Fluid Simulation with Adaptive Staggered Power Particles on GPUs

Xiao Zhai, Fei Hou*, Hong Qin*, *Senior Member, IEEE,* and Aimin Hao

Abstract—This paper extends the recently proposed power-particle-based fluid simulation method with staggered discretization, GPU implementation, and adaptive sampling, largely enhancing the efficiency and usability of the method. In contrast to the original formulation which uses co-located pressures and velocities, in this paper, a staggered scheme is adapted to the Power Particles to benefit visual details and computing efficiency. Meanwhile, we propose a novel facet-based power diagrams construction algorithm suitable for parallelization and explore its GPU implementation, achieving an order of magnitude boost in performance over the existing code library. In addition, to utilize the potential of Power Particles to control individual cell volume, we apply adaptive particle sampling to improve the detail level with varying resolution. The proposed method can be entirely carried out on GPUs, and our extensive experiments validate our method both in terms of efficiency and visual quality.

Index Terms—physically based modeling, fluid simulation, power diagrams, GPU parallelization, adaptive sampling.

1 Introduction and Motivation

FLUID motion is highly complex to reproduce in computer applications, game/film productions, and has gained much attention in the computer graphics community during the recent two decades. Modern fluid simulators are capable of generating vivid waves, splashes, bubbles, etc. However, the exact incompressibility continues to be challenging for most fluid simulators to date. As discussed in Ihmsen et al. [1], while Lagrangian methods preserve mass perfectly, the oscillations in density evaluation affect the particle volume, requiring further correction such as Implicit Incompressible Smoothed Particle Hydrodynamics (IISPH) [2]. On the other hand, incompressible Eulerian methods assume a rest density and compute a divergence-free velocity field, yet the mass fluctuations [3], [4] may lead to volume changes and possible artificial viscosity [5].

Hybrid methods that integrate the advantages of both have long been explored, such as the popular Particle-In-Cell (PIC) and FLuid Implicit Particle (FLIP) which were first adopted in [6] to simulate sand and later used to simulate liquids with abundant surface details [4], [7], [8], viscosity treatment [9], and multiple-phase setting [10]. Raveendran et al. [11] proposed to use SPH particles with an underlying Eulerian grid for pressure projection. These methods usually combine the Eulerian divergence-free solver and the Lagrangian material particles to minimize the impacts brought by either and to ensure the preservation of both density and mass.

De Goes et al. [12] recently presented Power Particles for incompressible fluids, which resort to power diagrams for domain partition on the basis of the particle distribution (hence called power particles). Despite its fully Lagrangian nature, their method makes use of a divergence-free solver as well as a precise volume control on particles to enforce strong incompressibility. Nevertheless, some limitations could still be found in this powerful method. First of all, the pressure solver of the Power Particles calculates the velocity divergence using co-located pressures/velocities and one-ring divergence operator. This could cause strong negative pressure near free surfaces and prevent the formation of individual sprays. The co-located scheme also brings a large number of non-zero entries into the Pressure Poisson Equation (PPE). Speed being a major impediment to the adoption of the Power Particles scheme, the original method fails to enjoy the GPU acceleration since few efforts have been made to construct power diagrams on modern GPUs. Moreover, despite Power Particles' potential to control individual cell volume, dynamic particle sampling still remains unexplored.

Accordingly, this paper is dedicated to resolving the aforementioned weaknesses within the existing power-particle-based fluid simulator. As opposed to the co-located scheme and one-ring divergence operator, we adapt the staggered discretization to the power-particle framework and apply the divergence condition using directional velocity perpendicular to cell borders. Such an arrangement not only eliminates the unwanted cluster behaviors but also reduces the non-zero entries of the PPE to a large extent. Regarding the construction of power diagrams, previous methods usually resort to the existing code libraries like CGAL [13] or VORO++ [14] for robustness and efficiency. Instead, we propose a novel facet-trim method which enables the parallelized construction on GPUs. We also apply adaptive particle sampling at run time to simulate fluid details up to different scales in various regions. Our extensive experiments prove that Power Particles produce better visual results with higher efficiency when using the staggered discretization and adaptive sampling. Additionally, the GPU-based implementation offers a significantly faster

- *X. Zhai and A. Hao are with State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China.*
- *F. Hou is with State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China.*
- *H. Qin is with Department of Computer Science, Stony Brook University, New York, USA.*
- ** corresponding authors.*

construction of power diagrams over the VORO++ library and tremendously enhances the performance of the fluid simulator, enabling its availability for designing, modeling, and other interactive applications.

The main contributions of this paper are:

- A staggered pressure projection on Power Particles to benefit visual details and computing efficiency;
- A GPU-based construction of power diagrams to boost the computational performance by an order of magnitude; and
- An adaptive sampling strategy for Power Particles to simulate fluids using varying resolution.

2 Background and Related Works

The topics of this paper cover the particle-based methods, grid-based methods, hybrid methods, Voronoi-based methods as well as the adaptive techniques in fluid simulation. In this section, we briefly review them in the following categories.

SPH techniques. Owing to its conceptual simplicity and ability to generate compelling visual results, SPH [15] has become one of the most popular fluid solvers nowadays. Such popularity is attributed partially to its kernel-based nature which brings decent trade-off between computational expenses and realism for unstructured methods. However, one of the drawbacks is its weakness in enforcing the incompressibility. Early works used the equation of state to calculate the interior pressure [16], [17], which often induced instabilities. Later on, iterative methods [18], [19] started to dominate as they greatly ameliorated the problem by using a dynamic stiffness. Recently, researchers managed to solve the PPE implicitly [2], and both the density and the velocity divergence were taken into account in some state-of-the-art solvers [20]. On the other hand, SPH stress points [21] and staggered SPH method [22] solved the tensile instabilities and improve boundary handling, using the staggered arrangement in a similar way to this paper.

Methods using staggered grids. In early Computational Fluid Dynamics, the decoupling of pressure and velocity was found in Cartesian grids inducing non-physical oscillations, also known as the checkerboard problem. The staggered grid [23] was designed to eliminate such problem and was later widely applied in grid-based fluid simulators, e.g., the octree approach [24]. The irregular grids also benefit from the staggered primal-dual formulation (or Discrete Exterior Calculus) in fewer non-zero entries in the Laplacian, smaller grid interval and simpler boundary handling. The tetrahedral meshes were naturally integrated with the staggered discretization [25] to simulate various phenomena, and the dynamic semi-structured [26] and unstructured tetrahedral grids [27] were also employed. Based on these, Mullen et al. [28] proposed an energy-preserving integration that was viscosity-free and time-reversible.

Hybrid methods. It is a tempting attempt to integrate two methods to neutralize each other's defect and produce more convincing results. Being one of these attempts, FLIP was employed to simulate sand [6], viscous fluids [9], multiphase fluids [10] and so on. Hong et al. [29] and Ando et al. [7], [30] put forward adaptive FLIP to drop the computational burden, and Ferstl et al. [31] came up with Narrow Band FLIP eliminating excess particles far beneath the surface. Moreover, FLIP was combined with IISPH [32] for a much more flexible usage. In addition to these FLIP-family methods, Losasso et al. [33] coupled SPH and particle level set to model the diffuse regions and dense liquid volume respectively. Raveendran et al. [11] proposed to employ a coarse Eulerian grid to provide a divergence-free background velocity and to use SPH interaction to counter the local density fluctuation. Chentanez et al. [34] simulated large-scale water phenomena by further combining particles, 3D grids and height fields, from fine to coarse.

Voronoi-based approaches. The Voronoi-based spatial discretization is relatively new in the fluid simulation territory because the non-stationary nature of fluids often requires frequent updates of Voronoi diagrams which are fairly time-consuming. Sin et al. [35] implemented a fluid solver and performed staggered pressure projection based on Voronoi diagrams. Brochu et al. [36] integrated a surface tracking process with Voronoi diagrams to capture thin features. English et al. [37] glued uniform grids of different resolution together with Voronoi diagrams. Power diagram, a generalization of Voronoi diagram with more control over the cell volume, was adopted to simulate bubble interactions in foam [38] as well as to build a fully functional fluid simulator [12]. It was also incorporated into a Sparse Paged Grids simulator to yield high-resolution adaptive liquids [39]. Most recently, meshless Voronoi is realized on GPUs [40], but neither the generalization to power diagram nor the possible adaptivity has been thoroughly studied. In this paper, we improve the existing fluid simulator of de Goes et al. [12] towards better visual result and efficiency.

Adaptivity in fluid simulation. Although the adaptive octree scheme in Eulerian methods [24], [39] is quite intuitive, the adaptivity is often difficult to be implemented with SPH as the insertion or deletion of particles could cause sudden large spring force, let alone the intricacy brought by non-uniform smoothing radii. Adams et al. [41] adjusted particle positions and the smoothing radii to reduce the pressure error, while Keiser et al. [42] simulated the interaction across particle resolutions. Zhang et al. [43] provided an early attempt of adaptive sampling for SPH on GPUs. To prevent the previously mentioned sudden changes in force, Orthmann and Kolb [44] introduced temporal blending to achieve a smooth transition from before to after the particle insertion or deletion, and Winchenbach et al. [45] further developed implicit temporal blending and achieved infinite continuous adaptivity. Alternatively, different levels of detail can be simulated independently and coupled on the common boundaries [46], [47].

3 Overview of Power Particles

This section provides a brief introduction of the power diagram and the original Power Particles solver, followed by an overview to highlight our main distinctions from the existing method.

