Physics-Based Simulation of Ocean Scenes in Marine Simulator Visual System

Haijiang Li *, Hongxiang Ren *, Shaoyang Qiu and Chang Wang

Key Laboratory of Marine Dynamic Simulation and Control for Ministry of Communications, Dalian Maritime University, Dalian 116026, China; qsydmu@163.com (S.Q.); sigdmu@gmail.com (C.W.)
* Correspondence: hai_nav@163.com (H.L.); rhx_dmu@163.com (H.R.);
Tel.: +86-1804-115-4069 (H.L.); +86-1894-092-9166 (H.R.)

Received: 22 November 2019; Accepted: 7 January 2020; Published: 12 January 2020

Abstract: The realistic simulation of ocean scenes is of great significance in many scientific fields. We propose an improved Smoothed Particle Hydrodynamics (SPH) framework to simulate the ocean scenes. The improved SPH combines nonlinear constant density constraints and divergence-free velocity field constraint. Density constraints adjust the particle distribution on position layer, so that the density is constrained to a constant state. The addition of the divergence-free velocity field constraint significantly accelerates the convergence of constant density constraint and further reduces the density change. The simulation results show that the improved SPH has high solution efficiency, large time steps, and strong stability. Then, we introduce a unified boundary handling model to simulate coupling scenes. The model samples the boundary geometry as particles by means of single layer nonuniform sampling. The contribution of the boundary particles is taken into account when the physical quantities of fluid particles are computed. The unified model can handle various types of complex geometry adaptively. When rendering the ocean, we propose an improved anisotropic screen space fluid method, which alleviates the discontinuity problem near the boundary and maintains the anisotropy of particles. The research provides a theoretical reference for the highly believable maritime scene simulation in marine simulators.

Keywords: ocean scenes; marine simulator; smoothed particle hydrodynamics; anisotropy

1. Introduction

Marine simulator has been widely applied to fields of marine education and training, engineering demonstration, scientific research, etc. The International Convention on Standards of Training, Certification and Watchkeeping for Seafarers and its amendments put forward a series of mandatory requirements and suggestions on the performance standards, scope of application, and rules of use of marine simulators. Improving the performance of marine simulator is required by both international conventions and navigation practice. The visual system is the most direct and the largest source of information for simulator operators, and is also an important part of marine simulator and one of the important indexes to evaluate the performance of marine simulator. Ocean waves account for about half of the whole visual system in the marine simulator. The fidelity of the ocean scene directly affects the immersion of the whole marine simulator. Moreover, the ocean waves are also an important object in the study of ship seakeeping property and maneuverability, and accurate ocean wave model can improve the fidelity of simulator manipulation. There are many modeling methods for ocean waves in computer graphics, and the most commonly used ones are the spectrum-based approaches and computational fluid dynamics (CFD) methods.
1.1. Spectrum-Based Approaches

Spectrum-based approaches are statistical models derived from the observed data of ocean waves in oceanography and were introduced into wave modeling in computer graphics by Tessendorf [1]. In these methods, the wave is regarded as a stationary normal process with ergodicity, and the fixed-point wave surface is described by the superposition of infinite random cosine waves [2]. Wave spectrums can be divided into frequency spectrums \( S(\omega) \) and directional spectrums \( S(\omega, \theta) \). Generally, the spectrums used for wave modeling includes Pierson–Moskowitz spectrum [2], Phillips spectrum [1,2], Joint North Sea Wave Observation Project (JONSWAP) spectrum [2,3], and Texel–Marsen–Arsloe (TMA) spectrum [4,5]. Directional spectrums are the extension of frequency spectrums. They are obtained by adding direction information \( D(\omega, \theta) \) on the basis of frequency spectrums. The directional spreading function mainly includes the positive cosine squared directional spreading function in [1], Hasselmann directional spreading function and Donelan–Banner directional spreading function in [2], etc. The spectrum-based approaches are characterized by their fast calculation speed. Therefore, they can simulate an ocean with a wide scale in real-time. Moreover, these methods have a strong oceanographic observation background. However, the spectrum-based approaches also have some limitations such as the inability to realistically interact with complex boundaries. After all, they are based on statistical derivation, so it is difficult to show the inherent physical laws of waves. Moreover, in the process of simulating wave breaking, it is often necessary to use auxiliary particles and the modeling process is complicated.

1.2. CFD Methods

The CFD methods are mainly based on the well-known Navier–Stokes equations. According to the domain discretization technique applied, the CFD methods can be divided into grid-based methods, meshfree methods, and combined methods. Grid-based method generally refers to Eulerian grid. The Eulerian grid used for wave modeling is the marker and cell (MAC) grid in [6], tall cell grid in [7,8], tetrahedral mesh in [9], etc. In terms of application scenarios, Alfonsi et al. [10] simulated the diffraction of a wave of viscous fluid impinging on a large-diameter vertical circular cylinder, which provided a good reference for the simulation of fluid–rigid coupling and viscous fluid scenes. However, mesh generation for the problem domain is a prerequisite for the Eulerian grid method. Constructing a regular grid for the irregular or complex geometry has never been an easy task, and usually requires additional complex mathematical transformation that can be even more expensive than solving the problem itself [11]. The difficulties and limitations of the grid-based methods are especially evident in the simulating process of large deformation scenes. Therefore, meshless methods such as SPH are widely used in ocean modeling at present. Since the computational frame in the meshfree methods is a set of arbitrarily distributed nodes rather than a system of predefined mesh/grid, the meshfree methods are attractive in dealing with problems that are difficult for traditional grid-based methods, such as spray splashing and fluid–rigid coupling. There are many studies on SPH methods, such as boundary handling, incompressibility, surface extraction/reconstruction, nearest neighbor search [12], surface tension [13], vortex/turbulent motion [14,15], multiphase flow [16–18], data-driven SPH methods [19–21], etc.

Boundary handling is one of the hot issues in SPH research, and the particle-based strategy is probably the most popular way. Akinci et al. [22] sampled the rigid object as one layer of nonuniform boundary particles. The boundary particles were taken into account when physical quantities of fluid particles, such as density, pressure force, friction force, etc., were computed. The particle volume \( V_b \) was adopted to compute the densities correctly regardless of the boundary particle sampling. It is a versatile and stable model to deal with arbitrary rigid objects. In terms of incompressibility, researchers have proposed a series of optimized SPH models in computer graphics, such as weakly compressible SPH (WCSPH) [23], predictive-corrective incompressible SPH (PCISPH) [24], local Poisson SPH (LPSPH) [25], position-based fluids (PBF) [26], implicit incompressible SPH (IISPH) [27], divergence-free SPH (DFSPH) [28], etc. WCSPH is not suitable for real-time modeling of maritime
scenes because of its small time step size. PBF and DFSPH are relatively large in time step size and have good incompressibility, thus they are widely used nowadays. Surface tracking and reconstruction of particle-based fluid are always some of the hot issues. The color field $c_S$ was used to identify surface particles by Müller et al. [29]. The point splatting techniques and marching cubes (MC) algorithm are adopted to generate a continuous surface. The MC algorithm adopted by Chentanez et al. [30] is a commonly used SPH surface reconstruction algorithm. However, under the fine scale, the number of meshes generated by the MC algorithm is huge, which results in a sharp increase in memory consumption and a slow calculation speed. Therefore, it is a preferred solution in offline rendering. At present, the screen space fluids (SSF) algorithm proposed by van der Laan et al. [31] has been widely used in real-time rendering. Green [32] elaborated on the specific implementation of the SSF algorithm and improved the smoothing algorithm to Gaussian bilateral filtering with good edge-preserving ability. The SSF algorithm is implemented on GPU, and the rendering speed is much faster than that of the MC. However, the isotropic depth map on which the smoothing process of SSF algorithm is performed cannot solve the surface smoothing problem fundamentally. The anisotropic algorithm proposed by Yu and Turk in [33] has the potential to solve the surface smoothing problem of the SSF algorithm. Truong and Yuksel proposed an improved narrow-range filter for the SSF algorithm [34], and an anisotropic depth map was used to generate a smooth surface and improved the edge-preserving performance near the discontinuous particles.

