COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds (2014)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1607

RESEARCH ARTICLE

Realistic and stable simulation of turbulent details behind objects in smoothed-particle hydrodynamics fluids

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

State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, No. 37 Xueyuan Road, 100191 Beijing, China

ABSTRACT

This paper presents a novel realistic and stable turbulence synthesis method to simulate the turbulent details generated behind objects in smoothed particle hydrodynamics (SPH) fluids. Firstly, by approximating the boundary layer theory on the fly in SPH fluids, we propose a vorticity production model to identify which fluid particles shed from object surfaces and which are seeded as vortex particles. Then, we employ an SPH-like summation interpolant formulation of the Biot–Savart law to calculate the fluctuating velocities stemming from the generated vorticity field. Finally, the stable evolution of the vorticity field is achieved by combining an implicit vorticity diffusion technique and an artificial dissipation term. Moreover, in order to efficiently catch turbulent details for rendering, we propose an octree-based adaptive surface reconstruction method for particle-based fluids. The experiment results demonstrate that our turbulence synthesis method provides an effect way to model the obstacle-induced turbulent details in SPH fluids and can be easily added to existing particle-based fluid–solid coupling pipelines. Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS
turbulence synthesis; SPH; fluid simulation; fluid–solid coupling; physically based animation

Supporting information may be found in the online version of this article.

***Correspondence**
Zhong Zhou, State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, No. 37 Xueyuan Road, 100191 Beijing, China.
E-mail: zz@vrlab.buaa.edu.cn

1. INTRODUCTION

Physically based simulations of the fluid motions have been gaining increased interest in a broad range of practical applications, for example, in commercial films and computer games. The inherent flow characteristics make the fluids interact with the solid objects all the time and create more visually interesting natural phenomena. For example, the object moving in the water generates turbulence, like the wake of a speedboat motor (Figure 1). However, due to both numerical dissipations and limited particle sampling resolutions, the recovering of solid-induced turbulent details in smoothed-particle hydrodynamics (SPH) fluids is still challenging.

One common method of reducing the numerical dissipations in Eulerian grid fluids is to adopt higher order advection schemes [1,2]. Although this technique can recover some small-scale details to greatly improve the realistic effect of the fluid simulation, it is not applicable to SPH fluids because the advection term is not required for particle-based methods. By refining visually important regions, adaptive particle sampling [3] and two-scale scheme [4] address the problem for SPH fluids to some degree. But when moving out of the high-resolution regions, the generated small-scale details are difficult to preserve in the global flow with the coarse particle sampling density. What is more, it is hard to select a suitable subdivision level of particles for the recovering of turbulent details.

A more simple and effective approach for recovering turbulence is to augment the basic low-resolution fluid solver with the turbulence synthesis models [5–9]. Recently, several researchers have successfully synthesized solid-induced turbulent details for SPH-based fluids, but the realistic turbulence formation and the stable vorticity evolution are still two crucial issues. For example, Yuan *et al.* [10] design a swirling incentive particle method to introduce random swirling-probability values to efficiently simulate the stochastic nature of the turbulence. But in some cases, unrealistic turbulence may be created from an

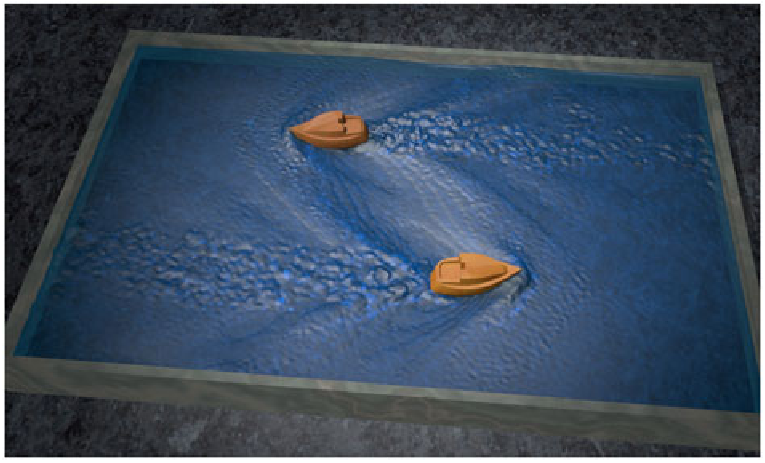


Figure 1. Turbulent details behind the boats sailing on the water.

inappropriate choice of parameters, especially when they simulate the turbulence induced by complex boundaries. Bo *et al*. [11] propose an effective method for creating and preserving the turbulent details around the moving objects in SPH fluids, which needs a high-resolution overlapping grid to be bounded to each object and translated with the objects. Because they use an attenuation coefficient instead of solving the vorticity diffusion term, it is difficult to achieve a stable vorticity evolution.

The formation of solid-induced turbulence is described by boundary layer theory of traditional computational fluid dynamics. By seeding vortex particles (VPs) where the precomputed artificial boundary layers separate from the object surface, Pfaff *et al*. [12] propose a physically plausible model to simulate solid-induced turbulence for Eulerian grid fluids. In this paper, for significantly improving our previous work [13] which is not stable and efficient enough, we propose a turbulence synthesis method which can simultaneously handle the physically plausible formation of solid-induced turbulence and the stable vorticity evolution. Our turbulence synthesis method approximates the boundary layer theory on the fly in SPH fluids and identifies which fluid particles shed from object surfaces and which are seeded as VPs. Then, an SPH-like summation approximation of Biot–Savart law is proposed to calculate the fluctuating velocities stemming from the generated vorticity field. Finally, to achieve the stable evolution of the generated vorticity field, we employ an implicit vorticity diffusion technique and introduce an artificial dissipation term when solving the Lagrangian vorticity equation.

Moreover, the rendering of turbulent details in SPH fluids needs the surface reconstruction method to catch fine details. In the surface reconstruction methods of particle-based fluids using Marching Cubes (MC) [3,14,15], the employed grid cell size significantly affects the quality of the reconstructed surface, the memory consumption, and the computation time. In order to catch the turbulent details on the fluid surfaces, small cells are required, which in turn scales the memory consumption and the computation time cubically. To improve the simulation efficiency for large fluid scenes, we propose an octree-based surface reconstruction method which constructs the adaptive distance field only in a narrow band around the fluid surface.

As the result shown in Figure 1, our turbulence synthesis method achieves the realistic and stable simulation of the turbulent details generated behind objects immersed in SPH fluids and can be regarded as a complementary of a particle-based fluid simulator. The main contributions of this paper can be described as follows:

(1) We propose a turbulence synthesis method for particle-based fluids, which simulates the realistic turbulent details generated behind the moving objects and solves the stable evolution of the vorticity field.
(2) We present an efficient surface reconstruction method for particle-based fluids using MC, which constructs the octree-based adaptive distance field only in a narrow band around the fluid surface.

2. RELATED WORK

In physically based computer animation, SPH-based simulation of fluids such as smoke and water has received considerable attention in recent years. Because Müller *et al*. [14] successfully employed SPH to simulate compelling fluids at interactive rates, SPH methods have been used to produce animations of many complex phenomena such as hairs [16], melting [17], sand [18], incompressible fluids [19,20], viscoelastic fluids [21], and fluid–solid coupling [15,22–25]. However, one of the prominent difficulties faced in physics-based fluid simulation, especially in the popular SPH method, is the loss of small-scale details such as turbulence due to inherent numerical diffusion.

For Eulerian grid methods, Fedkiw *et al*. [26] propose to use vorticity confinement to amplify existing vorticities in the fluids. Unfortunately, if the uniform simulation grid is not fine enough to catch the turbulent details, vorticity confinement is unable to recover them. Adopting higher order advection schemes [1,2] is another way to alleviate the numerical dissipation problem, but the underlying grid resolution still limits the turbulence represented by these methods.

By solving the fluid motion equations with higher resolution grids in visually important regions, adaptive grid methods [27] are employed to simulate the turbulence formation around moving objects. Similarly, adaptive particle sampling [3] and two-scale scheme [4] are incorporated into SPH fluids to catch small-scale details in a local concerned space. But for these methods, the generated vorticities are still difficult to be preserved when moving out of the high-resolution regions.

As one of the effective methods, procedure synthesis has been used to introduce turbulent details into the basic fluid solvers by using noise functions. Kim *et al.* [6] employ a wavelet decomposition to detect where turbulent details are being lost and to apply an incompressible turbulence function to reintroduce these details. Instead of enforcing the Navier–Stokes equations over all spatial scales, they enforce Navier–Stokes over low frequencies and Kolmogorov's turbulence spectrum over high frequencies. With the goal of getting closer to high-quality smoke simulation, Schechter *et al.* [7] present both a new sub-grid turbulence evolution model and a new predictor step for fluid simulation. From the large-scale flow simulated by the solver, Narain *et al.* [8] model the production and behavior of turbulent energy using a physically motivated energy model. Then, this energy distribution is used to synthesize an incompressible turbulent velocity field.

