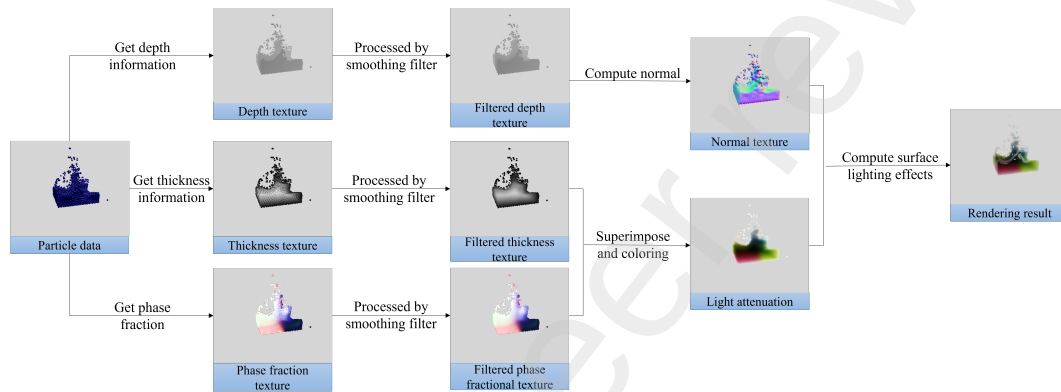


Graphical Abstract

Real-Time Screen Space Rendering Method for Particle-Based Multiphase Fluid Simulation

Yalan Zhang, Yuhang Xu, Yanrui Xu, Yue Hou, Xiaokun Wang, Guo Yu, Mohammad S. Obaidat, Life Fellow of IEEE, Fellow of AAIA, and Fellow of SCS, Xiaojuan Ban



Highlights

Real-Time Screen Space Rendering Method for Particle-Based Multiphase Fluid Simulation

Yalan Zhang, Yuhang Xu, Yanrui Xu, Yue Hou, Xiaokun Wang, Guo Yu, Mohammad S. Obaidat, Life Fellow of IEEE, Fellow of AAIA, and Fellow of SCS, Xiaojuan Ban

- Novel method for real-time rendering of multiphase flows, addressing miscible fluid challenges.
- Achieves rapid rendering of complex fluid dynamics with high computational efficiency.
- Outperforms existing methods for rendering immiscible multiphase fluids.

Real-Time Screen Space Rendering Method for Particle-Based Multiphase Fluid Simulation

Yalan Zhang^{a,b}, Yuhang Xu^{a,b}, Yanrui Xu^a, Yue Hou^a, Xiaokun Wang^{a,b},
Guo Yu^{a,b}, Mohammad S. Obaidat, Life Fellow of IEEE, Fellow of AAIA,
and Fellow of SCS^{c,1}, Xiaojuan Ban^{a,b,*}

^a*School of Intelligence Science and Technology, University of Science and Technology
Beijing, China*

^b*Shunde Innovation School, University of Science and Technology Beijing, Guangdong,
China*

^c*Chair and Professor, Computer Science Department, and Director of Cybersecurity
Center, University of Texas-Permian Basin, 4901 E. University Blvd., Odessa, TX
79762, USA*

^d*The King Abdullah II School of Information Technology, University of Jordan, Amman
11942, Jordan*

Abstract

Existing fluid simulation techniques mainly process single-phase fluids, and they have difficulties in accurately simulating and visualizing multiphase fluid dynamics. This paper proposes a new method for the real-time rendering of multiphase fluid simulations, which uses smoothed particle hydrodynamics in screen space. Meanwhile, the method employs phase fraction textures to differentiate various materials in multiphase fluid simulations, thereby portraying mixing and separation effects more realistically. Besides, efficient texture computation allows it to be integrated seamlessly into real-time simulation rendering workflows. Extensive testing confirms the effectiveness of the proposed method in rendering multiphase fluid behaviors with high visual fidelity and demonstrates its capability to process frames within 0.01 seconds, even in cases with up to 300,000 particles. This study enhances the

*Corresponding author.

Email addresses: zhangyl@ustb.edu.cn (Yalan Zhang), 18600250511@163.com (Yuhang Xu), xuyanruiedw@me.com (Yanrui Xu), 1586723707@qq.com (Yue Hou), wangxiaokun@ustb.edu.cn (Xiaokun Wang), guoyu@ustb.edu.cn (Guo Yu), m.s.obaidat@ieee.org (Mohammad S. Obaidat, Life Fellow of IEEE, Fellow of AAIA, and Fellow of SCS), banxj@ustb.edu.cn (Xiaojuan Ban)

Preprint submitted to Simulation Modelling Practice and Theory

February 7, 2024

fluid dynamics simulation field and provides a more accurate and efficient method for visualizing complex multiphase fluids in simulations.

Keywords: Visual Simulation, Real-Time Rendering, Multiphase Fluid, Screen Space Rendering

1. Introduction

Fluid dynamics simulations play a crucial role in various scientific and engineering applications, from environmental modeling to industrial processes(1). Traditionally, fluid simulation techniques mainly focused on single-phase fluids, which are beneficial but have limitations when it comes to more complex scenarios involving multiphase fluids (2; 3; 4). Characterized by the interaction of fluids with distinct properties, multiphase fluid dynamics bring about unique challenges in accurate simulation and visualization.

The emergence of smoothed particle hydrodynamics (SPH) has revolutionized the field of fluid dynamics simulations as it provides a more granular and interactive approach to modeling fluid behavior. However, the application of SPH in multiphase fluid simulations, especially in real-time contexts, is less explored and demands innovation(5). In the field of fluid rendering, three predominant methods have emerged: mesh-based, volume rendering, and screen-space methods. Mesh-based methods excel in delivering precise surface representations, essential for accurately modeling phase interactions in multiphase fluid simulations. However, they face significant computational and memory challenges, especially in complex scenarios, and tend to be less effective in miscible flows where distinct phase boundaries are absent. Volume rendering methods, capable of effectively visualizing complex interactions in multiphase fluid simulations, are constrained by their limited real-time performance due to high computational demands(6). Consequently, in dynamic and time-sensitive contexts, more efficient screen-space methods are often preferred, as they can provide the necessary real-time rendering capabilities, especially in simulations with numerous particles, though they require iterative refinement to achieve high visual fidelity. Despite these advancements, applying these methods to miscible multiphase fluid simulations remains a challenge, as current techniques often fail to fully capture the intricate interactions and visual subtleties of miscible fluids(7), highlighting a critical area for further research and development in the simulation field.

This paper introduces a novel method for real-time rendering of multiphase fluid simulations that exploits the capabilities of SPH in a screen-space context. The core of this approach is the use of phase fraction textures to differentiate various materials within a multiphase fluid system. This makes it possible to portray complex phenomena such as mixing and separation effects more realistically, which are central to the dynamics of multiphase fluids. One focus of this research is the efficiency of texture computation, which is critical for integrating this method into real-time simulation rendering workflows. The efficiency and scalability of the proposed method are demonstrated by its ability to process frames within 0.01 seconds, even in scenarios involving up to 300,000 particles.

Extensive testing confirms the effectiveness of the proposed method in rendering multiphase fluid behaviors with high visual fidelity. The advancements in this study significantly enhance the field of fluid dynamics simulations, providing a more accurate and efficient method for visualizing the intricate and dynamic nature of complex multiphase fluids.

The following sections introduce in detail the methodology, implementation, and evaluation of the proposed real-time rendering method, which sets a new benchmark in the field of multiphase fluid simulation.

2. Related work

2.1. Mesh-based methods

In particle-based multiphase fluid simulations, mesh-based methods are crucial for extracting and accurately reconstructing fluid surfaces directly from particle positions. These methods mainly employ algorithms like Marching Cubes (MC) to generate a smooth polygonal surface mesh. The fluid surface is usually represented implicitly by the zero level-set of a signed distance field, which is derived from kernel evaluations of particle distances.

