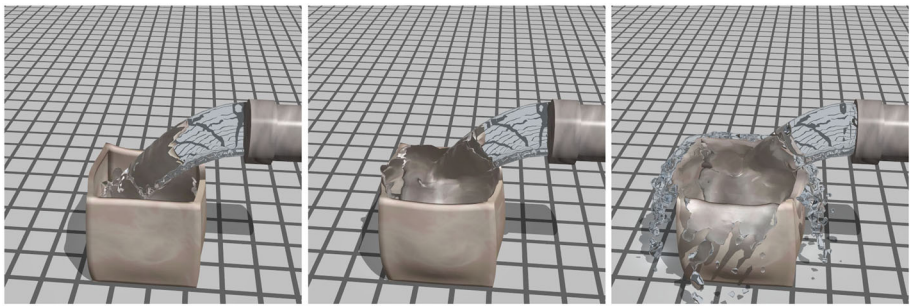

Figure 9. Fluids poured into an empty elastic box avoiding the penetrations.

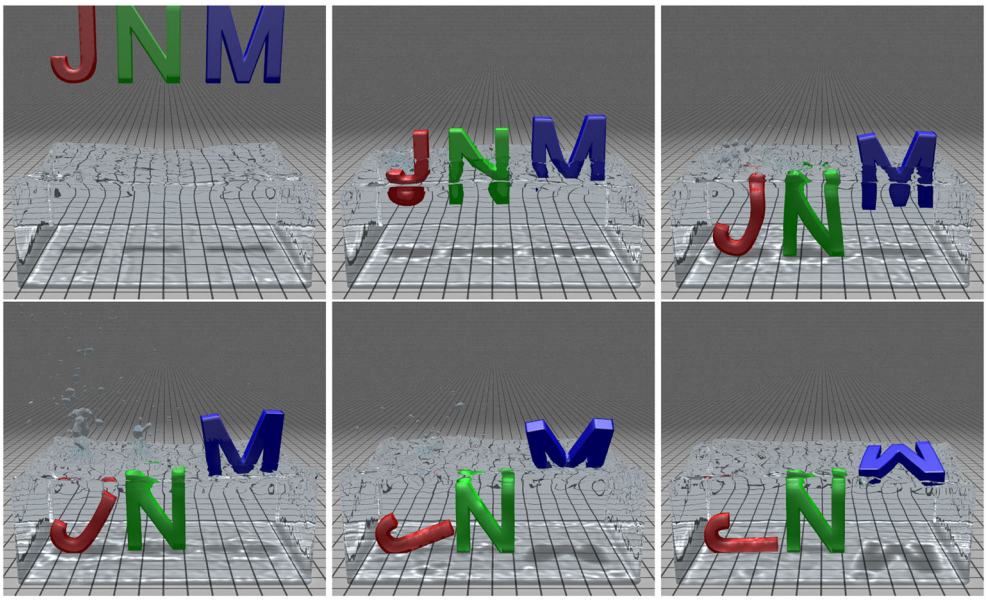

Figure 10. Letter-shaped objects of different densities drop into water.

Figure 10 displays the snapshots of a 3D solid-fluid coupling that three letter-shaped objects of different densities (J : $\rho_0 = 50000$ kg/m^3; N : $\rho_0 = 10000$ kg/m^3; M : $\rho_0 = 100$ kg/m^3) drop into water, which demonstrates that our coupling method can handle fluid–solid coupling with large density ratio up to 50.

Figure 11 shows that a water dam interacts with several rigid and deformable objects of different densities (Elastic ducks: $\rho_0 = 100, 10000$; Elastic torus: $\rho_0 = 200, 8000$; Rigid rabbits: $\rho_0 = 200, 9000$; Rigid spheres: $\rho_0 = 300, 10000$), which demonstrates our uniform particle-based framework can simulate fluid–solid interaction as well as solid-solid interaction.

6.2. *Stability analysis*

Figure 12 is a 2D scenario that four elastic objects of different densities drop into a pool of water (from left to right: $\rho_0 = 5000$ kg/m^3, 200 kg/m^3, 3000 kg/m^3, and 300 kg/m^3). The results show that our solid–fluid coupling method (Figure 12(b)) makes the simulation results well agree with the purely physical method proposed by Solenthaler *et al*. [5] (Figure 12(a)). However, when the velocity difference between fluid and solid is large, Solenthaler's method [5] produces penetration at solid–fluid interfaces. On the basis of the proposed collision handling scheme, our method prevents the penetration artifacts at the fluid–solid interfaces, which highly improves the stability of the fluid–solid simulation.