Seeding VPs into the basic fluid solvers is a more effective way to synthesize turbulence in fluids. Selle *et al.* [28] present a VP method which allows effective simulation by incorporating localized vorticity confinement in the grid. It directly computes the velocity for particles by trilinear interpolation and reduces the numerical loss because each VP stores a vorticity value. Park *et al.* [29] simulate gaseous phenomena by entirely distributing VPs on the whole grid and utilizing a pure Lagrangian simulation with a vorticity transport equation. There are some other grid-particle methods [9,30,31] which incorporate VPs into Eulerian grid fluid to model visually realistic turbulence effects. However, these approaches mainly deal with the preservation of turbulence already represented in the overall flow rather than the turbulence formation around solid objects.

A major source of turbulence is the interaction of fluids with solids, and several works have been focused on the simulation of turbulence behind objects immersed in the fluids. Based on the boundary layer theory, Pfaff *et al.* [12] seed VPs in Eulerian grid fluids to simulate obstacle-induced turbulent details. They identify areas where the separated boundary layers will transit into actual turbulence and seed VPs. But it is time-consuming to model a precomputed artificial boundary layer that captures the characteristics of turbulence generation around solid objects. Bo *et al.* [11] present a turbulence method to create and preserve the turbulent details generated around moving objects in SPH-based gaseous fluids. In the simulation, a high-resolution overlapping grid is bounded to each object and translates with the object, and the turbulence formation is modeled by resolving the local flow around objects using a hybrid SPH-FLIP method. To reduce numerical dissipations which consume turbulence in SPH fluids, Jang *et al.* [32] incorporate the Hermite-interpolation scheme into a particle-advection process. Aiming to capture multi-scale vortical motions efficiently, they propose large-scale kernels and a small-scale vorticity model. Zhu *et al.* [33] present a hybrid method, which integrates PIC/FLIP and VP methods into a unified work, to efficiently simulate vortex shedding that happens when fluids flow around objects immersed in fluids. And by seeding swirling incentive particles around internal obstacles, Yuan *et al.* [10] introduce random swirling-probability values to efficiently model the stochastic nature of the turbulence. Pan *et al.* [34] employ a 2D discrete vortex method to capture the detailed wake motions behind an obstacle, enriching the motion of shallow water equation simulation. In their method, by using a physically inspired procedural approach for particle seeding, discrete VPs are only created in the wake region.

3. OVERVIEW OF OUR PARTICLE-BASED METHOD

Figure 2 demonstrates the algorithmic flow of our particle-based turbulence synthesis method in each simulation loop. We detail the algorithm as follows.

For particle-based fluid simulation, we adopt the PCISPH method [19] which enforces incompressibility and allows larger time steps. As for the deformation simulation, the previous methods [15,22,35] have the instability problems for large deformations and require time steps that are too restrictive to handle stiff materials. Especially, these continuum mechanics-based methods are computationally complex and need to sample the whole volume of objects. In this paper, to improve the efficiency, we adopt the meshless shape matching method [36] to compute the dynamics of mesh-based objects, which is unconditionally stable for simulating rigid and elastic solids by solving a unified motion equation. We simulate the two-way rigid-fluid coupling based on [25]. And for the coupling of deformable objects and PCISPH fluids, we employ the coupling method [24] which considers the relative contributions of the adaptively sampled boundary particles to fluid particles. During the fluid–solid coupling simulation, the density model of [37] which allows larger density ratio is adopted to compute the particle density.

Then, our turbulence synthesis model is used to simulate the turbulent details behind the moving objects in SPH fluids. Firstly, the vorticity production module identifies which fluid particles shed from object surfaces and

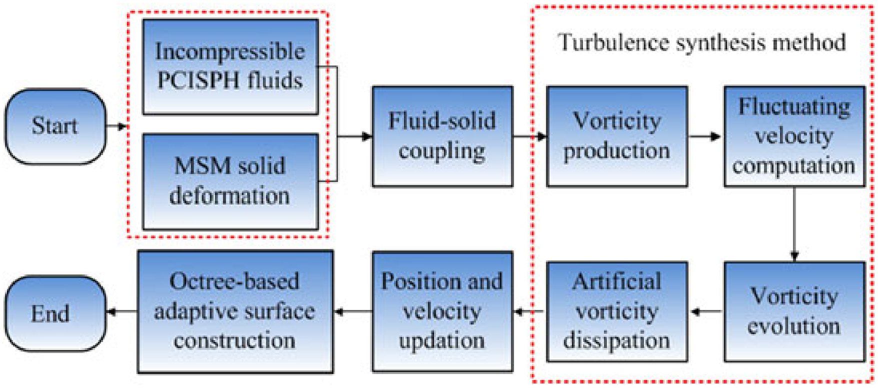

Start
Incompressible PCISPH fluids
MSM solid deformation
Fluid-solid coupling
Turbulence synthesis method
Vorticity production
Fluctuating velocity computation
Vorticity evolution
Artificial vorticity dissipation
Position and velocity updation
Octree-based adaptive surface construction
End

Figure 2. The pipeline of each time step.

which are seeded as VPs by approximating the boundary layer theory on the fly in SPH fluids. Secondly, the fluctuating velocity computation module calculates the fluctuating velocities stemming from the generated vorticity field using an SPH-like summation interpolant formulation of the Biot–Savart law. Finally, the stable evolution of the vorticity field is solved by the combination of an implicit vorticity diffusion technique and an artificial vorticity dissipation term.

The new positions and velocities of all particles are updated by using the leap-frog time integration scheme. Then, we track the fluid surface using our octree-based surface reconstruction method and employ Pov-Ray to render the triangle meshes extracted by MC.

4. TURBULENCE SYNTHESIS METHOD

4.1. Vorticity Production

Solid objects coupled to the fluids are obvious turbulence generators. The boundary layer theory [12] of computational fluid dynamics describes that the friction of objects enforces a tangential flow velocity of zero at the solid boundaries, which leads to the formation of a thin layer with reduced flow speed, called the boundary layer. In the boundary layer, the curl of tangential flow velocity $\mathbf{u}_{\tan}$ leads to the creation of a thin sheet of vorticity $\omega = \nabla \times \mathbf{u}_{\tan}$. When the boundary layers shed from the solid surfaces at the regions of high flow instability, vorticities are ejected from the boundary layers and enter the flow as turbulence. In order to model the physically plausible formation of solid-induced turbulence, as shown in Figure 3, we present a four step vorticity production model suitable to SPH fluids.

Firstly, based on the boundary layer theory, we make an approximate assumption that the boundary layer of an object immersed in SPH fluids consists of fluid particles whose distances to object surface are less than a small threshold d_T. We set $d_T = h$ in this paper, so the neighboring fluid particles of all solid surface particles constitute the thin boundary layer. We call each fluid particle of the boundary layer as BLFP which is colored green in Figure 3(a).

Secondly, we identify the separation points (SP) where BLFPs shed from the surfaces of objects immersed in fluids. Our method divides the identification process of SPs into three steps. First, we employ the gradient ∇C_i of the smoothed color field C_i, which estimates the normals of fluid–fluid interfaces [38], to compute the normal $\mathbf{n}_i$ of solid particles at the fluid–solid interfaces. Second, for each solid surface particle i, we use the formulation $\mathbf{u}_i^{diff} = \frac{\sum_j (\mathbf{u}_i - \mathbf{u}_j) W(\mathbf{x}_{ij}, h)}{\sum_j W(\mathbf{x}_{ij}, h)}$ to compute the smoothed velocity difference $\mathbf{u}_i^{diff}$ relative to all its neighboring fluid particles j. Third, the dot product $\mathbf{u}_i^{diff} \cdot \mathbf{n}_i$ between the velocity difference $\mathbf{u}_i^{diff}$ and the normal $\mathbf{n}_i$ is calculated. If $\mathbf{u}_i^{diff} \cdot \mathbf{n}_i < 0$, the solid particle i is identified as a SP (colored black in Figure 3(b)).

Thirdly, to simulate turbulence behind objects immersed in fluids, we identify the BLFPs which actually shed from its neighboring SPs. For a pair of neighboring particles BLFP j and SP i, we compute the dot product $\mathbf{x}_{ji} \cdot \mathbf{u}_{ji}$ between their relative position $\mathbf{x}_{ji}$ and their relative velocity $\mathbf{u}_{ji}$. If $\mathbf{x}_{ji} \cdot \mathbf{u}_{ji} > 0$, the BLFP j actually sheds from the object surface (colored pink in Figure 3(c)) and gains the vorticity $!_j$ which is formulated as

$$\omega_j = (1 - \alpha)\eta\kappa \left(\mathbf{u}_{ji}^{\tan} \times \mathbf{n}_i\right) + \alpha\omega_j \qquad (1)$$

where $\mathbf{u}_{ji}^{\tan}$ is the relative velocity along the tangent direction of the object surface, $\eta \in [0, 1]$ is a random number used to model the stochastic nature of turbulence, κ is a control constant evaluating the effects of fluid viscosity and object material, and α is a weight factor.

