Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 8971089, 11 pages
https://doi.org/10.1155/2019/8971089

Research Article

Incompressible Fluids Simulation by Relaxing the Density-Invariant Condition in a Marine Simulator

Xingfeng Duan[1,2] **Hongxiang Ren**[1] **and Haijiang Li**[1]

[1]*Navigation College, Dalian Maritime University, Dalian 116026, China*
[2]*Navigation College, Jimei University, Xiamen 361021, China*

Correspondence should be addressed to Hongxiang Ren; dmu_rhx@163.com

Received 17 October 2018; Revised 21 December 2018; Accepted 13 January 2019; Published 3 February 2019

Academic Editor: Giuseppina Colicchio

To achieve small-scale ocean scene simulation in a marine simulator, we present an incompressible SPH algorithm by relaxing the density-invariant condition for incompressible fluids. As there are larger density errors of the fluid particles near or on the boundary, more iteration numbers are required. Taking boundary handling to modify the density of the fluid particles, the relaxation method is used to optimize the iterative method to solve the density-invariant condition, which can reduce the iteration numbers and average density deviation and improve the accuracy. Our proposed approach can achieve incompressible SPH and solve the problem of the particle deficiency. While ensuring the stability, the approach can allow large time steps and control the density deviation below 0.01% and improve the efficiency by reducing the iteration numbers and optimizing the calculating procedure and the initial value selection.

1. Introduction

Ocean wave simulation is an important part of fluid simulation. The realistic and real-time characteristics and interactivity of the simulation directly affect the degree of realism of the sea scene in a marine simulator. At present, physically-based simulation methods are used to enhance environmental realism by means of the methods of computational fluid dynamics (CFD). The methods based on the Navier-Stokes equation can be divided into two categories: the Euler method, which is a grid-based method, and the Lagrange method, which is a particle-based method. Compared with the Euler method, the Lagrange method can effectively avoid the problem of numerical dissipation because there is no direct convection term, and it can suitably address large deformation and free surface problems. In the Lagrange method, the smoothed particle hydrodynamics (SPH) method is more realistic, stable, and efficient because it is easy to deal with the simulation of surface flow and the fluid-structure coupling of large deformation. At present, this method has become one of the most promising simulation methods. Desbrun et al. [1] first introduced the standard SPH (SSPH) to computer graphics (CG) for simulating deformable bodies. Since then, Müller et al. [2] have applied SSPH to simulate water with a free surface.

Currently, there are two main methods to simulate incompressible fluid by SPH: one is to regard incompressible fluid as a slightly compressible fluid and introduce a state equation to simulate weak compressible fluid, which is called weakly compressible SPH (WCSPH). The other way is to use a semi-implicit method to enforce incompressibility, called incompressible SPH (ISPH). The WCSPH method was proposed by Monaphan et al. [3] and Becker al. [4], which simulated the free surface flow. The equation of state was employed to obtain the fluid pressure, which is an explicit method, but smaller density fluctuations can lead to a sudden increase in pressure. It is often necessary to introduce a larger sound speed to limit the density fluctuation rate to less than 1%, which requires a smaller time step and affects the efficiency of the algorithm. A slightly larger time step causes instability.

The ISPH method is a better way to simulate incompressible fluids using the projection method [5–7], called the semi-implicit method. The fluid pressure is obtained by solving the

pressure Poisson equation (PPE). This method can allow a larger time step and strictly enforce incompressibility, but it needs to solve large linear equations. The coefficient matrix is very large, resulting in high computational cost. The methods for solving the PPE are usually grid-based methods and meshless particle methods. Grid-based methods [7–10] are traditional, usually taking a conjugate gradient, a successive over-relaxation (SOR), a multiple grid, or a Voronoi diagram method to solve the PPE on the background grid.

The meshless particle method is mainly used to solve the PPE by iteration methods such as the predictor-corrector method and Jacobi-style method to realize a density-invariant condition to simulate incompressible fluid. Solenthaler and Pajarola [11] proposed the PCISPH method, which uses the pressure field of the current time step to obtain the particle position of the next time step and then uses the density estimation to correct the current pressure field. The density error of the next time step is less than the given threshold to achieve the density-invariant condition by iterative method. The implicit incompressible SPH (IISPH) method proposed by Ihmsen et al. [12] is also strictly a semi-implicit incompressible SPH method. The relaxed Jacobi method is used to solve the PPE. The divergence-free SPH (DFSPH) method proposed by Bender et al. [13] is also a predictor-corrector method for solving the PPE. However, in addition to the density-invariant condition, the velocity divergence-free condition is used as the source term of the PPE, and two pressure solvers are combined to enforce incompressibility. Subsequently, Bender et al. [14] extend the method and applies it to the simulation of viscous flow.

There are two ISPH versions: CFD version and CG version. The two ISPH versions have obviously different features and scope of applications. In CFD version, the approach is developed for solving complex hydrodynamic problems such as modeling complex free surface flows, multiphase flows, and fluid-structure interactions and have widely applied in a wide range of hydrodynamic problems (e.g., water-body interactions, water impacts, water entries and exits, and flow fast bodies) and explosion problems (e.g., underwater explosions, contact explosions, etc.) [15]. In CG version, the approach is widely applied in computer graphics (e.g., the simulation of navigation, salvage, flying, virtual naval battlefield, game, and film), which is faster and easier for modeling the incompressible fluid. Different from CFD version, they simplified the physical model to save the computational cost. Their goal is to achieve a balance between realism and real-time. Raveendran et al. [16] proposed that solving the PPE on a coarse grid enforces a divergence-free velocity field and generates pressure forces that are then passed along to the WCSPH or predictive-corrective incompressible SPH (PCISPH) solver to achieve incompressible flow. This approach can be seen as the combination of Euler and SPH approaches. Losasso et al. [17] combined the particle level set (PLS) and SPH method, which uses the PLS method to model dense liquid volumes and the SPH method to simulate small-scale phenomena, such as spay and foam. The SPH solver also needs to solve the PPE on a background grid, not only to maintain the divergence-free flow field but also to maintain a predefined target density.

In ocean engineering, the breaking wave with spay and foam happened in the interaction of the high speed ship and wave. Marrone and Colagrossi [18] employed a 2D+t SPH approach to study the breaking wave pattern of a fast ship for different forward ship speeds. The vortical structures due to plunging of the bow wave are tracked using passive markers which are initially positioned on the undisturbed free surface. Sun and Colagrossi [19] presented algorithms for the detection of Lagrangian Coherent Structures in viscous flows through the Finite-Time Lyapunov Exponents (FTLEs). FTLEs introduced to Hydrodynamics are used for capturing the bow breaking wave feature. In order to make it easier, they adopt a simplified ship hull. In computer graphics, and Losasso et. al. [17] extended the SPH solver to simulate diffuse phenomena such as mixtures of spray and air, but foam is not based on physical model and a large number of spay particles can only be rendered offline. Ihmsen et al. [20] present a postprocessing model to simulate spay, foam, and air bubbles. This method enables efficient processing and versatile handling, but they ignored the effect of diffuse material on the fluid simulation and rendered the scene with mental ray 3.9, based on the ray casting and cannot meet the requirements of real-time rendering.