In Figure 13, we simulate the coupling between a water dam and a rigid empty box sampled with a band of black colored boundary particles. The initial densities of the water and the box are 1000 kg/m^3

DOI: 10.1002/jnm

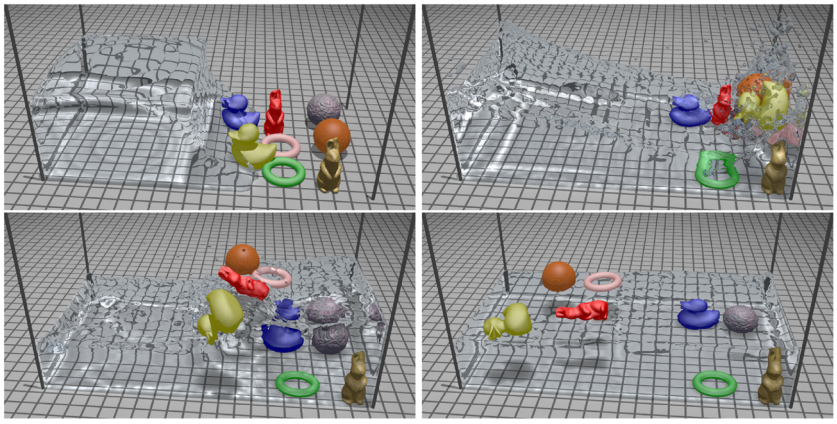

Figure 11. A water dam interacts with different material solids.

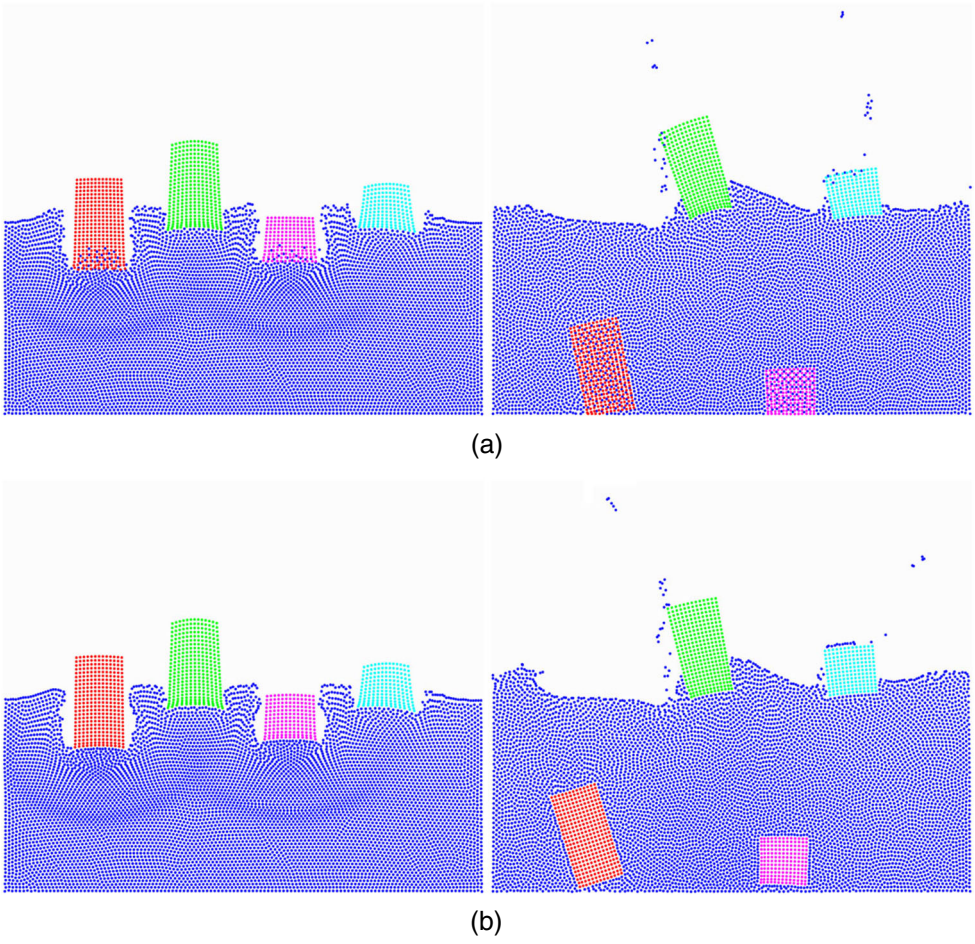

Figure 12. The penetration prevention using the momentum-conserving particle collision handling scheme. (a) Solenthaler's method [5] and (b) our method.

and 5000 kg/m^3, respectively. The fluid particle pressures are color-coded. Red to yellow to green to blue colors illustrate that the pressure changes from large to small. While the coupling method [12] leads to pressure noise and sticking artifacts at the solid–fluid interfaces (Figure 13(a)), our method avoids these problems in the coupling (Figure 13(b)).