Finally, we seed VPs into SPH fluids. When a BLFP carrying the vorticity does not have neighboring solid particles any more, it is labeled as a (VP) seeded into fluids. All VPs constitute the vorticity field induced by the objects immersed in the fluid flow. Figure 3(d) shows that VPs

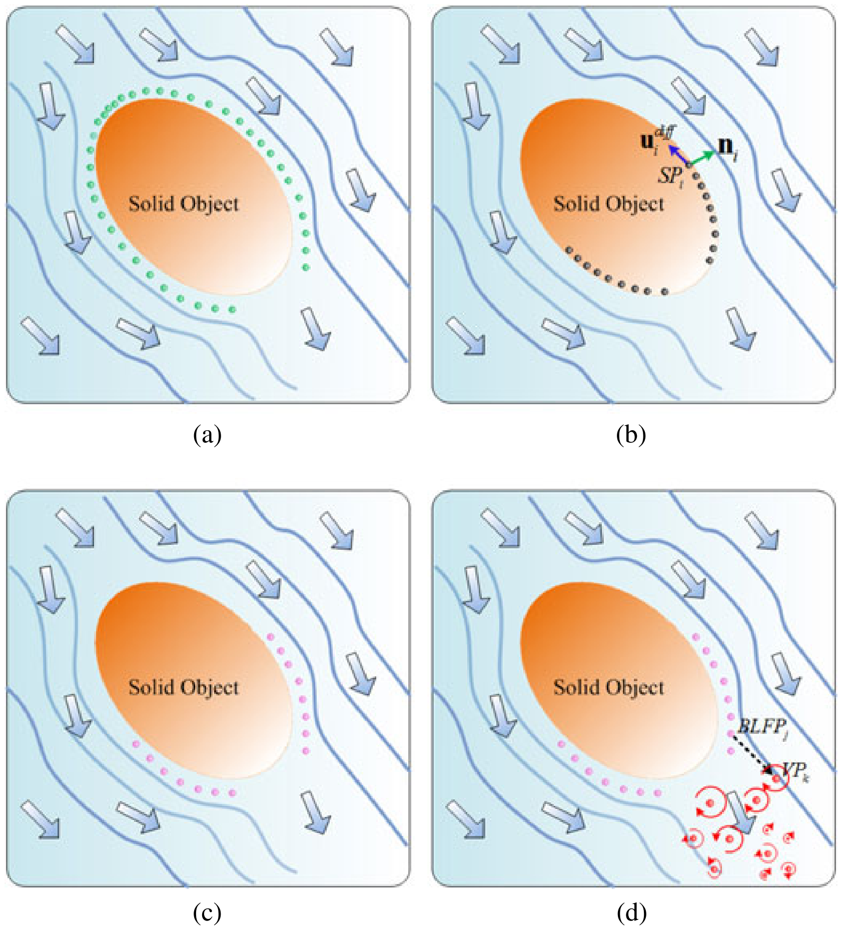


Figure 3. An overview of different steps of the vorticity production model. (a) identify fluid particles of artificial boundary layers, (b) identify SPs on the object surface, (c) compute the vorticities of shed BLFPs, and (d) vortex particle seeding.

are finally seeded behind the objects by our vorticity production method instead of being randomly seeded [10,28], which simulates the turbulent details induced by solid objects in a physically plausible way.

4.2. Fluctuating Velocity Computation

To simulate solid-induced turbulent details after the generation of vorticity field, by superposing the mean velocity field $\mathbf{U}$ with a rapidly fluctuating component $\mathbf{u}^v$, we compute the instantaneous velocity field $\mathbf{u}$ which describes the motions of fluid particles. $\mathbf{U}$ is calculated by solving the momentum equation using PCISPH method [19]; $\mathbf{u}^v$ stems from the vorticity field generated by our vorticity production model.

The velocity field $\mathbf{u}^v$ stemming from the vorticity field can be computed by the Biot–Savart law [39], which estimates $\mathbf{u}^v$ at the position $\mathbf{x}$ that is a distance $\mathbf{r} = \mathbf{x} - \mathbf{x}'$ from a vortex element $d\mathbf{x}'$ with vorticity ω by integrating over all the space:

$$\mathbf{u}^v(\mathbf{x}) = \frac{1}{4\pi}\int \frac{\omega(\mathbf{x}') \times \mathbf{r}}{|\mathbf{r}|^3} d\mathbf{x}' \tag{2}$$

In order to efficiently compute $\mathbf{u}_i^v$ for each fluid particle, we propose an SPH-like summation interpolant formulation of the Biot–Savart law. Specifically, we discretize the earlier equation and turn the integral into a summation formulation:

$$\begin{aligned} \mathbf{u}_i^v &= \frac{1}{4\pi}\sum_j \int_{\Omega_j} \frac{\omega(\mathbf{x}') \times \mathbf{r}}{|\mathbf{r}|^3} d\mathbf{x}' \\ &\approx \frac{1}{4\pi}\sum_j V_j \frac{\omega(\mathbf{x}_j) \times \mathbf{r}}{|\mathbf{r}|^3} \\ &= \sum_j \frac{m_j}{\rho_j}\left[\omega(\mathbf{x}_j) \times \mathbf{r}\right]\frac{1}{4\pi|\mathbf{r}|^3} \end{aligned} \tag{3}$$

If there are n VPs in the simulation, it is too computationally complex to employ the earlier straightforward summation taking $O(n^n)$ operations. To improve the efficiency, based on the summation interpolant formulation of

SPH [14], we only take the contributions of VPs within a certain distance h into account. We replace the term $\frac{1}{4\pi|\mathbf{r}|^3}$ with the radial symmetrical smoothing kernel $W(\mathbf{r},h)$ of SPH method for an approximation and obtain an SPH-like summation interpolant formulation of the Biot–Savart law:

$$\mathbf{u}_i^v = \sum_j \frac{m_j}{\rho_j}\left[\omega(\mathbf{x}_j)\times\mathbf{r}\right]W(\mathbf{r},h) \tag{4}$$

where the adopted W is the isotropic Gaussian kernel $W(\mathbf{r},h) = \frac{1}{(2\pi h^2)}^{\frac{3}{2}} \exp\left(-\frac{|\mathbf{r}|^2}{2h^2}\right)$.

4.3. Vorticity Evolution

We solve the evolution of the generated vorticity field by updating the vorticities of fluid particles according to the Lagrangian vorticity form of the Navier–Stokes equations:

$$\frac{d\omega}{dt} = (\omega\cdot\nabla)\mathbf{u} + \mu\nabla^2\omega \tag{5}$$

where $(\omega\cdot\nabla)\mathbf{u}$ denotes vorticity stretching, $\mu\nabla^2\omega$ denotes vorticity diffusion, and $\mathbf{u}$ is the instantaneous velocity.

We solve the vorticity stretching term by modifying its direction with $\delta t(\omega\cdot\nabla)\mathbf{u}$ [10]. This is easy to achieve in SPH summation interpolant formulation where the derivative of SPH kernel is predefined and used for gradient computation.

According to the vorticity diffusion term $\mu\nabla^2\omega$ in the earlier equation, a viscous flow rapidly dissipates vorticities on fluid particles. The previous particle-based turbulence methods [10,11,13,30] compute the vorticity diffusion term by using particle strength exchange method, which approximates the Laplacian operator ∇^2 by spreading vorticity from a particle to its neighboring particles. For each time step Δt, the vorticity increment of a particle i induced by the vorticity diffusion process is formulated as

$$\delta\omega_i = \mu\Delta t\nabla^2\omega_i = \mu\Delta t\sum_j \frac{m_j\omega_{ji}}{\rho_j}\nabla^2 W(\mathbf{x}_{ij},h) \tag{6}$$

where $\omega_{ji} = \omega_j - \omega_i$.

However, the earlier explicit vorticity diffusion is prone to instabilities when the time step Δt becomes too large (assuming μ is constant for a fluid), because the vorticity $\delta\omega_i$ spreading from j to i might be larger than their vorticity difference ω_{ji}. During the simulation using larger time steps, the vorticities of fluid particles oscillate and are dissipated too quickly.