3.1 Power Diagram

A power diagram [48] is a partition of the spatial domain Ω according to a set of sites $\{\mathbf{q}_i\}$ along with their associated scalar weights $\{w_i\}$, as shown in Figure 1. Each cell $\mathcal{V}_i$ is defined as

$$\mathcal{V}_i = \{\mathbf{x} \in \Omega \mid \|\mathbf{x} - \mathbf{q}_i\|^2 - w_i \le \|\mathbf{x} - \mathbf{q}_j\|^2 - w_j \ \forall j\}, \quad (1)$$

where $\|\cdot\|$ is the Euclidean distance. Two neighboring sites $\mathbf{q}_i$ and $\mathbf{q}_j$ share a facet $\mathcal{A}_{ij}$, which should be perpendicular to vector $\mathbf{q}_i - \mathbf{q}_j$. We denote V_i as the volume of $\mathcal{V}_i$ in 3D (area in 2D) and A_{ij} as the area of $\mathcal{A}_{ij}$ (length in 2D). The distance between two neighboring sites $\mathbf{q}_i$ and $\mathbf{q}_j$ is denoted as l_{ij}, while the distance from a site $\mathbf{q}_i$ to its facet $\mathcal{A}_{ij}$ is d_{ij}, thus $l_{ij} = d_{ij} + d_{ji}$.

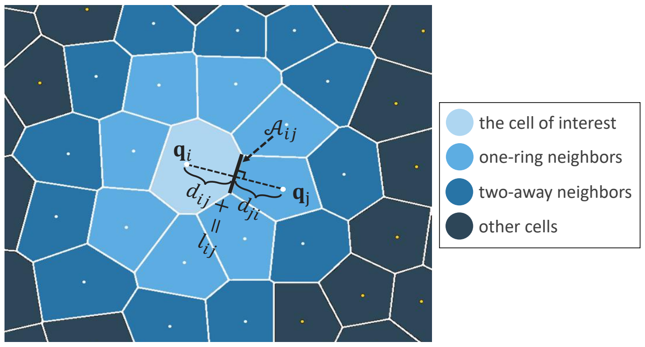

Fig. 1. Exemplary power diagram in 2D.

Aurenhammer et al. [48] pointed out that the volume of cells can be fully controlled by their power weights. Therefore the volume of each cell can be constrained to a target value $\bar{V}_i$. We use the same weight optimization scheme as [12] to iteratively update the weights towards direction δw following

$$\frac{1}{2}\boldsymbol{\Delta}\delta w = V - \bar{V}, \quad (2)$$

where $\boldsymbol{\Delta}_{ij} = A_{ij}/l_{ij}$.

3.2 Power Particles

The Power Particles method is a Lagrangian fluid solver whose particles are treated as the sites of a time-evolving power diagram. In each time step, the power diagram is updated according to the particles' locations and their corresponding weights, based on which the dynamics of fluids is calculated following the Navier-Stokes equation,

$$\frac{\mathrm{D}\mathbf{v}}{\mathrm{D}t} = -\frac{\nabla p}{\rho} + \nu\nabla^2\mathbf{v} + \mathbf{f}^{ext}, \quad (3)$$

where $\mathbf{v}$ denotes the velocity of particles, p their pressure, ρ the density, ν the kinematic viscosity, $\mathbf{f}^{ext}$ the body accelerations, $\frac{\mathrm{D}}{\mathrm{D}t}$ the material derivative, ∇ the gradient and ∇^2 the Laplacian. With the velocity determined, the particles are advected as all Lagrangian methods do. An algorithm that specifies such procedure is listed in Algorithm 1.

In each simulation iteration, the splitting strategy is applied for integration. The diffusion term and the external forces in the Navier-Stokes equation are first solved, giving an intermediate velocity $\mathbf{v}^*$ to each particle,

$$(1 - \nu\Delta t\nabla^2)\mathbf{v}^* = \mathbf{v}_{t-1} + \Delta t\mathbf{f}^{ext}, \quad (4)$$

Algorithm 1: The simulation loop of Power Particles.

```
1 Apply diffusion and external forces;
2 Apply pressure projection on power diagram;
3 Advect particles;
4 Update the power diagram and enforce volume
    constraints;
```

where $\mathbf{v}_{t-1}$ is the initial velocity of each particle and Δt the duration of the time step. Subsequently, the pressure projection solves the PPE, i.e.,

$$\frac{\Delta t}{\rho}\nabla^2 p = \nabla \cdot \mathbf{v}^*, \quad (5)$$

where $\nabla\cdot$ is the divergence operator. The velocity is updated afterwards by

$$\mathbf{v}_t = \mathbf{v}^* - \frac{\Delta t}{\rho}\nabla p, \quad (6)$$

where $\mathbf{v}_t$ is the resulting velocity used to advect the flow. In the previous Power Particles, de Goes et al. [12] proposed to use discrete divergence operator $\mathbf{D}$, gradient operator $\mathbf{G}$ and Laplacian operator $\mathbf{L}$ in calculating the fluid dynamics, where $\mathbf{D} = \nabla_{\mathbf{q}}V$, $\mathbf{G} = -\mathbf{D}^T$ and $\mathbf{L} = \mathbf{D}\mathrm{diag}(m)^{-1}\mathbf{G}$. Using the co-located discretization, $\mathbf{D}$ corresponds to the matrix:

$$\begin{cases} \mathbf{D}_{ij} := (\nabla_{\mathbf{q}_j} V_i)^T = \dfrac{A_{ij}}{l_{ij}}(\mathbf{q}_j - \mathbf{b}_{ij})^T, \\ \mathbf{D}_{ii} := (\nabla_{\mathbf{q}_i} V_i)^T = -\displaystyle\sum_{j\in N_i}(\nabla_{\mathbf{q}_i} V_j)^T, \end{cases} \quad (7)$$

where $\mathbf{b}_{ij}$ denotes the centroid of facet $\mathcal{A}_{ij}$ and N_i the set of one-ring neighboring particles of i. For more information about the power diagram or the original Power Particles, please refer to [12].

3.3 Overview of Our Method

In this paper, we make improvements in three aspects: integrating staggered discretization in pressure solve, designing a parallel power-diagram construction algorithm, and incorporating the adaptive particle sampling. The framework of our method is illustrated in Algorithm 2. Essentially different from the original Power Particles (Algorithm 1), our method is not only GPU friendly but also much faster and visually pleasing on CPU, and can produce even better surface details with lower computational cost with the adaptivity turned on.

Algorithm 2: The simulation loop of our method.

```
1 Apply diffusion and external forces;
2 Apply staggered pressure projection on power
    diagram (Section 4);
3 Advect particles;
4 Apply adaptive particle sampling (Section 6);
5 Update the power diagram on GPUs (Section 5) and
    enforce volume constraints;
```

The original Power Particles, in a fairly straightforward manner, use co-located pressure/velocity samples, one-ring discrete divergence/gradient operators $\mathbf{D}$ and $\mathbf{G}$, and two-away Laplacian operator $\mathbf{L}$. In the pressure step, the velocity is projected to achieve zero divergence, i.e. $\mathbf{D}\mathbf{v}_t = 0$. Given $\mathbf{D}$ is a one-ring operator, this zero divergence is enforced as an inter-particle velocity condition. However, it could lead to unnatural behaviors near free surfaces. As illustrated

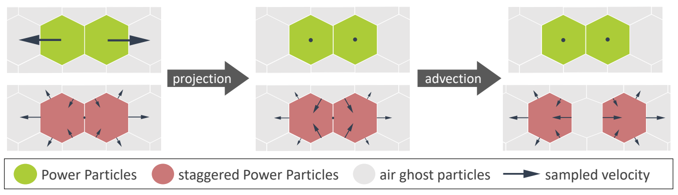

Fig. 2. The handling of parting particles using Power Particles and our staggered Power Particles.

in the top row of Figure 2, when nearby surface particles move apart from each other, the pressure projection would exert strong negative pressures to eliminate the relative movement between particle pairs in order to meet the zero-divergence condition. This causes particle clusters instead of individual sprays to form on disturbing surfaces.

The IISPH method of Ihmsen et al. [2], similar to the Power Particles, also used one-ring divergence and two-away Laplacian operator in their pressure solver. They reported this particle clustering as well albeit that they corrected the predicted density to a target value ρ_0 rather than enforced the zero-divergence condition directly. The predicted density ρ_i^{adv} is calculated as

$$\rho_i^{adv} = \rho_i - \Delta t \rho_i \nabla \cdot \mathbf{v}_i, \tag{8}$$

where ρ_i denotes the SPH density evaluation and $\nabla \cdot \mathbf{v}_i$ the velocity divergence. In other words, not only the underestimated SPH density but also the diverging particles could result in the drop of ρ_i^{adv} and then the consequent negative pressures. They referred to this artifact as "exaggerated cohesion effects" that noticeably absorbed the splashes, but failed to offer a fundamental solution other than clamping.

The Power Particles are free of underestimated particle density, however, the velocity divergence could still cause the clustering problem. In this paper, we present to use the staggered discretization to address this unnatural clustering on diverging particles. With this treatment, the zero divergence is no longer enforced with neighboring particle velocities but with directional velocities sampled at cell borders. The projection still makes the velocity field free of divergence but at the same time keeps the surface particles their tendency to form individual sprays in case of negative pressure, see Figure 2. Additionally, with the co-located scheme, the multiplication of large sparse matrices $\mathbf{D}$ and $\mathbf{G}$ is rather time-consuming, especially for GPU realization where the runtime speed matters a lot. Ihmsen et al. [2] proposed to bypass this multiplication by using a two-pass matrix-free method to solve the PPE, notwithstanding the nested two-away stencil still significantly increases the number of non-zero entries in the system. In contrast, the staggered scheme offers one-ring Laplacian operator which removes the necessity to multiply the large sparse matrices or to access the two-away neighbors, making the pressure projection cheaper to solve. It should be noted that the staggered discretization is only employed in the pressure projection, while in other steps the velocity is still sampled at the particles for convenience.

The construction of power diagrams, which is the foundation of Power Particles, is a well-studied spatial partitioning problem. Both the weight Delaunay triangulation, like CGAL [13], and the local cell-based method, like VORO++ [14], were extensively adopted for this task previously. However, those efforts failed to exploit modern GPUs for parallelization. Meanwhile, the update of the underlying structure is usually the most time-consuming step in the simulation loop. Therefore, we propose a GPU-based algorithm, specialized for fluid simulation, to construct the power diagram in parallel. To better enjoy the GPU parallelism, our new algorithm constructs power diagrams in a local facet-based fashion which differs from the local cell-based VORO++, see Figure 3.