There are also some hybrid methods that combine Eulerian grid method with meshfree methods, such as Fluid Implicit Particle (FLIP) [35], Narrow Band FLIP (NB-FLIP) [36,37], Lattice-Boltzmann Method (LBM) [38], Arbitrary Lagrangian–Eulerian (ALE) [39], Hybrid SPH [40], Eulerian-DFSPH [41], etc. These methods are relatively complex in modeling and limited in application scenarios. For these methods, special consideration needs to be given to stability, so they are rarely used in real-time maritime scene. Besides, some simplified methods are also used in wave modeling. Kass and Miller adopted the shallow water equation [42], which is a simplified form of the full Navier–Stokes equations, to model the waves. Jeschke et al. [43] proposed a model coupling spectrum-based approach and PDE method together, which improved the interaction of moving obstacles and reduced the limitations of the CFL condition and Nyquist limit. These simplified methods sacrifice the ability to simulate arbitrary fluid motion, and have many limitations in dealing with complex interactive scenes.

Therefore, for the application of virtual reality system, we need a kind of ocean motion and interaction modeling method which has strong stability, relatively fast solution speed, strong expansibility, and is realistic. We propose a modeling method and rendering technique for ocean motion and interaction in the visual system of the marine simulator. The main contributions are as follows:

1. We propose a novel hybrid SPH framework to discretize the governing equations of ocean. Our hybrid SPH method combines the nonlinear density constraints and divergence-free velocity field constraint for the first time, which has faster convergence speed and larger time steps.

2. Based on our novel SPH method, we introduce a unified boundary handling model for maritime scenes. Compared with spectrum-based approaches, which are widely used in marine simulator nowadays, our method has good expansibility and can deal with the boundary interaction of various complex maritime scenes adaptively.

3. We propose an improved anisotropic screen space fluid rendering method. A new anisotropic piecewise function is derived to deal with deformed particles. It can maintain sharper features in the splashing area.

2. Simulation Models

2.1. Governing Equations

According to the theories of physical oceanography and computational fluid dynamics, we assume that the ocean is a homogeneous incompressible fluid with constant temperature, i.e., the effects of
temperature and salinity can be ignored. The continuity equation and the momentum equation in Lagrangian form can be used to describe the motion of ocean:

\[
\frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{v} = 0, \quad (1)
\]

\[
\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho} \mathbf{\nabla} p + \nu \nabla^2 \mathbf{v} + \frac{f_{\text{ext}}}{\rho}, \quad (2)
\]

where \(\rho, \mathbf{v}, p,\) and \(f_{\text{ext}}\) denote the density, velocity, pressure, and body forces, respectively. \(\nu\) is kinematic viscosity, \(\nu = \mu/\rho,\) and \(\mu\) is viscosity. The Lagrangian form considers that the ocean consists of a set of small moving fluid elements. Each fluid particle has a mass \(m\) and carries attributes such as density \(\rho,\) pressure \(p,\) etc. This is consistent with the discrete idea of the SPH model.

2.2. SPH Discretization

In SPH framework, the problem domain is discretized into a series of interacting fluid particles, and these fluid particles carry various physical quantities, including mass, density, velocity, acceleration, etc. For particle \(i,\) the quantity \(A_i\) and its spatial derivative can be obtained by weighted summation of particles in its neighborhood [44]:

\[
A_i = \sum_j m_j \rho_j W_{ij}, \quad (3)
\]

\[
\nabla A_i = -\frac{1}{\rho_i} \sum_j m_j A_{ij} \nabla W_{ij}, \quad (4)
\]

where \(A_{ij} = A_i - A_j, W_{ij}\) is a smoothing kernel function, \(W_{ij} = W(x_{ij}/h),\) \(\nabla W_{ij} = \partial W_{ij}/\partial x_{ij}, x_{ij} = x_i - x_j, x_{ij} = |x_{ij}|, x\) is the particle position, and \(h\) is the support radius.

According to Equations (3) and (4), the items in Equation (2) can be divided into discrete components. For particle \(i,\) its density \(\rho_i\) can be obtained by employing Equation (3):

\[
\rho_i = \sum_j m_j W_{ij}, \quad (5)
\]

An equation of pressure is needed when the pressure term is discretized. Here we employ the Tait’s equation [28]:

\[
p_i = \kappa i \rho_0 \gamma \left( \left( \frac{\rho_i}{\rho_0} \right)^\gamma - 1 \right), \quad (6)
\]

where \(\kappa\) and \(\gamma\) are stiffness parameters. \(\kappa\) is solved below and \(\gamma = 1.\) \(\rho_0\) is the rest density. Then, the gradient of pressure can be expressed as:

\[
\nabla p_i = \kappa \nabla \rho_i = \kappa \sum_j m_j \nabla W_{ij}, \quad (7)
\]

2.2.1. Divergence-Free Solver

The continuity Equation (1) is the application of the law of mass conservation in hydrodynamics. In this paper, we assume that the ocean is a homogeneous incompressible fluid, thus the substantial derivative of density should be zero, i.e., \(\frac{D\rho}{Dt} = 0.\) Then, the continuity equation can be transformed into:

\[
\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v} = 0, \quad (8)
\]

where \(\nabla \cdot \mathbf{v}\) denotes the divergence of the velocity field. The incompressibility of the fluid can be further enhanced by fulfilling the divergence-free condition \(\nabla \cdot \mathbf{v} = 0.\) This constraint is mainly achieved by
converting the divergence-free velocity field to pressure force. According to Equation (4), the pressure force of particle $i$ is:

$$F^p_i = -\frac{m_i}{\rho_i} \nabla \rho_i = -\kappa \frac{m_i}{\rho_i} \sum_j m_j \nabla W_{ij},$$  \hspace{1cm} (9)

According to Newton’s third law, the pressure force that the particle $i$ acts on its neighboring particle $j$ is given by:

$$F^p_{j\rightarrow i} = \kappa m_i m_j \rho_i \nabla W_{ij},$$  \hspace{1cm} (10)

The goal of the divergence-free solver is to generate a set of symmetric pressure forces to enforce the substantial derivative of density to zero. Velocity changes caused by this set of symmetric pressure forces are

$$\Delta v_i = \Delta t \frac{F^p_i}{m_i}, \quad \Delta v_j = \Delta t \frac{F^p_{j\rightarrow i}}{m_i}.$$

Substituting Equation (4) into Equation (8) and combining the changes of velocity, we can get:

$$D\rho_i = \sum_j m_j v_{ij} \nabla W_{ij} = \Delta t \sum_j m_j \left( \frac{F^p_i}{m_i} - \frac{F^p_{j\rightarrow i}}{m_i} \right) \nabla W_{ij},$$ \hspace{1cm} (11)

Substituting Equations (9) and (10) into Equation (11) yields:

$$\kappa_i = \rho_i \frac{D\rho_i}{\Delta t} \alpha_i,$$  \hspace{1cm} (12)

where $\alpha_i = -1 / \left( |\sum_j m_j \nabla W_{ij}|^2 + |\sum_j m_j \nabla W_{ij}|^2 \right)$. Then, the total force of particle $i$ is:

$$F^p_{i,\text{total}} = F^p_i + F^p_{i\rightarrow j} = -m_i \sum_j m_j \left( \frac{\kappa_i}{\rho_i} + \frac{\kappa_j}{\rho_j} \right) \nabla W_{ij},$$ \hspace{1cm} (13)

The velocity changes caused by the pressure force $F^p_{i,\text{total}}$ are:

$$\Delta v_i = \Delta t \frac{F^p_{i,\text{total}}}{m_i} = -\Delta t \sum_j m_j \left( \frac{\kappa_i}{\rho_i} + \frac{\kappa_j}{\rho_j} \right) \nabla W_{ij},$$ \hspace{1cm} (14)

We describe the calculation flow of the divergence-free solver in detail in Appendix A.

