A Hybrid Method of Modelling Solid-Induced Turbulence for Incompressible Fluids

Xu-Qiang Shao[1] Rong-Hua Zhang[1] Ying Yang[2]

[1] School of Control and Computer Engineering, North China Electric Power University

Baoding 071003, Hebei, China

shaoxuqiang@163.com, zrh1015@sina.com

[2] Liren College, Yanshan University

Qinhuangdao 066004, Hebei, China

Ysu_yangying@163.com

Received 28 March 2015; Revised 27 May 2015 ; Accepted 10 August 2015

Abstract. The simulation of turbulent details generated around solid boundaries is crucial for visually realistic fluid animations. We present a new hybrid approach aimed at incorporating solid-induced turbulence into the particle-based SPH fluid solver. Based on a novel adaptive sampling method, we split particles in regions around solid boundaries to more finely capture solid-induced turbulence, and merge small particles in regions away from solid boundaries to promote efficiency. Furthermore, a turbulence production model is proposed to identify the small particles which separate from solid boundaries and obtain the vorticity information. We employ a hybrid scheme which combines coarse Euler grid and Lagrangian particle to enforce fluid incompressibility and solve turbulence evolution. Our method provides a physically plausible way to model turbulence details generated around solid boundaries in particle-based fluid solvers, from which the results demonstrate a significant improvement in the quality of visual details as compared to existing methods.

Keywords: turbulence, fluid simulation, smoothed particle hydrodynamics, particle-based method

1 Introduction

In computer graphics, physically based simulation of fluid is becoming increasingly popular to create realistic animation of many complex phenomena, such as water, smoke and fire. Actually, more interesting natural phenomena are the result of fluids interacting with solid objects immersed in a flow. For current fluid solvers, however, significant problems still remains in the capture of turbulence details induced by solids immersed in a flow. One of the prominent difficulties in the popular Eulerian grid fluid solver is the loss of small-scale details due to numerical diffusion caused by the low discretization resolution. To model the turbulence formation around objects immersed in a flow, high-resolution grids [1], [2], [3] are widely adopted in local concerned region to get vivid results. But the generated turbulence details are difficult to be preserved in the global flow with the coarse grid, when they move out of the high-resolution region.

Lagrangian particle method is becoming a competitive alternative to Eulerian grid method, in particular for animations with many small droplets and turbulence details of fluids. For example, the method in [4] used vortex particles to represent vortical details in fluids. Some researches attempt to combine the strengths of two different methods, and many hybrid grid-particle methods [5], [6], [7] which incorporate some vortex particles into Eulerian grid fluid solvers are presented to model turbulence effects. However, most of them deal with the preservation of vortices already existed in the overall flow rather than the turbulence formation around objects immersed in a flow. In [8], a physically plausible turbulence model is presented to seed vortex particles where the separated boundary layers will transit into actual turbulence. However, it is time-consuming to create a pre-computed artificial boundary layer that captures the characteristics of turbulence generation around objects.

To our knowledge, it is still a challenge to model the realistic turbulence formation and evolve turbulent details in particle-based fluid solvers like SPH, because the requirement of a higher particle resolution to capture turbulence details becomes the system bottleneck. Recently, Level of Detail technique [9], [10], [11] has been proposed to improve efficiency in the simulation process by focusing on visually important regions such as near the free surface or around objects. In this paper, we provide a physically plausible method to simulate turbulence induced by solid boundaries in particle-based fluid solvers. Based on a new adaptive sampling scheme, our approach split particles in regions around objects to more finely capture solid-induced turbulence, and merge parti-

cles in regions away from solid boundaries to avoid oversampling and lend efficiency to the simulation. The small particle which separates from solids and becomes unstable will obtain the vorticity information, and can be regarded as a vortex particle. The evolution of generated turbulence as well as fluid incompressibility is solved under the help of the coarse uniform grids. To sum up, our method is inspired by turbulence method using artificial boundary layers [8], the adaptive particle-based method [9], and the incompressible hybird SPH method [12]. It can be regarded as a complementary for a standard SPH simulator to simulate turbulence details induced by solid boundaries. The main contributions of this paper can be described as follows:

(1) Based on a new adaptive sampling scheme, we split particles near solid objects to allocate computing resources to regions with turbulence details, and merge small particles in regions away from solids.

(2) A turbulence production model is proposed to identify the small particles which separate from solid boundaries and would obtain the vorticity information.

(3) We employ a hybrid scheme which combines coarse Euler grid and Lagrangian particle to enforce fluid incompressibility and solve turbulence evolution.

2 Related Work

Over the past decades, physically based fluid animations such as smoke and water have received considerable attention in computer graphics. There exist two broad categories for simulation methods based on their different approaches to spatial discretization: Eulerian grid method [13], [14] and Lagrangian particle method [15], [16]. For a more thorough review, we refer readers to the recent paper [17].

The inherent damping of turbulence details due to numerical dissipation is one of the prominent difficulties in fluid simulation, especially in the popular Eulerian approach. To solve this problem, Fedkiw et al. [18] proposed vorticity confinement to amplify existing grid vorticity in the fluid. However, if the uniform simulation grid is not fine enough to capture the desired details, vorticity confinement can not recover them. The numerical dissipation problem can also be alleviated by using higher order advection schemes [19], [20], [21], but turbulence details represented by these methods are still inherently limited by the resolution of the underlying grid. By solving the Navier-Stokes equations with higher-resolution grids in a local concerned space, several approaches [1], [2], [3] can accurately model the small-scale turbulence formation around objects immersed in a flow. However, this assumes that only a small part of a simulation needs high detail. When moving out of the high-resolution region, these vortices are difficult to be preserved in the coarse grid.