In this context, various kernel types are utilized to improve the accuracy and efficiency of mesh construction. For instance, isotropic kernels (8) (9), adaptive size kernels (10), and anisotropic SPH kernels (11) have played an important role in improving the realism and performance of rendered fluid surfaces. These methods are crucial in multiphase flows, where precise surface representation is the key to accurate phase interaction modeling. Solenthaler et al. (12) proposed a novel surface reconstruction algorithm based on particle center movement, which is particularly effective for fluid-solid interaction simulations. This approach minimizes reconstruction errors in

concave surfaces and has a significant impact on the efficiency improvement of mesh-based methods. Additionally, Kazhdan and Hoppe(13) further optimized these reconstruction methods with a focus on enhancing edge precision and sharpness as they are crucial for accurately depicting fluid interfaces.

Mesh-based methods are advantageous in providing precise surface representations in multiphase fluid simulations, and this is crucial for the accurate modeling of phase interactions. However, these methods have significant limitations related to computational and memory demands, particularly in complex scenarios. Despite their capability to capture the dynamics of distinct fluid phases, they are less effective in handling miscible flows, where clear phase boundaries are not present. Therefore, for dynamic, time-sensitive scenarios, more efficient screen-space methods are usually preferred (14).

2.2. Volume rendering methods

Volume rendering methods adopt GPU-based techniques, and they help to render liquid surfaces from scalar fields defined by fluid particles. These methods are particularly suitable for multiphase fluid simulations, enabling the accurate depiction of complex fluid interactions. However, their large computational demand poses a significant challenge. To address this issue, existing studies focus on optimizing the efficiency of these techniques. Noteworthy advancements in volume rendering methods include the adaptive discretization of the view volume by Fraedrich et al. (15), the compression of the perspective grid by Zirr et al. (16), the implementation of ray-casting in narrow bands near boundary particles by Goswami et al.(17), along with subsequent refinements by Xiao et al. (18) and Biedert et al.(19), who further improved the technique for higher accuracy in multiphase fluid simulations.

Recently, Kawamura et al. (20) investigated the use of an interactive in situ steering framework for large-scale computational fluid dynamics (CFD) simulations on GPU supercomputers, including in situ particle-based volume rendering (PBVR). When it is applied to real-time plume dispersion analysis, this framework demonstrates the effectiveness of the human-in-the-loop approach in complex simulations. Another notable advancement by Valentine and Mukundan (21) is the use of OpenGL-4 shaders for parallel processing on the GPU in a fluid rendering pipeline. Their approach utilizes adaptive tessellation, polygon culling with the geometry shader, and surface lighting with the Phong-Blinn model to optimize the visualization of fluid surface flow, thereby enhancing both the framerate and capacity of the simulation to handle more primitives.

In the specific context of multiphase fluid simulations, though volume rendering can effectively visualize complex interactions, it has limited real-time performance due to high computational demands. Consequently, in dynamic, time-sensitive scenarios, more efficient screen-space methods are usually the preferred choice (6).

2.3. Screen-space methods

Screen-space methods are an indispensable part of real-time rendering, and they are particularly effective in particle-based multiphase fluid simulations within 2D image space. These methods utilize a smoothed depth buffer that is derived from the visible liquid surfaces defined by particles to facilitate various modes of surface representation. Also, Marching Squares-based triangle meshes or rasterization techniques are used in these methods. Since the rendering quality of liquid surfaces highly depends on the image-based filtering process of the depth buffer, key enhancements, including binomial filters by Cords and Staadt (22), curvature flow filters by Van der Laan et al. (23), bilateral Gaussian filters by Bagar et al. (24), and narrow-range filters by Truong and Yuksel (25), have been made to significantly improve rendering outcomes.

The foundational work of Müller et al. (26) in screen-space methods, particularly in generating and rendering surfaces from three-dimensional point clouds, indicated a significant shift towards efficiency in this field. Subsequent studies include the implementation of narrow-range filters by Truong and Yuksel (25) and overcoming specific rendering challenges such as the refinement of jagged edges for more precise fluid interface depiction. Further advancements by Xu et al. (5) in anisotropic screen space rendering and the introduction of a narrow-band filtering technique by Oliveira and Paiva (27) to achieve higher liquid surface smoothness further improved the capabilities of these methods. Moreover, the contribution by Wu et al. (28), involving the development of an adaptive ray marching method, has greatly augmented the rendering of fluid surfaces in complex simulation scenarios. Additionally, Brito et al. (7) made a notable contribution to the simulation of multiphase fluids by developing an SPH formulation with a shader-based rendering solution, which can efficiently handle high particle counts. However, it is noteworthy that their method focused exclusively on non-miscible fluids, thus highlighting the existing limitations in effectively simulating and rendering miscible multiphase fluids.

Screen-space methods in real-time rendering, especially for dynamic fluid simulations with numerous particles, need to perform iterative refinement to achieve the required visual fidelity. Despite advancements in rendering fluid surfaces, it is still challenging to apply these methods to miscible multiphase fluid simulations. Current techniques usually cannot fully capture the complex interactions and visual nuances of miscible fluids, and this is an important area that needs to be further developed in the simulation field. This paper addresses this gap by developing a new method for real-time screen space rendering, specifically for particle-based miscible multiphase fluid simulations. This method aims to improve both the simulation accuracy and efficiency in such intricate scenarios, contributing to the advancement of simulation techniques in the realm of fluid dynamics.

2.4. Recent developments and diverse applications

Recent developments in screen-space fluid rendering are characterized by a series of novel methods. For instance, Choi et al. (29) utilized Conditional Generative Adversarial Networks (cGAN) to generate refined normal maps to enhance visual realism in fluid rendering. This approach represents the ongoing effort to integrate advanced machine learning techniques with traditional rendering methods, thus improving the overall quality and detail of simulated fluid surfaces. Chan et al. (30) developed a novel method that integrates SPH with Finite Element Method (FEM) models. This approach was tailored for real-time simulations involving interactions between fluids and deformable bodies, and its core is a two-way coupling strategy that employs ray-traced collision detection and screen space refraction rendering. The approach not only improved the refraction effects in the simulations but also demonstrated high efficiency in handling complex scenarios with fluid-body interactions. Neto et al. (31) introduced new techniques to generate foam and droplets in fluid simulations, while Wang et al. (32) focused on adaptive surface smoothing to achieve realistic splash effects. Additionally, Xiao et al. (33) combined particle splatting, ray-casting, and surface normal estimation to produce high-quality fluid surfaces while preserving thin and sharp details in large-scale fluids.

3. Particle-based multiphase fluid simulation

Rendering the complex interactions between multiple fluid phases necessitates precise representation of the different fluid compositions. This re-

quirement originates from the simulation domain, leveraging the Lagrangian volume fraction scheme (34). This scheme is adept at modeling the simultaneous presence of various phases within a singular fluid parcel, establishing a foundation for numerous advanced multiphase simulation methodologies (2; 3; 35; 4). Its universal applicability and adaptability have made it a cornerstone for state-of-the-art developments in this field.

In the formulation of our screen-space multiphase fluid rendering approach, we have integrated this scheme as a fundamental component. This integration facilitates the accurate and detailed visualization of fluid interactions, capturing the nuanced dynamics of multiphase flows. Our method builds on this established groundwork, offering novel contributions to the rendering techniques that enable realistic and computationally efficient depictions of complex fluid phenomena.