### 2.2.2. Density Constraint Solver

In DFSPH method, a constant density solver is added to further maintain the incompressibility of the fluid and accelerate the convergence speed of the algorithm. The constant density solver uses the Pressure Poisson Equation (PPE) with density deviation as source term. In our novel SPH method, we propose a density constraint solver based on nonlinear constant density constraints, and try to replace the constant density solver in DFSPH method with a novel density constraint solver. We found that the divergence-free solver and density constraint solver could be well coupled. The density constraints are as follows:

$$C_i(x) = \frac{\rho_i}{\rho_0} - 1 \leq 0,$$ \hspace{1cm} (15)

where $x = [x_1, \ldots, x_n]$ and $n$ is the cardinality. It can be expanded in a Taylor series, and the final position correction $\Delta x_i$ for particle $i$ is:

$$\Delta x_i = \lambda_i \nabla C_i,$$ \hspace{1cm} (16)

where $\nabla C_i(x)$ is the constraint gradient. $\lambda_i$ is the scaling factor:

$$\lambda_i = \frac{-C_i(x)}{|\nabla C_i|^2}.$$

\hspace{1cm} (17)
The goal of the density constraint solver is to generate a set of symmetric constraint “forces” to enforce Equation (15) to be no more than zero. Therefore, the gradient of the constraint function $\nabla C_i(x)$ is composed of two symmetrical parts, which are $\nabla x_i C_i$ and $\nabla x_j C_i$. The gradient of the constraint function with respect to $x_i$ is:

$$\nabla x_i C_i = \frac{1}{\rho_0} \sum_j m_j \nabla W_{ij}, \quad (18)$$

The gradient with respect to $x_j$ is:

$$\nabla x_j C_i = -\frac{1}{\rho_0} m_j \nabla W_{ij}, \quad (19)$$

Inserting Equations (18) and (19) into Equation (17), we obtain:

$$\lambda_i = \rho_0^2 \alpha_i C_i(x), \quad (20)$$

where the form of $\alpha_i$ is the same as that in Equation (12). Therefore, the two solvers can be well coupled. Within a certain range of accuracy, $\alpha_i$ can be reused and no additional calculations are required. The position correction of the particle $i$ under the density constraint can be derived as:

$$\Delta x_i = \lambda_i \nabla C_i = \lambda_i \nabla x_i C_i + \sum_j \lambda_j \nabla x_j C_i = \frac{1}{\rho_0} \sum_j m_j (\lambda_i + \lambda_j) \nabla W_{ij}, \quad (21)$$

We describe the calculation flow of the density constraint solver in detail in Appendix B.

2.3. Boundary Handling

The boundary handling in the spectrum-based approaches in the current marine simulator is quite complicated. Regular boundaries such as cuboid and sphere are simple to handle, but complex boundaries such as ship and lighthouse are very difficult to deal with. Moreover, it usually needs to model the interaction additionally, and the model often lacks physical authenticity. In this paper, we introduce a particle-based strategy to deal with the rigid boundary in maritime scenes. That is, sampling the rigid boundary as boundary particles, and these particles are incorporated into the computation of density and pressure. The model can deal with regular boundary, irregular boundary, static boundary, and dynamic boundary in a unified way, thus it simplifies the complexity of modeling. Moreover, the model is a physical method, which can be directly integrated into the hybrid SPH model proposed above. In our unified model, rigid body objects are sampled as single layer nonuniform particles using the Poisson Disk Sampling (PDS) introduced in [45].

Suppose the quality of the boundary particle $x_{bi}$ is $m_{bi}$. After considering the contribution of boundary particles, the density of fluid particle $x_i$ in Equation (5) is deformed as follows:

$$\rho_i = m_i \sum_{fj} W_{ifj} + \sum_{bj} m_{bj} W_{ibj}, \quad (22)$$

where $fj$ is the fluid particle neighbor and $bj$ denotes the boundary particle neighbor. All fluid particles have the same size and rest density, which means all masses are equal: $m_i = m_{fj}$. We set the density of a boundary particle to be rest density of the fluid, i.e., $\rho_{bi} = \rho_0$. Then, the mass of boundary particle $bi$ can be written as:

$$m_{bi} = \rho_0 V_{bi} = \frac{\rho_0 \gamma_1}{\sum_{bj} W_{hbj}}, \quad (23)$$
The substantial derivative of the density of the fluid particles near the boundary is transformed from Equation (11) into:

\[
\frac{D\rho_i}{Dt} = m_i \sum_{fj} v_{ifj} \nabla W_{ifj} + \sum_{bj} m_{bj} v_{ibj} \nabla W_{ibj},
\]

(24)

The velocity changes in divergence-free solver and the position correction in density constraint solver are also deformed as:

\[
\Delta v_i = -\Delta t \cdot m_i \sum_{fj} \left( \frac{x_i}{\rho_i} + \frac{x_{fj}}{\rho_{fj}} \right) \nabla W_{ifj} - \Delta t \cdot 2\gamma_2 \frac{x_i}{\rho_i} \sum_{bj} m_{bj} \nabla W_{ibj},
\]

(25)

\[
\frac{D\rho_i}{Dt} = m_i \rho_0 \sum_{fj} \left( \lambda_i + \lambda_{fj} \right) \nabla W_{ifj} + 2\gamma_3 \frac{x_i}{\rho_0} \sum_{bj} m_{bj} \nabla W_{ibj},
\]

(26)

where the correcting coefficient \(\gamma_2 = 0.5\) and \(\gamma_3 = 1.0\). The factor \(\alpha_i\) is:

\[
\alpha_i = \frac{-1}{\left( \sum_j m_{fj} \nabla W_{ifj} \right)^2 + \sum_j \left| m_{fj} \nabla W_{ifj} \right|^2} + \frac{-1}{\left( \sum_{bj} m_{bj} \nabla W_{ibj} \right)^2 + \sum_{bj} \left| m_{bj} \nabla W_{ibj} \right|^2},
\]

(27)

### 2.4. Anisotropic Screen Space Fluid

Although the SPH method is very flexible and easy to extend, it is quite difficult to reconstruct a smooth surface. Especially in the marine simulator, surface reconstruction algorithm requires for high performance of real-time, thus most of the SPH surface tracking algorithms in computer graphics are not applicable. The screen space fluid (SSF) rendering algorithm is implemented on GPU. In fact, SSF does not reconstruct the surface, but smooths the depth map of particles to get a seemingly smooth ocean surface. Usually, the depth map needs to be smoothed by smoothing algorithm, such as curvature flow [31], bilateral Gaussian filter [32], narrow-range filter [33], etc. However, the method based on isotropic kernel cannot solve the surface smoothing problem fundamentally, and there is still a bumpy appearance in the details. Therefore, based on the work of Yu and Turk [33], we propose an improved anisotropic rendering method. The method fully considers the distribution of physical quantities of fluid particles in different directions, applies the weighted version Principal Component Analysis (WPCA) to the neighboring particle positions, and extracts a smoother depth map.