Macklin and Müller [21] presented a position based fluids (PBF) method which integrates an iterative density solver into the Position Based Dynamics framework (PBD) and applies to fluid simulation. Similar to the PCISPH method, the PBF method employs the predictor-corrector method to constrain the fluid position so as to enforce incompressibility. The position change is iteratively computed for each particle rather than the pressure so that PBF method allows larger time step, achieves better stability, and is suitable for real-time rendering. Because the particle position updating is based on geometry, PBF method is not physical simulation in its strict sense. In particular, collision detection and response of PBD are adopted for boundary handling and there is a larger density error at the interface between fluid particles and the ground or wall, which requires more iteration numbers, thus affecting the efficiency of the solver. In this work, the boundary particle method is introduced into the PBF method to correct the density error of fluid particles near or on the boundary. For density-invariant conditions, the relaxation method, optimal initial value, and iterative termination condition are adopted to accelerate convergence, reduce the number of iterations, improve the efficiency of predictor-corrector iteration, and increase the time step for improving the overall performance.

2. Position-Based Fluids Method

The incompressibility of fluid requires that the fluid particle densities remain constant. The idea of the PBF method is to modify the particle position when the density error of fluid particles exists, and the particle position is iteratively corrected so that the predicted density error is lower than a predefined value. The density constraint $C(\mathbf{x}_i)$ of a fluid particle i is defined as

$$C(\mathbf{x}_i) = \frac{\rho_i}{\rho_0} - 1, \tag{1}$$

where $\mathbf{x}_i$, ρ_i, and ρ_0 denote the position, density, and rest density of a fluid particle i, respectively.

The fluid particle density ρ_i, can be solved by density summation formula in SPH method as follows:

$$\rho_i = \sum_j m_j W\left(\mathbf{x}_i - \mathbf{x}_j, h\right) = \sum_j m_j W_{ij}, \tag{2}$$

where m_j and $\mathbf{x}_j$ denote the mass and position of a neighbor particle j, respectively, $\mathrm{W}(\mathbf{x}_i\text{-}\mathbf{x}_j, \mathrm{h})$ is a kernel function with support radius h, and $\mathrm{W}(\mathbf{x}_i\text{-}\mathbf{x}_j, \mathrm{h}) = \mathrm{W}_{ij}$.

In order to enforce incompressibility, the fluid density must be kept constant $\rho_i\text{-}\rho_0 = 0$ which is realized by correcting the fluid position. The position changes $\Delta\mathbf{x}$ should satisfy the following constraint:

$$C(\mathbf{x} + \Delta\mathbf{x}) = 0. \tag{3}$$

From (3) using the first-order Taylor expansion, we can get the following:

$$C(\mathbf{x} + \Delta\mathbf{x}) \approx C(\mathbf{x}) + \partial_{\mathbf{x}} C(\mathbf{x})\, \Delta\mathbf{x} = 0. \tag{4}$$

Considering that the constraints $\mathrm{C}(\mathbf{x})$ do not affect the motion state, the direction of partial derivative $\partial_{\mathbf{x}}\mathrm{C}(\mathbf{x})$ of $\mathrm{C}(\mathbf{x})$ with respect to $\mathbf{x}$ is perpendicular to the original direction of motion. The direction of $\Delta\mathbf{x}$ is the same as the direction of $\partial_{\mathbf{x}}\mathrm{C}(\mathbf{x})$. Importing a Lagrange multiplier λ, $\Delta\mathbf{x}$ is expressed as follows:

$$\Delta\mathbf{x} = \lambda \partial_{\mathbf{x}} C(\mathbf{x}). \tag{5}$$

Insert (5) into (4), and we obtain

$$\lambda = -\frac{C(\mathbf{x})}{\left|\partial_{\mathbf{x}} C(\mathbf{x})\right|^2}. \tag{6}$$

So (5) changes into

$$\Delta\mathbf{x} \approx -\frac{C(\mathbf{x})}{\left|\partial_{\mathbf{x}} C(\mathbf{x})\right|^2} \partial_{\mathbf{x}} C(\mathbf{x}). \tag{7}$$

The position change $\Delta\mathbf{x}_i$ and factor λ_i of a fluid particle i are computed as

$$\Delta\mathbf{x}_i \approx -\frac{C_i(\mathbf{x}_i)}{\left|\partial_{\mathbf{x}_i} C_i(\mathbf{x}_i)\right|^2} \partial_{\mathbf{x}_i} C_i(\mathbf{x}_i), \tag{8}$$

$$\lambda_i = -\frac{C_i(\mathbf{x}_1, \cdots, \mathbf{x}_n)}{\sum_k \left|\partial_{\mathbf{x}_k} C_i(\mathbf{x}_1, \cdots, \mathbf{x}_n)\right|^2}. \tag{9}$$

Derivative $\partial_{\mathbf{x}_k} C_i(\mathbf{x}_1, \cdots, \mathbf{x}_n)$ is given by

$$\partial_{\mathbf{x}_k} C_i(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \frac{1}{\rho_0} \begin{cases} \sum_j m_j \nabla W_{ij} & k = i \\ -m_j \nabla W_{ij} & k = j. \end{cases} \tag{10}$$

The denominator in (9) causes instability when the neighbors of the support domain are insufficient so we use a parameter ξ to avoid singularity and set ξ is 1×10^{-6}. The factor λ_i is now

$$\lambda_i = -\frac{C_i(\mathbf{x}_1, \cdots, \mathbf{x}_n)}{\sum_k \left|\partial_{\mathbf{x}_k} C_i(\mathbf{x}_1, \cdots, \mathbf{x}_n)\right|^2 + \xi}. \tag{11}$$

```
(1) while animating do
(2)   for all particles i do
(3)     apply forces v_i = v_i + Δt f_ext(x_i)
(4)     predict position x_i^* = x_i + Δt v_i
(5)   for all particles i do
(6)     find neighbor particles N_i(x_i^*)
(7)   while iter < solverIterations do
(8)     for all particles i do
(9)       calculate λ_i
(10)    for all particles i do
(11)      calculate Δx_i
(12)      perform collision detection and response
(13)    for all particles i do
(14)      update position x_i^* = x_i^* + Δx_i
(15)  for all particles i do
(16)    update velocity v_i = (x_i^* − x_i)/Δt
(17)    apply vorticity confinement and viscosity
(18)    update position x_i = x_i^*
```

Algorithm 1: Position based fluids algorithm.