For k fluid phases, the conservation of volume is paramount, with the volume fraction α_k at location $\mathbf{x}$ constrained by:

$$\sum_k \alpha_k(\mathbf{x}) = 1 \quad (1)$$

The velocity field $\mathbf{u}_k$ for phase k contributes to the mixture velocity $\mathbf{u}_m$, calculated as:

$$\mathbf{u}_m = \sum_k \alpha_k \mathbf{u}_k \quad (2)$$

which must satisfy the incompressibility condition $\nabla \cdot \mathbf{u}_m = 0$. The drift velocity $\mathbf{u}_{uk}$ is then given by:

$$\mathbf{u}_{uk} = \mathbf{u}_k - \mathbf{u}_m \quad (3)$$

enforcing the condition $\sum_k \alpha_k \mathbf{u}_{uk} = 0$. The mixture density combines the individual densities ρ_k :

$$\rho_m = \sum_k \alpha_k \rho_k \quad (4)$$

Mass conservation across phases is governed by the continuity equation:

$$\frac{\partial}{\partial t} (\alpha_k \rho_k) + \nabla \cdot (\alpha_k \rho_k \mathbf{u}_k) = 0 \quad (5)$$

and the rate of volume fraction change is dictated by the drift velocity:

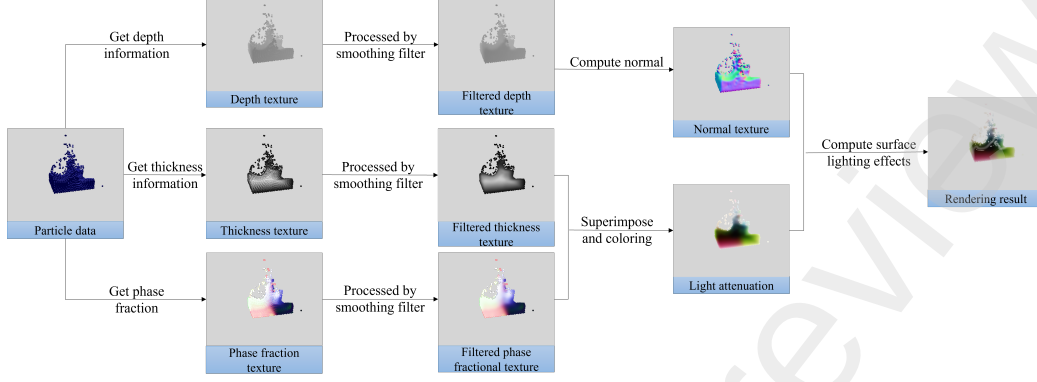


Figure 1: The procedure of the surface rendering algorithm for multiphase fluid simulation.

$$\frac{D}{Dt}\alpha_k = -\nabla \cdot (\alpha_k \mathbf{u}_{uk}) \quad (6)$$

The momentum equation for each phase, crucial for modeling the forces within the fluid, is expressed as:

$$\begin{aligned} \frac{\partial}{\partial t} (\alpha_k \rho_k \mathbf{u}_k) + \nabla \cdot (\alpha_k \rho_k \mathbf{u}_k \mathbf{u}_k) = \\ -\alpha_k \nabla p_k + \nabla \cdot (\alpha_k \boldsymbol{\tau}_k) + \alpha_k \rho_k \mathbf{g} + \mathbf{M}_k^I \end{aligned} \quad (7)$$

where the terms represent the pressure gradient, viscous stresses, gravitational forces, and interphase momentum exchange, respectively.

This advanced simulation framework thus provides a comprehensive tool for visualizing and analyzing the behavior of multiphase fluids in a variety of dynamic scenarios.

4. Multiphase screen space rendering

Fluid rendering in the screen space is conducted by drawing the 3D fluid directly into the 2D screen without generating surface meshes and is closely related to image processing algorithms (5). The procedure of the surface rendering algorithm for multiphase fluid simulation is shown in Fig. 1, and its details are described in the following.

4.1. Screen-space fluid rendering

Screen-space fluid rendering is a technique that estimates the fluid surface by projecting world-space particles onto the screen space. It reconstructs the fluid surface using the particles closest to the camera, as determined through depth testing. In this process, a depth texture is created and then smoothed to achieve a seamless fluid surface appearance. Meanwhile, a thickness texture is employed to represent the volume of the fluid that absorbs light. The screen-space rendering pipeline consists of several key steps, which are described as follows.

The depth texture is created by rasterizing particles that are represented as spheres with a radius r using point sprites. These point sprites are quadrilaterals oriented towards the camera. The depth texture obtains the depth information of the particles in camera space for each pixel on the screen. In this representation, pixels associated with particles closer to the camera have darker colors, while those farther away from the camera have lighter colors.

The initial depth texture, which represents the liquid surface via spheres, may have a spotty appearance due to the intersections of these spheres. To obtain a smooth and continuous surface, the depth texture is processed to minimize sharp variations in depth between particles. Existing methods, such as bilateral Gaussian filtering (which was proposed in 2010) and narrow-range filtering (which was developed specifically for screen-space rendering pipelines in 2018), can be utilized to smooth the surface while preserving the distinction between different fluid layers. This paper employs the narrow-range filtering approach, and the formula for calculating the filtered depth value is given by:

$$z'_i = \frac{\sum_j w_{ij} f(z_i, z_j)}{\sum_j w_{ij}} \quad (8)$$

where i denotes the position of the pixel currently being filtered, j denotes the coordinates of the sampled pixel within the filter kernel, z'_i denotes the depth value of the post-filter, w_{ij} denotes the weight of pixel j , z_i denotes the depth value of the pre-filter, and z_j denotes the depth value of the sampled pixel.

When filtering depth values within the kernel, adjustments are made based on the sample depth value z_j . For smaller sample depth values z_j , the depth value of the sample pixel is adjusted as follows:

$$f(z_i, z_j) = \begin{cases} z_j, & \text{if } z_j > z_i - \delta \\ z_i - \mu, & \text{otherwise} \end{cases} \quad (9)$$

where μ and δ are predefined parameters. δ controls the narrow range size, and μ adjusts the depth value manually to affect the curvature at the edge points of splash particles.

For larger sample depth values z_j , the weight of the sample pixel w_{ij} is modified as follows:

$$w_{ij} = \begin{cases} 0, & \text{if } z_j > z_i + \delta \text{ or } z_k > z_i + \delta \\ G(i, j, \sigma_i), & \text{otherwise} \end{cases} \quad (10)$$

where k represents the pixel in the filter kernel that is symmetrical to the sampled pixel j relative to the filtered pixel i . G denotes the chosen smoothing function, which can be a mean blur or a Gaussian function. σ_i represents the standard deviation of the Gaussian function, and it relies on the current filtered pixel i . Given the requirement of a uniform smoothing kernel size in world space, a unified standard deviation σ is set. From the perspective of the camera, the standard deviation for each pixel is dynamically computed based on the depth value of the filtered pixel i and the particle radius R :

$$\sigma_i = \left\lceil \frac{HR\sigma}{2|z_i|\tan(\alpha/2)} \right\rceil \quad (11)$$

where H denotes the number of pixels in the screen's height direction, and α represents the field of view angle of the camera.

Once the depth texture is smoothed, a three-dimensional smooth fluid surface is reconstructed through reverse projection transformation. To realize accurate shading, it is necessary to calculate the normal vector at each point. The surface normal is derived from the partial derivatives of the depth texture. For a point on the depth texture (u, v) , which is transformed into coordinates $p(u, v) = (x, y, z)^T$ in camera space, the normal vector at this point is calculated as the cross-product of the differentials:

$$\mathbf{n}_{uv} = \left(\frac{\partial \mathbf{P}}{\partial u} \right) \times \left(\frac{\partial \mathbf{P}}{\partial v} \right) \quad (12)$$

The partial derivatives are computed using the common first-order difference method:

$$\frac{\partial \mathbf{P}}{\partial u} = \frac{\mathbf{P}(u + \Delta u, v) - \mathbf{P}(u, v)}{\Delta u} \quad (13)$$

$$\frac{\partial \mathbf{P}}{\partial v} = \frac{\mathbf{P}(u, v + \Delta v) - \mathbf{P}(u, v)}{\Delta v} \quad (14)$$

The normal vector $\mathbf{n}_{uv}$ calculated for each pixel corresponding to a point in space is stored in the normal texture. The accompanying figure demonstrates the visualization of these normal vectors.