The improved model analyzes the spatial distribution of the neighboring particles of the fluid particle. According to the principle of anisotropy [33], a transformation matrix \(T\) is generated for each fluid particle, and then \(T\) is used to rotate and scale the particle sphere rendered by the point sprite. After this step, the particle sphere becomes an ellipsoid. Then, the depth map is extracted based on the ellipsoid to improve the smoothness of the surface. Therefore, a covariance matrix is first constructed according to the distribution of particle \(i\) and its neighboring particles:

\[
\text{Cov}_i = \frac{\sum_j w_{ij} (x_j - x_i^w)(x_j - x_i^w)^T}{\sum_j w_{ij}},
\]

(28)

\[
x_i^w = \sum_j w_{ij} x_j / \sum_j w_{ij},
\]

(29)

where \(w\) is an isotropic weight function:

\[
w_{ij} = \begin{cases} 
1 - \left( \frac{|x_i - x_j|}{h} \right)^3, & \text{if } |x_i - x_j| < h \\
0, & \text{otherwise} 
\end{cases}
\]

(30)
With each particle, the singular value decomposition (SVD) of the associated \( \text{Cov} \) gives:

\[
\text{Cov} = R \Sigma R^T,
\]

(31)

where \( R \) is a rotation matrix, which can be used to rotate the main axis of the particle sphere rendered by point sprite. \( \Sigma \) is a diagonal matrix containing three eigenvalues, and \( \Sigma = \text{diag}(\sigma_1, \sigma_2, \sigma_3), \sigma_1 \leq \sigma_2 \leq \sigma_3 \), which are used to scale the three axes of the particle sphere. Since the particle distribution is arbitrary and noisy, to ensure the robustness of the algorithm, the bilateral SVD method is used to decompose the covariance matrix. The SVD method is fast and stable in dealing with small-scale matrices. However, if the particle sphere is directly scaled by the eigenvalue generated after the decomposition, when the number of neighboring particles is insufficient, the extreme deformation of particle sphere will occur. Yu and Turk proposed an isotropic correction method [33] to set a threshold for the number of neighboring particles. If the number of neighboring particles is less than the threshold value, the eigenvalues in \( \Sigma \) will be set to a specific value. Although it is a very efficient method, in SSF, there may be significant discontinuities or rough boundary details in the boundary region where the neighboring particles are insufficient. Therefore, we propose a relatively fine deformation correction method:

\[
\tilde{\sigma}_i = \max(\sigma_i, \sigma_1/k_r),
\]

(32)

where \( k_r \) is a user-defined scaling factor and \( k_r = 5.0 \).

\[
\tilde{\Sigma} = \begin{cases} 
  k_s \cdot \text{diag}(\sigma_1, \tilde{\sigma}_2, \tilde{\sigma}_3), & \text{if } N \geq N_e \\
  \text{diag}(\xi_1, \xi_2, \xi_3), & \text{if } N_e > N \geq N_l \\
  \text{diag}(\zeta_1, \zeta_2, \zeta_3), & \text{if } N < N_l 
\end{cases}
\]

(33)

where \( k_s \) is a user-defined scaling factor and \( k_s = 560 \). \( N_e \) and \( N_l \) are thresholds of the number of neighboring particles, \( N_e = 20 \) and \( N_l = 5 \). \( \xi_i \) and \( \zeta_i \) are the main diagonal elements of the two correction matrices, and they can be given as:

\[
s_i = (1-l)k_n + l \tilde{\sigma}_i, \quad i = 1, 2, 3,
\]

(34)

where \( k_n \) is a scaling factor, \( k_n = 0.5 \) and \( l \in [0, 1] \), here \( l = 0.8 \), \( \xi_i = \min(1.0, s_i) \), and \( \zeta_i = \min(k_n, \tilde{\sigma}_i) \).

The deformation correction method preserves the anisotropy of particle spheres better. More neighboring particle segments can better maintain the continuity of the fluid particle size near the boundary. Finally, the transformation matrix is:

\[
T = R\tilde{\Sigma},
\]

Due to the characteristics of SPH method, particle distribution is relatively irregular. Although this distribution is beneficial to modeling, it will cause bumpy appearance near the surface, and then affect the smoothness of the surface rendering. To resolve the problem of irregular particle placement, a Laplacian smoothing is applied to adjust particle position.

3. Results and Discussion

We verified the stability and effectiveness of our SPH model by simulating various maritime scenes in marine simulator, such as dam break scenes and the coupling scenes of lighthouse, lifeboat, tanker, etc. The specific configurations of the computer used in our experiment are Intel(R) Core(TM) i7-6700 CPU (3.40 GHz), RAM 8.00 GB, and NVIDIA GeForce GTX 1070 GPU. The physical model was implemented using C++, the rendering part was based on modern OpenGL, and the integrated development environment was Microsoft Visual Studio 2019. Since our SPH model will be applied in the visual system of marine simulator, we paid more attention to the rapidity and stability than the accuracy. Although a highly accurate model can bring more accurate results, it is usually time-consuming and
the visual effects sometimes cannot be significantly improved. Therefore, the Taylor expansion in the density constraint solver only retains the first order, and the integration method of our SPH model only uses the stable Euler integrals. To verify the performance of our model, we also implemented some SPH models that are widely used in ocean modeling of virtual reality systems, such as PBF [21] and DFSPH [21].

3.1. Performance of the Presented SPH Model

In this section, we design a dam break scene to compare the performances of the presented SPH, DFSPH, and PBF models. Figure 1 is the design drawing of the dam break scene. Figure 1 shows a three-dimensional schematic diagram of the scene, as well as side and front views. The blue area represents the fluid and the thick wireframe represents the bounding box. The size of the bounding box on the three axes of x, y, z is 6 L × 5 L × 3 L, and all the dam break scenes have the same size in this paper. The boundary bounding box is a static boundary. The PDS method is used to sample single-layer non-uniform boundary particles to participate in the SPH estimation. The density of the boundary particles is the rest density of the fluid \( \rho_0 \), and the velocity of the static boundary particles is zero. The unit of length used in this paper is dimensionless, and its specific length can be set by L. The size of the fluid block is variable because we designed dam break experiments with different particle numbers. In Figure 2, we list four kinds of dam break scenes, and the corresponding fluid block sizes are different: 1.5 L × 2 L × L for 14.1 k, 1.5 L × 2 L × 1.5 L for 32.8 k, 1.5 L × 2 L × 2 L for 44.1 k, and 2 L × 2 L × 2 L for 59.3 k. Figure 1 shows the scene of 14.1 k fluid particles.

![Figure 1](image-url)

Figure 1. Design drawing of dam break scene. The unit of length used in dam break scene is dimensionless, and its specific length can be set by L. The figure shows a three-dimensional schematic diagram of the scene, as well as side and front views.

Figure 2 shows the time cost comparison of the first 2000 images in the dam break scene with different particle numbers. As can be seen in Figure 2, although the PBF model is stable, the single step time cost is high, and with the increase of the number of iterations, the solution time of PBF model will further increase. The time cost of DFSPH model is lower than that of PBF model, but the solution time of DFSPH model obviously increases after the 1750th time step (Figure 2a–c) when the fluid almost stops moving and covers the bottom of the bounding box. We carefully observed the time consumption of each solver in each time step and found that the constant density solver in DFSPH model needs extra effort to converge after the 1750th time step. In Figure 2d, due to the large number of fluid particles, the fluid is still moving at the 1750th time step, thus they do not appear there. In fact, the increase of solution time still exists. Due to the initial convergence of the divergence-free solver, as shown in Figure 2b–d, there is a rise of the time cost on the first frames in DFSPH model and our model. The rise of time cost does not exist in PBF model, because it does not have the divergence-free solver. Our SPH model combines the advantages of the above two models. In terms of running speed, it retains the fast solution speed of DFSPH model, thus its solution speed has also been improved to a
certain extent. In terms of stability, it inherits the advantage of PBF model and has strong stability in the later stage of the scene.

Figure 2. Time cost comparison of the first 2000 images in the dam break scene with different particle numbers. \( N_{\text{Fluid}} \) denotes the number of fluid particles. The green, orange, and blue curves in (a–d) represent the time cost of PBF model, DFSPH model, and our SPH model, respectively.

Compared with the PBF model, our method adds the divergence-free solver, and modifies the iteration condition of the density constraint solver by adding a density error threshold. If the density error is less than the threshold, no further iterations are required. The introduction of divergence-free solver seems to increase the amount of computation in each time step, but the runtime statistics (Figure 3a) show that it greatly accelerates calculation speed, and the acceleration ratio is stable at around 3–4 times. Because parameter \( \alpha_i \) is reused in our SPH model, the computation time of the density constraint solver is much less than that of PBF model. We set the solver iterations in the iterative cycle to 50 to ensure that the \( \rho_{\text{err}} \) in each iteration can meet the convergence condition (\( \rho_{\text{err}} < 0.01 \text{ g/m}^3 \)). We add the number of iterations in the 2000 time steps in the two methods and divide the sum by 2000 to get the average number of iterations, and the average number of iterations of our model is 9, while that of PBF model is 12. As can be seen in Figure 3b, the convergence of our model is better.
In theory, the position-based density constraint is more stable. The time is used in our SPH model, and the time step size in the first 2000 frames of the three SPH models. To maximize the time step size, the adaptive time step algorithm based on the CFL condition has been used. The density in our model has been kept at a relatively stable level.