Based on backward Euler diffusion adopted by Stam [40] in Eulerian grid method, we derive an unconditionally stable expression for the vorticity $\omega_i(t+\Delta t)$ of particle i at time $t+\Delta t$:

$$\omega_i(t+\Delta t) = \omega_i(t) + \mu\Delta t\sum_j \frac{m_j}{\rho_j}\omega_{ji}(t+\Delta t)\nabla^2 W_{ij} \tag{7}$$

Stam uses Gauss–Seidel iterative technique to solve the earlier equation, which is inefficient due to at least 20 iterations for each step. By adapting Monaghan's method [41], which uses an implicit formulation of the viscosity equation to compute drag forces between dust and air, we propose an implicit vorticity diffusion method. For simplicity, we treat each pair of particles individually. According to the earlier equation, we can determine how particle i and j exchange vorticity values at time t:

$$\omega_i(t+\Delta t) = \omega_i(t) + \mu\Delta t\frac{m_j}{\rho_j}\omega_{ji}(t+\Delta t)\nabla^2 W_{ij} \tag{8}$$

$$\begin{aligned}\omega_j(t+\Delta t) &= \omega_j(t) + \mu\Delta t\frac{m_i}{\rho_i}\omega_{ij}(t+\Delta t)\nabla^2 W_{ij}\\ &= \omega_j(t) - \mu\Delta t\frac{m_i}{\rho_i}\omega_{ji}(t+\Delta t)\nabla^2 W_{ij}\end{aligned} \tag{9}$$

Then, we formulate $\omega_{ji}(t+\Delta t)$ as

$$\begin{aligned}\omega_{ji}(t+\Delta t) &= \omega_j(t+\Delta t) - \omega_i(t+\Delta t)\\ &= \omega_j(t) - \mu\Delta t\frac{m_i}{\rho_i}\omega_{ji}(t+\Delta t)\nabla^2 W_{ij}\\ &\quad - \left[\omega_i(t) + \mu\Delta t\frac{m_j}{\rho_j}\omega_{ji}(t+\Delta t)\nabla^2 W_{ij}\right]\\ &= \omega_{ji}(t) - \mu\Delta t\left(\frac{m_i}{\rho_i}+\frac{m_j}{\rho_j}\right)\omega_{ji}(t+\Delta t)\nabla^2 W_{ij}\end{aligned} \tag{10}$$

By moving terms, we get the formulation of $\omega_{ji}(t+\Delta t)$:

$$\omega_{ji}(t+\Delta t) = \frac{\omega_{ji}(t)}{1+\mu\Delta t\left(\frac{m_i}{\rho_i}+\frac{m_j}{\rho_j}\right)\nabla^2 W_{ij}} \tag{11}$$

For a fluid particle i, we get $\omega_i(t+\Delta t)$ only depending on variables at time t, by substituting the earlier formulation of $\omega_{ji}(t+\Delta t)$ into Equation (8) or (9):

$$\omega_i(t+\Delta t) = \omega_i(t) + \frac{\mu\Delta t m_j\nabla^2 W_{ij}\omega_{ji}(t)}{\rho_j + \mu\Delta t\rho_j\left(\frac{m_i}{\rho_i}+\frac{m_j}{\rho_j}\right)\nabla^2 W_{ij}} \tag{12}$$

$$\omega_j(t+\Delta t) = \omega_j(t) + \frac{\mu\Delta t m_i\nabla^2 W_{ij}\omega_{ij}(t)}{\rho_i + \mu\Delta t\rho_i\left(\frac{m_i}{\rho_i}+\frac{m_j}{\rho_j}\right)\nabla^2 W_{ij}} \tag{13}$$

The earlier equations need to be solved by an implicit iterative method. So we replace $\omega_i(t)$ by $\omega_i(t+\Delta t)$ and repeat the process for the next neighboring particle j. Monaghan [41] proves that one sweep over all the particles does not give satisfactory convergence when the diffusion term is very large, and two sweeps will make the vorticity difference reduced by a factor close to 10. To improve the efficiency, we use two sweeps and implement the second time sweep using the reverse order of all the particles and their neighboring particles. Our sweep method is sufficiently rapid convergence to the state of zero vorticity difference. The results show that our implicit vorticity diffusion is unconditionally stable, which permits the use of a larger time step Δt.

4.4. Artificial Vorticity Dissipation

The evolution of the generated vorticity field solved by the vorticity equation (Equation (5)) has very little numerical dissipation in the simulation. All fluid particles finally have the same non-zero vorticity values under the effect of the implicit vorticity diffusion term. And the non-zero vortical velocity field $\mathbf{u}_v$ leads to that the simulated fluid keeps moving too long and cannot calm down, which makes fluid–solid coupling unrealistic and unstable. The same problem also exists in the previous particle-based turbulence simulation methods [10,11].

In order to address the earlier problem, an artificial vorticity dissipation term is employed to gradually decrease the vorticity strength of all fluid particles over time. To be specific, when a fluid particle i obtains the vorticity ω_i or becomes a VP, our method assigns i a lifetime value t_i for it to record how long the vorticity lasts. Then, a time-dependent dissipation function $\tau(t) = \exp\left(-\beta \int_{t_1}^{t_2} \phi(t)dt\right)$, which is an adaptation of the heat radiance function of [42], is proposed to compute the rest of the vorticity for particle i at time $t + \Delta t$:

$$\begin{aligned}\omega_i(t+\Delta t) &= \omega_i(0)\exp\left(-\beta\int_0^{t+\Delta t}\phi(t')dt'\right)\\ &= \omega_i(0)\exp\left(-\beta\int_0^{t}\phi(t')dt'\right)\exp\left(-\beta\int_t^{t+\Delta t}\phi(t')dt'\right)\\ &= \omega_i(t)\exp\left(-\beta\int_t^{t+\Delta t}\phi(t')dt'\right)\end{aligned} \quad (14)$$

where $\omega_i(0)$ denotes the generated initial vorticity and β is a user-defined control constant. We set $\beta = 0.04$ and adopt $\phi(x) = x$ for computational efficiency. Figure 4 shows that the value of our time-dependent dissipation function $\tau(t)$ changes over time.

Under the effect of the introduced time-dependent dissipation function $\tau(t)$, the vorticity strength of the fluid particles will dissipate quickly as time goes, which keeps the vorticity evolution stable and makes the fluid look more visually viscous and realistic.

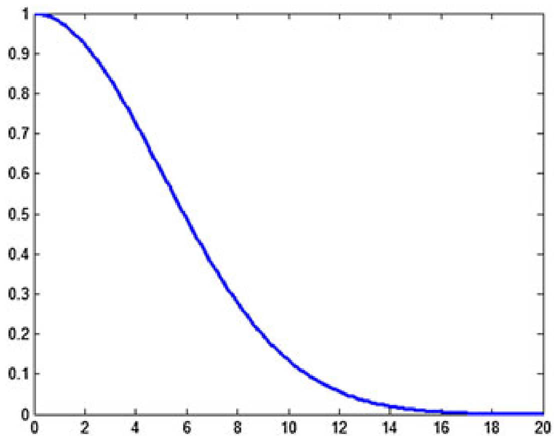


Figure 4. The time-dependent dissipation function $\tau(t)$.

5. FLUID SURFACE RECONSTRUCTION

To catch the small-scale turbulent details for rendering, we propose an efficient octree-based surface reconstruction method for particle-based fluids using MC. Our method constructs the adaptive distance field only in leaf nodes containing fluid surface particles, so the memory consumption and the computational complexity scale with the fluid surface instead of the volume. The proposed octree-based surface reconstruction method includes two stages: (i) finding the fluid surface particles and (ii) constructing the octree-based adaptive distance field. These two parts are detailed as follows.

5.1. Find Surface Particles

The surfaces of particle-based fluids can be tracked by the fluid surface particles. In this section, we use a modified version of the method in [43] to detect the surface particles of SPH fluids. For each particle i, we first compute the renormalization matrix B_i:

$$B_i = \left[\sum_j \nabla W(\mathbf{x}_{ij}, h) \otimes (\mathbf{x}_j - \mathbf{x}_i)V_j\right]^{-1} \quad (15)$$

where V_j is the volume of the jth neighboring particle, W is the renormalized Gaussian kernel used in [44], $h = 4d_0$, and d_0 is the initial spacing of fluid particles.

Then, we calculate the minimum eigenvalue $\lambda_i^{\min}$ of the matrix B_i and identify whether or not the particle i belongs to the fluid surface S via the following equation:

$$\begin{cases} i \in S, & \lambda_i^{\min} \le 0.75 \\ i \notin S, & \lambda_i^{\min} > 0.75 \end{cases} \quad (16)$$

Through the calculation of Equation (16), we obtain a particle set containing the surface particles of SPH fluids. In [43], a more precise and reliable control is performed in order to complete the surface particle detection. To improve the efficiency, we only detect the rough surface particle set using Equation (16), which is enough for constructing the octree-based adaptive distance fields.

5.2. Octree-based Adaptive Distance Field

According to the detected surface particles, we construct an octree-based adaptive distance field in a narrow band around the surface of particle-based fluids, which features saving of storage and computation.