A great variety of adaptive sampling techniques have been presented in the field of fluid simulation. Among them, the recent one from Winchenbach et al. [45] is rather spectacular due to its decent effectiveness and excellent performance. Although this scheme is originally designed for SPH, we adopt its idea of particle classification and the $(n{+}1){:}n$ particle merging strategy in Power Particles. We get rid of the data dependencies in the original method and eliminate the atomic operations to fit GPU implementation. Besides, some minor modifications to the particle classification also help it fit into the Power Particles as we desire.

4 Pressure Projection using Staggered Discretization

Pressure solve is normally the most expensive step in incompressible fluid simulators. Previously, the Power Particles assemble the one-ring divergence and gradient operators to achieve the two-away Laplacian. However, the pressure projection with co-located scheme could cause unnatural particle clusters and is expensive to solve, as discussed in Section 3.3.

To improve, we make use of the staggered discretization previously seen in octree [24], tetrahedral [26], [27], hybrid [25] and Voronoi [35] solvers. Essentially, this scheme defines the divergence and gradient as facet-particle/particle-facet operators instead of particle-particle operators. Therefore the nested Laplacian becomes particle-facet-particle style rather than the particle-particle-particle form. The details of the staggered pressure solver are specified in Section 4.1, while Section 4.2 presents the boundary handling of our method.

4.1 Staggered Pressure Projection

Starting with the intermediate particle velocity $\mathbf{v}^*$ after the non-pressure forces applied, we first assume the staggered velocity v_{ij}^*, namely the directional velocity perpendicular to each facet $\mathcal{A}_{ij}$ of the power diagram, is constant within the facet, and calculate it using a linear interpolation as

$$v_{ij}^* = \frac{d_{ij}\mathbf{v}_j^* + d_{ji}\mathbf{v}_i^*}{l_{ij}} \cdot \hat{\mathbf{n}}_{ij}, \tag{9}$$

where $\hat{\mathbf{n}}_{ij} = (\mathbf{q}_j - \mathbf{q}_i)/\left\|\mathbf{q}_j - \mathbf{q}_i\right\|$ is the normal vector of facet $\mathcal{A}_{ij}$. We then define the divergence of a particle cell following the divergence theorem as

$$(\nabla \cdot \mathbf{v}^*)_i = \frac{1}{V_i} \sum_{j \in N_i} A_{ij} v^*_{ij}, \tag{10}$$

where V_i is the volume of the i-th particle cell and N_i denotes the set of one-ring neighboring particles of i. Meanwhile, the scalar pressure gradient of a facet is written as

$$(\nabla p)_{ij} = \frac{p_j - p_i}{l_{ij}}, \tag{11}$$

and the Laplacian operator can be assembled in a one-ring operator as

$$\begin{aligned}(\nabla^2 p)_i &= (\nabla \cdot \nabla p)_i \\ &= \frac{1}{V_i} \sum_{j \in N_i} A_{ij} (\nabla p)_{ij} \\ &= \frac{1}{V_i} \sum_{j \in N_i} \frac{A_{ij}}{l_{ij}} (p_j - p_i).\end{aligned} \tag{12}$$

Equation 10 and 12 are substituted into the PPE (Equation 5) to form the resulting linear system to solve. Although this Laplacian operator ∇^2 shares a very similar form with the finite-volume Laplacian $\mathbf{\Delta}$ in the volume constraints, it is still compatible with the divergence and gradient operator under staggered discretization. Besides, this Laplacian operator is also used in the diffusion calculation, namely Equation 4.

With the pressure solved, the directional pressure gradients on facets can be calculated through Equation 11. However, since the power cells are usually irregularly shaped, to evaluate the pressure gradients at particles is not so intuitive as interpolation. We propose to minimize the objective function below to recover the vector pressure gradients $(\nabla p)_i$ on particles:

$$\arg\min_{(\nabla p)_i} \sum_{j \in N_i} ((\nabla p)_i \cdot \hat{\mathbf{n}}_{ij} - (\nabla p)_{ij})^2. \tag{13}$$

Essentially, this objective function calculates an average vector influenced by all the facets related to $\mathbf{q}_i$, and its solution on each cell is obtained by solving the equivalent 3×3 equation locally:

$$(\sum_{j \in N_i} \hat{\mathbf{n}}_{ij} \hat{\mathbf{n}}_{ij}^T)\ (\nabla p)_i = \sum_{j \in N_i} ((\nabla p)_{ij} \hat{\mathbf{n}}_{ij}). \tag{14}$$

This vector conversion from facets to particles shares an essentially similar least-squares fitting approach with the tetrahedral/Voronoi solvers [27], [49]. Although it does not perfectly match the interpolation from particles to facets, our experiments on the kinetic energy demonstrate that the artificial viscosity it brings is rather negligible, see Figure 14. Finally, the particle velocity is updated using Equation 6.

Equipped with the staggered scheme, Power Particles are free from the particle clustering artifact and are capable of generating more abundant ricocheting splashes with higher efficiency, please see the comparisons on visual results in Figure 8, 11 and on performance in Section 7.2.

Our pressure projection is similar to the staggered Voronoi method from Sin et al. [35] in the definition of cell divergence (Equation 10) and facet gradient (Equation 11). The major difference lies in the conversion of vectors between particles and facets. In their work, velocities and pressure gradients on particles are calculated using smooth-kernel-based fitting and moving least-squares fitting respectively. These are common techniques to define continuous vector fields based on sample points, but they inevitably induce numerical damping. To alleviate this problem, we replace their back-and-forth vector conversions with linear interpolation (Equation 9) and local least-squares fitting (Equation 14) defined on power diagrams. A comparison in our experiments demonstrates that the decrease in numerical damping with our modification is quite noticeable, see Figure 12. Besides, we apply a different boundary handling strategy with solid particles and ghost air particles, which will be explained in Section 4.2. There are also some minor distinctions between the two methods, e.g., we apply the viscosity implicitly in Equation 4 while they ignore the viscous term completely in their solver, we merge their two-step Laplacian calculation into one operator (Equation 12) to slightly decrease memory accessing time, and we do not use cell volumes as weights in PPE since the substantially varying volumes in power diagrams could affect the convergence of solution via iterative methods.

4.2 Boundary Handling of Our Method

Boundary handling usually plays an indispensable role in fluid simulation. Our approach supports two kinds of boundaries: solid boundary and free surfaces. Specifically, we dynamically sample solid particles as the symmetrical mirror of the fluid particles through the solid boundaries at run time to clip the power diagrams according to the exact boundary shape, see Figure 4. These clipping particles do not have pressure values and are excluded in the PPE. They do not have power weights either since the solid shape is already known. For solid-fluid coupling at facet $\mathcal{A}_{is}$, the boundary flux $A_{is}v_s$ is included in the divergence of fluid particle i in Equation 10, where v_s denotes the velocity of the moving boundary along the normal direction $\hat{\mathbf{n}}_{is}$ of facet $\mathcal{A}_{is}$.

For free surfaces, ghost particles are adopted for fluid-air interaction. These particles are populated around the fluid particles in each time step, clipping the fluid power cells on the outer layer and fulfilling the free-surface boundary condition. They are excluded from the PPE since the incompressibility of these cells is less meaningful. Though they hold no determined velocity or pressure, they offer these values in a ghost-fluid fashion [50] at run time. For example, when a ghost particle $\mathbf{q}_g$ is accessed through facet $\mathcal{A}_{ig}$, its pressure is defined as $p_g = -(d_{gi}/d_{ig})p_i$ and velocity $\mathbf{v}_g = \mathbf{v}_i$. Furthermore, their power weights are set uniformly to offer a good boundary reference to other particles.

5 Parallel Power Diagram Construction

Without doubt, the construction of power diagram is the most computationally intensive and time-consuming step in the power-particle simulation. Previous works usually resort to existing libraries, such as CGAL [13] or VORO++ [14], for robust implementation of power diagram

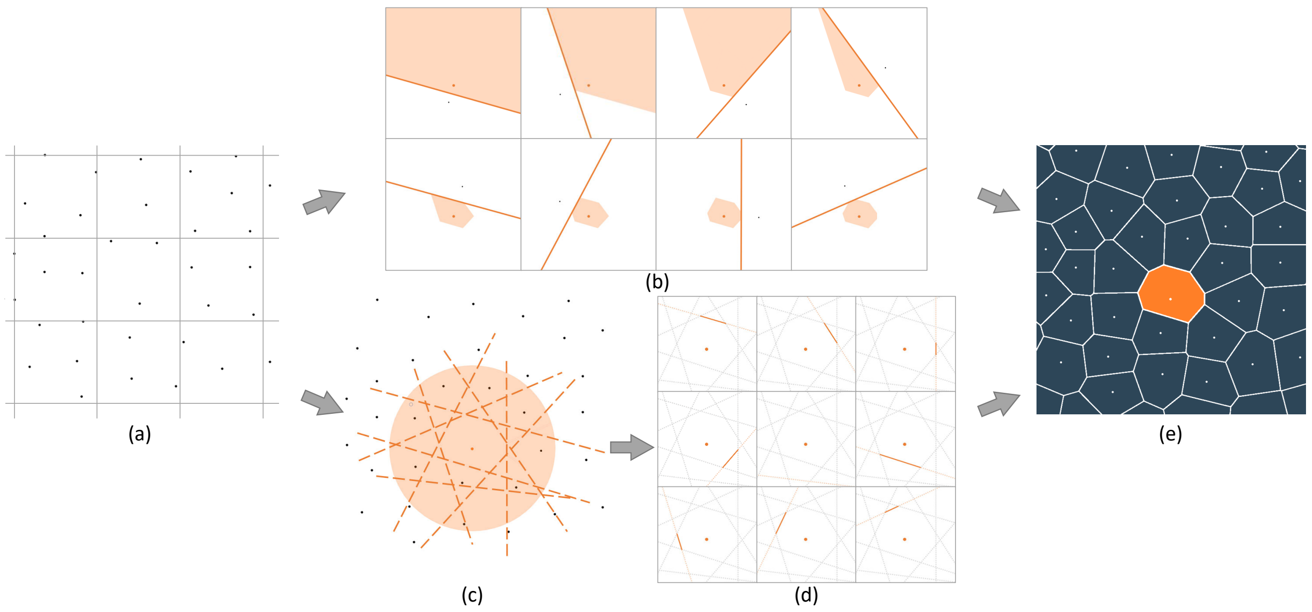

Fig. 3. The VORO++ style cell-based method (a-b-e) and our facet-based method (a-c-d-e) to construct power diagrams. (a) All particles are mapped to an auxiliary grid for the accelerated neighborhood search; (b) For each particle i (the orange dot), the corresponding power cell is constructed by sequential cell cuts from nearby sites; (c) For particle i the nearby sites within a maximum radius r_m are found, and the possible facets from all site pairs $(\mathbf{q}_i, \mathbf{q}_j)$ are generated; (d) The possible facets are used to trim each other in parallel; (e) The remaining facets are collected to form the power cells.