DOI: 10.1002/jnm

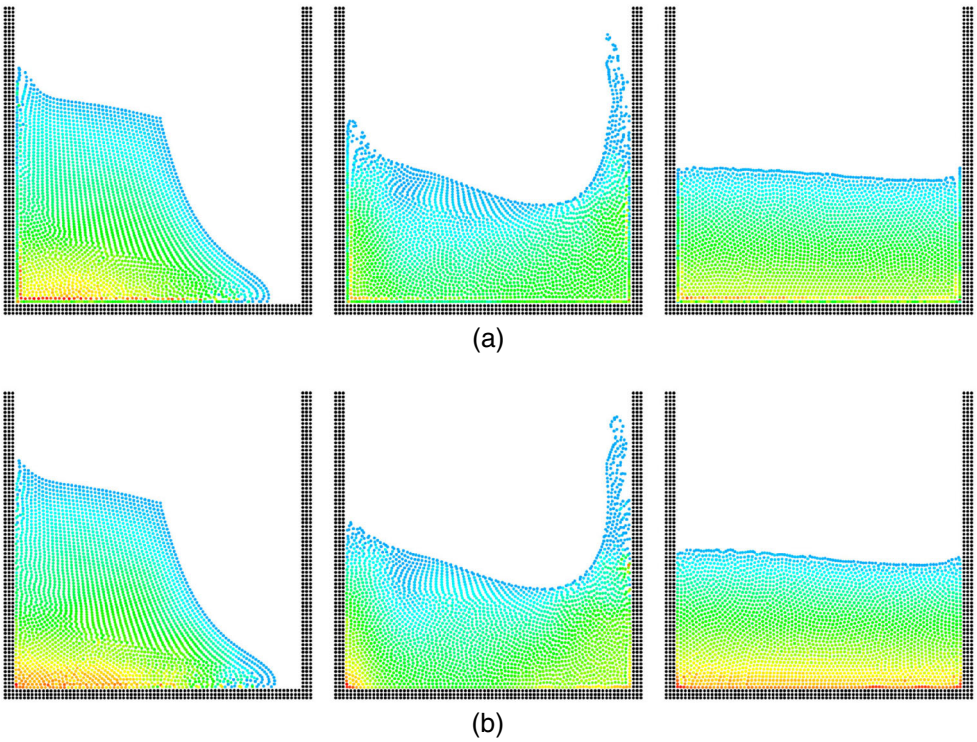

Figure 13. Fluid pressure distribution in the coupling. (a) The coupling method [12] and (b) our coupling method.

Table II. Executive time.

Scene	$\#FP$	$\#RSP$	$\#ESP$	CPU time	GPU time	Speedup
Figure 7	40 k		3.2 k	913.9 msec	35.7 msec	25.6
Figure 8	20 k	3 k		546.8 msec	25.2 msec	21.7
Figure 9	35 k		5 k	758.2 msec	32.4 msec	23.4
Figure 10	62 k		7.4 k	1665.3 msec	54.6 msec	30.5
Figure 11	80 k	22 k	24 k	3852.2 msec	93.5 msec	41.2

6.3. Time performance

Compared with the coupling methods [1–3], the time performance of each time step is highly improved by using an entire GPU implementation of our uniform particle-based formulation. Table II shows statistics for the average time cost in milliseconds of each simulation step (each animation frame only includes one time step), which states that the time cost increases with the increase of fluid particles and solid particles. The number of fluid particles, rigid solid particles, and elastic solid particles for each 3D scenario are noted as $\#FP$, $\#RSP$, and $\#ESP$, respectively. As shown in Table II, our method can achieve the realtime simulation of fluid–solid couplings. Even for up to 126 k particles, the frame rate can be achieved 10.7 FPS. Our entire GPU implementation shows significative improvements on performance in comparison with the CPU implementation: impressive speedups of about 41.2 times could be achieved for 126 k particles.