To speed up the updating of the particle position changes, the position change $\Delta\mathbf{x}_i$ is updated by the factor λ_j of the neighbors in the support domain of particle i as

$$\Delta\mathbf{x}_i = \frac{1}{\rho_0} \sum_j m_j \left(\lambda_i + \lambda_j\right) \nabla W_{ij}. \tag{12}$$

The whole algorithm is shown in Algorithm 1.

3. Incompressible SPH Method

3.1. Relaxation Method. With the increase of the accuracy requirement of fluid incompressibility and the number of fluid particles, the computational cost time for solving the PPE is dramatically increased. To solve the PPE quickly and accurately and save the computational cost time, we adopt relaxation method to accelerate convergence. So the predicted position is adjusted by

$$\mathbf{x}_i^* = \left(\mathbf{x}_i^* + \Delta\mathbf{x}_i\right) + \alpha\Delta\mathbf{x}_i = \mathbf{x}_i^* + (1 + \alpha)\, \Delta\mathbf{x}_i, \tag{13}$$

where α is the relaxation factor. If α is chosen properly, the convergence can be greatly accelerated. We use different relaxation factors to test the maximum number of iterations when the maximum volume compression is set to in 0.1% or 0.01%, respectively. The maximum number of iterations in different relaxation factors is shown in Table 1.

When α is set to 0, our method is equivalent to the PBF method and the maximum iteration number is 47. When α is greater than 0.5, although the maximum number of iterations is still decreasing, the simulation tends to be unstable. When $\alpha \in [0, 0.5]$, the maximum number of iterations decreases with the increase of α. Therefore the relaxation factor α is set to 0.5.

3.2. Boundary Handling. Recently, the boundary handling methods of SPH fluid simulation mainly include penalty

Table 1: Comparison of the maximum iteration number under different relaxation factors.

relaxation factor	max. iter. number	volume compression	max. iter. number	volume compression
0.0	47	0.01%	17	0.1%
0.1	40	0.01%	14	0.1%
0.2	34	0.01%	12	0.1%
0.3	27	0.01%	11	0.1%
0.4	20	0.01%	10	0.1%
0.5	17	0.01%	10	0.1%
0.6	16	0.01%	9	0.1%
0.7	16	0.01%	9	0.1%
0.8	14	0.01%	8	0.1%
0.9	13	0.01%	7	0.1%

method, ghost particle method, and boundary particle method. In penalty method, distance-based penalty forces, such as Lennard-Jones forces [3] and normal repulsion forces [22], are exerted for preventing the fluid particles at the fluid-solid interface from penetrating the solid boundary. But a large penalty force is needed, which results in large pressure fluctuation of the fluid particles on the boundary and momentum conservation is not guaranteed. Becker et al. [23] employed direct force to correct the position and velocity of the fluid particles for two-way fluid-rigid coupling and nonpenetration. Different slip conditions are realized by including a nonsymmetric friction force. But penalty force and direct force method have the problem of the fluid particles adhering to the boundary, which is caused by the insufficient number of the fluid particles at the fluid-solid interface.

In the ghost particle method [24, 25], ghost particles are dynamically generated, which is symmetrical to the fluid particle outside the boundary. A ghost particle gets the same density, pressure, and velocity at opposite direction of its corresponding fluid particle, which participates in the density estimation. This method was successfully employed to simulate the different slip conditions and ensure conservation of momentum, but it is difficult to deal with the complex boundary problem. Aiming at the defect of standard ghost particle method in modeling complex geometries, Marrone et al. [18] develop the fixed ghost particle method. Differently from the standard ghost particle method where at each time step any particle nearby the solid boundary is mirrored into a ghost with respect to that boundary; the method fixed the mirror particles to the outside of the fluid at the boundary. The interpolation of these particles is carried out by using the method of higher order difference to compute the physical quantities of these particles. Finally, the corresponding fixed ghost particles are assigned by the mirror rule. This method solves the problem of particle defects and overlap caused by standard ghost particle method and is suitable for dealing with complex geometries and gives quite good and accurate results. Subsequently, Sun et al. [26] introduced it into the numerical simulation of the self-propulsive motion of a fishlike swimming foil. However, this method is complicated in computing the physical quantities attributed to each ghost particle; an interpolation point is associated with it. This interpolation point is obtained by mirroring the position of the fixed ghost particle into fluid domain, which results in high computational cost and requires high accuracy for interpolation methods.

In boundary particle method [12, 27, 28], layer, two-layer, or multi-layer boundary particles are sampled at the solid boundary. The number of boundary particle layer is determined by the support domain. The boundary particles have the same rest density of the fluid particles. The velocity of the boundary particles is zero for the static boundary, while the velocity is the same as the velocity of the dynamic boundary. The density and pressure of the boundary particles also participate in the interpolation of the density, pressure, and internal force of the fluid particles.

Therefore, we employ the boundary particle method to solve the problem of the particle deficiency, sample the solid triangles by Poisson disk sampling method [25], and generate single layer boundary particles which participate in the density estimation of the fluid particles. In order to reduce the error of density estimation caused by unreasonable mass or nonuniform distribution of boundary particles in single layer boundary. The mass of boundary particles is adaptively changed according to the distribution of boundary particles. Assuming that the density of a boundary particle b is equal to the density of a fluid particle $\rho_b = \rho_0$, the mass $\Psi_b(\rho_0)$ of a boundary particle b is computed as

$$\Psi_b(\rho_0) = \rho_0 V_b, \tag{14}$$

where V_b denotes the volume of a boundary particle. Its value is $V_\mathrm{b} = 1/\sigma_\mathrm{b}$, and the number density of a boundary particle $\sigma_\mathrm{b} = \sum_{\mathrm{k}\in\mathrm{rigid}} \mathrm{W}_{\mathrm{bk}}$. After the correction of the continuity equation, the predicted density $\rho_i(t + \Delta t)$ of a fluid particle is computed as

$$\begin{aligned}\rho_i(t+\Delta t) = &\sum_{j\in fluid} m_j \nabla W_{ij}(t) \\ &+ \sum_{b\in rigid} \Psi_b(\rho_0)\nabla W_{ib}(t) \\ &+ \Delta t \sum_{j\in fulid} m_j\left(\mathbf{v}_i - \mathbf{v}_j\right)\nabla W_{ij}(t) \\ &+ \Delta t \sum_{b\in rigid} \Psi_b(\rho_0)\left(\mathbf{v}_i - \mathbf{v}_b\right)\nabla W_{ib}(t).\end{aligned} \tag{15}$$

In this way, the density fluctuation caused by under sampling or over sampling can be corrected to improve stability. Similarly, after the correction of the momentum equation, the position change of a fluid particle is computed as

$$\Delta \mathbf{x}_i = \frac{1}{\rho_0}\left(\sum_{j\in fluid} m_j\left(\lambda_i + \lambda_j\right)\nabla W_{ij}(t) + \sum_{b\in rigid}\Psi_b\left(\rho_0\right)\left(\lambda_i\right)\nabla W_{ib}(t)\right). \tag{16}$$