In screen-space rendering, representing light absorption by the fluid volume is crucial for obtaining visually realistic images. This is accomplished using a thickness texture, which quantifies the cumulative length of particles that overlap with a pixel. The thickness of an individual particle is defined as the length of the particle intercepted by the pixel. When obtaining the thickness texture, particles are rendered as point sprites. The intercepted length of a particle at a pixel is twice the z-component of the relative coordinates corresponding to the point sprite. This relationship is represented as:

$$z = \sqrt{x^2 + y^2} \quad (15)$$

Using this equation, the thickness Thick of a particle is calculated by:

$$\text{Thick} = 2z = 2\sqrt{x^2 + y^2} \quad (16)$$

Through this calculation, each particle's contribution to the pixel's thickness can be determined, which represents the absorption of light by the fluid volume in the thickness texture.

Because of the representation of particles as spheres, it is necessary to smooth the thickness texture to achieve a continuous variation in thickness. In this study, a simple Gaussian filter is adopted to smooth the thickness values.

Once the thickness texture is smoothed, the next step is to obtain light absorption based on thickness. This process adheres to the Beer-Lambert Law and is formulated as:

$$\overline{\mathbf{rgb}} = \mathbf{rgb} \cdot \exp(-\kappa l) \quad (17)$$

where l denotes the path length of light traveling through the fluid and is approximated using the thickness texture. κ is a constant coefficient associated with the fluid type, and it can vary for **rgb** channels to produce more realistic effects.

The final image is obtained by calculating pixel colors based on the light attenuation and normal buffer. The calculation exploits the Fresnel reflection effect along with the Blinn-Phong specular highlight model. Additional effects like shadows and caustics can be integrated to achieve better realism.

4.2. Multiphase fluid rendering solution

After simulating multiphase fluids using a particle-based approach, the data of all particles in the fluid simulation is gathered, including the position p , size r , and the volume fractions c_k of different fluid phases within each particle. This fulfills the conditions $\sum_k c_k = 1$ and $c_k \geq 0$, with the order of particles being arbitrary. Based on the rendering pipeline discussed in Section 4.1, this paper proposes a rendering pipeline for multiphase fluids. The lighting model applicable for screen-space rendering and oriented towards multiple fluid phases is represented as follows:

$$C_{\text{out}} = C_{\text{refract}}(1 - R(\theta)) + C_{\text{reflect}}R(\theta) + k_s(n \cdot h)^\alpha \quad (18)$$

where C_{refract} denotes the color obtained by sampling the skybox with refracted rays; C_{reflect} denotes the color obtained by sampling the background texture, multiplied by the attenuation coefficient for refracted light; $R(\theta)$ represents the previously discussed Fresnel reflectance, and the last term is the specular highlight from the Blinn-Phong model. k_s and α are constants in the specular term; n is the reconstructed normal vector for the pixel, and h is the half-angle vector. For electrolytic fluids, the Fresnel reflectance varies slightly but does not significantly affect the final color, so it is not interpolated in this study.

The reflective color component C_{reflect} is obtained by sampling the skybox with reflected rays, and it depends only on the surface normal and thus requires no modification. The refractive component equals the value obtained by sampling the background texture (excluding the fluid) after light attenuation through the fluid. This attenuation involves two factors: the light path within the fluid and the fluid's material properties.

Based on the Beer-Lambert Law 17 and considering the proportions of different fluid particles, this paper proposes a rendering solution for multiphase fluids, which dynamically calculates the attenuation coefficient κ at

each pixel. Meanwhile, an additional phase fraction texture is introduced to represent the weighted cumulative volume fraction of fluid phases contributing to each pixel. The weight is the thickness of the particles, and it is calculated as follows:

$$M_k^i = \sum_{n \in p(i)} \text{Thick}_n \times c_k^n \quad (19)$$

where k denotes a specific fluid phase in the multiphase fluid, i denotes the current pixel, M_k^i denotes the phase fraction of phase k at pixel i , $p(i)$ represents the set of particles contributing to pixel i , n iterates over the particle set $p(i)$, Thick_j represents the thickness of particle n at pixel i , and c_k^n denotes the volume fraction of phase k in particle n . Once the phase fraction texture is obtained, the light attenuation coefficients of different fluids are combined to calculate the light attenuation coefficient for each pixel, and the phase fractions are normalized beforehand:

$$\kappa = \sum_i \kappa_i \times M_i \quad (20)$$

$$\sum_i M_i = 1 \quad (21)$$

Then, the newly calculated κ is used to determine the light attenuation for each pixel according to the Beer-Lambert Law. The final illumination image is obtained by implementing the lighting model proposed by Lann(23).

5. Results and analysis

The experimental evaluations were carried out on a system outfitted with an NVIDIA GeForce GTX 1650 graphics card. We adopted the SPH-based multiphase fluid simulation methodology as developed by Xu et al(2). This method is adaptable to various particle-based fluid simulations. Depth texture smoothing in our experiments used one-dimensional narrow-range filtering with parameters $\delta = 10.5R$ and $\mu = 1R$, where R is the particle radius. The computational costs associated with the fluid simulation were excluded from the performance results of our experiments.

5.1. Miscible multiphase fluid rendering

The efficacy of the rendering approach introduced in this work has been substantiated through a series of simulations involving diverse multiphase fluid dynamics. This method extends the capabilities of current real-time rendering technologies to include miscible multiphase fluid simulations(7), addressing a challenging aspect that has seen limited exploration in real-time contexts. The rendering technique was applied to three distinct simulation scenarios: crown splash simulation, quadruple dam-break simulation, and three-phase fluid interaction, each demonstrating the practical application of the method. The parameters underpinning these simulations are systematically documented in Table 1.

The *Crown Splash Simulation* shown in Figure 2 is rendered to illustrate a two-phase miscible flow, where a denser fluid block, depicted in red with a density of 3000 kg/m^3 , descends into a less dense fluid pool, illustrated in green with a density of 1000 kg/m^3 . The resultant dynamic is a splash that evokes the form of a crown, with the distinct fluids gradually blending into a unified body. The rendering method in this paper captures the gradational blending of the two miscible fluids, showcasing the higher-density red block integrating progressively with the green fluid body, a phenomenon articulated through the proposed rendering method. This seamless blending is indicative of the efficacy of the rendering approach in simulating and visualizing real-time miscible multiphase fluid interactions.

As shown in Figure 3, in the *Quadruple Dam-break Simulation*, a scenario with four different densities of miscible fluids is presented. These fluids undergo a dam break, leading to complex interactions and blending, resulting in a unified mixture. Traditional screen-space rendering techniques fall short in accurately depicting the scenario where a single screen pixel must represent multiple densities and colors of miscible fluids (7). The method proposed in this study effectively renders this challenging scenario, producing visually accurate results that showcase the blending of multiple fluid phases within the same region. This capability is a significant advancement in the field of multiphase fluid simulation, addressing the complexities inherent in rendering miscible fluids in real time.

The *Three-Phase Fluid Interaction Simulation* presented in Figure 4 demonstrates the proposed method's capability to render scenarios involving fluids with varying densities impacting a static pool. The first block, a blue fluid with a density of 1000 kg/m^3 , descends, followed by a purple fluid at 3000 kg/m^3 , and finally a white fluid at 5000 kg/m^3 . Each subsequent fluid

impacts the tranquil surface, creating dynamic interactions. The initial state of the simulation captures the three fluids poised to fall, highlighting the distinct densities and colors as they prepare to interact with the pool. This is shown in the sub-figure (a). As the simulation progresses, sub-figures (b) and (c) reveal the impact of the purple fluid on the blue, illustrating the beginning of their intermingling, while regions untouched by the collision retain their distinct colors and outlines. Sub-figures (d) and (e) depict the high-density white fluid as it impacts and begins to blend with the already mixed purple and blue fluids, with the blending region's color boundary becoming increasingly indistinct. Finally, sub-figure (f) displays the complete amalgamation of the three fluid types into a single, cohesive body. The experimental outcomes affirm the effectiveness of the rendering method in depicting the intermingling phenomena inherent in miscible multiphase fluid simulations.