Recently, procedural methods for adding further details in sub-grid by using noise have been introduced. Kim et al. [22] used wavelet method and synthesized missing turbulent flow components with band-limited wavelet noise. This method helps to show turbulence details in a coarse grid. Schechter and Bridson [23] presented a sub-grid turbulence evolution model for fluid simulation. They tracked bands of turbulent energy using a simple linear model, and created the turbulent velocity using flow noise. They also added a predictor step to the usual time splitting of the incompressible Euler equation, and corrected the additional vorticity dissipation caused by time splitting of the pressure and advection. Narain et al. [24] introduced a technique coupling a procedural turbulence model with a numerical fluid solver. They used an energy function and a Lagrangian approach to advect noise when synthesizing an incompressible turbulent velocity field. In addition, they applied the method to the simulation of liquids with free surfaces.

The vortex particle method is more suitable to model small-scale details in fluids, and was introduced to the graphics community by Gamito et al [25]. Selle et al. [5] made use of vortex particles to introduce additional vorticity into the grid fluid for highly turbulent flows. But their method requires the artist to specify where these particles are injected into the flow, and does not reproduce the decay of turbulence when the force is removed. Thus, it is suited for scenarios where a constant source of turbulence vorticity is known in advance. Some other grid-particle methods [6], [7], [26] incorporating vortex particles into Eulerian grid fluid solvers are used to model turbulence effects. However, most of them mainly deal with the preservation of vortices already represented in the overall flow rather than the turbulence formation around objects immersed in a flow. In addition, Park and Kim [4] model gaseous phenomena by entirely distributing vortex particles on the whole grid and utilizing a pure Lagrangian simulation with vorticity transport equation.

While the preceding approaches achieve interesting turbulent behavior in fluids, a major source of turbulence is the interaction of fluids with solids. By carrying vorticity information on SPH particles, the method in [27] created the turbulence details around the objects immersed in flows and preserved these details in particle-based SPH smoke. The visual effects obtained by their methods are vivid, but perhaps not physically plausible. The method in [8] employed vortex particles in a grid-based fluid solver to introduce obstacle-induced turbulence and obtained physically plausible results. During a fluid simulation, the method identifies areas where the separated layers will transit into actual turbulence and seeds vortex particles. After seeding, the method solves energy transport equations to determine when the particles should increase or reduce their chaotic agitation, and correspondingly heuristic rules are developed for particle merging and splitting. However, it is time-consuming to

create a pre-computed artificial boundary layer that captures the characteristics of turbulence generation around objects. In this paper, we proposed a novel adaptive sampling method for SPH fluid. Our method splits particles in regions around solid objects to more finely capture turbulence, and merges small particles in regions away from solids to promote efficiency. Then a turbulence model is proposed to identify the small particles which separate from solid boundaries and will obtain the vorticity information.

Incompressibility property of fluids plays an important role in the formation of important visual effects like splashes and dynamic turbulent motions. SPH method was originally designed to model compressible or weakly compressible flow. In order to simulate incompressible fluids in SPH, instead of preserving a uniform particle density [28], Karthik et al. [12] proposed a hybrid method which uses a Poisson solve on an auxiliary coarse grid to enforce a divergence free velocity field and permits a significantly larger time step. Greatly inspired by their work, we use an auxiliary coarse grid to enforce incompressibility for our splittable particle-based SPH fluid solver.

3 Particle-Based Fluid Simulation

In this section, we will briefly introduce our SPH-based fluid solver, and show how we calculate the incompressibility of fluid and solid-fluid interaction.

3.1 SPH Method Summary

Generally, the governing Navier-Stokes equations of Lagrangian fluid solvers are given as:

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{f} \quad , \tag{1}$$

$$\frac{d\rho}{dt} = -\rho\nabla\cdot\mathbf{u} \ , \tag{2}$$

which are composed of two equations, i.e., momentum equation and continuity equation, respectively. The value $\mathbf{u}$ denotes the velocity field of the fluid, ρ denotes the density, p the pressure, $\mathbf{f}$ the external forces and μ viscosity coefficient.

Among Lagrangian methods, SPH is one of the most effective model in computational fluid dynamic (CFD), which represents fluid as a collection of particles. Field quantities are expressed as a summation of physical values each of which is weighted by smoothing kernel product in the vicinity of each particle. A physical quantity $A(\mathbf{x}_i)$ at the position $\mathbf{x}_i$ can be computed by the equation

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

where m_j is the mass of neighboring particle j, $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. The function $W(\mathbf{x})$ is called the smoothing kernel with the support radius h which is a scaling factor that controls the smoothness or roughness of the kernel. Here we use different kernel functions in [15] for evaluating different field quantities in order to enhance the precision and the robustness of our algorithm.

In SPH approach, the derivatives only affect the smoothing kernel. The gradient and Laplacian of the smooth attribute function $A(\mathbf{x})$ are

$$\nabla A(\mathbf{x}_i) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(\mathbf{x}_{ij}, h) \ , \tag{4}$$

$$\nabla^2 A(\mathbf{x}_i) = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(\mathbf{x}_{ij}, h) \ . \tag{5}$$

Since our method adopts a new adaptive sampling scheme which will be described in next section, a particle may have other neighborhood particles with different radius. Following the method in [9], we substitute Equation

(4) and Equation (5) into the Navier-Stokes momentum equation and obtain the symmetric pressure and viscosity forces acting from particle p_j on p_i:

$$\mathbf{f}_{ij}^{pres} = -V_i V_j (P_i + P_j) \frac{\nabla W(\mathbf{x}_{ij}, h_i) + \nabla W(\mathbf{x}_{ij}, h_j)}{2} , \tag{6}$$

$$\mathbf{f}_{ij}^{visc} = -\mu V_i V_j (\mathbf{u}_j - \mathbf{u}_i) \frac{\nabla^2 W(\mathbf{x}_{ij}, h_i) + \nabla^2 W(\mathbf{x}_{ij}, h_j)}{2} \tag{7}$$

with the particle volume $V_i = m_i / \rho_i$.

3.2 Fluid-Solid Interaction

In general, SPH-based fluid solvers calculate wall boundaries by generating wall particles [29]. Muller et al. developed a method in which fluid particles interact with deformable solids [30]. This method computes particle interaction with solid meshes by generating temporary particles on the surface of the mesh at every time step. Since the density of generated particles varies among polygons which do not have a uniform surface area, it does not guarantee that the constant particle density is near the wall boundary. In addition, the number of wall particles is proportional to the surface area of the boundary. Thus the shape of the boundary gets much complex, and the number of wall particles also increases.