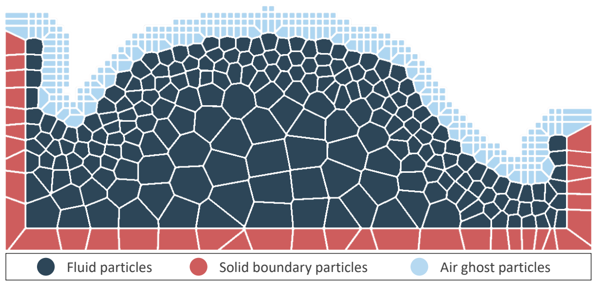

Fig. 4. A 2D example of power-particle spatial partition during fluid simulation. The solid particles are calculated as the symmetrical mirror of fluid particles through the exact boundary, while ghost particles are populated around the free surface to clip the cells on the outer layer.

construction on CPUs. However, the algorithms are not adequate to parallelize. Instead, we propose a parallel power diagram construction algorithm and implement it on GPU to accelerate the construction significantly. We first elaborate the ideas of our algorithm in Section 5.1 and then some details of implementation in Section 5.2.

5.1 The Algorithm of Power Diagram Construction

Employing a local strategy in the construction of power diagrams, VORO++ treats every particle cell $\mathcal{V}_i$ individually by iterating through all of its neighboring particles and repeatedly cutting the cell using the possible weighted planes $\mathcal{A}_{ij}$. This idea requires repeatedly updating polyhedral cells with the number of facets and vertices unpredictable beforehand, however, this is extremely time-consuming on GPUs. In contrast to the cell-based method of VORO++, our method constructs all the facets first and assembles power cells with the corresponding facets afterwards. This strategy not only avoids the expensive dynamic memory management but also takes full advantage of memory coalescing in every parallel step to maximize the power of multi-threading.

Our construction is also founded upon the fact that in fluid-simulation circumstance any two neighboring fluid particles should be close enough to some extent, or they should be separated by other fluid cells or ghost solid/air cells. As a result, the possible neighbors of a site are confined to a surrounding narrow spatial range.

Algorithm 3: GPU-based power diagram construction.

```
input  : weighted sites $(\mathbf{q}, w)$, maximum spacing $r_m$
output: power cells $\mathcal{V}$
1  Insert boundary/ghost particles;
2  Set up the auxiliary grid for neighborhood search;
   /* allocate possible facets */
3  for the i-th site in parallel do
4      for the j-th neighboring site within the radius of $r_m$ do
5          Generate possible facets $\hat{\mathcal{A}}_{ij}$;
   /* trim facets */
6  for the possible facets $\hat{\mathcal{A}}_{ij}$ in parallel do
7      for the possible facets $\hat{\mathcal{A}}_{ik}, k \neq j$ do
8          trim $\hat{\mathcal{A}}_{ij}$ with $\hat{\mathcal{A}}_{ik}$;
   /* collect facets to form cells */
9  for the i-th site in parallel do
10     for the possible facets $\hat{\mathcal{A}}_{ij}$ do
11         Collect $\hat{\mathcal{A}}_{ij}$ as $\mathcal{A}_{ij}$ if $A_{ij} > 0$;
12     Form the power cell $\mathcal{V}_i$;
```

Our algorithm to construct power diagrams can be seen in Algorithm 3 and Figure 3. First of all, the boundary/ghost particles are sampled on fluid-solid or free boundaries to clip the power diagrams. Inspired by the index-sort neighborhood search widely used in VORO++ and various SPH methods [1], our method also utilizes an underlying uniform grid for efficient particle accessing. With this auxiliary grid established, all sites within a grid cell are placed continuously in the array and can be accessed efficiently in a coalesced manner by GPUs. For each site $\mathbf{q}_i$, after finding all nearby sites within a maximum spacing r_m, we generate its possible facets $\hat{\mathcal{A}}_{ij}$ from all possible site pairs $(\mathbf{q}_i, \mathbf{q}_j)$. These possible facets are then used to trim each other, and during this process, some may be nullified if they are completely eliminated by other possible facets. Finally, the remaining facets $\mathcal{A}_{ij}$ are collected as the faces of the local power cell. Please see our supplementary video for an animated illustration of the power construction.

5.2 Details of Implementation

Compared with the cell-cut operation in VORO++, our facet-trim process circumvents the dynamic memory management, which is difficult to handle for GPUs, by allocating memory for all possible facets $\hat{\mathcal{A}}_{ij}$ in advance before the generation and trimming. However, this pre-allocation scheme consumes much more memory than enough since the exact amount to store the valid facets can hardly be predicted beforehand, and may easily exhaust all on-chip GPU memory when millions of particles are simulated simultaneously. To cope with this problem, the power cells are not computed all at once, but batch by batch. In each iteration, a batch of particles is processed through the steps of generating possible facets, trimming facets and forming power cells (line 3 - line 12 in Algorithm 3), then the memory buffers are refreshed for the next batch. The sequential construction of power cells in VORO++ can be seen as a special case where the batch size equals to 1, while the typical batch size used in our implementation is 10k to offer a balance between memory overhead and calculation expense.

Within the construction process, the most time-consuming step is the facet trimming whose time complexity is $O(nk^2)$, while both allocating possible facets and collecting facets have the complexity of $O(nk)$, where n denotes the total number of fluid cells and k the average number of possible facets for each cell. The complexity of trimming appears to be unacceptable at first glance. However, if we allocate a GPU thread for each possible facet, the complexity on each thread is only $O(k)$. In addition, the GPU memory coalescing further substantially decreases the accessing cost during computation. Meanwhile, a reasonable choice of r_m is crucial to keep the computational cost low while maintaining the accurate construction of the power diagram. In our experiments, r_m is empirically set as twice the largest particle diameter and k never exceeds 128, hence the time complexity of these three steps can be assumed to have a linear upper bound. Other components like ghost particle insertion or neighborhood search are commonly seen in most SPH solvers and are extremely fast when implemented properly on GPUs.

6 Adaptive Particle Sampling

This section covers the adaptive sampling strategy for dynamically adjusting the particle size and distribution during simulation. We draw inspiration from Winchenbach et al. [45] and propose an adaptive sampling method for Power Particles free of atomic operations and data dependencies. Illustration examples for both 3D and 2D can be found in Figure 5 and Figure 4. Section 6.1 introduces the sizing function we use as well as the particle classification process, followed by the particle splitting and merging strategy in Section 6.2 and Section 6.3 respectively.

6.1 Particle Classification

The most interesting parts of fluid flows lie in the vicinity of free surfaces and should be simulated with the finest resolution, while far beneath the boundary the subtle fluid movement is less noticeable to human eyes. Winchenbach et al. used Level-set functions to calculate the SPH particles' distance to free surfaces. In Power Particles, however, the air boundaries are well specified, therefore we approximate the distance to surface d_i in a cheaper flood-fill fashion $d_i = min(d_j) + l_{ij}, j \in N_i$, where the particles near free surfaces have $d_i = 0$ initially. We then use a sizing function to figure out a reference volume V_i^{ref} for each fluid particle,

$$V_i^{ref} = min(\frac{h + d_i}{d^{max}}, 1)V^{max}, \tag{15}$$

where h is the diameter of the finest particle and d^{max} controls the field of interests beyond which the reference volume is constantly set as V^{max}. In all demonstrated experiments, d^{max} is set to $16h$ and V^{max} is set to $16h^3$. Particles are classified according to the ratio of their volumes to the reference volumes $V_i^{rel} = V_i/V_i^{ref}$ into following categories,

$$C_i = \begin{cases} T & V_i^{rel} < 0.5 \\ S & 0.5 \le V_i^{rel} < 1.5 \\ O & 1.5 \le V_i^{rel} < 4 \\ L & 4 \le V_i^{rel} \end{cases}, \tag{16}$$

depending on which they should be split, merged or left intact: L-class particles are too large and should split into smaller ones while T-class particles are tiny and are likely to be merged and deleted; S-class and O-class particles have appropriate volume, but S-class particles are relatively smaller so they can receive more when a nearby particle is merged, while O-class particles are left intact.

6.2 Particle Splitting

When a particle is categorized as L-class, it splits into several smaller particles which have equal volume and are distributed within the original cell. In our implementation, an L-class particle i is split into $n_i = min(\lfloor V_i/V^{opt} \rfloor, 8)$ particles, where $\lfloor \cdot \rfloor$ is the floor function. Note that we set 8 as the upper limit of n_i only for coding feasibility. If a newly allocated particle is categorized as L-class again, it can be further split in the following time steps. For each n_i, we use a pre-computed pattern (see Figure 6) as sites for new particles, and the volume and velocity of new particles are $V_{new} = V_i/n_i$ and $\mathbf{v}_{new} = \mathbf{v}_i$, accordingly.