5.2. Immiscible multiphase fluid rendering

In addition to the previously discussed methods for rendering miscible multiphase flows, this section presents a comparative analysis to demonstrate the superiority of our proposed method for rendering simulations of immiscible multiphase fluids. Our method was rigorously tested across three distinct simulation scenarios: crown splash simulation, quadruple dam-break simulation, and three-phase fluid interaction, showcasing its versatility and effectiveness in accurately rendering complex fluid dynamics. The experimental setup and parameters for these simulations are meticulously detailed in Table 1, underscoring the method's robustness and the precision of the rendering outcomes. The comparative evaluation illustrates that our approach not only achieves real-time performance but also significantly enhances the visual fidelity of the multiphase fluid simulations compared to existing techniques (7).

In *Crown Splash Simulation* shown in Figure 5 for immiscible two-phase fluid dynamics, a block of water with a density of 3000 kg/m^3 is simulated as it impacts a pool with a density of 1000 kg/m^3 , resulting in a crown-shaped splash. The red water block, under the influence of gravity and the interaction with the green water body, disperses and then reconvenes due to the immiscible forces at play. The top image displays the refined rendering achieved by our method, whereas the bottom image depicts the comparatively blurred rendering by Brito et al. method (7) due to oversaturation. The middle row zooms in on both rendering results. Our method results in a rendering with a consistent light source, as opposed to the layered rendering

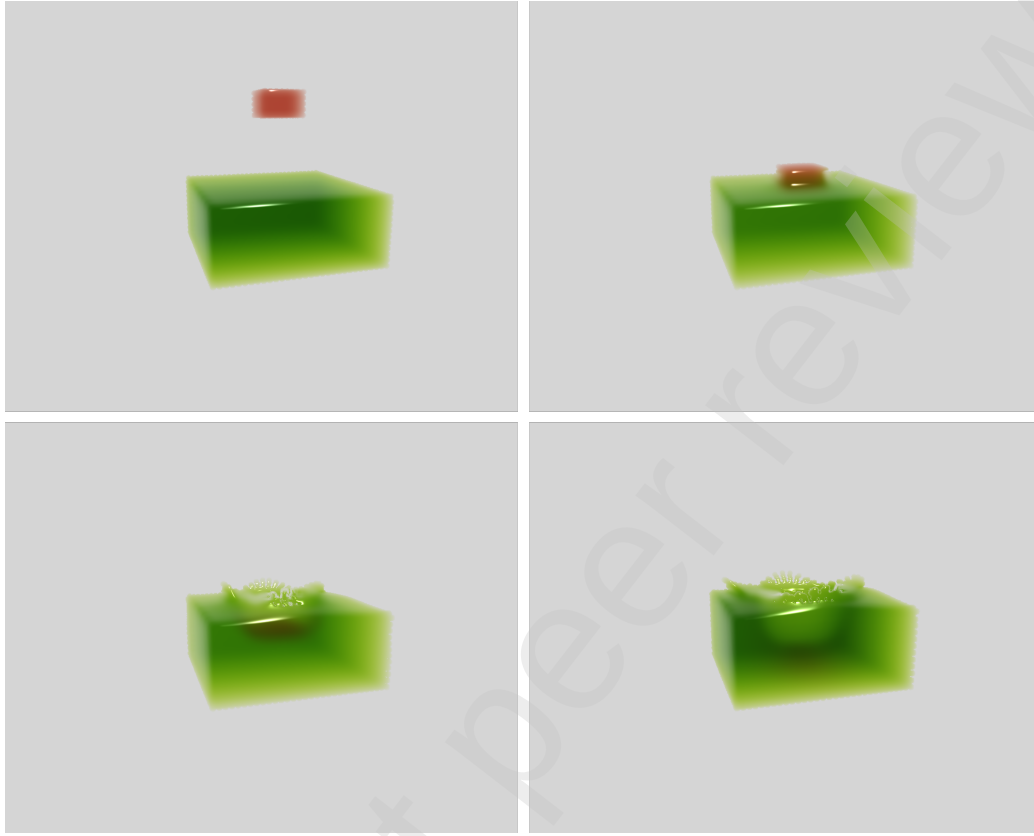


Figure 2: Rendering for miscible crown splash simulation. A water cube with a density of 3000kg/m^3 falls into a pool with a density of 1000kg/m^3 . The two fluids are miscible.

approach of Brito et al., which manifests multiple light sources and consequently, the red fluid within the green body appears unnaturally highlighted, leading to a lack of visual coherence in the image.

In *Crown Splash Simulation*, four fluids with distinct densities were simultaneously released, resulting in an intricate interplay and collision that produced splashing effects. As the simulation progressed, the fluids reached a stable stratification due to their density differences under the influence of gravity, settling from top to bottom in order of increasing density. The comparative visualization reveals that the rendering clarity is markedly improved using our method compared to the Brito et al. method (7), which appears oversaturated and consequently blurrier. Focused examination of the rendering processes (see middle column of images) indicates a unified light source

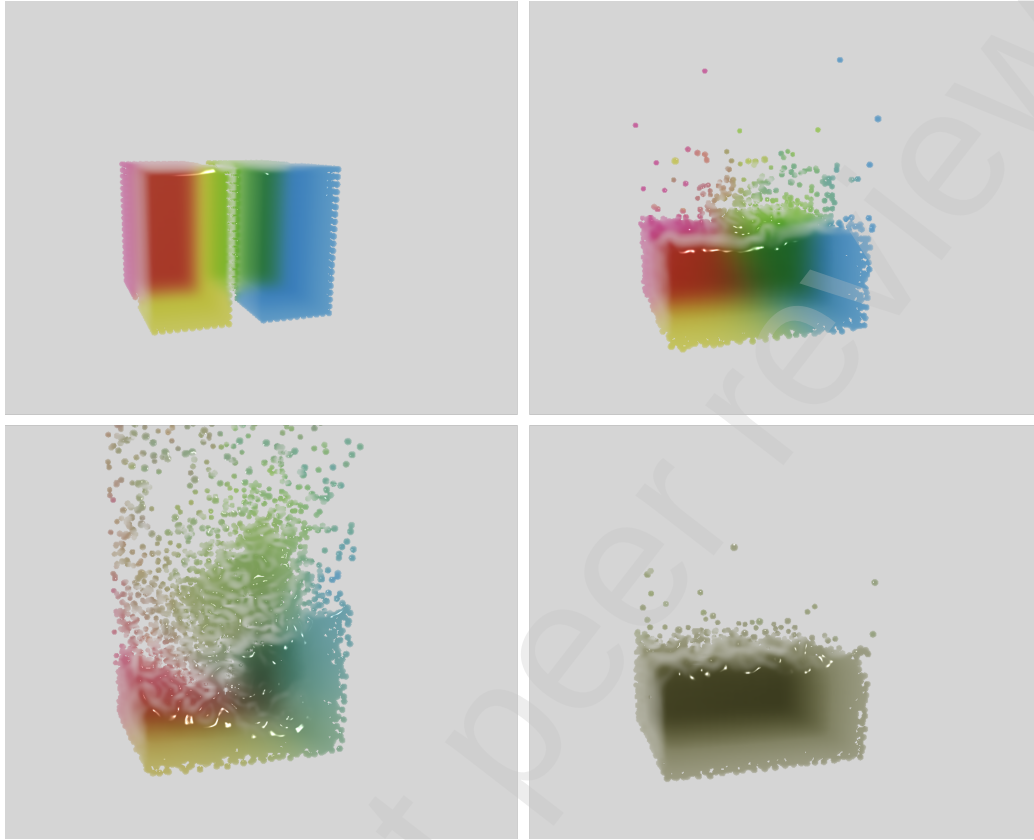


Figure 3: Rendering for miscible quadruple dam-break simulation. Four blocks of miscible fluids with different densities burst the dam simultaneously.

in our method, enabling distinct and naturally transitioning layers between fluids. In contrast, the Brito et al. method (7), which renders each fluid independently before layering, results in the presence of four separate light sources. This multiplicity leads to each fluid’s individual highlight, culminating in an incongruent overall visual effect.