In our method, each octree node contains two particle sets: **SurfaceParticles** which contains surface fluid particles belonging to the node and **FluidParticles** which

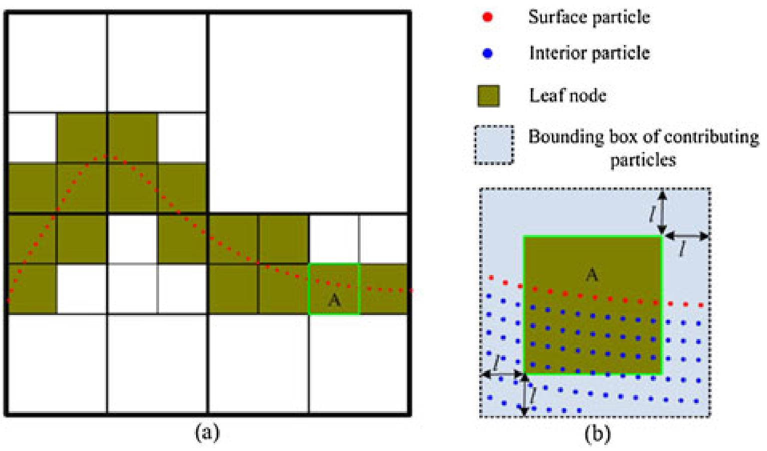


Figure 5. The construction of the octree-based adaptive distance field. (a) The octree building process and (b) contributing particles of the octree nodes.

contains all fluid particles belonging to the node. We divide octree nodes into three types: exterior nodes, boundary nodes, and interior nodes. For exterior nodes, both **SurfaceParticles** and **FluidParticles** are empty. Boundary nodes are the octree nodes whose **SurfaceParticles** are non-empty. Interior nodes have empty **SurfaceParticles** and non-empty **FluidParticles**. As shown in Figure 5(a), the building process of the octree-based adaptive distance field is divided into three stages. (i) We first create a cubic root node for the whole fluid volume. For the root node, **SurfaceParticles** contains all surface fluid particles, and **FluidParticles** contains all fluid particles. (ii) Starting from the root node, we recursively divide each node till the maximum depth is reached or its **SurfaceParticles** is empty. (iii) For the leaf nodes whose **SurfaceParticles** are non-empty, we use the method of [15] to compute the distance values for its eight vertices:

$$\phi(\mathbf{x}) = |\mathbf{x} - \bar{\mathbf{x}}(\mathbf{x})| - fr \tag{17}$$

where $\bar{\mathbf{x}}(\mathbf{x}) = \frac{\sum_j \mathbf{x}_j W(|\mathbf{x}-\mathbf{x}_j|,R)}{\sum_j W(|\mathbf{x}-\mathbf{x}_j|,R)}$, $r = d_0$ is the particle radius, $R = 4r$, and the factor $f \in [0...1]$ is computed as

$$f = \begin{cases} 1, & EV_{\max} < t_{low} \\ \gamma^3 - 3\gamma^2 + 3\gamma, & \text{otherwise} \end{cases} \tag{18}$$

where $EV_{\max}$ is the largest eigenvalue of $\nabla_{\mathbf{x}}(\bar{\mathbf{x}})$ to detect fast movements and γ is computed according to the method of [45].

As shown in Figure 5(b), when identifying **SurfaceParticles** and **FluidParticles** for each node of the octree, we consider all the particles residing in the bounding box which is $2l$ larger than the node size in each axial direction. This guarantees that the fluid surface is accurately reconstructed to avoid holes. In our parameter settings, l equals R of Equation (17).

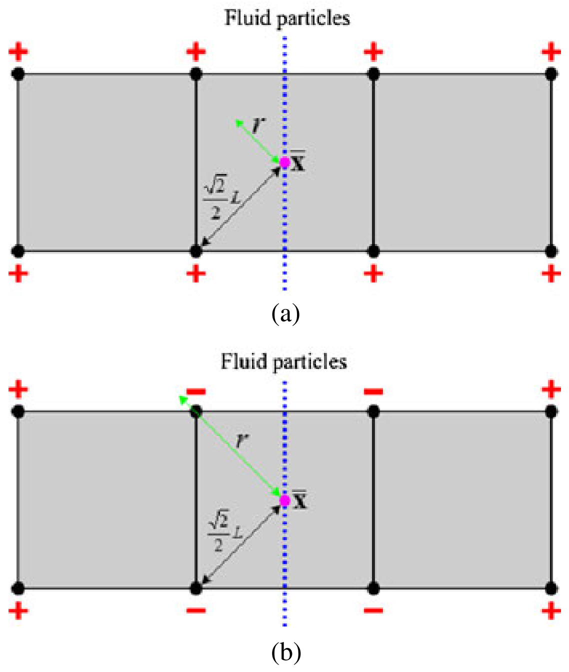


Figure 6. The relationship of size between the particle radius R and the leaf node size L. (a) Reconstruction failure and (b) successful reconstruction.

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

After the octree-based adaptive distance field is built, we traverse the octree and employ MC to extract the triangle meshes in the leaf nodes. Then, in post-processing stage, we delete the redundant indices of mesh vertices and use Pov-Ray to render the mesh surface.

6. RESULTS AND DISCUSSION

6.1. Surface Reconstruction Results

In Figure 7, we demonstrate our octree-based surface reconstruction method for particle-based fluids using MC. The volume of the armadillo model is uniformly sampled with $12.1k$ particles whose initial spacing is 0.018 m (Figure 7(a)). Then, $3.5k$ surface particles are identified by our surface particle detection method (Figure 7(b)). According to the found surface particles, we construct the octree-based adaptive distance field (Figure 7(c)), and the surface cells are yellow colored. Figure 7(d) shows the extracted triangle surface using MC.

We use three different methods to reconstruct the surfaces of the sampled particles of the armadillo model and compare their memory consumptions. We scale the memory consumption with the number of cells which contain the distance field values and are processed by MC. Table I shows the memory consumptions for the examples adopting different particle radii and cell sizes. The traditional uniform distance field method (ReconsMethod1) [3,14,15] computes and stores the scalar values for the vertices of all cells, which may become infeasible for the large-scale fluid simulation or the high-quality surface reconstruction

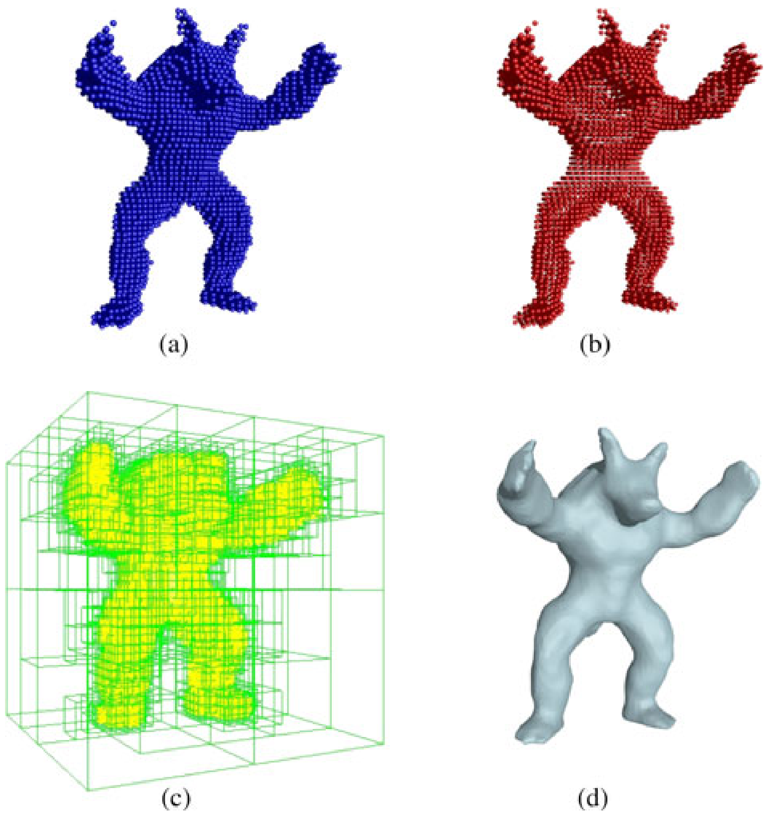


Figure 7. Our octree-based surface reconstruction method. (a) The sampled particles, (b) the detected surface particles, (c) the constructed octree-based adaptive distance field, and (d) the extracted triangle mesh using Marching Cubes.

Table I. Memory consumptions of the surface reconstruction methods.

		Cell num		
r	*L*	ReconsMethod1	ReconsMethod2	ReconsMethod3
0.08	0.064	16^3	573	518
0.04	0.032	32^3	2352	2177
0.02	0.016	64^3	9075	8601
0.01	0.008	128^3	36590	34437

Table II. Surface reconstruction information of the presented 3D scenes. The data are given as average per frame.

Scene	*#FP*	*#SP*	$\frac{\#FP_{surface}}{\#FP}$	$\frac{\#Cell_{octree}}{\#Cell_{uniform}}$	$\frac{T_{octree}}{T_{uniform}}$	*#triangles*
Figure 1	722.3k	18.4k	0.172	0.136	0.53	359.6k
Figure 10(a)	235.8k	5.7k	0.139	0.113	0.57	140.3k
Figure 10(b)	235.8k	5.7k	0.134	0.102	0.56	148.7k
Figure 11(a)	359.1k	9.2k	0.142	0.106	0.59	196.3k
Figure 11(b)	641.6k	12.6k	0.137	0.095	0.55	343.8k
Figure 11(c)	359.1k	9.2k	0.153	0.124	0.57	204.5k
Figure 13	187.1k	8.5k	0.135	0.107	0.63	158.2k

Table III. Parameters in the turbulence simulation.