7. CONCLUSION AND FUTURE WORK

With the purpose of eliminating the need to define an interface for coupling different models, we have proposed an interactive unified particle-based method on GPUs to simulate fluid–solid interaction with large density ratio. In the method, the solid object is treated as a special fluid constrained to solid

DOI: 10.1002/jnm

motions; therefore, fluid–solid interaction as well as solid–solid interaction could be computed directly by the multiphase SPH solver. For stability and accuracy, our method handles the discontinuities at interfaces and employ a surface tension model that can obtain a stable and accurate surface curvature to capture small-scale details in the fluid–solid interfaces. Accelerated by GPU, the fluid–solid interaction with the number of particles up to 126 k can be performed at interactive frame rates. Our method is beneficial to interactive applications such as games and virtual surgery.

Our method still has several limitations. Firstly, for stability reasons the maximum time-step of our method is limited by several time-step criteria such as the CFD condition, the viscous condition, and the surface tension condition. Secondly, the realistic rendering of water is not included in the GPU implementation. Thirdly, our method does not handle the generation of bubbles during fluid–solid interaction. We believe that the bubbles can be simulated by incorporating the method of [36] into our framework. In the future, our uniform particle method will be extended to simulate more complex phenomena such as interaction between fluid and permeable or erodible solid.

ACKNOWLEDGEMENT

This work is supported by the National 863 Program of China (grant no. 2012AA011803) and the National Natural Science Foundation of China (grant no. 61300066). We thank Zhaohui Wu for writing the rendering code and helping the image production. We also thank the anonymous reviewers for their constructive comments.

REFERENCES

1. Chentanez N, Goktekin TG, Feldman BE, O'Brien JF. Simultaneous coupling of fluids and deformable bodies, *Proceedings of the 2006 ACM SIGGRAPH/ Eurographics Symposium on Computer Animation*, Vienna, Austria, 2006; 83–89.
2. G'enevaux O, Habibi A, Dischler JM. Simulating fluid–solid interaction, *Proceedings of Graphics Interface*, Halifax, Nova Scotia, Canada, 2003; 31–38.
3. Mosher AR, Shinar T, Gretarsson J, Su J, Fedkiw R. Two-way coupling of fluids to rigid and deformable solids and shells, *Proceedings of ACM SIGGRAPH*, New York, NY, USA, 2008.
4. Keiser R, Adams B, Gasser D, Bazzi P, Dutre P, Gross M. A unified lagrangian approach to solid-fluid animation, *Proceedings of the Second Eurographics / IEEE VGTC conference on Point-Based Graphics*, Stony Brook, NY, USA, 2005.
5. Solenthaler B, Schlafli J, Pajarola R. A unified particle model for fluid–solid interactions. *Computer Animation and Virtual Worlds* 2007; **18**(1):69–82.
6. Attardo EA, Francavilla1 MA, Vipiana F, Vecchi G. Investigation on accelerating FFT-based methods for the EFIE on graphics processors. *Int J Numer Modell Electron Networks Devices Fields* 2013; **26**:324–336.
7. Donno DD, Esposito A, Monti G, Catarinucci L, TArricone L. GPU-based acceleration of computational electromagnetics codes. *Int J Numer Modell Electron Networks Devices Fields* 2013; **26**:309–323.
8. Van de Wiele B, Vansteenkiste A, Van Waeyenberge B, Dupré L, De Zutter D. Implementation of a finite-difference micromagnetic model on GPU hardware. *Int J Numer Modell Electron Networks Devices Fields* 2013; **26**:366–375.
9. Hamada S. Performance comparison of three types of GPU-accelerated indirect boundary element method for voxel model analysis. *Int J Numer Modell Electron Networks Devices Fields* 2013; **26**:337–354.
10. Zhang F, Shen X, Long X, Hu L, Zhao B. A particle model for fluid simulation on the multi-graphics processing unit. *Int J Numer Model* 2013; **26**: 397–414.
11. Carlson M, Mucha PJ, Turk G. Rigid fluid: Animating the interplay between rigid bodies and fluid, *Proceedings of ACM SIGGRAPH*, New York, NY, USA, 2004; 377–384.
12. Müller M, Schirm S, Teschner M, Heidelberger B, Gross M. Interaction of fluids with deformable solids. *Comput Anim Virtual Worlds* 2004; **15**(3-4):151–171.
13. Müller M, Keiser R, Nealen A, Pauly M, Gross M, Alexa M. Point based animation of elastic, plastic and melting objects, *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Grenoble, France, 2004; 141–151.
14. Toon L, Philip D. Unified sph model for fluid-shell simulations, *Proceedings of ACM SIGGRAPH*, New York, NY, USA, ACM, 2008.
15. Iwasaki K, Uchida H, Dobashi Y, Nishita T. Fast particle-based visual simulation of ice melting. *Computer Graphics Forum* 2010; **29**(7):2215–2223.
16. Müller M, Solenthaler B, Keiser R, Gross M. Particle-based fluid-fluid interaction, *Proceedings of the 2005 ACM SIGGRAPH/ Eurographics symposium on Computer animation*, New York, NY, USA, 2005; 237–244.
17. Solenthaler B, Pajarola R. Density contrast sph interfaces, *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Aire-la-Ville, Switzerland, 2008; 211–218.
18. Desbrun M, Cani MP. Smoothed particles: A new paradigm for animating highly deformable bodies, *Proceedings of Eurographics Workshop on Computer Animation and Simulation'96*, 1996; 61–76.