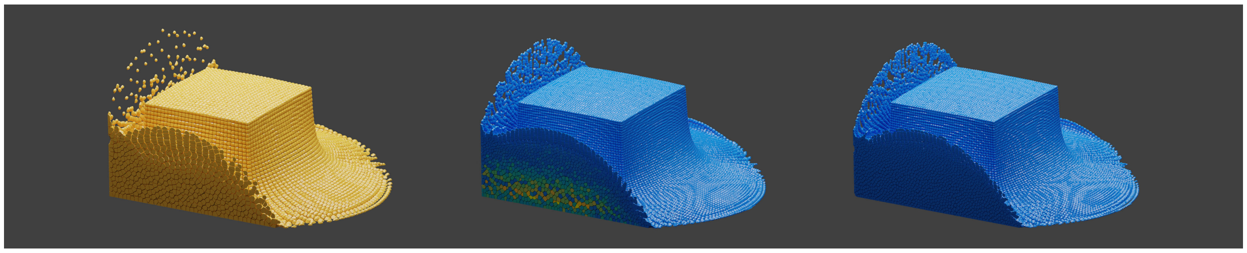
Fig. 5. A 3D example of the particle adaptive sampling with the cells color-coded according to their volumes. From left to right: 36k uniform particles, 70k particles with adaptivity, 147k uniform particles. The example of 70k particles with adaptivity manages to achieve similar surface resolution as the 147k uniform example while maintaining a coarse particle distribution beneath the surface.

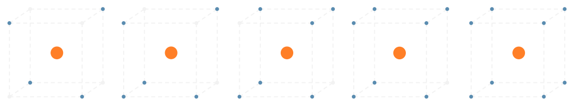
Fig. 6. The pattern used to split the orange particle into the blue sites.

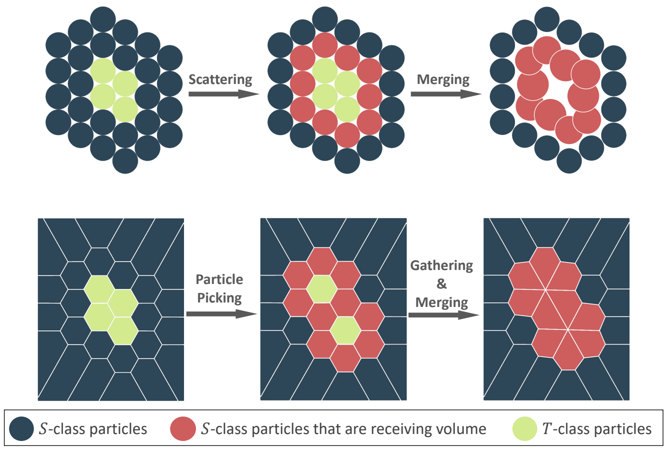

Fig. 7. Previous merging strategy should use atomic scattering operations and would contradict the data dependencies if the merging is applied at once, while ours uses gathering instead and is thread-safe.

6.3 Particle Merging

T-class particles are too fine for their local resolution and therefore can be merged into their neighborhood. Inspired by Winchenbach et al. [45], we accomplish the merging in an $(n+1){:}n$ pattern, where T-class particles are removed and their volumes are evenly divided and scattered to the neighboring S-class particles. However, when parallelized, this approach has two main drawbacks. First, if several T-class particles share one S-class neighbor, the divide-and-scatter operation towards this target should be atomic, which is totally feasible but not efficient on GPUs. Second, simply removing all T-class particles at once may contradict data dependencies and cause problems, see the top row of Figure 7. During the merging procedure, when one T-class particle is removed, its neighboring T-class particles should also receive their shares of volume just like S-class particles do. Yet in the previous method, these T-class particles are removed simultaneously, which induces the faulty distribution of fluids.

To correct these drawbacks, we set a restriction that T-class particles should not be next to each other, see the bottom row of Figure 7. By doing so, T-class particles do not have to transfer volume to other T-class particles which are removed simultaneously. Besides, we can eliminate the atomic operations by using the S-class particles to gather their shares from neighboring T-class particles rather than applying the divide-and-scatter operations.

In terms of implementation, we determine a subset of T-class particles that is unconnected. This is equivalent to a graph coloring problem which is NP-hard to find an optimal solution. Nonetheless, we are not attempting to seek the perfect solution, but only to work out a fast and convenient one suitable to be solved on GPUs. We propose to use a simple random picking method iteratively to determine the subset, as shown in Algorithm 4. In each pass, particles are randomly picked and tested, and those without T-class neighbors are combined into the target subset. Those candidates who are not categorized as T-class particles after several passes are tagged as S-class for receiving volume during the particle merging. This picking strategy is extremely fast on GPUs, and the small particles which miss all the passes can be further merged in the upcoming time steps. The picking probability p_{pick} and number of passes n_{pass} can be increased to allow quicker adaption, but values too high would undermine the chance of finding the unconnected candidates (line 6 in Algorithm 4), thus making the picking less effective. In our experiments, 10 passes of random picking with 0.1 probability is used.

With T-class particles satisfying the unconnected restriction, the i-th S-class particle can update its volume, position and velocity using a volume-weighted average as

$$\begin{aligned} V_i^{new} &= V_i + \sum_{j\in N_i \cap \mathcal{P}_T} \frac{V_j}{n_j}, \\ \mathbf{q}_i^{new} &= \frac{1}{V_i^{new}}(V_i\mathbf{q}_i + \sum_{j\in N_i \cap \mathcal{P}_T} \frac{V_j}{n_j}\mathbf{q}_j), \\ \mathbf{v}_i^{new} &= \frac{1}{V_i^{new}}(V_i\mathbf{v}_i + \sum_{j\in N_i \cap \mathcal{P}_T} \frac{V_j}{n_j}\mathbf{v}_j), \end{aligned} \tag{17}$$

where $\mathcal{P}_T$ denotes the selected subset of T-class particles and n_j is the surrounding S-class particles count of the j-th T-class particle. Finally, the merged particles are removed from the simulation.

7 Experimental Results

In this section, we validate our method through experiments and comparisons. The visual results are presented in

Algorithm 4: Picking valid T-class particles.

```
input : a set of candidate $T$-class particles $\mathcal{P}_{cand}$,
        picking probability $p_{pick}$, number of passes
        $n_{pass}$, two empty sets $\mathcal{P}_T$ and $\mathcal{P}_{temp}$
output: unconnected $T$-class particles $\mathcal{P}_T$
1  $k = 0$;
2  for $k < n_{pass}$ do
3      Empty $\mathcal{P}_{temp}$;
4      Randomly pick particles from $\mathcal{P}_{cand}$ into $\mathcal{P}_{temp}$
         under probability $p_{pick}$;
5      for the $i$-th particle in $\mathcal{P}_{temp}$ in parallel do
6          if $\boldsymbol{q}_i$ has no neighbors in $\mathcal{P}_{temp}$ and $\mathcal{P}_T$ then
7              $\mathcal{P}_T = \mathcal{P}_T \cup \{\mathbf{q}_i\}$
8      $k = k + 1$;
9  $\mathcal{P}_{cand} = \mathcal{P}_{cand} - \mathcal{P}_T$;
10 Tag particles within $\mathcal{P}_T$ as $T$-class;
11 Tag particles within $\mathcal{P}_{cand}$ as $S$-class;
```

Section 7.1, while the quantitative analysis and discussion are provided in Section 7.2. We compared the proposed method with the pressure projection from Sin et al. [35] as well as the original Power Particles [12] based on our own implementation. All of our experiments were implemented by CUDA C/C++ (staggered methods) or by OpenMP and VORO++ [14] (staggered methods and co-located methods) and were tested on a PC with an Intel Core i7-6700k CPU and an Nvidia Geforce GTX1070 graphics card. The visual results were achieved by extracting fluid surfaces through Marching Cubes and offline rendering with Cycles in Blender.

To test the capabilities of our method and to make comparisons, we build several scenarios, including a colliding spheres scenario (Figure 8), a dam break scenario (Figure 9), a colliding fountains scenario (Figure 10), a sphere drop scenario (Figure 11), a double dam break scenario (Figure 12), a wave generator (Figure 13) and a rotating vortices scenario (Figure 14). All of these test cases were simulated within a cube whose edge length was set to $1m$ and the acceleration of gravity $g = 9.8m/s^2$. Furthermore, the timesteps for low-resolution ($< 100k$ particles) and high-resolution ($\geq 100k$ particles) examples are $0.004s$ and $0.001s$ respectively to limit the CFL number $\alpha = \Delta t \frac{max\|\mathbf{v}\|}{h}$ no greater than 1, where h is the diameter of the finest particle.

7.1 Visual Results

Figure 8, 9, 10 and 11 juxtapose the visual renderings using our proposed staggered Power Particles and the original Power Particles. Similarities in the overall water movement, at a glance, can be found; however, the proposed approach is capable of yielding results with finer features and more details on free surfaces. In the colliding spheres scenario (Figure 8), two water balls collide towards each other in a zero-gravity space. Our staggered Power Particles generate a substantially greater amount of sprays during this collision, while the original Power Particles suffer severely from the particle clustering problem and fail to form highly detailed sprays even if the viscosity is set to zero. In the colliding fountains scenario (Figure 10), two columns of water collide to form splashes. As it drops onto the ground, the water sheet starts to wobble shortly after its formation (see the accompanying video) due to the perturbation in dynamics, where the original Power Particles wobble much more frequently than the proposed staggered method. In the sphere drop scenario (Figure 11), a water ball plummets to calm water, creating a crown-like splash and then some water spikes. Noticeably, our method manages to produce more abundant ricocheting droplets in contrast to particle clusters with the original Power Particles.

In Figure 9 and 11, the same scenarios are simulated with low-resolution and high-resolution examples. Similar fluid motion can be observed, with the high-resolution examples standing out with detailed surface behaviors. Besides, we incorporate the adaptive particle sampling method in Figure 10 and 11, which allows the simulation using merely half the particles to achieve the same visual results as the full resolution examples without noticeable difference.

To test the ability of our method in handling large-scale simulation, we use 1 million staggered Power Particles in the double dam break scenario as shown in Figure 12. We also compare our pressure projection method against the one from Sin et al. [35] under this setting since both methods share a similar staggered discretization. In comparison, their method displays more numerical damping while ours generates more highly detailed water sheets and splashes. For solid-fluid coupling, a moving wall is used in Figure 13 to periodically push the water in the tank, resulting in a desirable breaking-wave effect.