Properties	2D scene	3D scene	Unit
Time step (Δt)	0.03	0.03	s
Initial spacing (d_0)	0.008	0.02	m
Support radius (h)	0.016	0.04	m
Density (ρ_0)	10^2–10^4	10^2–10^4	kg/m^3
Viscosity (μ)	0.6	0.3	Pa · s

due to the limited memory. The method of [45] (ReconsMethod2) constructs the uniform scalar field only in a narrow band around the surface, so the memory consumption scales with the fluid surface instead of the volume. Our surface reconstruction method (ReconsMethod3) adopts a new surface particle detection method and constructs the octree-based adaptive distance field only in leaf nodes containing fluid surface particles. As shown in Table I, our method reduces the memory consumption to the level of ReconsMethod2.

Table II shows the detailed octree-based surface reconstruction information of the presented 3D examples. In the table, $\#FP, \#FP_{surface}, \#SP$, and $\#triangles$ denote the number of fluid particles, fluid surface particles, solid particles, and reconstructed triangles, respectively. $\#Cell_{octree}$ and $\#Cell_{uniform}$ denote the number of cells used in our octree-based surface reconstruction method and the uniform surface method [3,13–15], respectively. $\frac{T_{octree}}{T_{uniform}}$ denotes the performance ratio, and the results show that our method greatly improves the time performance of the surface reconstruction. In our method, the fluid surface particles which occupy a small fraction (<0.2) of the total fluid particles are accurately found, and the distance field is only constructed in a narrow band around the fluid surface.

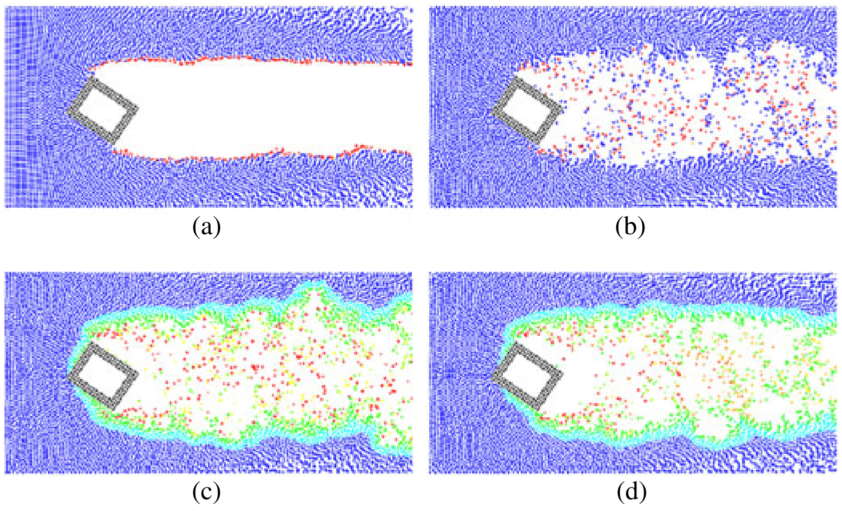
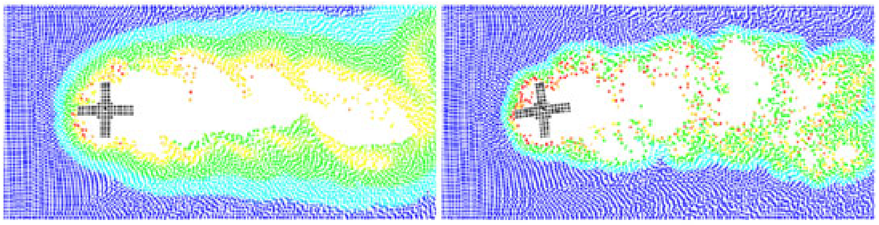
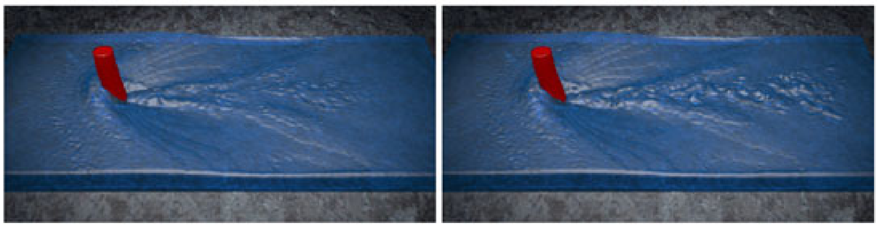


Figure 8. Four steps of our turbulence model. (a) Vortex particle seeding, (b) fluctuating velocity computation, (c) implicit vorticity diffusion, and (d) artificial dissipation.



(a)

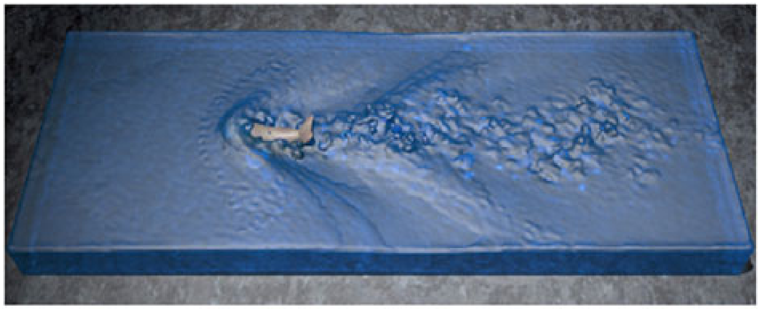


(b)

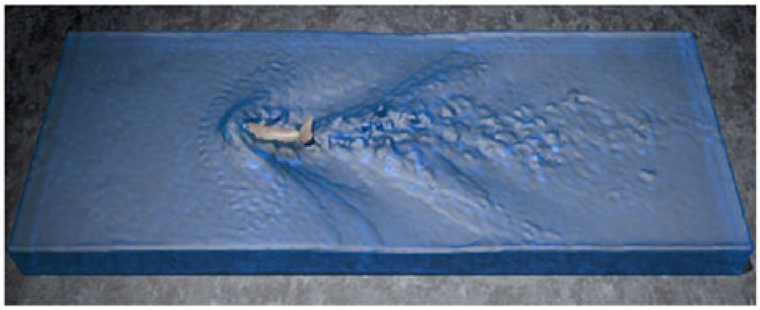
Figure 9. The comparison between explicit diffusion (left) and implicit diffusion (right) under larger time step $\Delta t = 0.03$. (a) A 2D scenario that a fluid flow interacts with a rigid propeller and (b) a 3D scenario that an elastic cylinder moves in fluids.

Table IV. Averaged per frame timings of explicit vorticity diffusion and implicit vorticity diffusion.

	Time [ms]	
Scene	Explicit vorticity diffusion	Implicit vorticity diffusion
Figure 1	1984	2969
Figure 10	609	938
Figure 11(c)	899	1478
Figure 13	458	745



(a)



(b)

Figure 10. A dolphin swimming in the water. (a) Without artificial vorticity dissipation and (b) with artificial vorticity dissipation.

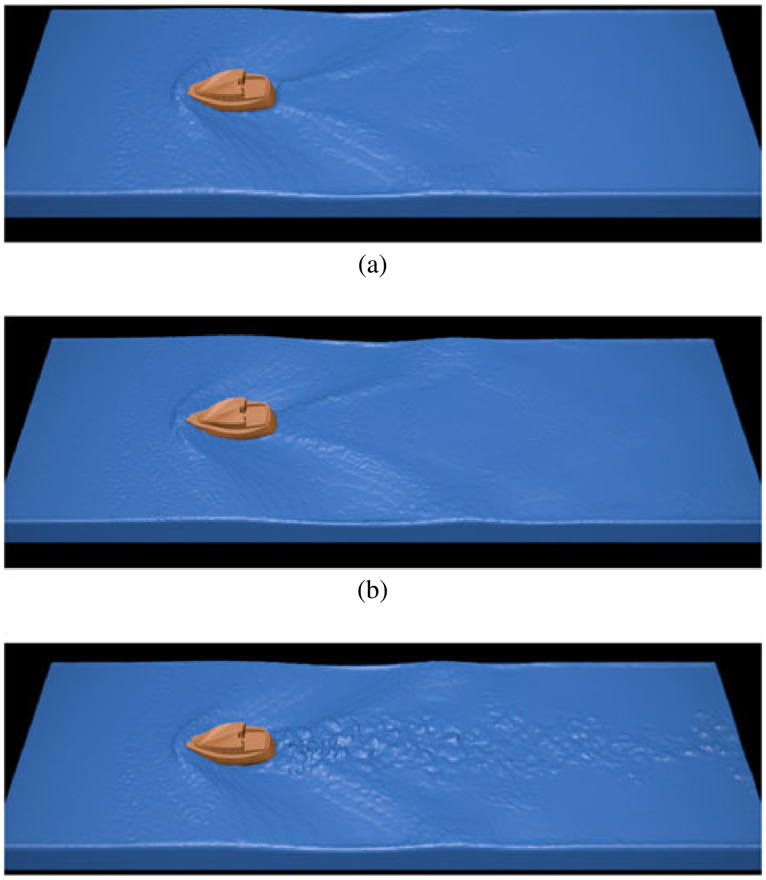


Figure 11. A boat sailing on the water. (a) Traditional SPH fluids, (b) adaptive SPH fluids [3], and (c) our turbulence synthesis method.