Compared with the DFSPH model, our SPH method also has two solvers, a divergence-free solver and a density constraint solver. The density constraint solver in our SPH method uses the concept of position-based dynamics. In theory, the position-based density constraint is more stable. The time cost of the divergence-free solver is shown in Figure 4a, and that of the constant density solver in Figure 4b. The data come from the first 2000 time steps of the dam break scene. As shown in Figure 4a, the performance of the divergence-free solver in the two SPH models is similar, and, due to the initial convergence of the divergence-free solver, there is a rise of the time cost on the first frames in DFSPH model and our model. The performance of the constant density solver shown in Figure 4b is quite different. The statistical results show that the average time cost of our model in each time step is 147.72 ms, whereas that of DFSPH is 160.83 ms. After the 1750th time step in Figure 4b, the time cost of DFSPH model is significantly increased, and the density fluctuates greatly, prone to instability, whereas the density in our model has been kept at a relatively stable level.

In SPH-based methods, time step is one of the important bases to measure the stability of the models. To maximize the time step size, the adaptive time step algorithm based on the CFL condition is used in our SPH model, and the time step size in the first 2000 frames of the three SPH models...
in dam break scene is collected (Figure 5). As mentioned in [26,28], PBF and DFSPH have greater advantages in time step compared with other SPH methods, but the performance of our SPH model is more prominent. The time step of our model is larger, and the performance of our model is slightly better than PBF model and DFSPH model. As shown in Figure 5, in the first 1500 frames, the time step size of PBF and DFSPH models is relatively close, whereas that of our SPH is slightly larger than the two models. However, in the last 500 frames, the time step size of our SPH and the PBF models is different from that of DFSPH model. The reason for this is that both our SPH and the PBF models use the concept of position-based dynamics method, which is more controllable than the constant density solver in DFSPH model. Table 1 summarizes the average, maximum, and minimum time steps of the first 2000 frames. To keep the stability, the maximum time step is usually limited to specific value according to the CFL condition. Therefore, the maximum time steps of the three models are the same. However, the value of the average time step shows that our SPH model has stable performance and a larger step size.

![Time step size](image)

**Figure 5.** The time step statistics of the three methods in the first 2000 frames of dam break scene.

| Methods   | Average  | Maximum   | Minimum   |
|-----------|----------|-----------|-----------|
| Our Method| $1.868 \times 10^{-3}$ s | $3.162 \times 10^{-3}$ s | $1.088 \times 10^{-3}$ s |
| DFSPH     | $1.561 \times 10^{-3}$ s | $3.162 \times 10^{-3}$ s | $1.015 \times 10^{-3}$ s |
| PBF       | $1.810 \times 10^{-3}$ s | $3.162 \times 10^{-3}$ s | $0.907 \times 10^{-3}$ s |

The research on the incompressibility of SPH methods is mainly to observe the fluid particle density level. Since the influence of salinity and other factors have not been considered, the rest density of fluid particles is set to 1000 kg/m$^3$. We analyze the incompressibility of the model based on the dam break scene. The data in Figure 6 come from the first 2000 time steps of the dam break scene. In Figure 6a, we counted the maximum density of all fluid particles in each time step. The maximum density of fluid particles is stably maintained near the rest density. The maximum density value of DFSPH model oscillates in the first frames. However, due to the density constraint based on Position-Based Dynamics (PBD) in our method, this kind of oscillation is eliminated. Compared with DFSPH model, our method is more stable in the first 250 time steps. As shown in Figure 6b, we calculated the average density of all fluid particles in each time step. Since the interaction between air and fluid surface is not considered, the particle density near the free surface is underestimated, resulting in the average density slightly lower than the rest density. Compared with DFSPH model, from about the 500th time step, the average density of fluid particles in our model is closer to the rest density.
The velocity and density configuration of the lighthouse particles is the same as the bounding box particles. The velocity and density configuration of the lighthouse particles is the same as the bounding box particles. Therefore, the selection of rigid body objects with boundary bounding box is a static boundary, and the velocity of the static boundary particles is all zero.

The lighthouse is also a static boundary. The PDS algorithm is used to sample the lighthouse. The front and side view show the detailed parameters of the bounding box, lighthouse, and fluid block. In the side view, the lighthouse is located behind the fluid block, thus the lighthouse is represented by dashed lines.

As we can see, models such as lighthouses (Table 2) are much more complicated than regular objects such as cuboids, cylinders, spheres, etc. Therefore, the selection of rigid body objects with different complexity has certain representativeness. The traditional interaction methods such as collision detection have difficulty dealing with complex rigid mesh, thus the selection of scenes such as lighthouse can fully demonstrate the flexibility, stability, and applicability of the unified boundary handling model.

**Figure 6.** The comparison of incompressibility between DFSPH and our SPH models. The experimental scene is dam break, and the data are taken from the first 2000 time steps. (a) The maximum density of fluid particles in each time step. The orange and the blue curves in (a) are the maximum densities of DFSPH method and our method, respectively. (b) The average density of fluid particles in each time step. The orange and the blue curves in (b) are the average densities of DFSPH method and our method.

### 3.2. Boundary Handling Simulation Result

To validate the unified boundary handling model, various maritime fluid–rigid coupling scenes were designed, including ones coupling fixed objects such as lighthouse and with floating objects such as lifeboat and tanker. The detailed parameters of the lighthouse scene are shown in Figure 7. The boundary bounding box is a static boundary, and the velocity of the static boundary particles is all zero. The lighthouse is also a static boundary. The PDS algorithm is used to sample the lighthouse particles. The velocity and density configuration of the lighthouse particles is the same as the bounding box particles.

**Figure 7.** Design drawing of waves crashing against the lighthouse. The front and side view show the detailed parameters of the bounding box, lighthouse, and fluid block. In the side view, the lighthouse is located behind the fluid block, thus the lighthouse is represented by dashed lines.
handling model. When the fluid block breaks the dam, the flow velocity in the front of the fluid block is large, which is consistent with Figure 8d–f, and, when the flow is blocked by the lighthouse, its speed will slow down. We can see in Figure 8a that the velocity of the fluid particles impacting the lighthouse is obviously slowed down, while the velocity of fluid particles flowing to both sides of the lighthouse is faster, which has a very obvious interaction effect.

Table 2. The number of vertices, faces, and boundary particles sampled by PDS in different 3D rigid models.

| Rigid Models   | Vertices | Faces  | Sampling Time | Sampled Particles |
|----------------|----------|--------|---------------|-------------------|
| Lighthouse     | 17,317   | 34,168 | 1.47 s        | 3.8 k             |
| Lifeboat       | 24,859   | 45,789 | 3.11 s        | 5.1 k             |
| Tanker         | 24,443   | 43,782 | 6.94 s        | 4.4 k             |
| Bounding Box   | 8        | 12     | 11.08 s       | 312.5 k           |

Figure 7. Design drawing of waves crashing against the lighthouse. The front and side view show the detailed parameters of the bounding box, lighthouse, and fluid block. In the side view, the lighthouse is located behind the fluid block, thus the lighthouse is represented by dashed lines.

As we can see, models such as lighthouses (Table 2) are much more complicated than regular objects such as cuboids, cylinders, spheres, etc. Therefore, the selection of rigid body objects with different complexity has certain representativeness. The traditional interaction methods such as collision detection have difficulty dealing with complex rigid mesh, thus the selection of scenes such as lighthouse can fully demonstrate the flexibility, stability, and applicability of the unified boundary handling model. When the fluid block breaks the dam, the flow velocity in the front of the fluid block is large, which is consistent with Figure 8d–f, and, when the flow is blocked by the lighthouse, its speed will slow down. We can see in Figure 8a that the velocity of the fluid particles impacting the lighthouse is obviously slowed down, while the velocity of fluid particles flowing to both sides of the lighthouse is faster, which has a very obvious interaction effect.