The whole algorithm of incompressible fluid simulation based on relaxing density-invariant condition is illustrated in Algorithm 2. We modify the PBF algorithm as follows: first of all, the convergence speed is accelerated by relaxation method and the relaxation factor is set to 0.5 (line (16) of Algorithm 2). Then, collision detection and response are replaced with boundary handling (line (14) of Algorithm 2). In (2), the density equation may lead to density underestimation and cause particle clustering near a free surface where kernel truncation occurs. We solved this issue simply by clamping density to rest density of a fluid particle. Considering the effect of relative velocity on density computing, (15) and (16) are used to replace (2) and (12) for computing the density and position change. Finally, if *solverIterations* is set too small (line (7) of Algorithm 1), the average density error in simulation may not reach the required target but the iteration terminates, which makes the density estimation inaccurate. The limit of the maximum number of iterations is replaced with the average density error less than the predefined threshold η as the termination condition of iteration (line (7) of Algorithm 2), so that the average density error can satisfy the requirements of accuracy.

```
(1) while animating do
(2)   for all particles i do
(3)     apply forces v_i = v_i + Δt f_ext(x_i)
(4)     predict position x_i^* = x_i + Δt v_i
(5)   for all particles i do
(6)     find neighbor particles N_i(x_i^*)
(7)   while (ρ_avg − ρ_0 > η) ∪ (iter < 2) do
(8)     for all particles i do
(9)       calculate ρ_i
(10)      calculate C_i
(11)      calculate λ_i
(12)    for all particles i do
(13)      calculate Δx_i
(14)      perform boundary handling
(15)    for all particles i do
(16)      update position x_i^* = x_i^* + (1 + α)Δx_i
(17)  for all particles i do
(18)    update velocity v_i = (x_i^* − x_i)/Δt
(19)    apply vorticity confinement and viscosity
(20)    update position x_i = x_i^*
```

Algorithm 2: Incompressible SPH algorithm based on relaxing the density-invariant condition.

4. Foam

The effect of broken waves, such as foam and spray, directly affects the visual realism of the whole scenario. The key to simulate foam and spray is the determination of foam generation area, the advection, and dissolution of foam and spray. The potential value of each fluid particle calculated by geometric characteristics, kinetic energy, and relative velocity is compared with a given threshold to determine the area of foam generation. The detailed small-scale effects are achieved by a postprocessing technology and modeled with unified particles which are called diffuse particles. We classify the particle into foam and spay. In our experiments, the diffuse particles with less than 8 fluid neighbors are considered as spray particle and, in all other cases, the particles are considered to be foam. Physically motivated rules are employed to generate and advect and dissipate spay and foam, but interparticle forces and the influence of spay and foam particles onto the fluid particles are neglected. The rendering of the foam and spay is employed by the screen space foam rendering [29].

4.1. Generation. The probability of a fluid particle generating diffuse particles is expressed by the control function $\phi(I, \tau_{\min}, \tau_{\max})$ which is defined as

$$\phi\left(I, \tau_{\min}, \tau_{\max}\right) = \frac{\min\left(I, \tau_{\max}\right) - \min\left(I, \tau_{\min}\right)}{\tau_{\max} - \tau_{\min}}, \tag{17}$$

where $\tau_{\min}$, $\tau_{\max}$ is the minimum and maximum threshold and I is the potential for the generation of diffuse material.

Geometric Characteristics. The wave crest is unstable and easy to produce diffused particles. At the wave crest, the surface curvature is relatively high and the local surface is convex. The surface curvature k can be approximated with

$$k_i = \sum_i k_{ij} = \sum_j \left(1 - \hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j\right) W\left(\mathbf{x}_{ij}, h\right), \tag{18}$$

where $\hat{\mathbf{n}}_i$ is a normalized surface normal. In order to distinguish convex from concave regions, considering the vector product between $\hat{\mathbf{n}}_i$ and $\hat{\mathbf{x}}_{ji}$, wave crests are identified using

$$\hat{k}_i = \sum_i \hat{k}_{ij} \tag{19}$$

with

$$\hat{k}_{ij} = \begin{cases} 0 & \hat{\mathbf{x}}_{ji} \cdot \hat{\mathbf{n}}_i \geq 0 \\ k_{ij} & \hat{\mathbf{x}}_{ji} \cdot \hat{\mathbf{n}}_i < 0. \end{cases} \tag{20}$$

The condition of the identification of wave crests is adjusted as

$$\delta_i = \begin{cases} 0 & \hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}_j < 0.6 \\ 1 & \hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}_j \geq 0.6. \end{cases} \tag{21}$$

Finally, the likelihood of a particle to be at the crest of a wave is computed as

$$I_{wc} = \phi\left(\hat{k}_i \cdot \delta_i, \tau_{wc}^{\min}, \tau_{wc}^{\max}\right) \tag{22}$$

with user-defined $\tau_{wc}^{\min}$ and $\tau_{wc}^{\max}$.

Kinetic Energy. The kinetic energy $E_{k,i} = 0.5 m_i \mathbf{v}_i^2$ is used as a measurement for the generation of diffuse particles. The higher the velocity of a fluid particle, the greater the energy and the easier the diffusive particles will be generated at the position. The potential to generate diffuse particles due to kinetic energy is computed as

$$I_k = \phi\left(E_{k,i}, \tau_k^{\min}, \tau_k^{\max}\right) \tag{23}$$

with user-defined $\tau_k^{\min}$ and $\tau_k^{\max}$.

Relative Velocity. The relative velocity is used to determine regions where air is potentially trapped since these are large for impacts and vortices. The larger the relative velocity of fluid particles is, the easier it is to generate diffuse particles. The potential to generate diffuse particles due to relative velocity is computed as

$$I_{ta} = \phi\left(v_i^{diff}, \tau_{ta}^{\min}, \tau_{ta}^{\max}\right) \tag{24}$$

with user-defined $\tau_{ta}^{\min}$ and $\tau_{ta}^{\max}$, where v_i^{diff} is the

$$v_i^{diff} = \sum_j \left\|\mathbf{v}_{ij}\right\| \left(1 - \widehat{\mathbf{v}}_{ij} \cdot \widehat{\mathbf{x}}_{ij}\right) W\left(\mathbf{x}_{ij}, h\right) \tag{25}$$

with the normalized relative velocity $\widehat{\mathbf{v}}_{ij} = (\mathbf{v}_i - \mathbf{v}_j)/\|\mathbf{v}_i - \mathbf{v}_j\|$ and normalized distance vector $\widehat{\mathbf{x}}_{ij} = (\mathbf{x}_i - \mathbf{x}_j)/\|\mathbf{x}_i - \mathbf{x}_j\|$, $W(\mathbf{x}_{ij}, h)$ is a radially symmetric weighting function defined as

$$W\left(\mathbf{x}_{ij}, h\right) = \begin{cases} 1 - \dfrac{\left\|\mathbf{x}_{ij}\right\|}{h} & \left\|\mathbf{x}_{ij}\right\| \le h \\ 0 & otherwise. \end{cases} \tag{26}$$