In order to avoid generating too many wall particles when calculating the interaction between fluid and solid boundaries, we use a modified version of the method in [31]. When coupling fluid and solid, we interpolate the distance d_{ik} of fluid particle p_i to the solid object S_k and the normal $\mathbf{n}_{ik}$ based on the pre-computed signed distance fields [32]. If $d_{ik} < 2h_i$, we add a temporary virtual solid particle p_k whose distance to p_i along the direction of $\mathbf{n}_{ik}$ is d_{ik}. Then the normal force $\mathbf{f}_{normal}$ and the tangent force $\mathbf{f}_{\tan gent}$ exerted on fluid particle p_i are computed as:

$$\mathbf{f}_i^{normal} = k_a m_i \nabla W(\mathbf{x}_{ik}, h_i) \mathbf{n}_{ik} , \tag{8}$$

$$\mathbf{f}_i^{\tan gent} = k_b m_i \nabla^2 W(\mathbf{x}_{ik}, h_i) \frac{\mathbf{u}_{ki} - (\mathbf{u}_{ki} \cdot \mathbf{n}_{ik}) \mathbf{n}_{ik}}{\left| \mathbf{u}_{ki} - (\mathbf{u}_{ki} \cdot \mathbf{n}_{ik}) \mathbf{n}_{ik} \right|} , \tag{9}$$

where $\mathbf{u}_{ki} = \mathbf{u}_k - \mathbf{u}_i$, k_a and k_b are user-defined elasticity and friction coefficients respectively. The kernel function W is defined as:

$$W(\mathbf{x}_{ik}, h_i) = \frac{1}{\left| \mathbf{x}_{ik} \right|} \begin{Bmatrix} \frac{2}{3}, & 0 < q < \frac{2}{3} \\ 2q - \frac{3}{2} q^2, & \frac{2}{3} \le q < 1 \\ \frac{1}{2}(2-q)^2, & 1 \le q < 2 \\ 0, & otherwise \end{Bmatrix} , \tag{10}$$

where $q = \left| \mathbf{x}_{ik} \right| / h_i$.

3.3 Incompressibility Enforcement Method

The most popular Lagrangian fluid simulation method SPH was originally designed to model compressible flow. However, incompressibility plays an important role in creating realistic animations of fluids such as water, where this property leads to the formation of important visual effects like splashes and dynamic turbulent motions. In order to simulate incompressible fluids with SPH, researchers have proposed various techniques that either enforce a divergence free velocity field or preserve a uniform particle density. The weakly compressible SPH

method (WCSPH) attempts to enforce uniform particle density by introducing a stiff equation of state, which imposes a severe timestep restriction on the simulation [33]. In PCISPH [28], pressures are determined through a predictor-corrector approach so that the maximum density does not exceed a defined threshold, but the cost per timestep is much higher.

A different strategy that makes SPH-based fluid impressible is to enforce a divergence free velocity field. However, solving a divergence free velocity equation on each particle [34] usually makes the simulation of a large number of particles intractable. Karthik et al. [12] proposed a hybrid SPH method which samples particle velocities on a coarse uniform grid and solves for pressure values that enforce a divergence free velocity field on this grid. In this paper, we adopt the idea of Karthik's method and optimize it. Our overall incompressibility scheme is:

(1) Integrate viscosity forces and other external forces such as gravity and boundary forces to obtain intermediate particle velocities.

(2) Construct a coarse staggered Marker-and-Cell (MAC) grid around the particles.

(3) Interpolate particle immediate velocities onto grid faces.

(4) Classify cells as solid, fluid or air.

(5) Construct linear system and solve Poisson equation.

(6) Interpolate pressure values back onto particles.

(7) Compute the pressure force between each pair of particles using SPH kernels.

Different from the integration process of Karthik's method, our method uses time splitting of the incompressible fluid equations, which integrate viscosity forces and other external forces to obtain intermediate particle velocities. In addition, Karthik's method is initially designed to enforce the incompressibility of SPH method with uniform particle size. Since our SPH fluid solver adopts an adaptive sampling scheme, each particle has a different mass and radius. In our framework, we optimize the hybrid SPH method and extend it to our new adaptive SPH fluid solver. When particle velocity is transferred to the auxiliary uniform grid, the contribution of a particle to a grid point is weighted by the mass of fluid represented by the particle. Given a particle distribution, the fluid velocity $\mathbf{u}_g$ of grid point at location $\mathbf{x}_g$ is computed as:

$$\mathbf{u}_g = \frac{\sum_i m_i W(\mathbf{x}_{gi}, d_g)\mathbf{u}_i}{\sum_i m_i W(\mathbf{x}_{gi}, d_g)}, \tag{11}$$

where d_g is the grid cell width, $\mathbf{x}_{gi} = \mathbf{x}_g - \mathbf{x}_i$. The weighting kernel W is formulated as:

$$W(\mathbf{r}, h) = \begin{Bmatrix} 1 - \|\mathbf{r}\|^2 / h^2, & 0 \le \|\mathbf{r}\| \le h \\ 0, & otherwise \end{Bmatrix}. \tag{12}$$

4 Turbulence Evolution

4.1 Adaptive Sampling Scheme

The physical and visual quality of SPH fluid solvers is defined by the number of particles that are used to discretize the fluid. Generally, the more particles that are used, the smaller the damping artifacts and the more small-scale details like droplets and vortices can be reproduced. However, using a large number of small particles to obtain a finer animation leads to enormous computation cost, which is often unnecessary. Greatly inspired by some previous Level of Detail techniques [9], [10], [11] which allocate computing resources to regions where complex flow behavior emerges, our approach splits large particles in regions around solid objects to more finely capture solid-induced turbulence, and merges small particles in regions away from solid to promote efficiency. Each particle p_i is defined by its position $\mathbf{x}_i$, mass m_i, independent radius $r_i = \sqrt[3]{3m_i / 4\pi\rho_0}$, and support radius $h_i = \xi\sqrt[3]{m_i / \rho_0}$. In our system, the initial density ρ_0 equals to 1000, and ξ is valued according to the average number of neighboring particles we want to obtain around p_i. The initial particles are the largest ones used in our fluid algorithm, and have mass m_0 and support radius h_0. Whether a particle p_i should be split into finer

ones or be merged is determined by its distance to the solid boundaries d_i interpolated from the pre-computed signed distance field. In practice, we split or merge particles according to the following rules:

Splitting If $d_i < \alpha h_0$, the large particle p_i will be split into several small particles (we use $\alpha = 2$ in all our simulations), so there are more particles in regions around solid objects to more finely capture turbulence details in fluid-solid interaction. As the splitting scheme illustrated in 2D in Fig. 1 (a), at each step eight candidate sub-particles are placed on vertices of an axial bounding cube around the splittable parent particle p_i. The mass m_c and support radius h_c of each candidate sub-particle can be calculated by $m_c = m_i / 8$ and $h_c = \sqrt[3]{m_c / m_i} h_i$. However, to improve stability and maintain a uniform particle distribution, the candidate particles cannot be inserted naively because these candidates are usually too closer to solid boundaries and existing particles or within solid volume. To avoid the introduction of large pressure forces, we choose the candidate sub-particles which are not closer to any other particle or solid boundary than $h_c / 4$ to be the truly created sub-particles. Once the number n of true sub-particles of the parent particle p_i is determined, the mass and support radius of each sub-particle are calculated by $m = m_i / n$ and $h = \sqrt[3]{1/n} h_i$ respectively. According to a given compromise between precision and computational efficiency, the particle whose mass is less than $m_0 / 64$ would not be split any more. Our cubical splitting scheme can keep the volume of fluid the same as it was, and results in a regular particle distribution around solids compared to a 2-particles in [9].

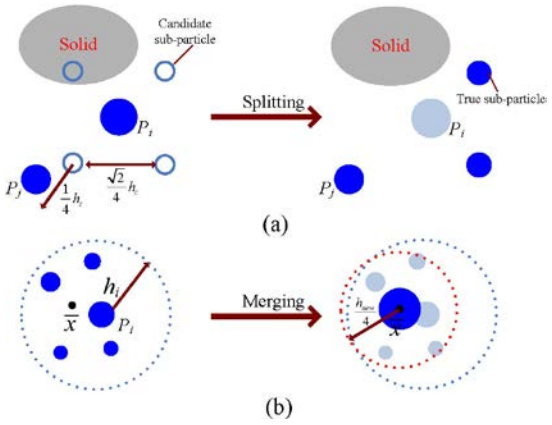


Fig. 1. Particle splitting and merging

Merging As shown in Fig. 1(b), if $d_i > \beta h_0$ (we use $\beta = 4$ in all our simulations), the small particle p_i whose mass is less than m_0 searches for other small particles p_j among its neighborhood particles in the order of the distance increasing to merge. When the total mass m_{new} of p_i and current found particles is larger than m_0, the search process is suspended. A new particle with a support radius $h_{new} = \xi \sqrt[3]{m_{new} / \rho_0}$ is created at the center of gravity $\overline{\mathbf{x}} = \dfrac{m_i \mathbf{x}_i + \sum_j m_j \mathbf{x}_j}{m_i + \sum_j m_j}$ of the original small particles if this position is within the fluid volume and there is no other particles within the distance $h_{new} / 4$. The velocity of the new particle is set to $\dfrac{m_i \mathbf{u}_i + \sum_j m_j \mathbf{u}_j}{m_i + \sum_j m_j}$. In addition, we merge particle vorticities produced by our vorticity production model which will be described in the next section. The new particle's vorticity $\boldsymbol{\omega}$ is set to $\dfrac{m_i \boldsymbol{\omega}_i + \sum_j m_j \boldsymbol{\omega}_i}{m_i + \sum_j m_j}$. Compared to the merging scheme in [9], our method can merge particles with different masses.

4.2 Vorticity Production Model

Solid boundaries are obvious turbulence generators. According to boundary layer theory [35], the friction of solid boundaries enforces a tangential flow velocity of zero at the solid boundary. This leads to the formation of a thin layer with reduced flow speed, called the boundary layer. The gradient of tangential flow velocity $\mathbf{u}_{\tan}$ in the boundary layer leads to the creation of a thin sheet of vorticity $\boldsymbol{\omega} = \nabla \times \mathbf{u}_{\tan}$. At regions of high flow instability, the boundary layer is separated from the solid boundary, and vorticity is ejected from the boundary layer and enters the flow as turbulence. In graphics, this effect was modeled in Euler fluid through pre-computing an artificial boundary layer considering the tangential flow profile and layer separation [8]. However, the method in [8] is time-consuming to create a pre-computed artificial boundary layer that captures the characteristics of turbulence generation around objects.

In fluid mechanics, the Reynolds Number R_e, which is the ratio of inertial force to viscous force, can be used to identify the boundary layer. Different from pre-computing an artificial boundary layer in Euler fluid [8], to identify the small SPH particles in the boundary layer, we define a factor named R_{BL} roughly proportional to R_e:

$$R_{BL} = \frac{(\mathbf{u} \cdot \nabla)\mathbf{u}}{\mu \nabla^2 \mathbf{u}} \tag{13}$$

by taking a ratio of the convection term and the diffusion term in the Navier-Stokes equation. And the convection term is computed by taking finite difference on an associated grid.