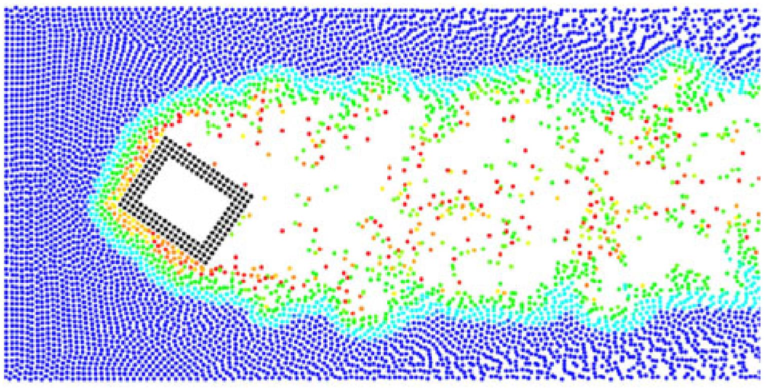
6.2. Turbulence Simulation Results

We have generated several both 2D and 3D animations to assess the advantages of our turbulence simulation method. The testing cases have been implemented on a PC with an Intel Dual-Core 2.8 CPU, Geforce GTX580 GPU, and 8 GB RAM. In 3D animations, the implicit surface of the fluid is reconstructed by our octree-based surface reconstruction method, and then, the extracted triangle meshes are rendered by Pov-Ray. In 2D animations, we directly use particles to represent fluids and solids and ignore the effect of gravity. The parameter values of the simulation are documented in Table III.

Figure 8 is an animation scenario that the fluid at the velocity of 20 m/s flows past a fixed black object, which demonstrates four steps of our turbulence synthesis model. For fluid particles, the color transition from red to yellow to blue illustrates that the magnitude of $\omega_i(t)$ changes from large to small. Figure 8(a) shows that our vorticity production model seeds VPs behind the object by approximating boundary layer theory on the fly in PCISPH fluids. In Figure 8(b), the fluctuating velocity $\mathbf{u}^v$, which is calculated by the proposed SPH-like summation interpolant formulation of the Biot–Savart law, is enforced on the mean velocity field $\mathbf{U}$. The spread of vorticities between fluid particles is solved by our stable implicit vorticity diffusion in Figure 8(c). Figure 8(d) shows that our artificial vorticity dissipation term is added to gradually reduce the vorticity strengths over time.

In Figure 9, we give the comparisons of the explicit vorticity diffusion [10,11,13,22] (left) and the implicit vorticity diffusion (right) under the larger time step $\Delta t = 0.03$. From the vorticity distributions given by the 2D scenario (Figure 9(a)), the vorticity field evolved by the explicit diffusion method oscillates and is dissipated too quickly. This is because that the incremented vorticity of a particle i might be larger than its vorticity difference ω_{ji} with neighboring particle j. As a result, in the 3D simulation (Figure 9(b)), the turbulence behind the elastic cylinder is hard to be observed when we use the explicit vorticity diffusion method. Compared to the explicit vorticity diffusion, the implicit vorticity diffusion method introduces some extra computations (Table IV) but improves the stability of the vorticity evolution.

Figure 10 shows an animation scenario that a dolphin swims across the water pool. Compared with the sim-


(a)

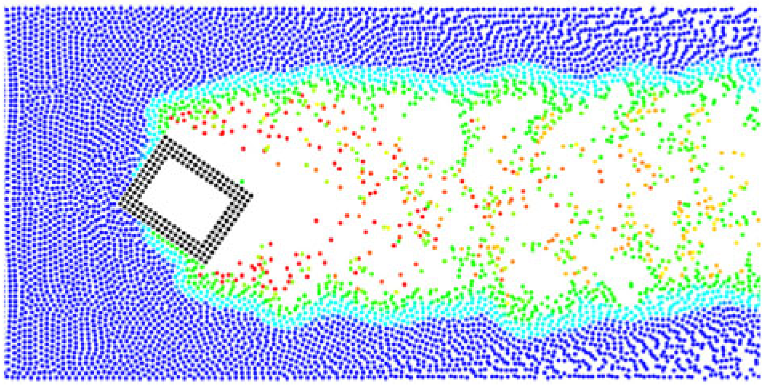

(b)

Figure 12. The comparison of vortex particle seeding between Yuan's turbulence method [10] (a) and our method (b).

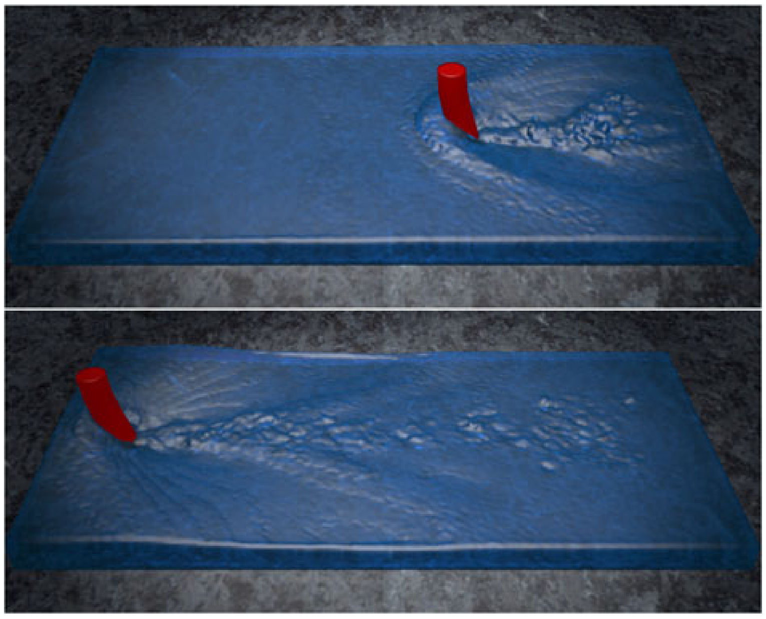


Figure 13. The turbulence behind an elastic cylinder moving in the water.

Table V. Averaged per frame timings of physical simulation, turbulence synthesis, surface particle extraction, and octree-based distance field construction.

Scene	t_{ps}[s]	t_{ts}[s]	t_{spe}[ms]	t_{ode}[s]
Figure 1	30.9	4.2	879	6.32
Figure 10(a)	10.3	1.43	340	2.17
Figure 10(b)	10.2	1.41	326	2.13
Figure 11(a)	15.6	/	492	3.25
Figure 11(b)	27.9	/	864	5.39
Figure 11(c)	15.5	2.69	498	3.31
Figure 13	8.3	1.05	294	1.73

ulation only computing the vorticity stretching and the implicit vorticity diffusion (Figure 10(a)), the simulation with artificial vorticity dissipation term (Figure 10(b)) makes the turbulence behind the dolphin more stable and plausible.

Figure 11 is an animation scenario that a boat sails on the water at the velocity of 20 m/s. Figure 11(a) shows that traditional SPH fluids without the turbulence model produce too smooth water surface and lack small-scale details; Figure 11(b) shows the results of adopting adaptive sampling method [3] in the simulation. Although the fluid particles near the free surface are split into one tenth of the original size, the simulation only creates bigger waves and more water splashes and still cannot recover the turbulent details; as shown in Figure 11(c), our turbulence synthesis method produces visually plausible turbulence behind the boat by approximating the boundary layer theory in SPH, which significantly improves the visual realism of the fluid–solid coupling. For a better comparison, please see the supplemental video.

In Figure 12, we compare our turbulence synthesis method (Figure 12(b)) with Yuan's turbulence method [10] (Figure 12(a)). Due to the stochastically seeded VPs in the fluid–solid coupling, Yuan's method creates some turbulence ahead of the fixed object, which makes the simulation unrealistic. Our method produces visually plausible turbulence behind the object immersed in SPH fluids. In addition, the implicit vorticity diffusion and the artificial vorticity dissipation adopted in our method make the turbulence evolution more stable and realistic.

Figure 13 displays the snapshots of an elastic cylinder moving in the water at the velocity of 15 m/s. The turbulent details are physically plausible generated behind the moving cylinder, which greatly improves the simulation reality of fluid–solid coupling.

For the presented 3D examples, Table V shows averaged per frame timings of physical simulation, turbulence synthesis, surface particle extraction, and octree-based distance field construction. Because our turbulence method introduces small extra computational load, the high efficiency is a remarkable advantage of our turbulence synthesis method. Compared with Yuan's method [10], our turbulence method runs a little slower due to the implicit vorticity diffusion and the artificial vorticity dissipation, but generates more stable and realistic turbulence.

7. CONCLUSIONS

We have presented a realistic and stable method to simulate the turbulent details behind the solid objects immersed in SPH fluids. By approximating the boundary layer theory on the fly in SPH, a turbulence production model is designed to create the obstacle-induced vorticity field. We employ an SPH-like summation interpolant formulation of the Biot–Savart law to enforce the fluctuating velocities stemming from the vorticity field on the global flow. And under the combined effect of both the implicit vorticity diffusion and the artificial vorticity dissipation, the generated vorticity field is stably evolved. Moreover, in order to catch fine turbulent details at the fluid surface for rendering, we have proposed an efficient octree-based adaptive surface reconstruction method for particle-based fluids.