In the *Three-Phase Fluid Interaction Simulation*, densities of 1000 kg/m^3 , 3000 kg/m^3 , and 5000 kg/m^3 are attributed to descending water blocks colored blue, purple, and white, respectively. The serene state of the previously settled blocks is disrupted by the introduction of a new block, which is meticulously rendered and analyzed. The comparative results are evident: our method delivers a distinct and vivid rendering, as shown at the top of the image, in contrast to the Brito et al. method (7) depicted at the bottom,

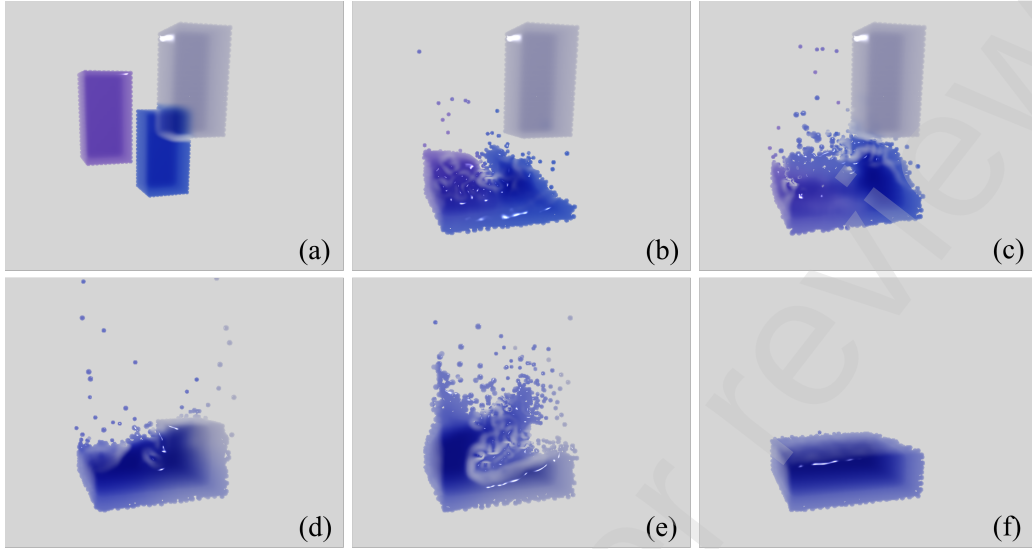


Figure 4: Rendering for three-phase miscible fluid interaction simulation. Three blocks of immiscible fluids with different densities burst the dam one after another.

where over-saturation leads to a loss of detail. A closer inspection, provided in the middle row of the image, reveals the superior definition of fluid particles and interfaces in our rendering, unlike the faintly depicted particles and blurred boundaries in the Brito et al. method (7), which obscure the fluid motion and state.

5.3. Quantitative Evaluation

The experimental analysis conducted in this section aims to determine the real-time rendering capabilities of the proposed method for multiphase fluid simulations and to quantify the additional performance overhead compared to single-phase fluid rendering.

A fixed-volume scene was rendered, varying particle counts to assess the impact on computational time. In each case, 89 frames were rendered and the average time per frame was calculated.

Figure 8 represents the rendering times for single-phase and multiphase fluids, with the latter being subdivided into miscible and immiscible categories. The rendering times for all fluid types did not exceed the 0.01-second threshold, indicating that the method satisfies real-time rendering requirements.

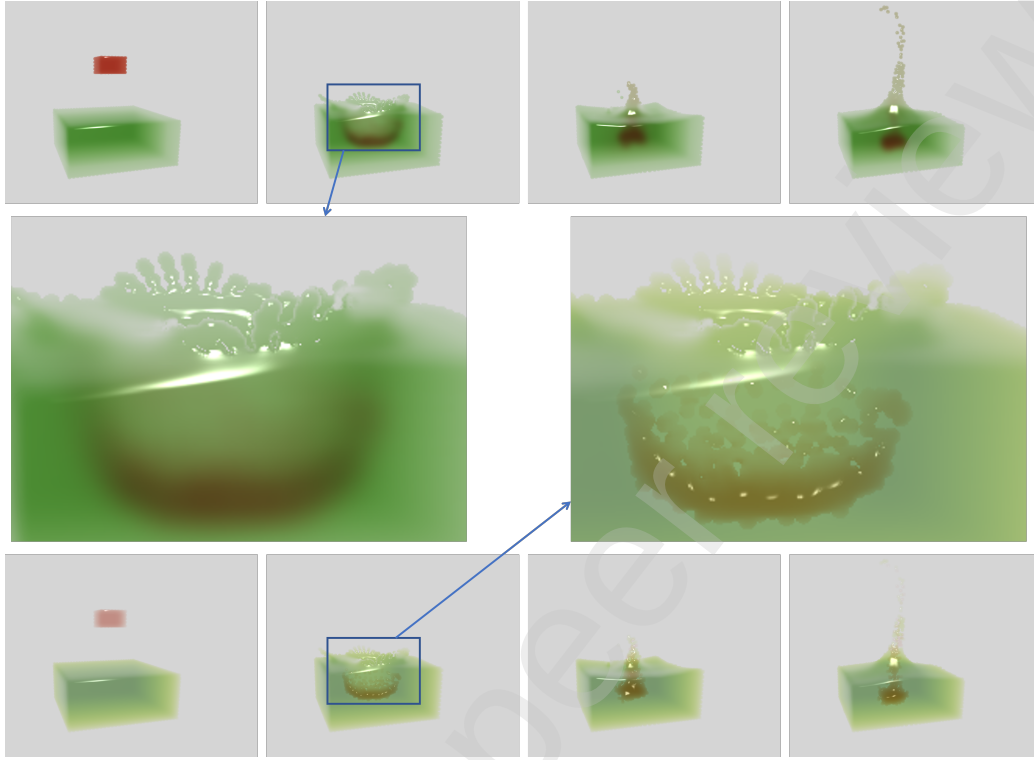


Figure 5: Comparison of the rendering for immiscible crown splash simulation between our method (top) and Brito et al.'s method (7) (bottom). A block of water with a density of 3000kg/m^3 falls into a pool with a density of 1000kg/m^3 .

An in-depth examination of Figure 9 provides insights into the computational effort associated with texture processing. The graph delineates six distinct trends corresponding to different processing stages for single-phase, miscible, and immiscible fluid simulations. The light blue lines represent single-phase fluid calculations for thickness texture and its smoothing. The orange and yellow lines correlate with the miscible two-phase fluid simulations, denoting the calculation and smoothing of texture thickness and phase fraction, respectively. Similarly, the dark blue and gray lines are associated with the immiscible two-phase fluid simulations, illustrating the computation times for the same texture processing as miscible fluids. Notably, the times for miscible and immiscible fluids are nearly identical, indicating a consistent processing overhead for both types of multiphase fluids.

The analysis reveals that the proposed method incurs an additional but

manageable overhead for multiphase fluid rendering, with the primary contributions being the acquisition and smoothing of phase fraction textures. Despite this, the method maintains real-time performance, demonstrating its potential for practical applications in dynamic fluid simulations.

Table 1: Comparison of rendering times for multiphase fluid simulations.

Simulation scenario	Fluid density (kg/m ³)	Number of fluid particles	Average rendering time per frame (s)			
			Brito et al. (7)		Our method	
			Miscible	Immiscible	Miscible	Immiscible
Crown splash	$\rho_{\text{green}} = 1000$ $\rho_{\text{orange}} = 3000$	12813	0.022987	/	0.009164	0.009291
Three-phase fluid interaction	$\rho_{\text{blue}} = 1000$ $\rho_{\text{purple}} = 3000$ $\rho_{\text{white}} = 5000$	7623	0.018487	/	0.009592	0.009676
Quadruple dam-break	$\rho_{\text{green}} = 500$ $\rho_{\text{blue}} = 1000$ $\rho_{\text{red}} = 3000$ $\rho_{\text{yellow}} = 5000$	10164	0.01393	/	0.012753	0.013007

5.4. Adaptability of Rendering Method

This section evaluates the integration process of the proposed multiphase fluid rendering approach within existing screen-space fluid rendering frameworks. A significant aspect of this new method is the independent management of phase fraction textures, distinct from depth textures typically used. Unlike conventional screen-space rendering techniques, our method necessitates the acquisition and subsequent smoothing of a novel phase fraction texture. This texture functions analogously to depth and thickness textures, encoding phase fraction data at the pixel level. The flexibility of our approach is exemplified by its compatibility with standard smoothing techniques already applied to existing texture types.