7.2 Quantitative Analysis and Discussion

Numerical Viscosity. In staggered Power Particles, the recovery of velocity gradients from facets to particles (Equation 14) does not perfectly match the interpolation from particles to facets (Equation 9), hence the conversion could give rise to numerical viscosity. Figure 14 demonstrates an example of four rotating vortices in a unit cube. After 1000 timesteps, both the proposed method and the original Power Particles maintain the vortices well with a very similar decay of kinetic energy with or without the viscosity. This result offers strong evidence that the numerical viscosity brought by the staggered discretization, particularly the unmatched velocity conversion, is rather negligible.

Performance. Owing to the GPU implementation, our method gains significant performance boost compared with the CPU-based Power Particles. The speedup of power diagrams construction lays a good foundation for the fluid simulator since updating the underlying spatial partition has always been the most time-consuming part. A speed comparison is conducted across several scenarios and the results are shown in Table 1 and Figure 15, where the data indicate that our method outperforms the original method over 10 times. We also implement our method on CPU to test the minor speed gain brought solely by the staggered discretization, which mainly influences the speed of pressure calculation. With the achieved speedup, the time cost of high-resolution Power Particles is cut from days to hours, while the low-resolution simulation with the proposed method could even serve as the preview demo for interactive designing or similar applications.

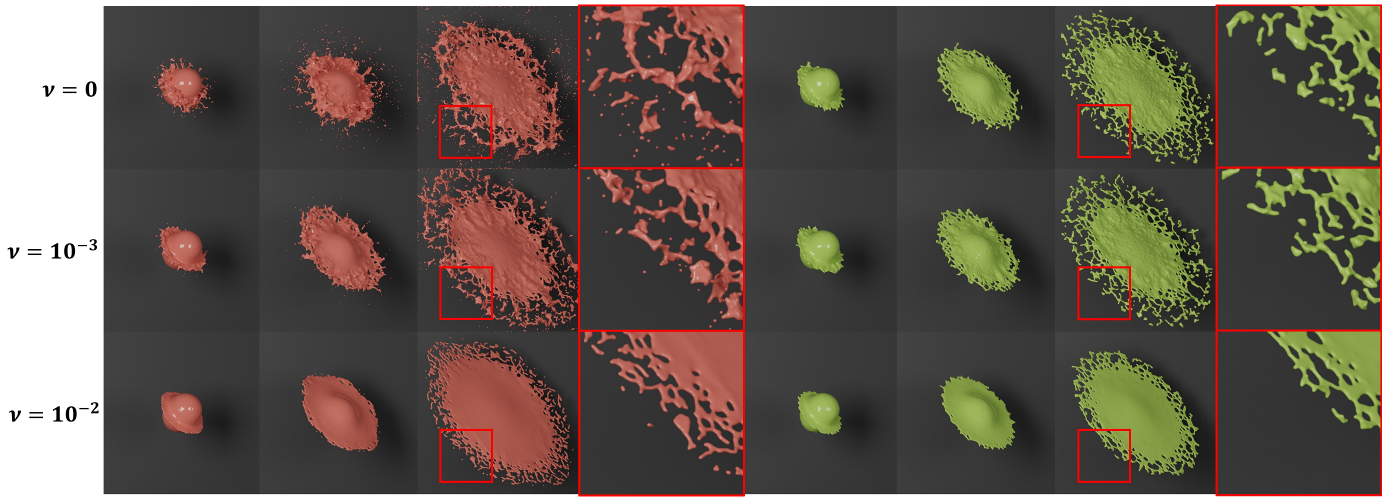

Fig. 8. The rendered results of the colliding spheres scenario under different kinematic viscosity ($\nu = 0, 1e^{-3}$ and $1e^{-2}$, respectively) with 60k staggered Power Particles on the left and 60k Power Particles on the right. Particle clusters are formed with Power Particles due to negative pressure on surfaces, while the staggered Power Particles are free from this artifact, creating abundant tiny splashes in the simulation.

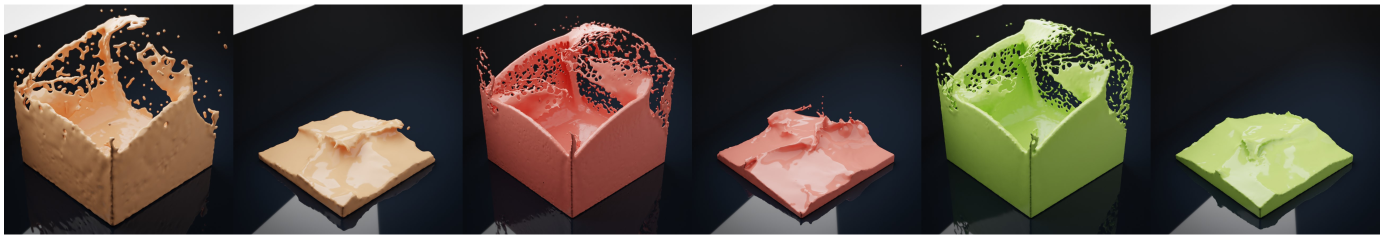

Fig. 9. The rendered results of the dam break scenario. From left to right: 18k staggered Power Particles, 147k staggered Power Particles, 147k Power Particles. In this test, the 147k staggered Power Particles is roughly 14 times faster than the 147k Power Particles.

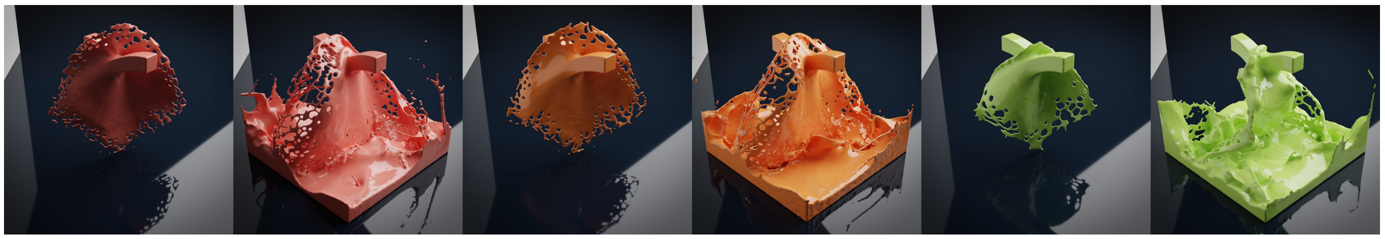

Fig. 10. The rendered results of the colliding fountains scenario. From left to right: 320k staggered Power Particles, 165k adaptive staggered Power Particles, 320k Power Particles. In this test, the 320k staggered Power Particles and 165k adaptive staggered Power Particles are respectively 8 and 15.5 times faster than the 320k Power Particles.

Volume error. To manifest the ability of our algorithm in handling uneven cell volume arrangement, Figure 16 shows a hydrostatic test on a narrow tank of liquid with various initial cell volumes over the domain. After 1000 timesteps, the position and shape of the cells hold well with barely perceptible volume error. This ability to accurately maintain the cell volume ensures a stable and even density/mass distribution in our method.

Adaptivity. Figure 17 offers statistics of particle numbers over time in the colliding fountains scenario. With the adaptivity turned off, the water is injected into the scene at a constant rate hence the particle number grows linearly over time up to 320k. When the adaptivity is enabled, the particle number increases rapidly at first and slows down when the water accumulates at the bottom. The particle number reaches at most 165k particles, about half the number of the full-resolution example, without having noticeable discrepancy in surface details visually.

8 Conclusion and Future Works

By integrating Power Particles with the staggered discretization as well as our newly proposed parallel power-diagram construction, this paper has detailed a fully GPU-based method for fluid simulation, staggered Power Particles, which not only outperforms the previous method with highly detailed surface effects but also yields a significant performance gain in both the construction of power diagrams and the overall fluid simulation. By continuing to equip it with the ability of adaptive particle sampling, our method has become an efficient and flexible fluid simulator, capable of producing vivid and photo-realistic fluid motion

TABLE 1
The statistics of calculation overhead (in seconds) of each step. From the second column, the overhead of the power-diagram construction, the pressure projection, the volume enforcement, the ghost-particle generation, the diffusion, other procedures (including advection and adaptive sampling) as well as the total expense are displayed respectively.

scenario	power diagram	pressure	volume	ghost particle	diffusion	others	total
dam break 18k SPP	0.062	0.038	0.038	0.001	0.004	0.000	0.145
dam break 147k SPP	0.415	0.065	0.066	0.004	0.011	0.000	0.563
dam break 147k PP	5.360	1.209	0.355	1.154	0.307	0.038	8.424
dam break 147k SPP (CPU)	5.352	0.819	0.365	1.145	0.307	0.035	8.025
fountains 320k SPP	2.002	0.083	0.077	0.007	0.021	0.000	2.19
fountains 165k adaptive SPP	1.098	0.056	0.052	0.005	0.017	0.005	1.233
fountains 320k PP	12.54	3.138	0.937	2.747	0.915	0.038	20.344
fountains 320k SPP (CPU)	12.44	1.948	0.969	2.680	1.008	0.038	19.089
sphere drop 52k SPP	0.119	0.039	0.038	0.001	0.004	0.000	0.203
sphere drop 418k SPP	0.966	0.098	0.091	0.009	0.021	0.000	1.187
sphere drop 216k adaptive SPP	0.247	0.064	0.060	0.006	0.012	0.009	0.400
sphere drop 418k PP	20.65	4.772	1.202	3.287	0.880	0.025	30.824
sphere drop 418k SPP (CPU)	20.56	2.440	1.253	3.227	0.865	0.026	28.379
double dam break 1M SPP	2.569	0.119	0.168	0.016	0.041	0.000	2.913
waves 576k SPP	1.545	0.109	0.105	0.009	0.034	0.000	1.804

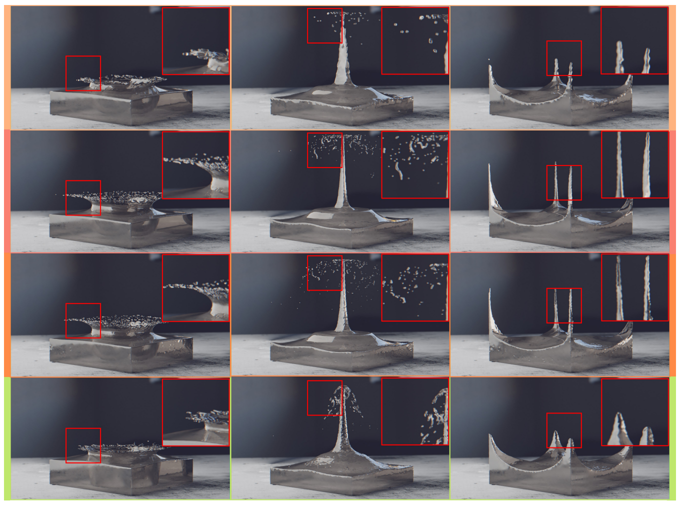
Fig. 11. The rendered results of the sphere drop scenario. From top to bottom: 52k staggered Power Particles, 418k staggered Power Particles, 216k adaptive staggered Power Particles, 418k Power Particles. The adaptive particle sampling helps demonstrating similar visual details with merely half the particles. Additionally, the staggered scheme manages to produce more abundant ricocheting droplets in contrast to particle clusters with the original Power Particles. In this test, the 418k staggered Power Particles and 216k adaptive staggered Power Particles are respectively 25 and 76 times faster than the 418k Power Particles.

even with a relatively small number of particles. The extensive experiments have validated both efficacy and adequacy of our research, and the sharp comparisons with the previous method have also demonstrated the advantages of our method computationally and visually.