The generated turbulence is visually plausible and stable, which is beneficial to interactive applications such as VR and games. However, the turbulence synthesis method is not physically completely accurate because the energy is not conserved in the generation and dissipation of the vorticity field. So our method cannot be applied to virtual surgery systems which require physically accurate coupling results. Our immediate efforts are geared toward capturing bubbles and foams in the fluid–solid coupling and reconstructing the more smooth fluid surface using anisotropic kernels [46]. In addition, the unified particle framework will be entirely implemented on GPUs for interactive frame rates.

ACKNOWLEDGEMENTS

This work is supported by the National 863 Program of China (Grant No. 2012AA011801), the National Key Technology R&D Program of China (Grant No. 2012BAI06B01). We thank the anonymous reviewers for their constructive comments.

REFERENCES

1. Kim B, Liu Y, Llamas I, Rossignac JR. Flowfixer: using bfecc for fluid simulation. In *Proceedings of Eurographics Workshop on Natural Phenomena*, Dublin, 2005; 51–56.
2. Molemaker J, Cohen JM, Patel S, Noh J. Low viscosity flow simulations for animation. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Dublin, 2008; 9–18.
3. Adams B, Pauly M, Keiser R, Guibas LJ. Adaptively sampled particle fluids. *ACM Transactions on Graphics* 2007; **26**(3): 48–54.
4. Solenthaler B, Gross M. Two-scale particle simulation. In *Proceedings of SIGGRAPH*, Vancouver, Canada, 2011; 811–818.
5. Bridson R, Houriham J, Nordenstam M. Curl-noise for procedural fluid flow. In *Proceedings of SIGGRAPH*, San Diego, California, USA, 2007.
6. Kim T, Thürey N, James D, Gross M. Wavelet turbulence for fluid simulation. In *Proceedings of SIGGRAPH*, Los Angeles, 2008; 27–33.
7. Schechter H, Bridson R. Evolving sub-grid turbulence for smoke animation. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation*, Dublin, 2008; 1–7.
8. Narain R, Sewall J, M MC, Lin MC. Fast animation of turbulence using energy transport and procedural synthesis. *Transactions on Graphics* 2008; **27**(5): 1–8.
9. Yoon JC, Kam HR, Hong JM, Kang SJ, Kim CH. Procedural synthesis using vortex particle method for fluid simulation. *Computer Graphics Forum* 2009; **28**(7): 1853–1859.
10. Yuan Z, Zhao Y, Chen F. Incorporating stochastic turbulence in particle-based fluid simulation. *The Visual Computer* 2012; **28**(5): 435–444.
11. Zhu B, Yang X, Fan Y. Creating and preserving vortical details in sph fluid. *Computer Graphics Forum* 2010; **29**(7): 2207–2214.
12. Pfaff T, Thurey N, Selle A, Gross M. Synthetic turbulence using artificial boundary layers. In *Proceedings of SIGGRAPH ASIA*, Yokohama, Japan, 2009; 1–10.
13. Shao X, Zhou Z, Zhang J, Wu W. Synthesizing solid-induced turbulence for particle-based fluids. In *Proceedings of 13th International Conference on Computer-Aided Design and Computer Graphics*, Hong Kong, China, 2013; 252–259.
14. Müller M, Charypar D, Gross M. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, San Diego, California, USA, 2003; 154–159.
15. Solenthaler B, Schlafli J, Pajarola R. A unified particle method for fluid-solid interactions. *Computer Animation and Virtual Worlds* 2007; **18**(1): 69–82.
16. Hadap S, Magnenat-Thalmann N. Modeling dynamic hair as a continuum. *Computer Graphics Forum* 2001; **20**(3): 329–338.
17. Iwasaki K, Uchida H, Dobashi Y. Fast particle-based visual simulation of ice melting. In *Proceedings of Pacafic Graphics*, Hangzhou, China, 2010; 2215–2223.
18. Ihmsen M, Wahl A, Teschner M. A lagrangian framework for simulating granular material with high detail. *Computer and Graphics* 2013; **37**(7): 800–808.
19. Solenthaler B, Pajarola R. Predictive-corrective incompressible sph. In *Proceedings of SIGGRAPH*, New Orleans, Louisiana, USA, 2009; 1–6.
20. Bao K, Zhang H, Zheng L, Wu E. Pressure corrected sph for fluid animation. *Computer Animation and Virtual Worlds* 2009; **20**(2-3): 311–320.
21. Clavet S, Beaudoin P, Poulin P. Particle-based viscoelastic fluid simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Los Angeles, California, USA, 2005; 219–228.
22. Müller M, Schirm S, Teschner M, Heidelberger B, Gross M. Interaction of fluids with deformable solids. *Computer Animation and Virtual Worlds* 2004; **15**(3-4): 159–171.
23. Schechter H, Bridson R. Ghost sph for animating water. In *Proceeding of SIGGRAPH*, Los Angeles, California, USA, 2012; 41–50.
24. Akinci N, Cornelis J, Akinci G, Teschner M. Coupling elastic solids with sph fluids. *Computer Animation and Virtual Worlds* 2013; **24**(3-4): 195–203.
25. Akinci N, Ihmsen M, Akinci G, Solenthaler B, Teschner M. Versatile rigid-fluid coupling for incompressible sph. *Transactions on Graphics* 2012; **31**(4): 62:1–62:8.
26. Fedkiw R, Stam J, Jensen HW. Visual simulation of smoke. In *Proceedings of SIGGRAPH*, Los Angeles, California, USA, 2001; 15–22.
27. Losasso F, Gibou F, Fedkiw R. Simulating water and smoke with an oc-tree data structure. In *Proceedings of SIGGRAPH*, Los Angeles, California, USA, 2004; 457–462.
28. Selle A, Rasmussen N, Fedkiw R. A vortex particle method for smoke, water and explosions. *Transactions on Graphics* 2005; **24**(3): 910–914.
29. Park SI, Kim MJ. Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Los Angeles, California, USA, 2005; 261–270.
30. Yu Q, Neyret F, Bruneton E, Holzschuch N. Scalable real-time animation of rivers. In *Proceedings*

of EUROGRAPHICS, Munich, Germany, 2009; 125–131.
31. Pfaff T, Thuerey N, Cohen J, Tariq S, Gross M. Scalable fluid simulation using anisotropic turbulence particles. *Transactions on Graphics* 2010; **29**(6): 174–182.
32. Jang T, Hwang H, Cha S, You M. Simulating water turbulence in sph fluids. In *Proceedings of Computer Graphics International*, Bournemouth, UK, 2012; 213–220.
33. Zhu J, Liu Y, Chang Y, Wu E. Animating turbulent water by vortex shedding in pic/flip. *SCIENCE CHINA Information Sciences* 2013; **56**(3): 1–11.
34. Pan Z, Huang J, Tong Y, Bao H. Wake synthesis for shallow water equation. *Computer Graphics Forum* 2012; **31**(7): 2029–2036.
35. Becker M, Ihmsen M, Teschner M. Corotated sph for deformable solids. In *Proceedings of Eurographics Workshop on Natural Phenomena*, Munich, Germany, 2009; 27–34.
36. Müller M, Heidelberger B, Teschner M, Gross M. Meshless deformations based on shape matching. In *Proceedings of SIGGRAPH*, Los Angeles, California, USA, 2005; 471–478.
37. Solenthaler B, Pajarola R. Density contrast sph interfaces. In *Proceedings of the 2008 ACM/ Eurographics Symposium on Computer Animation*, Dublin, 2008; 211–218.
38. Müller M, Solenthaler B, Keiser R, Gross M. Particle-based fluid-fluid interaction. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Los Angeles, California, USA, 2005; 237–244.
39. Cottet GH, Koumoutsakos PD. *Vortex Methods: Theory and Practice*. Cambridge University Press: Cambridge, United Kingdom, 2000.
40. Stam J. Stable fluids. In *Proceedings of SIGGRAPH*, Los Angeles, California, USA, 1999; 121–128.
41. Monaghan JJ. Implicit sph drag and dusty gas dynamics. *Journal of Computational Physics* 1997; **138**(2): 801–820.
42. Hasinoff SW, Kutulakos KN. Photo-consistent reconstruction of semitransparent scenes by density-sheet decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2007; **29**(5): 870–885.
43. Marrone S, Colagrossi A, Touze DL, Graziani G. Fast free-surface detection and level-set function definition in sph solvers. *Journal of Computational Physics* 2010; **229**(10): 3652–3663.
44. Colagrossi A., Landrini M. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics* 2003; **191**(2): 448–475.
45. Akinci G, Ihmsen M, Akinci N, Teschner M. Parallel surface reconstruction for particle-based fluids. *Computer Graphics Forum* 2012; **31**(6): 1797–1809.
46. Yu J, Turk G. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Madrid, 2010; 217–225.

SUPPORTING INFORMATION

Supporting information may be found in the online version of this article.

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 and distributed virtual reality.

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 and distributed virtual reality.