19. Stora D, Agliati PO, Cani MP, Neyret F, Gascuel JN. Animating lava flows, *Proceedings of Graphics Interface'99*, Kingston, Canada, 1999; 203–210.
20. Müller M, Charypar D, Gross M. Particle-based fluid simulation for interactive applications, *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, San Diego, CA, USA, 2003; 154–159.
21. Clavet S, Beaudoin P, Poulin P. Particle-based viscoelastic fluid simulation, *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2005; 219–228.
22. Paiva A, Petronetto F, Lewiner T, Tavares G. Particle-based viscoplastic fluid/solid simulation. *Journal of Computer-Aided Design* 2009; **41**(4):306–314.
23. Becker M, Teschner M, Weakly compressible sph for free surface flows, *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, CA, USA, 2007; 209–217.
24. Sin F, Bargteil AW, Hodgins JK. A point-based method for animating incompressible flow, *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New Orleans, Louisiana, USA, 2009; 209–217.
25. Solenthaler B, Pajarola R. Predictive-corrective incompressible sph. *ACM Transactions on Graphics* 2009; **28**(3): 251–276.
26. Adams B, Pauly M, Keiser R, Guibas L. Adaptively sampled particle fluids. *ACM Transactions on Graphics* 2007; **26**(3): 48–56.
27. Goswami P, Schlegel P, Solenthaler B, Pajarola R. Interactive sph simulation and rendering on the gpu, *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Madrid, Spain, 2010; 55–64.
28. Zhang Y, Solenthaler B, Pajarola R. Adaptive sampling and rendering of fluids on the gpu, *Proceedings of the Fifth Eurographics/IEEE VGTC Conference on Point-Based Graphics*, Los Angeles, CA, USA, 2008; 137–146.
29. Becker M, Ihmsen M, Teschner M. Corotated sph for deformable solids, *Proceedings of Eurographics Workshop on Natural Phenomena*, Munich, Germany, 2009; 27–34.
30. Monaghan J. Smoothed particle hydrodynamics. *Reports on Progress in Physics* 2005; **68**(8): 1703–1759.
31. Wang HM, Mucha PJ, Turk G. Water Drops on Surfaces. *ACM Transactions on Graphics* 2005; **24**(3):921–929.
32. Adami S, Hu XY, Adams NA. A new surface-tension formulation for multi-phase SPH using a reproducing divergence approximation. *Journal of Computer Physics* 2010; **13**(6): 5011–5020.
33. Morris JP, Fox PJ, Zhu Y. Modeling low reynolds number incompressible flows using sph. *Journal of Computer Physics* 1997; **13**(6):214–226.
34. Espanol P, Revenga M. Smoothed dissipative particle dynamics. *Physical Review E* 2003; **67**(2): 219–231.
35. Atish N, Harris M, Garland M. Designing Efficient Sorting Algorithms for Manycore Gpus. NVIDIA Corporation, 2008.
36. Cleary PW, Pyo SH, Prakash M, Koo BK. Bubbling and frothing liquids, *Proceedings of ACM SIGGRAPH*, New York, NY, USA, 2007; 971–976.

AUTHORS' BIOGRAPHIES

Xuqiang Shao, born in 1982, is a PhD student of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China. He is a student member of China Computer Federation. His main research interests include computer graphics and virtual reality.

Zhong Zhou, born in 1978, is an associate professor of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China. He is a senior member of China Computer Federation. His main research interests include computer vision, distributed virtual reality, and visualization technology.

DOI: 10.1002/jnm

Jinsong Zhang, born in 1987, is a PhD student of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China. He is a student member of China Computer Federation. His main research interests include computer graphics and virtual reality.

Wei Wu, born in 1961, is a professor of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China. He is a senior member of China Computer Federation. His main research interests include wireless sensor network, distributed virtual reality, and visualization technology.

DOI: 10.1002/jnm