Figure 8. Scene of waves crashing against the lighthouse with 53.2 k fluid particles: (a–c) the different stages of the lighthouse scene, which are selected from three different time steps; and (d–f) the velocity distributions of fluid particles. The time instances in (a–c) are the same as in (d–f), respectively. The color bars on the right side of (d–f) denote the modulus length of the velocity (velocity is a three-dimensional vector). The unit of velocity in (d–f) is m/s.

Table 2 lists the number of vertices and faces of rigid body objects including lighthouse, lifeboat, and tanker. It objectively shows the complexity of the models. In the initial stage, PDS is used to sample rigid body objects as single layer nonuniform boundary particles. PDS has good blue noise characteristics and is currently widely used in surface sampling. Table 2 also shows the number of boundary particles after sampling. Figure 9 shows the 3D mesh model before sampling (Figure 9a) and the particle model after sampling (Figure 9b,c). Another advantage of the unified model is that both complex rigid meshes and regular rigid meshes can be sampled based on PDS, that is, the algorithm is not sensitive to the complexity of the 3D meshes. The 3D meshes of different complexity only slightly affect the sampling time under certain conditions (Table 2). Since the deformation of rigid body is not considered, the sampling process only occurs in the initialization stage, thus it does not affect the solution speed of the physical model.
To verify the effectiveness of the unified boundary handling model in dealing with floating objects, the scenes of lifeboat entering water and tanker sailing were simulated. The detailed parameters of the two experimental scenes are shown in Figures 10 and 11. The boundary bounding box is a static boundary. The density of the sampled boundary particles is the rest density of the fluid $\rho_0$, and the velocity of the static boundary particles is zero. The lifeboat and tanker have floating boundaries. The density of the floating boundary particles is also $\rho_0$, whereas the velocity is no longer zero; in addition to the forward velocity of the lifeboat and tanker, it will also be subject to forces from fluid particles.

Figure 9. The three-dimensional mesh model of lifeboat (a); and the side view (b) and top view (c) of the boundary particle model involved in the fluid–solid coupling calculation. Single layer nonuniform particle boundary is based on PDS.

Figure 10. Design drawing of lifeboat entering water scene. The front and side view show the detailed parameters of the bounding box, lifeboat, and ocean. In the experiment, seawater (blue area) fills the bottom of the bounding box.

Figure 11. Design drawing of tanker sailing scene. The front and side view show the detailed parameters of the bounding box, tanker, and ocean. The gap between seawater and the bounding box is used to generate swell waves.
Figure 12 shows the interaction between lifeboat and ocean. In Figure 12, the lifeboat enters the water at an angle of 45 degrees to the calm ocean surface. We designed the specific experiment to simulate and record the entire entry process. Figure 12a–c represents the different stages of entry process of lifeboat, which are selected from three different time steps. Figure 12d–f presents the velocity distributions of fluid particles. When the lifeboat enters the water, the velocity field of fluid particles near the lifeboat changes due to interaction with the lifeboat. It can be seen in Figure 12d–f that the closer the fluid particles are to the lifeboat, the greater the change in the velocity field, which is consistent with the actual water entry process. The number of fluid particles in the scene is 312.8 k, and the number of lifeboat particles sampled using PDS is 5.1 k. PDS algorithm can adaptively sample the complex boundary without any additional operation. The sampled boundary particles can be directly added to the unified boundary handling model introduced in Section 2.3, and the interactive calculation can be completed automatically. Therefore, the unified boundary model is very suitable for industrial applications.

![Scene of lifeboat entering water](image)

**Figure 12.** Scene of lifeboat entering water with 312.8 k fluid particles. The lifeboat enters the water at an angle of 45 degrees to the calm ocean surface: (a–c) the different stages of entry process of lifeboat, which are selected from three different time steps; and (d–f) the velocity distributions of fluid particles. The time instances in (a–c) are the same as in (d–f), respectively. The color bars on the right sides of (d–f) denote the modules length of the velocity. The unit of velocity in (d–f) is m/s.

Figure 13 shows the tanker sailing scene. The images in Figure 13a–c are selected from three different time steps in the tanker sailing scene in Figure 13. In Figure 13a, the ship’s wake left on the sea surface after the tanker sailed is obvious. In Figure 13b, the bow of the tanker meets the swell wave, which results in the green water. In Figure 13c, the scene of the tanker passing through the swell wave is shown. The number of fluid particles in the scene is 312.8 k, and the number of boundary particles sampled using PDS is 4.4 k. The velocity distribution during the corresponding motion can be seen in Figure 13d–f, and the velocity of fluid particles around the tanker is relatively large because of their interaction with the sailing tanker, especially near the bow and stern. This is consistent with the real green water scene and the ship’s wake.
3.3. Performance of the Improved Anisotropic SSF

We applied the improved anisotropic SSF method to the dam break scene, and compared the rendering results with those of Yu’s work [33]. In the dam break scene in Figures 14 and 15, the number of fluid particles is 59.3 k. The experimental scene is the one shown in Figure 1. The size of the fluid block is \(2 \times L \times 2 \times L\).

**Figure 13.** Scene of tanker sailing in swell wave, with 312.8 k fluid particles: (a–c) selected three different time steps; and (d–f) velocity distributions of fluid particles. The time instances in (a–c) are the same as in (d–f), respectively. The color bars on the right sides of (d–f) denote the modules length of the velocity. The unit of velocity in (d–f) is m/s.

3.3. Performance of the Improved Anisotropic SSF

We applied the improved anisotropic SSF method to the dam break scene, and compared the rendering results with those of Yu’s work [33]. In the dam break scene in Figures 14 and 15, the number of fluid particles is 59.3 k. The experimental scene is the one shown in Figure 1. The size of the fluid block is \(2 \times L \times 2 \times L\).
The particles near the boundary region are sparsely distributed, which results in insufficient neighboring particles. This will affect the SVD result of the covariance matrix $\text{Cov}$ in Equation (32), leading to the deformation of particles. As shown in Figure 14a, although the shapes of most particles are acceptable, there are still some particles for which the scale of some axis $\vec{\sigma}_i$ is extremely large. Yu and Turk prevented particle deformation by setting the threshold $N_\varepsilon$ of neighboring particles [33] (Figure 14b). The scaling vector of fluid particles whose number of neighborhood particles is smaller than the threshold $N_\varepsilon$ was set to a constant isotropic value $\vec{\Sigma} = k_n I$, where $I$ is a unit matrix. However, Yu’s method has a limitation, which is obvious discontinuity in the boundary region with few fluid particles. There will be many artificial isotropic particles near the boundary region. Although particle deformation can be avoided, the visual effect of the fluid is rough, which affects the reality of the splashing scenes. Figure 14c shows the rendering result of the improved anisotropic particle deformation correction method. As can be seen in Figure 14a, except for a few deformed particles, the shapes of most other particles are normal and there is no unacceptable deformation. Therefore, in Yu’s method, once the threshold $N_\varepsilon$ of the number of the neighboring particles is set unreasonably, the shapes of many normal particles will also be artificially changed.

The improved method is to set the threshold of the number of the neighboring particles as a piecewise function to maintain the continuity of the fluid particle shape. Since it is difficult to quantitatively give the conditions under which particle deformation occurs, the fluid particles should be kept as anisotropic as possible within each neighborhood segment. Experiment results have shown that not all axial scaling of all deformed particles is anomalous, thus it is only necessary to control the axial scaling in which a deformation occurs. Therefore, our method considers the contribution of the original scaling when we correct the deformed particles. This can be achieved by adjusting the value of $l$ in Equation (35). To maintain the anisotropy, while retaining the original scaling contribution, the min function is used to correct the abnormal scaling beyond the threshold. By processing each axis separately, the particle shape near the boundary region has been greatly improved. To maintain
the continuity of the shapes of the fluid particles with few neighboring particles and simulate the fine details of splashing, the min function is used to directly correct the scaling of each axis, and the final results are shown in Figure 15. Figure 16 shows the final shading effect of the fluid particles in Figure 15 using OpenGL.