To detect where boundary layer separates, we calculate the value $\mathbf{u}_i \cdot \mathbf{n}_i$ for each particle p_i in the boundary layer, and $\mathbf{n}_i$ is interpolated from the pre-computed signed distance fields. If $\mathbf{u}_i \cdot \mathbf{n}_i > 0$, particle p_i is selected as a separated particle whose position is chosen as the location of boundary layer separating from solids. Next we need to identify regions where the separated boundary layer becomes unstable and transits to free turbulence. As described in [8], the anisotropic component a_{ij} of the Reynolds stress tensor R_{ij}, which is responsible for the production of turbulence, is a good indicator for such transition regions. As a modified version of the method in [8], we therefore define a transition probability density P_T for each separated particle in our particle-based fluid, which is used to seed turbulence,

$$P_T = c_p \Delta t \frac{\left\| a_{ij} \right\|}{\left| \overline{\mathbf{u}} \right|^2} \approx 2 c_p \Delta t l_m^{\ 2} \frac{\left| \boldsymbol{\omega}_s \right|}{\left| \overline{\mathbf{u}} \right|^2} , \tag{14}$$

where $\left\| \cdot \right\|$ denotes the Euclidean matrix norm, c_p is a parameter to control the seeding granularity, $\overline{\mathbf{u}}$ the average velocity, and for each particle $\boldsymbol{\omega}_s = \mathbf{u}_{\tan} \times \mathbf{n}_i$. The turbulent viscosity can be expressed in terms of a so called mixing length l_m, which in near-wall regions is given by the distance to the wall.

If P_T is larger than a user-defined threshed value, an initial swirling vorticity with value $\boldsymbol{\omega}_s$ is computed for each separated particle. The SPH particles carrying vorticities are regarded as vortex particles, and form the vorticity field of the fluid flow.

4.3 Turbulence Synthesis

The SPH particles are convected with the velocity field. We consider two kinds of velocity fields: One is $\mathbf{u}$ from the calculation of the momentum equation using SPH method, which is described before; the other is vorticity velocity $\mathbf{u}_v$ from vorticity. The total velocity field $\mathbf{u}_{total}$ is obtained by superposing these two velocity fields: $\mathbf{u}_{total} = \mathbf{u} + \mathbf{u}_v$.

Then we exploit a second-order Adams-Bashforth scheme for time integration as follows:

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \Delta t \left(\frac{3}{2} \mathbf{u}_i^n - \frac{1}{2} \mathbf{u}_i^{n-1} \right) , \tag{15}$$

where $\mathbf{x}_i^n$ is the position of particle i at time n and $\mathbf{u}_i^n$ is the velocity at $\mathbf{x}_i^n$.

The velocity field $\mathbf{u}_v$ induced by the vorticity field can be recovered via the Biot-Savart law, which computes $\mathbf{u}_v$ at the position $\mathbf{x}$ that is a distance $\mathbf{r} = \mathbf{x} - \mathbf{x}'$ from a vortex element $d\mathbf{x}'$ with vorticity $\boldsymbol{\omega}$ by integrating over all space:

$$\mathbf{u}_v(\mathbf{x}) = \frac{1}{4\pi}\int \frac{\boldsymbol{\omega}(\mathbf{x}')\times\mathbf{r}}{|\mathbf{r}|^3} d\mathbf{x}' \quad . \tag{16}$$

Discretizing the above equation, the integral becomes a simple summation:

$$\mathbf{u}_v(\mathbf{x}) = \frac{1}{4\pi}\sum_i \frac{\boldsymbol{\omega}_i \times \mathbf{r}_i}{|\mathbf{r}_i|^3} \quad . \tag{17}$$

If the simulation has n vortex particles, a straightforward summation would take $O(n^2)$ operations, which is too slow. So instead, based on the method in [36], we approximate the influence of clusters of multiple distance vortex particles as a single vortex particle. As long as those clusters are far away enough, this approximation works well for visual effects. Fig. 2 shows ways in which we can represent vortex particle clusters using the hierarchical data structure KD-Tree. The bottom row of nodes represents actual vortex particles. In the layer above that, each node represents clusters of vertex particles, and the layer above that represents clusters of clusters, and so on. To compute $\mathbf{u}_v$ at each vertex particle's position, our method traverses the tree from top down, as Fig. 2 shows: visit each node and ask whether the query location (colored red) is inside the region that node represents. If not, apply the influence of that cluster (shaded regions) as though it were a single vertex particle, whose position is the average position of all constituent vertex particles in the cluster: $\overline{\mathbf{x}} = \sum_i \omega_i \mathbf{x}_i$. If the query location lies within the cluster region, then descend that branch (following the arrows) and repeat.

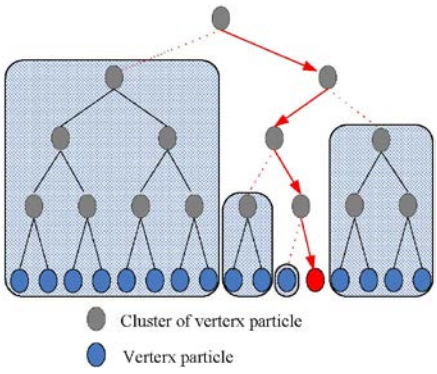


Fig. 2. KD-Tree of vertex particles

4.4 Development and Spreading

By taking the curl of Equation (1), the momentum equation of Navier-Stokes equations can be put into vorticity form

$$\frac{D\boldsymbol{\omega}}{Dt} = (\boldsymbol{\omega}\cdot\nabla)\mathbf{u} + \mu\nabla^2\boldsymbol{\omega} + \nabla\times\mathbf{f} \quad , \tag{18}$$

where $(\boldsymbol{\omega}\cdot\nabla)\mathbf{u}$ is a vortex stretching term, and the pressure term vanishes for constant density fluids.

In our simulation process, each vortex particle stores a vorticity value $\boldsymbol{\omega}$ which includes both a magnitude and direction. The fluid flow evolves as the particles move around and their vorticity values change. The solution of Equation (18) for each particle requires a velocity field, which can be calculated from the vorticity values stored on the individual particles using Equation (17). What is more, the vortex stretching term $(\boldsymbol{\omega}\cdot\nabla)\mathbf{u}$ involves spatial derivatives of the total velocity. In our method, a coarse uniform grid is employed to solve the spatial derivative of velocity for each particle. As we have done in Incompressibility Enforcement section, we first construct a coarse uniform grid around all vortex particles. Then interpolate particle velocities onto grid points. When parti-

cle velocity is transferred to the auxiliary grid, the weight of a particle's effect is determined by the mass of fluid represented by the particle. Finally, the value of the vortex stretching term for each particle is solved by the spatial derivatives computation of the velocity field on the grid with central differences, then the trilinear interpolation of them to the particle location, and finally the augmentation of the vorticity on the particle with $\boldsymbol{\omega} += \Delta t(\boldsymbol{\omega} \cdot \nabla)\mathbf{u}$.