Our present work still has much room for improvement. For example, in the experiments of the colliding fountains scenario, despite the fact that the adaptivity we incorporate enhances the visual results significantly, the holes within the water sheet have yet to be completely eliminated. A possible solution for remedy is to make use of some geometrical techniques such as anisotropic particle repositioning [7], [51]. Another thing to improve would be to exploit the cut-cell boundary handling by Brochu et al. [36] to better capture the surface details. Besides, we are considering coupling the Power Particles with the SPH method and grid-based methods, aiming at taking the advantages of all in a single simulation framework.

Acknowledgments

This project was partially supported by the NSFC Grants (61872347, 61532002, 61661146002), Special Plan for the Development of Distinguished Young Scientists of ISCAS (Y8RC535018), USA NSF IIS-1715985, and CAS Key Research Program of Frontier Sciences (QYZDY-SSW-JSC041).

References

[1] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner, "SPH fluids in computer graphics," in *Eurographics 2014 - State of the Art Reports, Strasbourg, France, April 7-11, 2014*, S. Lefebvre and M. Spagnuolo, Eds. Eurographics Association, 2014, pp. 21–42.
[2] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath, and M. Teschner, "Implicit incompressible SPH," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 3, pp. 426–435, 2014.
[3] N. Chentanez and M. Müller, "Mass-conserving eulerian liquid simulation," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 1, pp. 17–29, 2014.
[4] D. Gerszewski and A. W. Bargteil, "Physics-based animation of large-scale splashing liquids," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 185:1–185:6, Nov. 2013.
[5] J. Stam, "Stable fluids," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1999, Los Angeles, CA, USA, August 8-13, 1999*, W. N. Waggenspack, Ed. ACM, 1999, pp. 121–128.
[6] Y. Zhu and R. Bridson, "Animating sand as a fluid," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 965–972, Jul. 2005.
[7] R. Ando, N. Thürey, and R. Tsuruno, "Preserving fluid sheets with adaptively sampled anisotropic particles," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 8, pp. 1202–1214, 2012.
[8] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin, "The affine particle-in-cell method," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 51:1–51:10, Jul. 2015.
[9] C. Batty and R. Bridson, "Accurate viscous free surfaces for buckling, coiling, and rotating liquids," in *Proceedings of the 2008 Eurographics/ACM SIGGRAPH Symposium on Computer Animation, SCA 2008, Dublin, Ireland, 2008*, M. H. Gross and D. L. James, Eds. Eurographics Association, 2008, pp. 219–228.
[10] L. Boyd and R. Bridson, "Multiflip for energetic two-phase fluid simulation," *ACM Trans. Graph.*, vol. 31, no. 2, pp. 16:1–16:12, Apr. 2012.
[11] K. Raveendran, C. Wojtan, and G. Turk, "Hybrid smoothed particle hydrodynamics," in *Proceedings of the 2011 Eurographics/ACM SIGGRAPH Symposium on Computer Animation, SCA 2011, Vancouver, BC, Canada, 2011*, A. W. Bargteil and M. van de Panne, Eds. Eurographics Association, 2011, pp. 33–42.

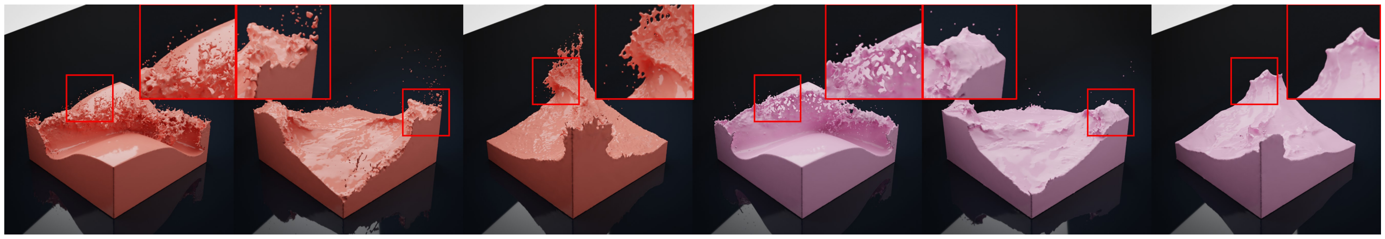

Fig. 12. The rendered results of the double dam break scenario with 1 million staggered Power Particles. The left sequence is simulated with our standard method and the right one with the pressure projection proposed in Sin et al. [35]. Noticeably, our method demonstrates less numerical damping in this comparison.

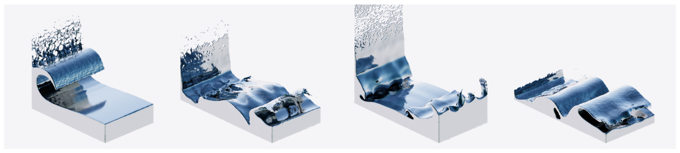

Fig. 13. The rendered results of the waves scenario with 576k staggered Power Particles.

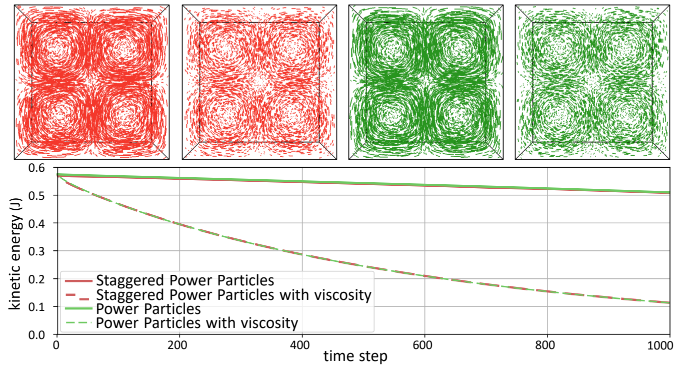

Fig. 14. The rotating vortices scenario. Both the staggered Power Particles and the original Power Particles are tested with or without viscosity ($\nu = 0.005$), and the resulting energy profiles are comparable. Top: the streamlines after 1000 timesteps. Bottom: the corresponding energy statistics.

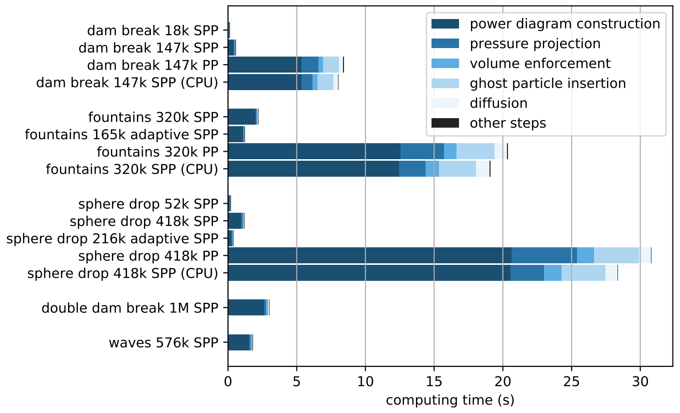

Fig. 15. The comparison of simulation overhead per time step between the staggered Power Particles and the original Power Particles [12]. Not only our GPU-based algorithms provide a significant improvement on efficiency, our staggered pressure solver is also much faster on CPU.

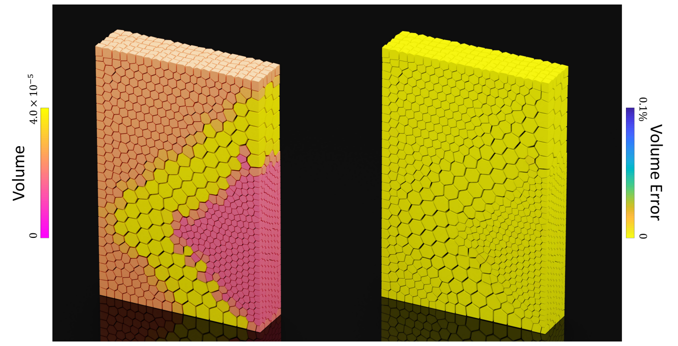

Fig. 16. The 1000-th timestep of a hydrostatic test with various cell volumes. The volume of each cell is shown on the left, and the percentage error, with respect to the initial volume, is demonstrated on the right.

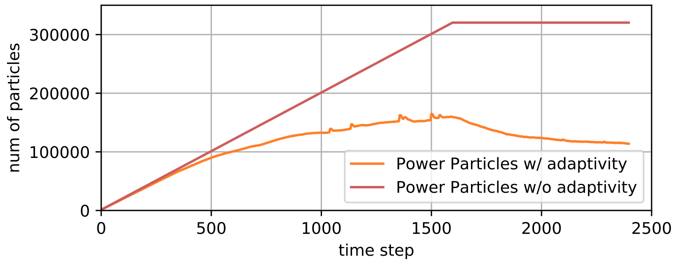

Fig. 17. The statistics of particle number in the colliding fountains scenario (Figure 10) with adaptivity enabled and disabled. The adaptive particle sampling allows using about half the particles to achieve the similar surface effect.