Therefore, the number of diffuse particles generated by a fluid particle is computed as

$$n_d = I_k \left(k_{ta} I_{ta} + k_{wc} I_{wc}\right) \Delta t. \tag{27}$$

Similar to [20], we sample the diffuse particles in a cylinder which becomes a rectangle in two dimension (see Figure 1). The position and velocity of a diffuse particle d are computed as

$$\mathbf{x}_d = \mathbf{x}_f + \Delta t \cdot r \cos\theta \mathbf{e}' + h\widehat{\mathbf{v}}_f \tag{28}$$

and

$$\mathbf{v}_d = \mathbf{v}_f + r \cos\theta \mathbf{e}', \tag{29}$$

where three uniformly distributed random variables $X_r, X_\theta, X_h \in [0, \cdots, 1]$, the distance to the cylinder axis $r = r_V \sqrt{X_r}$, the azimuth $\theta = 2\pi X_\theta$, the distance $h = X_h \|\Delta t \mathbf{v}_f\|$, and normalized vector $\mathbf{e}'$ is perpendicular to the velocity direction of the fluid particle.

4.2. Advection. Spay particles are affected by gravity $\mathbf{g}$ and external force $\mathbf{F}_{ext}$, and using the Euler method, the velocity and position are updated as

$$\mathbf{v}_{spay}(t + \Delta t) = \mathbf{v}_{spay}(t) + \Delta t \left(\mathbf{g} + \frac{\mathbf{F}_{ext}(t)}{m}\right) \tag{30}$$

$$\mathbf{x}_{spay}(t + \Delta t) = \mathbf{x}_{spay}(t) + \Delta t \mathbf{v}_{spay}(t + \Delta t) \tag{31}$$

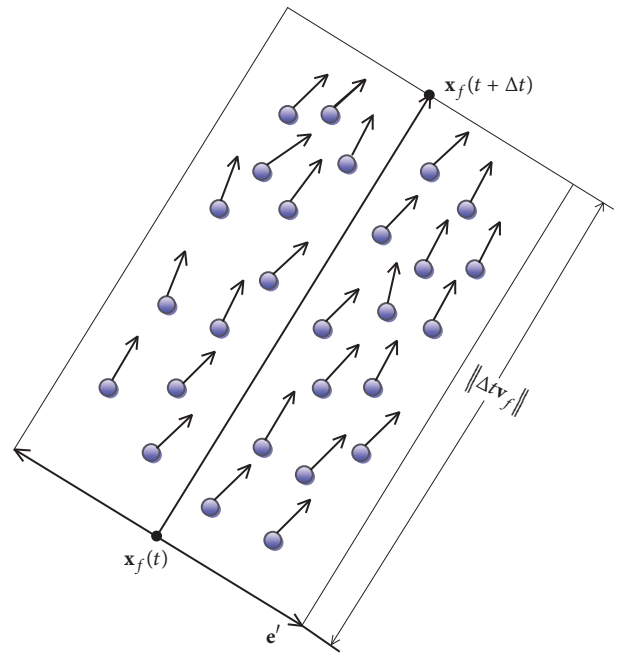


Figure 1: The 2D model of diffuse particles generated by a fluid particle f. $\mathbf{x}_f(t)$ is the current position, $\mathbf{x}_f(t + \Delta t)$ is the position, and $\mathbf{v}_f$ is the velocity of a fluid particle.

But foam particles are mainly influenced by the fluid particles of the supporting domain. The velocity of the foam particle d is computed by the average velocity of the fluid particles of the supporting domain which is computed as

$$\widehat{\mathbf{v}}_f(t + \Delta t) = \frac{\sum_f \mathbf{v}_f(t + \Delta t) W_{df}}{\sum_f W_{df}}, \tag{32}$$

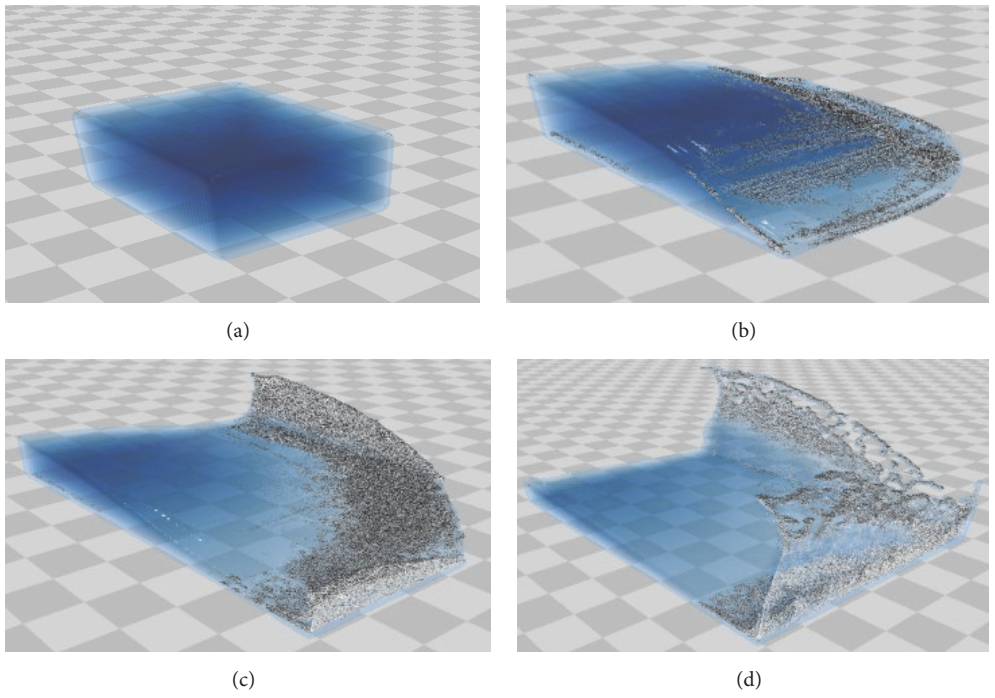
where $\mathbf{v}_f(t + \Delta t) = (\mathbf{x}_f(t + \Delta t) - \mathbf{x}_f(t))/\Delta t$. Then the position of the foam particles is updated as

$$\mathbf{x}_{foam}(t + \Delta t) = \mathbf{x}_{foam}(t) + \Delta t \widehat{\mathbf{v}}_f\left(\mathbf{x}_d, t + \Delta t\right). \tag{33}$$

4.3. Dissipation. Similar to the particle system, the diffuse particles' lifetime is initialized with a predetermined value. In each simulation step, the time step is subtracted from the lifetime of foam particles, whereas the lifetime of spray particles is not reduced. Foam particles are finally deleted when the lifetime is smaller or equal to zero.