Viscous flow rapidly dissipates vorticity according to the $\mu\nabla^2\boldsymbol{\omega}$ term in Equation (18). In particle-based fluid, this term spreads vorticity from a particle to its neighbors. So we implement this term as an exchange of vorticity between particles. In SPH fluid solver, the $\mu\nabla^2\boldsymbol{\omega}$ term for each particle is solved by

$$\mu\nabla^2\boldsymbol{\omega} = \mu\sum_i m_j \frac{\boldsymbol{\omega}_j - \boldsymbol{\omega}_i}{\rho_j}\nabla^2 W(\mathbf{x}_{ij}, h) \quad . \tag{19}$$

5 Results and Discussion

In this section, we implement our new method for the simulation of turbulence induced by solid boundaries in particle-based SPH fluid flow, and discuss comparisons of our approach to previous work. The simulation and rendering parts of our system are implemented on a Microsoft Windows XP PC with dual Intel Core 2.8 GHz CPUs, 2.0 GB RAM, and NVIDIA GeForce GTX 480 GPU. The parameter values of the simulation are documented in Table 1.

Table 1. Parameter values in the experiments

Properties	Values	Unit
Time step (Δt)	0.003	*s*
Initial spacing (r_0)	0.02	*m*
Support radius (h)	0.05	*m*
Density (ρ_0)	100-5000	*kg/m3*
Particle mass (m)	0.00054-0.013	*kg*
Viscosity (μ)	0.05	$Pa \cdot s$

For rendering 3D results, we construct a density volume from the particles using the method in [37], and apply the Marching Cubes method to generate surface mesh, which is then visualized with POV-Ray 3.7 rendering engine. The cell size for the Marching Cube method is set to 0.7*d*, and d is the rest particle spacing. The flowchart of our system is described in Fig. 3.

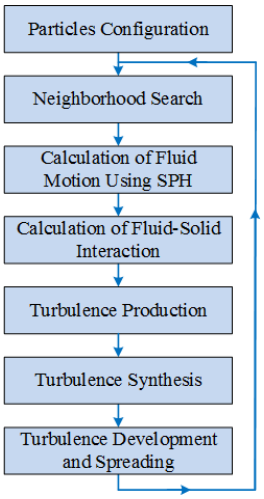


Fig. 3. Flowchart of the turbulence simulation

Fig. 4 shows the results of our new adaptive sampling method for particle-based fluid. Our method splits particles in regions around solid boundaries to more finely capture solid-induced turbulence, and merges small particles in regions away from solid boundaries to promote efficiency. The particle size difference is a user-defined value and can be chosen to be arbitrarily large. In our simulations, according to a given compromise between precision and computational efficiency, the particle whose mass is less than $m_0 / 64$ would not be split any more. Our cubical splitting scheme can keep the volume of fluid the same as it was, and results in a regular particle distribution around solids compared to Adams' 2-particles splitting method in [9] and spherical splitting method in [10]. In addition, our splitting method does not need sphere-sphere Boolean operations in spherical coordinates, so it runs twice as fast as Adams' splitting method in the water dam break simulation displayed by Fig. 5 and Fig. 6. Compared to the Adams' merging scheme, our method can merge particles with different masses.

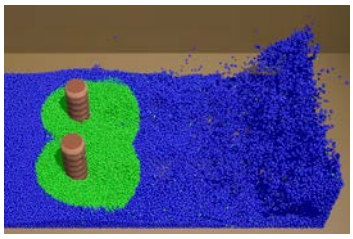


Fig. 4. Our adaptive sampling method which allocates computing resources to regions where turbulence emerges. the region around cylindrical obstacles is simulated with high resolution particles (green), the major remaining part of the fluid is computed with low resolution (blue)

Fig. 5 shows the distribution of particle's vorticity created by our turbulence model in the simulation of a water dam break passing through two cylinders. A color spectrum is used to dye the particles with the evolving swirling vorticity ω_i. Red to yellow to blue colors illustrate that the magnitude of ω_i changes from large to small. The color distribution of particles' vorticity indicates that the magnitude of particle's vorticity is determined by the tangential velocity, and our method can simulate the development and spreading of particles' vorticity.

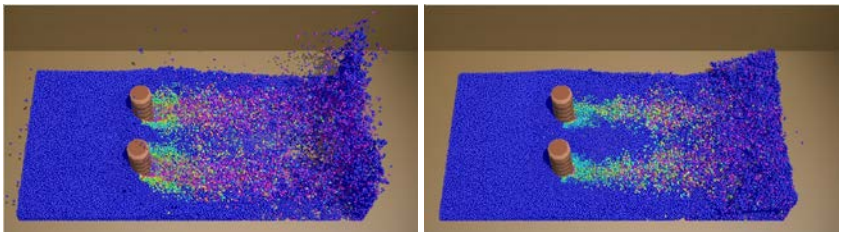
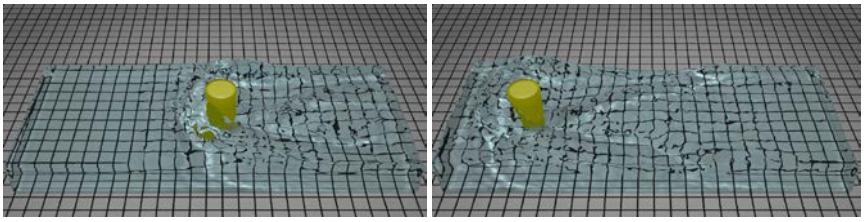
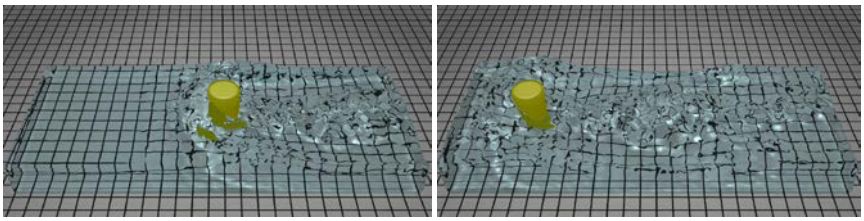