The integration simplicity becomes apparent during the light attenuation computation phase, where both the thickness and phase fraction textures are utilized. At this juncture, the method seamlessly dovetails with established screen-space rendering protocols, affirming its ease of implementation.

Experiments conducted to test the independence of the phase fraction texture involved applying both Bilateral Gaussian and Narrow Range filters to the depth texture. Outcomes, as presented in Figure 10, indicated that variations in depth texture filtering exerted negligible influence on the resultant rendering, substantiating the successful decoupling of the phase fraction texture.

The experimental findings validate the proposed method's adaptability and straightforward incorporation into prevalent screen-space fluid rendering

frameworks. This underscores its viability as a practical tool for enhanced fluid dynamics visualization.

6. Conclusion and Future Works

This work presented a novel real-time screen space rendering method for particle-based multiphase fluid simulations, which significantly enhances visual fidelity and computational efficiency. The method's robustness was validated across various simulation scenarios, demonstrating its superiority over existing techniques. Future works may explore the extension of this method to handle larger simulation scales and integrate more complex physical phenomena. Further research will also focus on optimizing the algorithm for different hardware architectures and exploring its applicability in virtual reality environments.

7. Acknowledgements

This research was supported by National Natural Science Foundation of China (62306032, 62376025, U22A2022), Guangdong Basic and Applied Basic Research Foundation (2022A1515110350, 2023A1515030177), Interdisciplinary Research Project for Young Teachers of USTB (FRF-IDRY-22-025). The computing work is partly supported by MAGICOM Platform of Beijing Advanced Innovation Center for Materials Genome Engineering.

References

- [1] Y. Zhang, X. Ban, Y. Guo, M. S. Obaidat, Simulation system for collisions and two-way coupling of non-newtonian fluids and solids, *Simulation Modelling Practice and Theory* 106 (2021) 102212.
- [2] Y. Xu, X. Wang, J. Wang, C. Song, T. Wang, Y. Zhang, J. Chang, J. J. Zhang, J. Kosinka, A. Telea, X. Ban, An implicitly stable mixture model for dynamic multi-fluid simulations, in: *SIGGRAPH Asia 2023 Conference Papers*, SA '23, Association for Computing Machinery, New York, NY, USA, 2023. doi:10.1145/3610548.3618215.
- [3] W. Li, D. Liu, M. Desbrun, J. Huang, X. Liu, Kinetic-based multi-phase flow simulation, *IEEE Transactions on Visualization and Computer Graphics* 27 (7) (2021) 3318–3334.

- [4] Y. Jiang, Y. Lan, A dynamic mixture model for non-equilibrium multi-phase fluids, *Computer Graphics Forum* 40 (7) (2021) 85–95.
- [5] Y. Xu, Y. Xu, Y. Xiong, D. Yin, X. Ban, X. Wang, J. Chang, J. J. Zhang, Anisotropic screen space rendering for particle-based fluid simulation, *Computers & Graphics* 110 (2023) 118–124.
- [6] F. I. Quispe, A. Paiva, Counting particles: a simple and fast surface reconstruction method for particle-based fluids, in: *2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, Vol. 1, IEEE, 2022, pp. 145–149.
- [7] C. J. dos Santos Brito, M. W. S. Almeida, A. L. B. Vieira-e Silva, J. M. X. N. Teixeira, V. Teichrieb, Screen space rendering solution for multiphase sph simulation, in: *2017 19th Symposium on Virtual and Augmented Reality (SVR)*, IEEE, 2017, pp. 309–318.
- [8] M. Müller, D. Charypar, M. Gross, Particle-based fluid simulation for interactive applications, in: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2003, pp. 154–159.
- [9] Y. Zhu, R. Bridson, Animating sand as a fluid, *ACM Transactions on Graphics (TOG)* 24 (3) (2005) 965–972.
- [10] B. Adams, M. Pauly, R. Keiser, L. J. Guibas, Adaptively sampled particle fluids, in: *ACM SIGGRAPH 2007 papers*, 2007, pp. 48–es.
- [11] J. Yu, G. Turk, Reconstructing surfaces of particle-based fluids using anisotropic kernels, *ACM Transactions on Graphics (TOG)* 32 (1) (2013) 1–12.
- [12] B. Solenthaler, J. Schläfli, R. Pajarola, A unified particle model for fluid–solid interactions, *Computer Animation and Virtual Worlds* 18 (1) (2007) 69–82.
- [13] M. Kazhdan, H. Hoppe, Screened poisson surface reconstruction, *ACM Transactions on Graphics (ToG)* 32 (3) (2013) 1–13.
- [14] F. Incahuanaco, A. Paiva, Surface reconstruction method for particle-based fluids using discrete indicator functions, *Computers & Graphics* 114 (2023) 26–35.

- [15] R. Fraedrich, S. Auer, R. Westermann, Efficient high-quality volume rendering of sph data, *IEEE transactions on visualization and computer graphics* 16 (6) (2010) 1533–1540.
- [16] T. Zirr, C. Dachsbacher, Memory-efficient on-the-fly voxelization of particle data., in: *EGPGV@ EuroVis*, 2015, pp. 11–18.
- [17] P. Goswami, P. Schlegel, B. Solenthaler, R. Pajarola, Interactive sph simulation and rendering on the gpu (2010).
- [18] X. Xiao, S. Zhang, X. Yang, Fast, high-quality rendering of liquids generated using large-scale sph simulation, *Journal of Computer Graphics Techniques Vol 7* (1) (2018).
- [19] T. Biedert, J.-T. Sohns, S. Schröder, J. Amstutz, I. Wald, C. Garth, Direct raytracing of particle-based fluid surfaces using anisotropic kernels., in: *EGPGV@ EuroVis*, 2018, pp. 1–11.
- [20] T. Kawamura, Y. Hasegawa, Y. Idomura, Interactive steering on in situ particle-based volume rendering framework, *Journal of Visualization* (2023) 1–19.
- [21] W. Valentine, R. Mukundan, Gpu accelerated modelling and real-time rendering of fluid motion, in: *2023 38th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2023, pp. 1–6. doi:10.1109/IVCNZ61134.2023.10343884.
- [22] H. Cords, O. G. Staadt, Interactive screen-space surface rendering of dynamic particle clouds, *Journal of Graphics, GPU, and Game Tools* 14 (3) (2009) 1–19.
- [23] W. J. van der Laan, S. Green, M. Sainz, Screen space fluid rendering with curvature flow, in: *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, 2009, pp. 91–98.
- [24] F. Bagar, D. Scherzer, M. Wimmer, A layered particle-based fluid model for real-time rendering of water, in: *Computer Graphics Forum*, Vol. 29, Wiley Online Library, 2010, pp. 1383–1389.
- [25] N. Truong, C. Yuksel, A narrow-range filter for screen-space fluid rendering, *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1 (1) (2018) 1–15.