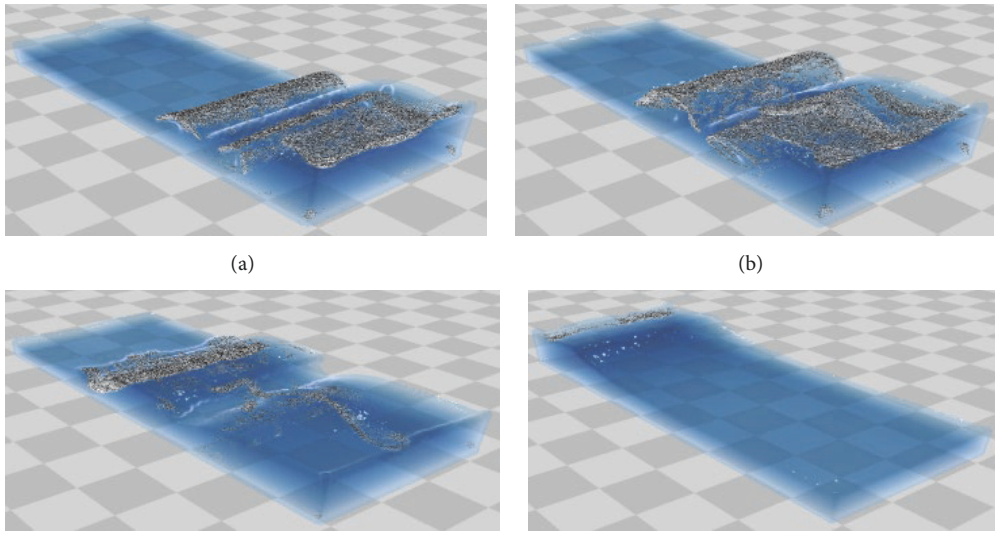
5. Results

To verify the validity of the algorithm proposed in this paper, we performed simulations of several small-scale ocean scenes, such as breaking dam, wave heaving, rolling, and breaking using 262k fluid particles on two Intel (R) Core (TM) i5-6400 processors with 2.2 GHz CPU, 8 g memory, and NVIDIA GeForce GTX960 graphics. The visual effect is shown in Figures 2 and 3. The boundary handling was performed using our proposed method, the neighborhood search algorithm using the space hash function [30], smoothing kernels function with the cubic B-spline function [31],



(a) (b) (c) (d)

Figure 2: The simulation results of two-way dam breaking with foam: (a) the initial state of two-way dam breaking; (b) the water flowed in two directions from the corner; (c) the water lifted over the border; (d) the breaking water with spay and foam.



(a) (b) (c) (d)

Figure 3: The simulation of wave heaving, rolling and breaking: (a) the rolling wave with foam generated in the wave rests; (b) the steeper height of wave; (c) the heaving wave with foam; (d) the calm wave and dissipating foam.

viscous force with the method proposed by Schechter and Bridson [25], and vorticity confinement with the method of Macklin and Müller [21]. To render the fluid surface, we employed the screen space fluid rendering method [32].

6. Discussion

6.1. Convergence. To compare the convergence and overall performance of the proposed approach and PBF method, we simulated a breaking dam scenario with 4500 fluid particles and 18630 boundary particles. The particle radius r is 0.025 m and the average density change rate η set to 0.01% or 0.1%, and the fixed time step Δt is 0.005 s. The average iteration numbers of each physical time step can be found in Figure 4.

When the maximum density error is set to 0.1%, the average iteration of PBF method is 10.6 times and our approach is 6.2 times, and when the maximum density error is set to 0.01%, the average iteration number of PBF method is 29.0

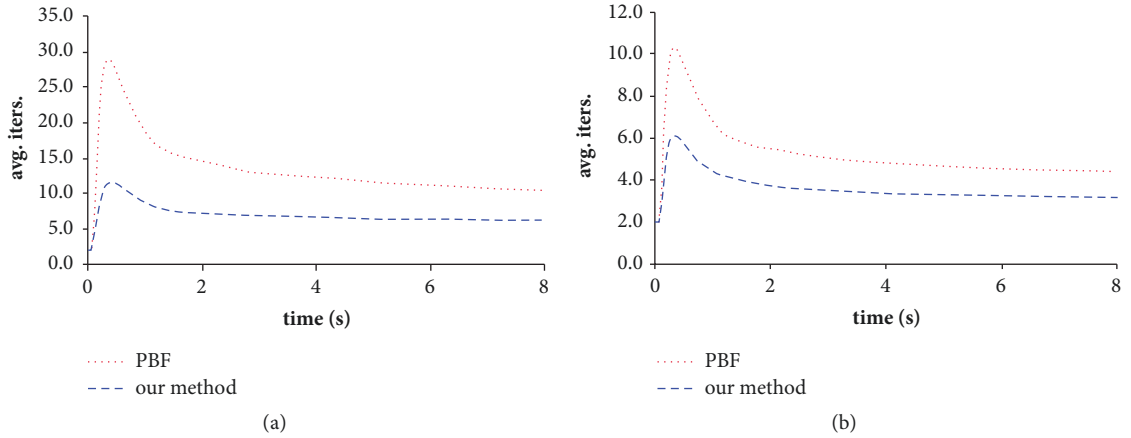


Figure 4: Average number of iterations comparison of our method with the PBF method: (a) the maximum density error with a threshold of 0.01%; (b) the maximum density error with a threshold of 0.1%.

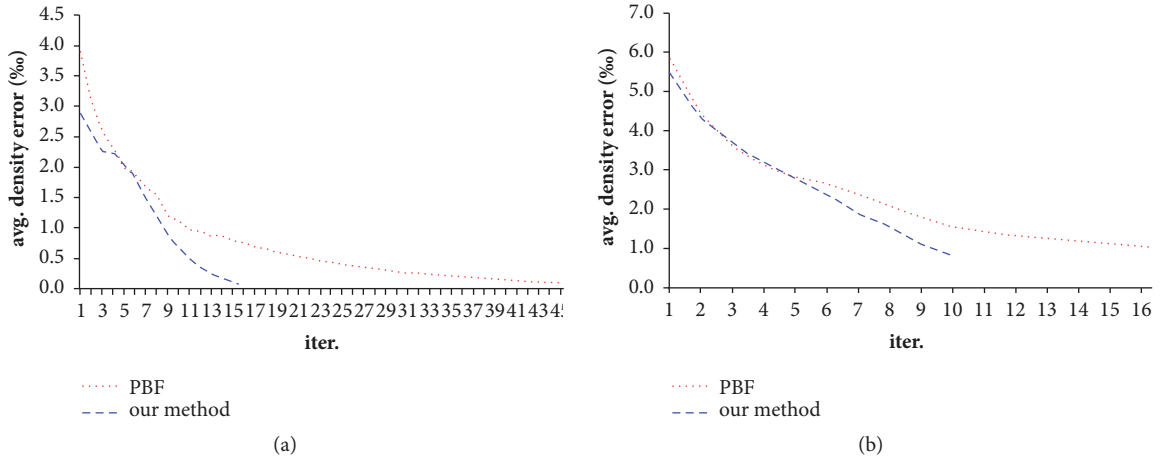


Figure 5: Comparison of average iteration numbers of our method with the PBF method in time t = 0.2 s: (a) the maximum density error with a threshold of 0.01%; (b) the maximum density error with a threshold of 0.1%.