[12] F. de Goes, C. Wallez, J. Huang, D. Pavlov, and M. Desbrun, "Power particles: An incompressible fluid solver based on power diagrams," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 50:1–50:11, Jul. 2015.
[13] P. Alliez, A. Fabri, and E. Fogel, "Computational geometry algorithms library," in *ACM SIGGRAPH 2008 Classes*, ser. SIGGRAPH '08. New York, NY, USA: ACM, 2008, pp. 22:1–22:358.
[14] C. Rycroft, "Voro++: A three-dimensional voronoi cell library in c++," *Chaos: An Interdisciplinary Journal of Non-linear Science*, vol. 19, no. 4, 2009.
[15] J. J. Monaghan, "Smoothed particle hydrodynamics," *Annual review of astronomy and astrophysics*, vol. 30, no. 1, pp. 543–574, 1992.
[16] M. Müller, D. Charypar, and M. H. Gross, "Particle-based fluid simulation for interactive applications," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, San Diego, CA, USA, July 26-27, 2003*, R. Parent, K. Singh, D. E. Breen, and M. C. Lin, Eds. The Eurographics Association, 2003, pp. 154–159.
[17] M. Becker and M. Teschner, "Weakly compressible SPH for free surface flows," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2007, San Diego, California, USA, August 2-4, 2007*, M. Gleicher and D. Thalmann, Eds. Eurographics Association, 2007, pp. 209–217.
[18] B. Solenthaler and R. Pajarola, "Predictive-corrective incompressible sph," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 40:1–40:6, Jul. 2009.

[19] M. Macklin and M. Müller, "Position based fluids," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 104:1–104:12, Jul. 2013.
[20] J. Bender and D. Koschier, "Divergence-free smoothed particle hydrodynamics," in *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation, SCA 2015, Los Angeles, CA, USA, August 7-9, 2015*, J. Barbic and Z. Deng, Eds. ACM, 2015, pp. 147–155.
[21] P. W. Randles and L. D. Libersky, "Normalized sph with stress points," *International Journal for Numerical Methods in Engineering*, vol. 48, pp. 1445–1462, Aug. 2000.
[22] X. He, N. Liu, G. Wang, F. Zhang, S. Li, S. Shao, and H. Wang, "Staggered meshless solid-fluid coupling," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 149:1–149:12, 2012.
[23] F. H. Harlow and J. E. Welch, "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface," *Physics of Fluids*, vol. 8, pp. 2182–2189, Dec. 1965.
[24] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating water and smoke with an octree data structure," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 457–462, 2004.
[25] B. E. Feldman, J. F. O'Brien, and B. M. Klingner, "Animating gases with hybrid meshes," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 904–909, 2005.
[26] N. Chentanez, B. E. Feldman, F. Labelle, J. F. O'Brien, and J. R. Shewchuk, "Liquid simulation on lattice-based tetrahedral meshes," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2007, San Diego, California, USA, August 2-4, 2007*, M. Gleicher and D. Thalmann, Eds. Eurographics Association, 2007, pp. 219–228.
[27] B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'Brien, "Fluid animation with dynamic meshes," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 820–825, 2006.
[28] P. Mullen, K. Crane, D. Pavlov, Y. Tong, and M. Desbrun, "Energy-preserving integrators for fluid animation," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 38:1–38:8, 2009.
[29] W. Hong, D. H. House, and J. Keyser, "Adaptive particles for incompressible fluid simulation," *The Visual Computer*, vol. 24, no. 7-9, pp. 535–543, 2008.
[30] R. Ando, N. Thürey, and C. Wojtan, "Highly adaptive liquid simulations on tetrahedral meshes," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 103:1–103:10, Jul. 2013.
[31] F. Ferstl, R. Ando, C. Wojtan, R. Westermann, and N. Thuerey, "Narrow band FLIP for liquid simulations," *Comput. Graph. Forum*, vol. 35, no. 2, pp. 225–232, 2016.
[32] J. Cornelis, M. Ihmsen, A. Peer, and M. Teschner, "IISPH-FLIP for incompressible fluids," *Comput. Graph. Forum*, vol. 33, no. 2, pp. 255–262, 2014.
[33] F. Losasso, J. O. Talton, N. Kwatra, and R. Fedkiw, "Two-way coupled SPH and particle level set fluid simulation," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 4, pp. 797–804, 2008.
[34] N. Chentanez, M. Müller, and T. Kim, "Coupling 3d eulerian, heightfield and particle methods for interactive simulation of large scale liquid phenomena," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 10, pp. 1116–1128, 2015.
[35] F. Sin, A. W. Bargteil, and J. K. Hodgins, "A point-based method for animating incompressible flow," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2009, New Orleans, Louisiana, USA, August 1-2, 2009*, D. W. Fellner and S. N. Spencer, Eds. ACM, 2009, pp. 247–255.
[36] T. Brochu, C. Batty, and R. Bridson, "Matching fluid simulation elements to surface geometry and topology," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 47:1–47:9, 2010.
[37] R. E. English, L. Qiu, Y. Yu, and R. Fedkiw, "Chimera grids for water simulation," in *The ACM SIGGRAPH / Eurographics Symposium on Computer Animation, SCA '13, Anaheim, CA, USA, July 19-21, 2013*, J. Chai, Y. Yu, T. Kim, and R. W. Sumner, Eds. ACM, 2013, pp. 85–94.
[38] O. Busaryev, T. K. Dey, H. Wang, and Z. Ren, "Animating bubble interactions in a liquid foam," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 63:1–63:8, Jul. 2012.
[39] M. Aanjaneya, M. Gao, H. Liu, C. Batty, and E. Sifakis, "Power diagrams and sparse paged grids for high resolution adaptive liquids," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 140:1–140:12, Jul. 2017.
[40] N. Ray, D. Sokolov, S. Lefebvre, and B. Lévy, "Meshless voronoi on the GPU," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 265:1–265:12, 2018.
[41] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas, "Adaptively sampled particle fluids," *ACM Trans. Graph.*, vol. 26, no. 3, p. 48, 2007.
[42] R. Keiser, B. Adams, P. Dutre, L. J. Guibas, and M. Pauly, "Multiresolution particle-based fluids," *Tech. rep., ETH Zurich*, 2006.
[43] Y. Zhang, B. Solenthaler, and R. Pajarola, "Adaptive sampling and rendering of fluids on the GPU," in *Proceedings of the Eurographics / IEEE VGTC Workshop on Volume Graphics 2008, Los Angeles, California, USA, 2008*, H. Hege, D. H. Laidlaw, R. Pajarola, and O. G. Staadt, Eds. Eurographics Association, 2008, pp. 137–146.
[44] J. Orthmann and A. Kolb, "Temporal blending for adaptive SPH," *Comput. Graph. Forum*, vol. 31, no. 8, pp. 2436–2449, 2012.
[45] R. Winchenbach, H. Hochstetter, and A. Kolb, "Infinite continuous adaptivity for incompressible sph," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 102:1–102:10, Jul. 2017.
[46] B. Solenthaler and M. Gross, "Two-scale particle simulation," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 81:1–81:8, Jul. 2011.
[47] C. J. Horvath and B. Solenthaler, "Mass preserving multi-scale sph," *Pixar Technical Memo 13-04, Pixar Animation Studios*, 2013.
[48] F. Aurenhammer, "Power diagrams: Properties, algorithms and applications," *SIAM J. Comput.*, vol. 16, no. 1, pp. 78–96, 1987.
[49] S. Elcott, Y. Tong, E. Kanso, P. Schröder, and M. Desbrun, "Stable, circulation-preserving, simplicial fluids," *ACM Trans. Graph.*, vol. 26, no. 1, p. 4, 2007.
[50] D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw, "Using the particle level set method and a second order accurate pressure boundary condition for free surface flows," in *ASME/JSME 2003 4th Joint Fluids Summer Engineering Conference*, 2003, pp. 337–342.
[51] X. He, H. Wang, F. Zhang, H. Wang, G. Wang, and K. Zhou, "Robust simulation of sparsely sampled thin features in sph-based free surface flows," *ACM Trans. Graph.*, vol. 34, no. 1, pp. 7:1–7:9, Dec. 2014.

Xiao Zhai received his BS degree in computer science and engineering from Beihang University in 2013. He is currently a PhD candidate at State Key Laboratory of Virtual Reality Technology and Systems, Beihang University. His research interests include physics-based fluid simulation, data-driven fluid animation, and all the relevant topics in computer graphics.

Fei Hou received his PhD degree in computer science from Beihang University in 2012. He is currently a research associate professor of Institute of Software, Chinese Academy of Sciences. He was a Postdoctoral researcher at Beihang University from 2012 to 2014 and a research fellow in School of Computer Science and Engineering, Nanyang Technological University from 2014 to 2017. His research interests include geometry processing, image-based modeling, data vectorization, medical image processing, etc.

Hong Qin received his BS and MS degrees in computer science from Peking University, and his PhD degree in computer science from the University of Toronto. He is a professor of computer science in Department of Computer Science at Stony Brook University. His research interests include geometric and solid modeling, graphics, physics-based modeling and simulation, computer-aided geometric design, human-computer interaction, visualization, and scientific computing. Currently, he serves as an associate editor for The Visual Computer, Graphical Models, and Journal of Computer Science and Technology. He is a senior member of the IEEE and the IEEE Computer Society.

Aimin Hao received his BS, MS, and PhD degrees in computer science at Beihang University. He is a professor in School of Computer Science and Engineering, Beihang University and the associate director of State Key Laboratory of Virtual Reality Technology and Systems. His research interests include virtual reality, computer simulation, computer graphics, geometric modeling, image processing, and computer vision.