![Figure 15](image1)

![Figure 16](image2)

**Figure 15.** The rendering results of the improved method from different time steps of the experimental dam break scene: (a, b) the isotropic particle rendering results; and (c, d) the improved anisotropic particle rendering results. Obviously, in the edge region, the images in (c, d) have sharp features.

**Figure 16.** Shading effect of dam break scene using the improved anisotropic screen space fluids technique. The shading effect is implemented using OpenGL. (a–d) Images selected from four different time steps in a dam break scene.

### 4. Conclusions and Future Work

In this paper, we propose a novel hybrid SPH simulation method for ocean scene in marine simulator. Compared to the state-of-the-art methods, the combination of divergence-free condition and position-based density constraint leads to a faster convergence speed, stronger stability, and better incompressibility. However, the scope of scene simulated by our model is limited, and our method is more suitable for simulating large deformation scenes such as splashes, as well as some special effects of scene of ocean. When developing marine simulator, our model needs to be solved in parallel to ensure the real-time performance. In the development of interactive scenes, our unified boundary handling model is more flexible than the existing methods in the marine simulator. It can deal with complex boundary adaptively. In the interactive scene of ship and ocean, our current model is somewhat simple. In the future, ship mathematical motion models such as Maneuvering Modeling Group (MMG) can be coupled with our model to simulate more realistic ship motion. The improved anisotropic screen space fluid rendering method alleviates the discontinuity of fluid particle shape near the boundary region. It maintains sharper features and produces smoother surfaces compared to the state-of-the-art work in [33]. In this paper, we use OpenGL to implement the improved SSF method. In the development of marine simulator, developers need to make corresponding changes according to the rendering engine used by the simulator.

**Author Contributions:** H.L. led the writing of the manuscript, contributed to data analysis and research design and served as the corresponding author. H.R. supervised this study, contributed to the funding acquisition, and served as the corresponding author. S.Q. contributed to investigation. C.W. contributed to investigation and implementation of part of the code. All authors have read and agreed to the published version of the manuscript.
Funding: This research was funded by the National High Technology Research and Development Program of China (863 Program), grant number 2015AA016404; by the Ministry of Transport Applied Basic Research Funding Project, grant number 2015329225240; and by the Key scientific research and cultivation project of Dalian Maritime University, grant number 017190129.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Calculation Flow of the Divergence-Free Solver

Algorithm A1 shows the calculation flow of the divergence-free solver. The first step in the divergence-free solver loop is to compute density change rate \( \frac{D\rho}{Dt} \) according to Equation (11), and then the divergence-free condition is iteratively solved. We improve the stopping criteria for the iterative solver by adding the average density change \( \rho_{err} \). This improvement enables our solver to stop the iteration early when the average density change \( \rho_{err} \) is smaller than the density change threshold \( \eta_v \), thus speeding up the calculation. In each iteration step, according to Equation (12), the stiffness parameter \( \kappa_i \) of each particle is updated with the calculated density change rate \( \frac{D\rho_i}{Dt} \), and then \( \kappa_i \) is substituted into Equation (14) to obtain the velocity change \( \Delta v_i \) and update the velocity \( v_i \). When the velocity \( v_i \) is updated, according to Equation (11), the density change rate \( \frac{D\rho_i}{Dt} \) of the fluid is updated again. At this time, the average density change \( \rho_{err} \) of the current fluid particles can be updated by the new density change rate \( \frac{D\rho_i}{Dt} \). Loop continues until the average density change \( \rho_{err} \) is smaller than the threshold \( \eta_v \), or the number of iteration steps \( iter \) is greater than the threshold \( solver_iterations \).

Algorithm A1 Divergence-Free Solver

01: compute density substantial derivative: \( \frac{D\rho}{Dt} \)
02: while \( \rho_{err} > \eta_v \) and \( iter < solver_iterations \):
03:   for \( i \) in particles:
04:      compute stiffness parameter \( \kappa_i \) and \( \kappa_j \)
05:      update velocity \( v_i \leftarrow v_i + \Delta v_i \)
06:   for \( i \) in particles:
07:      update density substantial derivative: \( \frac{D\rho}{Dt} \)
08:      update \( \rho_{err} \) and \( iter \)

Appendix B. Calculation Flow of the Density Constraint Solver

Algorithm A2 shows the calculation flow of the density constraint solver. We use the same iterative stopping criteria as used in divergence-free solver. In each iteration step, the particle density \( \rho_i \) needs to be calculated first, which is then used to update the constraint \( C_i \) according to Equation (15). If \( C_i \) is greater than zero, it means that the particle distribution is too dense, and then we need to update the scaling factor \( \lambda_i \) using \( C_i \). The position correction \( \Delta x_i \) of each particle can be calculated by Equation (21). The particle position \( x_i \) can be updated using \( \Delta x_i \). At the end of the iteration step, the average density error \( \rho_{err} \) is updated according to the particle density \( \rho_i \). In each iteration step, the particle position \( x_i \) is updated and the distribution of the particles changes. Therefore, in the next iteration step, the particle density \( \rho_i \) needs to be updated first. The loop continues until the density error \( \rho_{err} \) is smaller than the threshold \( \eta_c \), or the number of iteration steps \( iter \) is greater than the threshold \( solver_iterations \).
Algorithm A2. Density Constraint Solver.

01: while $\rho_{\text{err}} > \eta_c$ and $\text{iter} < \text{solver\_iterations}$:
02:   for $i$ in particles:
03:     compute density $\rho_i$
04:     compute constraint $C_i$
05:     if $C_i > 0$:
06:        compute $\lambda_i$
07:   for $i$ in particles:
08:     compute $\Delta x_i$
09:   for $i$ in particles:
10:     update position $x_i \leftarrow x_i + \Delta x_i$
11: update $\rho_{\text{err}}$ and $\text{iter}$

References

1. Tessendorf, J. Simulating Ocean Water; 2004 Course Notes; ACM SIGGRAPH: Los Angeles, CA, USA, 2004.
2. Horvath, C.J. Empirical directional wave spectra for computer graphics. In Proceedings of the 2015 Symposium on Digital Production, Los Angeles, CA, USA, 8 August 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 29–39.
3. Fréchot, J. Realistic simulation of ocean surface using wave spectra. In Proceedings of the First International Conference on Computer Graphics Theory and Applications, Setúbal, Portugal, 25–28 February 2006; Springer: Berlin/Heidelberg, Germany, 2007; pp. 76–83.
4. Lee, N.; Baek, N.; Ryu, K.W. A real-time method for ocean surface simulation using the TMA model. Int. J. Comput. Inf. Syst. Ind. Manag. Appl. 2009, 1, 25–26.
5. Hopper, R.; Wolter, K. The Water Effects of Pirates of the Caribbean: Dead Men Tell no Tales; 2017 Talks; ACM SIGGRAPH: Los Angeles, CA, USA, 2017.
6. Foster, N.; Metaxas, D. Realistic animation of liquids. Graph. Models Image Process. 1996, 58, 471–483. [CrossRef]
7. Irving, G.; Guendelman, E.; Losasso, F.; Fedkiw, R. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. ACM Trans. Graph. 2006, 25, 805–811. [CrossRef]
8. Chentanez, N.; Müller, M. Real-time Eulerian water simulation using a restricted tall cell grid. ACM Trans. Graph. 2011, 30, 82. [CrossRef]
9. Chentanez, N.; Feldman, B.E.; Labelle, F.; O’Brien, J.F.; Shewchuk, J.R. Liquid simulation on lattice-based tetrahedral meshes. In Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, San Diego, CA, USA, 2–4 August 2007; Eurographics Association: Goslar, Germany, 2007; pp. 219–228.
10. Alfonsi, G.; Lauria, A.; Primavera, L. Proper orthogonal flow modes in the viscous-fluid wave-diffraction case. J. Flow Vis. Image Process. 2013, 20, 227–241. [CrossRef]
11. Liu, G.R.; Liu, M.B. Smoothed Particle Hydrodynamics: A Meshfree Particle Method; World Scientific Publishing Company: Singapore, 2003.
12. Ihmsen, M.; Akinci, N.; Becker, M.; Teschner, M. A parallel SPH implementation on multi-core CPUs. Comput. Graph. Forum 2011, 30, 99–112. [CrossRef]
13. Akinci, N.; Akinci, G.; Teschner, M. Versatile surface tension and adhesion for SPH fluids. ACM Trans. Graph. 2013, 32, 182. [CrossRef]
14. Bender, J.; Koschier, D.; Kugelstadt, T.; Weiler, M. A micropolar material model for turbulent SPH fluids. In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Los Angeles, CA, USA, 28–30 July 2017; Association for Computing Machinery: New York, NY, USA, 2017; p. 4.
15. Bender, J.; Koschier, D.; Kugelstadt, T.; Weiler, M. Turbulent Micropolar SPH fluids with foam. IEEE Trans. Vis. Comput. Graph. 2018, 25, 2284–2295. [CrossRef]
16. Ren, B.; Li, C.; Yan, X.; Lin, M.C.; Bonet, J.; Hu, S.M. Multiple-fluid SPH simulation using a mixture model. ACM Trans. Graph. 2014, 33, 171. [CrossRef]
17. Yan, X.; Jiang, Y.T.; Li, C.F.; Martin, R.R.; Hu, S.M. Multiphase SPH simulation for interactive fluids and solids. ACM Trans. Graph. 2016, 35, 79. [CrossRef]
18. Yang, T.; Chang, J.; Lin, M.C.; Martin, R.R.; Zhang, J.J.; Hu, S.M. A unified particle system framework for multi-phase, multi-material visual simulations. *ACM Trans. Graph.* 2017, 36, 224. [CrossRef]