Fig. 5. Particle's vorticity induced by solid boundaries

In Fig. 6, we compared our method with the latest rigid-fluid coupling method [38]. In the simulation, the initial number of particles is 89,293. As illustrated Fig. 6(a), the rigid-fluid coupling method [38] generates large-scale waves and some droplets, but fails to capture turbulence. Fig. 6(b) shows that our turbulence simulation method can produce physically plausible turbulent details behind the solid obstacles. In addition, our adaptive sampling method adaptively splits particles into small ones near solids (maximum up to 105,665), so more splash details are produced. The average time per frame for these two methods is 1.7*s* and 2.1*s* respectively.



(a)



(b)

Fig. 6. The comparison between the rigid-fluid coupling method [38] and our method. (a) The rigid-fluid coupling method [38] without turbulence. (b) Our method with more droplets and turbulence

Our method can also simulate the turbulence of particle-based smoke. Fig. 7 shows the comparison between Bo Zhu's method [27] and our method. Fig. 7(a) shows the results of the rising smoke passing through the cylinder obstacles, which are simulated by Bo Zhu's method. The maximum number of particles is 100 *k*, and the average time per frame is 2.3 *s*. And Fig. 7(b) shows the results produced by our method. The maximum number of particles is 112 *k*, and the average time per frame is 2.7 *s*. From the comparison results, we can conclude that our method can calculate the fluid-solid interaction more robustly, and generate more physically plausible turbulence induced by the solid obstacles.

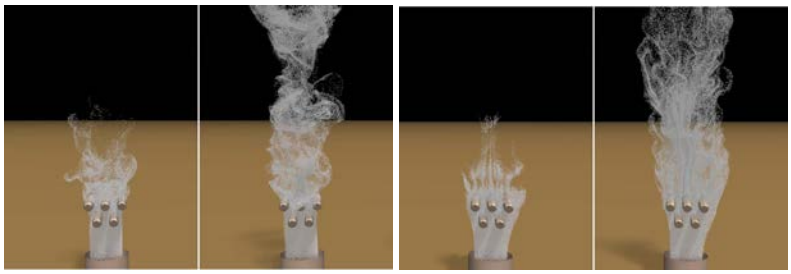


Fig. 7. Turbulence in smoke-solid interaction. (a) Bo Zhu's method [27]. (b) Our method

Figure 8 displays the snapshots of the water at the velocity of 5 *m/s* flows past a rigid tower model. The turbulent details are physically plausible generated behind the rigid tower, which greatly improves the simulation reality of fluid-solid coupling. In the simulation, the initial number of particles is 183 *k*, and the maximum number of particles is 212 *k*. The average time per frame throughout the simulation is 4.9 *s*.

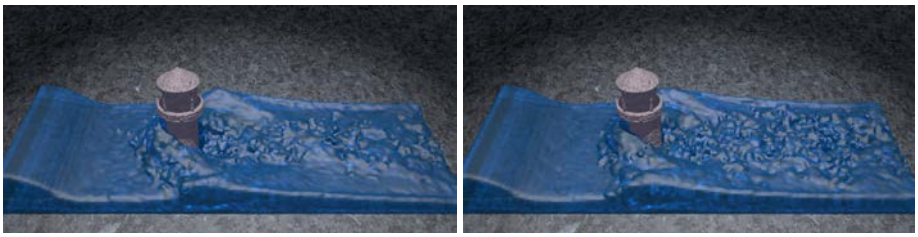


Fig. 8. The turbulence behind the rigid tower model immersed in the water

6 Conclusion

In this paper, we provide a new hybrid numerical method to simulate turbulence induced by solid boundaries in SPH fluid flow, which has not been well developed for the well-known Lagrangian particle scheme before. We propose a new adaptive sampling scheme, which splits particles near solid objects to allocate computing resources to regions with turbulence details, and merge small particles in regions away from solids to lend efficiency to the simulation. According to our plausible turbulence production model, the small particle which separates from solids and becomes unstable obtains the solid-induced vorticity information. And coarse uniform grids are

employed to enforce fluid incompressibility and solve turbulence evolution. The results show that our method can efficiently produce turbulence details generated around solid boundaries in particle-based fluid, and get a significant improvement in the quality of visual details over existing methods.

In future, we will extend our method to simulate the turbulence induced by deformable solids, and take advantage of GPGPU (General-Purpose computation on Graphics Processing Units) techniques to further promote the performance.

7 Acknowledgement

This work is supported by the National Natural Science Foundation of China (Grant no. 61502168) and the Fundamental Research Funds for the Central Universities (Grant no. 2015MS126). We thank the anonymous reviewers for their valuable comments and suggestions which greatly helped to improve the quality of the paper.

References

[1] F. Losasso, F. Gibou, R. Fedkiw, "Simulating Water and Smoke with An Octree Data Structure, " in *Proceedings of the 2004 ACM SIGGRAPH*, pp. 457-462, 2004.

[2] B.M. Klingner, B.E. Feldman, N. Chentanez, "Fluid Animation with Dynamic Meshes," *ACM Transactions on Graphics*, Vol. 25, pp. 820-825, 2006.

[3] Y. Dobashi, Y. Matsuda, T. Yamamoto, "A Fast Simulation Method Using Overlapping Grids for Interactions between Smoke and Rigid Objects," *Computer Graphics Forum*, Vol. 27, pp. 477-486, 2008.

[4] I. Pars, M.J. Kim, "Vortex Fluid for Gaseous Phenomena," in *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pp. 261-270, 2005.

[5] A. Selle, N. Rasmussen, R. Fedkiw, "A Vortex Particle Method for Smoke, Water and Explosions," *ACM Transactions on Graphics*, Vol. 24, pp. 910-914, 2005.