- [26] M. Müller, S. Schirm, S. Duthaler, Screen space meshes, in: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2007, pp. 9–15.
- [27] F. Oliveira, A. Paiva, Narrow-band screen-space fluid rendering, in: Computer Graphics Forum, Vol. 41, Wiley Online Library, 2022, pp. 82–93.
- [28] T. Wu, Z. Zhou, A. Wang, Y. Gong, Y. Zhang, A Real-Time Adaptive Ray Marching Method for Particle-Based Fluid Surface Reconstruction, in: A. Ghosh, L.-Y. Wei (Eds.), Eurographics Symposium on Rendering, The Eurographics Association, 2022. doi:10.2312/sr.20221157.
- [29] M. Choi, J.-H. Park, Q. Zhang, B.-S. Hong, C.-H. Kim, Deep representation of a normal map for screen-space fluid rendering, Applied Sciences 11 (19) (2021) 9065.
- [30] K.-H. Chan, W. Ke, S.-K. Im, Particle-mesh coupling in the interaction of fluid and deformable bodies with screen space refraction rendering, Computer Animation and Virtual Worlds 29 (1) (2018) e1787.
- [31] L. d. S. R. Neto, A. L. Apolinário Jr, Real-time screen space cartoon water rendering with the iterative separated bilateral filter, SBC Journal on Interactive Systems 8 (1) (2017) 20–32.
- [32] P. Wang, Y. Zhang, D. Kong, B. Yin, A real-time fluid rendering method with adaptive surface smoothing and realistic splash, in: Advances in Multimedia Modeling: 19th International Conference, MMM 2013, Huangshan, China, January 7-9, 2013, Proceedings, Part II, Springer, 2013, pp. 468–478.
- [33] X. Xiao, S. Zhang, X. Yang, Real-time high-quality surface rendering for large scale particle-based fluids, in: Proceedings of the 21st ACM siggraph symposium on interactive 3D graphics and games, 2017, pp. 1–8.
- [34] M. Manninen, V. Taivassalo, S. Kallio, On the mixture model for multiphase flow, VTT Publications (288) (1996) 3–67.
- [35] A. P. Tampubolon, T. Gast, G. Klár, C. Fu, J. Teran, C. Jiang, K. Museth, Multi-species simulation of porous sand and water mixtures, ACM Trans. Graph. 36 (4) (jul 2017).

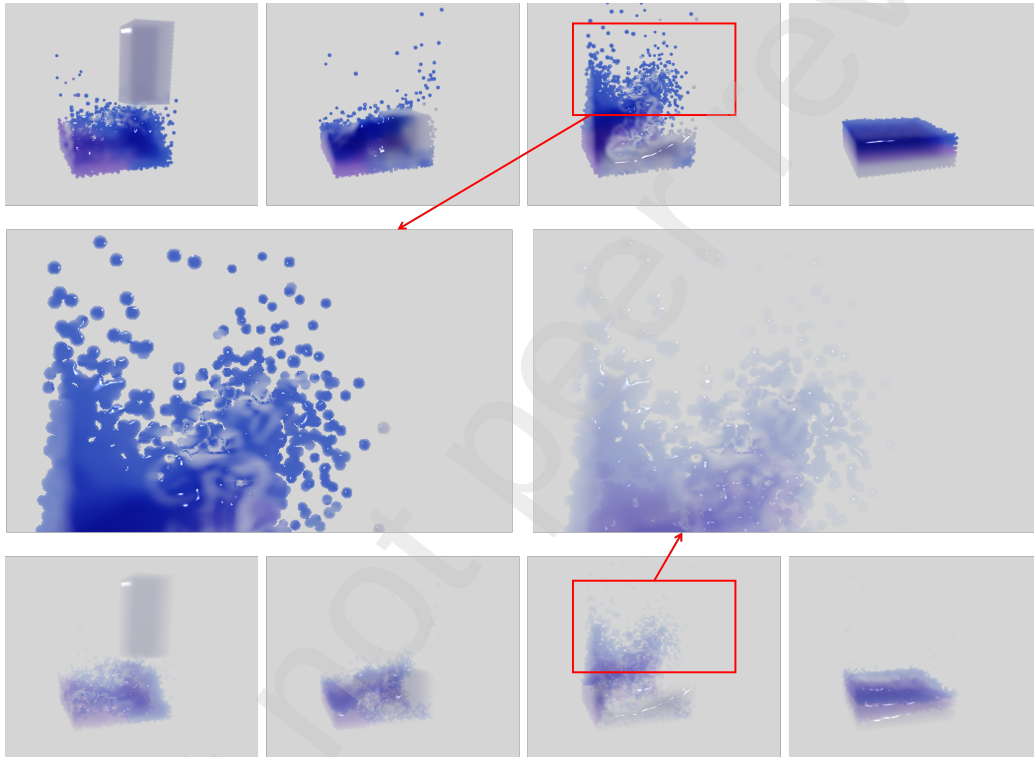


Figure 7: Comparison of the rendering for three-phase immiscible fluid interaction simulation between our method (top) and Brito et al.'s method (7) (bottom). Three blocks of immiscible fluids with different densities burst the dam one after another.

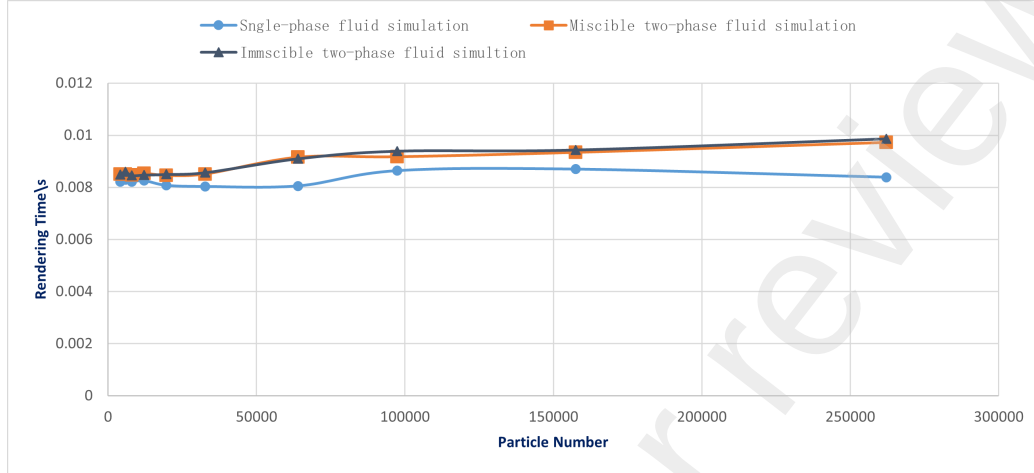


Figure 8: Rendering times for single-phase (light blue), miscible two-phase (orange), and immiscible two-phase (dark blue) fluid simulations, verifying the method's real-time rendering capability.

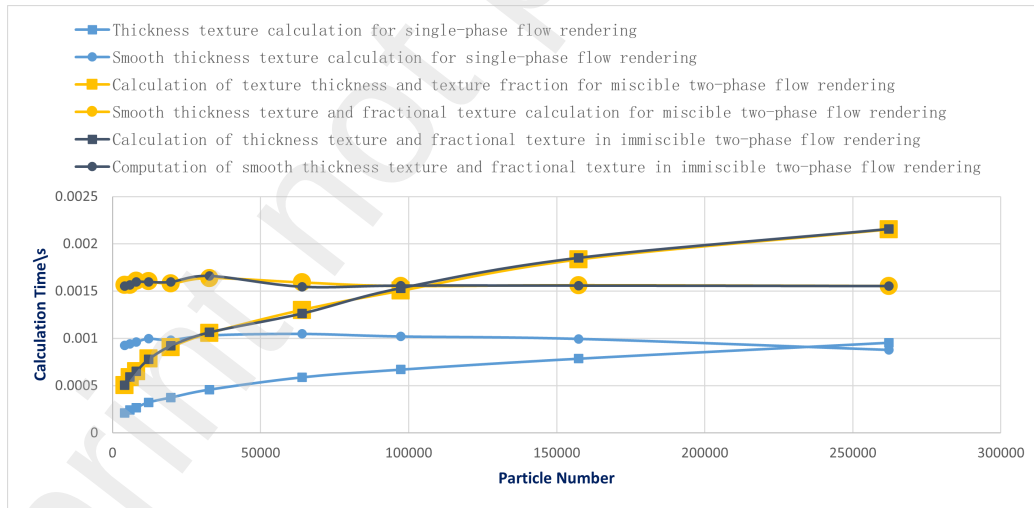


Figure 9: Detailed computational times for texture processing stages across single-phase (light blue), miscible (orange and yellow), and immiscible (dark blue and gray) two-phase fluids, highlighting the efficiency of the proposed rendering pipeline.



Figure 10: Depicting the impact of various depth texture filtering techniques on the rendering outcome, demonstrating the decoupled nature of the phase fraction texture within the rendering pipeline.