19. Ladicky, L.; Jeong, S.H.; Solenthaler, B.; Pollefeys, M.; Gross, M. Data-driven fluid simulations using regression forests. *ACM Trans. Graph.* 2015, 34, 1–9. [CrossRef]

20. Um, K.; Hu, X.; Thuerey, N. Liquid splash modeling with neural networks. *Comput. Graph. Forum* 2018, 37, 171–182. [CrossRef]

21. Schenck, C.; Fox, D. Spnets: Differentiable fluid dynamics for deep neural networks. In Proceedings of the 2nd Conference on Robot Learning, Zürich, Switzerland, 29–31 October 2018; Microtome Publishing: Brookline, MA, USA, 2018. Available online: https://arxiv.org/pdf/1806.06094.pdf (accessed on 11 January 2020).

22. Akinci, N.; Ihmsen, M.; Akinci, G.; Solenthaler, B.; Teschner, M. Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans. Graph.* 2012, 31, 62. [CrossRef]

23. Becker, M.; Teschner, M. Weakly compressible SPH for free surface flows. In Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, San Diego, CA, USA, 2–4 August 2007; Eurographics Association: Goslar, Germany, 2007; pp. 209–218.

24. Solenthaler, B.; Pajarola, R. Predictive-corrective incompressible SPH. *ACM Trans. Graph.* 2009, 28, 40. [CrossRef]

25. He, X.; Liu, N.; Li, S.; Wang, H.; Wang, G. Local poisson SPH for viscous incompressible fluids. *Comput. Graph. Forum* 2012, 31, 1948–1958. [CrossRef]

26. Macklin, M.; Müller, M. Position based fluids. *ACM Trans. Graph.* 2013, 32, 104. [CrossRef]

27. Ihmsen, M.; Cornelis, J.; Solenthaler, B.; Horvath, C.; Teschner, M. Implicit incompressible SPH. *IEEE Trans. Vis. Comput. Graph.* 2013, 20, 426–435. [CrossRef]

28. Bender, J.; Koschier, D. Divergence-free smoothed particle hydrodynamics. In Proceedings of the 14th ACM SIGGRAPH/Eurographics symposium on computer animation, Los Angeles, CA, USA, 7–9 August 2015; Eurographics Association: Goslar, Germany, 2015; pp. 147–155.

29. Müller, M.; Charypar, D.; Gross, M. Particle-based fluid simulation for interactive applications. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, San Diego, CA, USA, 26–27 July 2003; Eurographics Association: Goslar, Germany, 2003; pp. 154–159.

30. Chentanez, N.; Müller, M.; Macklin, M.; Kim, T.K. Fast grid-free surface tracking. *ACM Trans. Graph.* 2015, 34, 148. [CrossRef]

31. Van der Laan, W.J.; Green, S.; Sainz, M. Screen space fluid rendering with curvature flow. In Proceedings of the 2009 symposium on Interactive 3D graphics and games, Boston, MA, USA, 27 February–1 March 2009; Association for Computing Machinery: New York, NY, USA, 2009; pp. 91–98.

32. Green, S. Screen space fluid rendering for games. In Proceedings of the Game Developers Conference, San Francisco, CA, USA, 9–13 March 2010.

33. Yu, J.; Turk, G. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Trans. Graph.* 2013, 32, 5. [CrossRef]

34. Truong, N.; Yuksel, C. A narrow-range filter for screen-space fluid rendering. *Proc. ACM Comput. Graph. Interact. Tech.* 2018, 1, 1. [CrossRef]

35. Boyd, L.; Bridson, R. MultiFLIP for energetic two-phase fluid simulation. *ACM Trans. Graph.* 2012, 31, 16. [CrossRef]

36. Ferstl, F.; Ando, R.; Wojtan, C.; Westermann, R.; Thuerey, N. Narrow band flip for liquid simulations. *Comput. Graph. Forum* 2016, 35, 225–232. [CrossRef]

37. Sato, T.; Wojtan, C.; Thuerey, N.; Igarashi, T.; Ando, R. Extended narrow band FLIP for liquid simulations. *Comput. Graph. Forum* 2018, 37, 169–177. [CrossRef]

38. Guo, Y.; Liu, X.; Xu, X. A unified detail-preserving liquid simulation by two-phase lattice Boltzmann modeling. *IEEE Trans. Vis. Comput. Graph.* 2016, 23, 1479–1491. [CrossRef] [PubMed]

39. Lee, M.; Hyde, D.; Li, K.; Fedkiw, R. A robust volume conserving method for character-water interaction. In Proceedings of the 2019 ACM SIGGRAPH/Eurographics symposium on computer animation, Los Angeles, CA, USA, 26–28 July 2019; Eurographics Association: Goslar, Germany, 2019; p. 3. [CrossRef]

40. Raveendran, K.; Wojtan, C.; Turk, G. Hybrid smoothed particle hydrodynamics. In Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation, Vancouver, BC, Canada, 5–7 August 2011; Association for Computing Machinery: New York, NY, USA, 2011; pp. 33–42.
41. Roy, B.; Poulin, P. A hybrid Eulerian-DFSPH scheme for efficient surface band liquid simulation. *Comput. Graph.* 2018, 77, 194–204. [CrossRef]

42. Kass, M.; Miller, G. Rapid, stable fluid dynamics for computer graphics. In Proceedings of the ACM Siggraph Computer Graphics, Dallas, TX, USA, 6–10 August 1990; Association for Computing Machinery: New York, NY, USA, 1991; pp. 49–57.

43. Jeschke, S.; Skřivan, T.; Müller, M.; Chentanez, N.; Macklin, M.; Wojtan, C. Water surface wavelets. *ACM Trans. Graph.* 2018, 37, 94. [CrossRef]

44. Ihmsen, M.; Orthmann, J.; Solenthaler, B.; Kolb, A.; Teschner, M. *SPH Fluids in Computer Graphics*; The Eurographics Association: Strasbourg, France, 2014.

45. Bowers, J.; Wang, R.; Wei, L.Y.; Maletz, D. Parallel poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph.* 2010, 29, 166. [CrossRef]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).