times and our approach is 11.7 times, which greatly improved convergence. Solenthaler and Pajarola [11] indicated that the peak of average iteration number appears when fluid particles collide with the ground and wall. The reason for this is that there exist boundary deficiencies between the fluid particles and the interface of the ground and wall, and thus a large density error is predicted. In our proposed approach, a layer of boundary particles is placed in the boundary position and participates in the density calculation to correct the boundary deficiency, and the peak of the average iteration number appears in the position of larger surface curvature changes.

A convergence comparison of one physical time step at the time of 0.2 s is shown in Figure 5.

When the maximum density error is set to 0.01%, the maximum number of iterations for the PBF method is 47 and the maximum average density change rate is 3.91%, while the maximum number of iterations for our approach is 16, and the maximum average density change rate is 2.90%; when the maximum density error is set to 0.1%, the maximum number of iterations for the PBF method is 17 and the maximum average density change rate is 5.83%, while the maximum number of iterations for our approach is 10, and the maximum average density change rate is 5.49%. Solenthaler and Pajarola [11] showed that the maximum density change rate can be controlled within 1% and, at the time of 0.2 s, the maximum average density change rate is 6.2%. On one hand, in our approach, a boundary particle is adopted to correct the fluid particle density near or on the boundary, which greatly reduces the predicted maximum density error; on the other hand, it uses the relaxation method and optimizes the selection of the initial value to significantly decrease the maximum iteration number and improve convergence of iteration.

6.2. Performance. For predictive-corrective iteration, as long as the iteration number is sufficient, the precision of the result can be guaranteed; however, the convergence speed is sometimes slow, which causes a heavy calculation burden. Its convergence also depends on the selection of the initial value. We adopt the relaxation method and optimize the selection of

Table 2: Performance comparison of PBF and our approach using different time step for a breaking dam, when the maximum density error was set to 0.1%.

	PBF			our approach			PBF / our approach		
		time			time			speedup	
Δt	avg. iter.	solver	total	avg. iter.	solver	total	avg. iter.	solver	total
0.001	2.0	35.4	68.4	2.0	32.1	62.5	1.0	1.1	1.1
0.002	2.6	50.2	80.9	2.1	37.3	65.2	1.2	1.3	1.2
0.003	3.8	75.2	106.4	2.7	48.2	74.6	1.4	1.6	1.4
0.004	4.9	95.6	122.4	3.3	59.6	85.6	1.5	1.6	1.4
0.005	5.8	113.4	140.5	4.0	71.3	95.7	1.5	1.6	1.5
0.006	6.8	131.4	158.7	4.4	78.4	104.1	1.5	1.7	1.5
0.007	7.5	151.2	174.2	4.7	81.5	109.9	1.6	1.9	1.6

Table 3: Performance comparison of PBF and our approach using different time step for a breaking dam, when the maximum density error was set to 0.01%.

	PBF			our approach			PBF / our approach		
		time			time			speedup	
Δt	avg. iter.	solver	total	avg. iter.	solver	total	avg. iter.	solver	total
0.001	5.8	111.6	147.0	3.8	66.7	98.3	1.5	1.7	1.5
0.002	11.7	237.4	271.3	7.0	130.5	158.5	1.7	1.8	1.7
0.003	13.8	275.9	316.5	7.5	137.5	167.5	1.8	2.0	1.9
0.004	14.5	289.2	313.6	7.5	133.0	158.3	1.9	2.2	2.0
0.005	15.6	300.9	331.0	7.5	130.0	156.4	2.1	2.3	2.1
0.006	16.9	326.2	351.7	7.7	131.8	158.8	2.2	2.5	2.2
0.007	17.8	345.6	371.9	7.8	131.1	159.7	2.3	2.6	2.3

the initial value with 0.5 times the value from the last time step to reduce iteration numbers and accelerate the convergence speed. We used different fixed time step to control the density change rate within the range of 0.1% or 0.01%, and the tested pressure solver and the overall time were used to illustrate the time step and convergence effect on the overall performance. The measurements are shown in Tables 2 and 3.

As shown in Tables 2 and 3, under the condition of a larger time step, our proposed approach exhibits better performance than the PBF method. As shown in Table 2, by adopting our approach, the maximum density error is controlled within 0.1%, the pressure solver obtained a speedup of 1.9, and the total time obtained a speedup of 1.6 times that of the PBF method. Table 3 shows that the maximum density error is controlled by 0.01%, the pressure solver obtained a speedup of 2.6 times, and the total time obtained a speedup of 2.3 times that of the PBF method. The improvement in the overall performance is related not only to the reduction of iterations but also to the decrease of the consuming time of each physical time step which saves data storage and operation times.

7. Conclusions and Future Work

In this paper, we have proposed an incompressible fluid simulation approach based-on relaxing density-invariant condition, realized the simulation of small-scale ocean scenes, such as breaking dam and wave heaving, rolling, and breaking, and solved the problem of the boundary particle deficiency problem. The execution efficiency is improved by reducing the number of iterations and optimizing the calculation process. Compared with the PBF method, the pressure solver and the total calculation of our approach obtained a speedup of 2.6 times and 2.3 times, respectively, when the maximum density error is controlled within 0.01%.

The density filtering using the well-known MLS interpolation [33, 34] or the Shepard kernel interpolation [35] is an more efficient way to solve the problem of the particle deficiency near the free surface. It is our next research. Based on the ISPH by relaxing the density-invariant condition method, we plan to simulate the interaction between a vessel underway and water, including spray and foam, on the bow of the vessel in a marine simulator.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was funded by the Ministry of Science and Technology of People's Republic of China via National 863 Project No. SS2015AA010504. Funding was also provided by

the China Department of Education of Fujian Province under Award no. JA15269.

References

[1] M. Desbrun and M. P. Gascuel, "Smoothed particles: a new paradigm for animating highly deformable bodies," in *Proceedings of the Computer Animation and Simulation '96*, pp. 61–76, New York, NY, USA, 1996.