[6] J.C. Yoon, H.R. Kam, J.M. Hong, "Procedural Synthesis Using Vortex Particle Method for Fluid Simulation," *Computer Graphics Forum*, Vol. 28, pp. 1853-1859, 2009.

[7] Q. Yu, F. Neyret, E. Bruneton, "Scalable Real-time Animation of Rivers," in *Proceedings of Eurographics*, pp. 125-131, 2009.

[8] T. Pfaff, N. Thurey, A. Selle, "Synthetic Turbulence Using Artificial Boundary Layers," in *Proceedings of ACM SIGGRAPH Asia*, 2009.

[9] B. Adams, M. Pauly, R. Keiser, "Adaptively Sampled Particle Fluids," *ACM Transactions on Graphics*, Vol. 26, pp. 48-54, 2007.

[10] H. Yan, Z.Y. Wang, J. He, "Real-time Fluid Simulation with Adaptive SPH," *Computer Animation and Virtual Worlds*, Vol. 20, pp.417-426, 2009.

[11] B. Solenthaler, M. Gross, "Two-scale Particle Simulation," in *Proceedings of the ACM SIGGRAPH*, 2011.

[12] K. Raveendran, C. Wojtan, G. Turk, "Hybrid Smoothed Particle Hydrodynamics," in *Proceedings of Eurographics/SIGGRAPH Symposium on Computer Animation*, 2011.

[13] N. Foster, D. Metaxas, "Realistic Animation of Liquids," *Graphical Models and Image Processing*, Vol. 58, No. 5, pp.471-483, 1996

[14] J. Stam, "Stable Fluids," in *Proceedings of ACM SIGGRAPH*, pp.121-128, 1999.

[15] M. Muller, D. Charypar, M. Gross, "Particle-based Fluid Simulation for Interactive Applications," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer Animation*, pp.154-159, 2003.

[16] S. Premoze, T. Tasdizen, J. Bigler J, "Particle-based Simulation of Fluids," *Computer Graphics Forum*, Vol. 22, pp. 401-410, 2003.

[17] R. Bridson. Fluid simulation for computer graphics. AK Peters/CRC Press, 2008.

[18] R. Fedkiw, J. Stam, H.W. Jensen, "Visual Simulation of Smoke," in *Proceedings of Eurographics/SIGGRAPH Symposium on Computer Animation 2001*, pp. 154-159, 2001.

[19] J. Molemaker, J.M. Cohen, S. Patel S, "Low Viscosity Flow Simulations for Animation," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 9-18, 2008.

[20] Y. Zhu, R. Bridson, "Animating Sand as Fluid," in *Proceedings of the ACM SIGGRAPH*, pp. 965-972, 2005.

[21] D. Kim, S.O. Young, H.S. Ko, "A Semi-Lagrangian Cip Fluid Solver without Dimensional Splitting," in *Proceedings of the 2008 Eurographics*, pp. 467-475, 2008.

[22] T. Kim, N. Thurey, D. James, "Wavelet Turbulence for Fluid Simulation," in *Proceedings of the 2008 ACM SIGGRAPH*, pp. 1-6, 2008.

[23] H. Schechter, R. Bridson, "Evolving Sub-grid Turbulence for Smoke Animation," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 1-8, 2008.

[24] R. Narain, J. Sewall, M. Carlson M, "Fast Animation of Turbulence Using Energy Transport and Procedural Synthesis," in *Proceedings of SIGGRAPH Asia*, pp. 1-8, 2008.

[25] M.N. Gamito, P.F. Lopes, M.R. Gomes, "Two-dimensional Simulation of Gaseous Phenomena Using Vortex Particles," in *Proceedings of the 6th Eurographics Workshop on Computer Animation and Simulation*, Vol. 28, pp. 315-319, 1995.

[26] T. Praff, N. Thuerey, J. Cohen, "Scalable Fluid Simulation Using Anisotropic Turbulence Particles," *ACM Transactions on Graphics*, Vol. 29(6), 2010.

[27] B. Zhu, X. Yang, Y. Fan, "Creating and Preserving Vortical Details in SPH Fluid," *Computer Graphics Forum*, Vol. 21, pp. 2207-2214, 2010.

[28] B. Solenthaler, R. Pajarola, "Predictive-corrective Incompressible SPH," *ACM Transactions on Graphics*, Vol. 28, No. 3, pp. 40-49, 2009.

[29] J. Morris, P. Fox, Y. Zhu, "Modeling Low Reynolds Number Incompressible Flows Using SPH," *Computational Physics*, Vol. 136, pp. 214-226, 1997.

[30] M. Muller, S. Schirm, M. Teschner, "Interaction of Fluids with Deformable Solids," *Computer Animation and Virtual Worlds*, Vol.15, No. 3, pp. 159-171, 2004.

[31] T. Harada, S. Koshizuka, Y. Kawaguchi, "Smoothed Particle Hydrodynamics in Complex Shapes," in *Proceedings of Spring conference on computer graphics*, pp.26-28, 2007.

[32] M.W. Jones, J.A. Baerentzen, M. Sramek, "3D Distance Fields: A Survey of Techniques and Applications," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12, No. 4, pp. 581-599, 2006.

[33] M. Becker, M. Teschner, "Weakly Compressible SPH for Free Surface Flows," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 209-217, 2007.

[34] X. Hu, N. Adams, "An Incompressible Multi-phase SPH Method," *Computational Physics*, Vol. 27, No. 1, pp. 264-278, 2007.

[35] S.B. Pope, "Turbulent Flows," *Cambridge University Press*, 2000.

[36] M.J. Gourlay, "Fluid Simulation for Video Games," http://software.intel.com/en-us/articles/.

[37] J. Yu, G. Turk, "Reconstructing Surfaces of Particle-based Fluids Using Anisotropic Kernels," *ACM Transactions on Graphics*, Vol. 32, Issue 1, No. 5, 2013.

[38] N. Akinci, M. Ihmsen, G. Akinci, "Versatile Rigid-fluid Coupling for Incompressible SPH," *ACM Transactions on Graphics*, Vol. 31, Issue 4, No. 62, 2012.