[2] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 154–159, California, Calif, USA, July 2003.

[3] J. J. Monaghan, "Simulating free surface flows with SPH," *Journal of Computational Physics*, vol. 110, no. 2, pp. 399–406, 1994.

[4] M. Becker and M. Teschner, "Weakly compressible SPH for free surface flows," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 209–217, California, Calif, USA, August 2007.

[5] S. J. Cummins and M. Rudman, "An SPH projection method," *Journal of Computational Physics*, vol. 152, no. 2, pp. 584–607, 1999.

[6] S. Koshizuka, A. Nobe, and Y. Oka, "Numerical analysis of breaking waves using the moving particle semi-implicit method," *International Journal for Numerical Methods in Fluids*, vol. 26, no. 7, pp. 751–769, 1998.

[7] S. Shao and E. Y. M. Lo, "Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface," *Advances in Water Resources*, vol. 26, no. 7, pp. 787–800, 2003.

[8] N. Foster and R. Fedkiw, "Practical animation of liquids," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 23–30, New York, NY, USA, 2001.

[9] N. Chentanez, "Real-time Eulerian water simulation using a restricted tall cell grid," *ACM Transactions on Graphics*, vol. 30, pp. 1–10, 2011.

[10] F. Sin, A. W. Bargteil, and J. K. Hodgins, "A point-based method for animating incompressible flow," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 247–255, Louisiana, La, USA, August 2009.

[11] B. Solenthaler and R. Pajarola, "Predictive-corrective incompressible SPH," *ACM Transactions on Graphics*, vol. 28, pp. 1–6, 2009.

[12] M. Ihmsen, N. Akinci, M. Gissler, and M. Teschner, "Boundary handling and adaptive time-stepping for PCISPH," in *Proceedings of the 7th Workshop on Virtual Reality Interactions and Physical Simulations*, pp. 79–88, Copenhagen, Denmark, November 2010.

[13] J. Bender and K. Dan, "Divergence-free smoothed particle hydrodynamics," in *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 147–155, California, Calif, USA, August 2015.

[14] J. Bender and D. Koschier, "Divergence-free SPH for incompressible and viscous fluids," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 3, pp. 1193–1206, 2016.

[15] A.-M. Zhang, P.-N. Sun, F.-R. Ming, and A. Colagrossi, "Smoothed particle hydrodynamics and its applications in fluid-structure interactions," *Journal of Hydrodynamics*, vol. 29, no. 2, pp. 187–216, 2017.

[16] K. Raveendran, C. Wojtan, and G. Turk, "Hybrid smoothed particle hydrodynamics," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 33–42, British Columbia, Canada, August 2011.

[17] F. Losasso, J. O. Talton, N. Kwatra, and R. Fedkiw, "Two-way coupled SPH and particle level set fluid simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 797–804, 2008.

[18] S. Marrone, A. Colagrossi, M. Antuono, C. Lugni, and M. P. Tulin, "A 2D+t SPH model to study the breaking wave pattern generated by fast ships," *Journal of Fluids and Structures*, vol. 27, no. 8, pp. 1199–1215, 2011.

[19] P. N. Sun, A. Colagrossi, S. Marrone et al., "Detection of Lagrangian coherent structures in the SPH framework," *Computer Methods Applied Mechanics and Engineering*, vol. 305, pp. 849–868, 2016.

[20] M. Ihmsen, N. Akinci, G. Akinci, and M. Teschner, "Unified spray, foam and air bubbles for particle-based fluids," *The Visual Computer*, vol. 28, no. 6-8, pp. 669–677, 2012.

[21] M. Macklin and M. Müller, "Position based fluids," *ACM Transactions on Graphics*, vol. 32, pp. 1–5, 2013.

[22] J. J. Monaghan and A. Kos, "Solitary waves on a cretan beach," *Journal of Waterway, Port, Coastal, and Ocean Engineering*, vol. 125, no. 3, pp. 145–155, 1999.

[23] M. Becker, H. Tessendorf, and M. Teschner, "Direct forcing for Lagrangian rigid-fluid coupling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 3, pp. 493–503, 2008.

[24] L. D. Libersky, A. G. Petschek, T. C. Carney, J. R. Hipp, and F. A. Allahdadi, "High strain lagrangian hydrodynamics: a three-dimensional SPH code for dynamic material response," *Journal of Computational Physics*, vol. 109, no. 1, pp. 67–75, 1993.

[25] H. Schechter and R. Bridson, "Ghost SPH for animating water," *ACM Transactions on Graphics*, vol. 31, pp. 1–8, 2012.

[26] P. N. Sun, A. Colagrossi, and A. M. Zhang, "Numerical simulation of the self-propulsive motion of a fishlike swimming foil using the δ^+-SPH model," *Theoretical and Applied Mechanics Letters*, vol. 8, no. 2, pp. 115–125, 2018.

[27] B. Solenthaler, J. Schläfli, and R. Pajarola, "A unified particle model for fluid-solid interactions," *Computer Animation and Virtual Worlds*, vol. 18, pp. 69–82, 2010.

[28] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, and M. Teschner, "Versatile rigid-fluid coupling for incompressible SPH," *ACM Transactions on Graphics*, vol. 31, pp. 1–8, 2012.

[29] N. Akinci, A. Dippel, G. Akinci, and M. Teschner, "Screen space foam rendering," *Journal of WSCG*, vol. 21, no. 3, pp. 173–182, 2013.

[30] M. Ihmsen, N. Akinci, M. Becker, and M. Teschner, "A parallel SPH implementation on multi-core CPUs," *Computer Graphics Forum*, vol. 30, pp. 99–112, 2015.

[31] J. J. Monaghan and J. C. Lattanzio, "A refined particle method for astrophysical problems," *Astronomy & Astrophysics* , vol. 149, pp. 135–143, 1985.

[32] W. J. van der Laan, S. Green, and M. Sainz, "Screen space fluid rendering with curvature flow," in *Proceedings of the Symposium on Interactive 3D Graphics and Games*, pp. 91–98, Massachusetts, Mass, USA, Feburary 2009.

[33] A. Colagrossi and M. Landrini, "Numerical simulation of interfacial flows by smoothed particle hydrodynamics," *Journal of Computational Physics*, vol. 191, no. 2, pp. 448–475, 2003.

[34] A.-M. Zhang, X.-Y. Cao, F.-R. Ming, and Z.-F. Zhang, "Investigation on a damaged ship model sinking into water based on three dimensional SPH method," *Applied Ocean Research*, vol. 42, pp. 24–31, 2013.

[35] B. Ren, M. He, P. Dong, and H. Wen, "Nonlinear simulations of wave-induced motions of a freely floating body using WCSPH method," *Applied Ocean Research*, vol. 50, pp. 1–12, 2015.