Turbulent Details Simulation for SPH Fluids via Vorticity Refinement

Liu, Sinuo; Wang, Xiaokun; Ban, Xiaojuan; Xu, Yanrui; Zhou, Jing; Kosinka, Jiří; Telea, Alexandru C.

Published in:
COMPUTER GRAPHICS FORUM

DOI:
10.1111/cgf.14095

Publication date:
2021

DOI: 10.1111/cgf.14095

COMPUTER GRAPHICS *forum*
Volume 40 (2021), number 1 pp. 54–67

Turbulent Details Simulation for SPH Fluids via Vorticity Refinement

Sinuo Liu,[2,†] Xiaokun Wang,[1,†] Xiaojuan Ban,[1] Yanrui Xu,[1] Jing Zhou,[2] Jiří Kosinka[3] and Alexandru C. Telea[4]

[1]Institute of Artificial Intelligence, University of Science & Technology, Beijing, China
xuyanruiedw@me.com
[2]School of Computer & Communication Engineering, University of Science & Technology, Beijing, China
787666673@qq.com,zhoujing_me_cn@hotmail.com
[3]Bernoulli Institute, University of Groningen, Groningen, the Netherlands
j.kosinka@rug.nl
[4]Department of Information and Computing Sciences, Utrecht University, Utrecht, the Netherlands
a.c.telea@rug.nl

Abstract

A major issue in smoothed particle hydrodynamics (SPH) approaches is the numerical dissipation during the projection process, especially under coarse discretizations. High-frequency details, such as turbulence and vortices, are smoothed out, leading to unrealistic results. To address this issue, we introduce a vorticity refinement (VR) solver for SPH fluids with negligible computational overhead. In this method, the numerical dissipation of the vorticity field is recovered by the difference between the theoretical and the actual vorticity, so as to enhance turbulence details. Instead of solving the Biot-Savart integrals, a stream function, which is easier and more efficient to solve, is used to relate the vorticity field to the velocity field. We obtain turbulence effects of different intensity levels by changing an adjustable parameter. Since the vorticity field is enhanced according to the curl field, our method can not only amplify existing vortices, but also capture additional turbulence. Our VR solver is straightforward to implement and can be easily integrated into existing SPH methods.

Keywords: physically based animation, animation, fluid modeling, animation, particle systems, animation

ACM CCS: • Computing methodologies → Physical simulation

1. Introduction

Fluid simulation is a hot topic in computer graphics, with huge research and application demands. Within this context, the smoothed particle hydrodynamics (SPH) method simulates fluids with large deformations accurately and efficiently, showing abundant details and vivid motion. In the past decades, several solutions have been proposed to enforce incompressibility [SP09, ICS*14, BK15]. However, numerical dissipation problems still remain and cause a significant loss of turbulence details [JST17, FGG*17]. For instance, vorticity dissipation is one of the major issues causing the loss of details on the fluid surface and in overall dynamic effects [KBST19].

To maintain complex turbulence and vortex details on the fluid surface, some methods proposed to increase the apparent resolution by seeding over surface points [MBT*15], or use an adaptive volumetric mesh for grid-based fluids [EB14]. However, these methods only add details over a coarse discretization, without considering the inner volume. Vorticity confinement (VC) methods add vortices from the perspective of the entire flow field [FSJ01, MM13] to recover dissipated details. However, VC methods tend to add more energy than is dissipated, and can amplify only existing vortices. Lagrangian vortex methods, such as vortex particles [PK05] and vortex filaments [WP10], have been used to effectively simulate turbulent fluids. While these methods maintain a divergence-free velocity field and have theoretically no numerical dissipation, they require solving the equivalent of three Poisson equations to obtain velocity from vorticity, which is computationally expensive.

To alleviate the above-mentioned problems and obtain more realistic turbulent flows, we introduce a turbulence refinement scheme by correcting the vorticity field. In continuum mechanics, vorticity is a pseudovector field that describes the local spinning motion of a continuum. It can be defined as the curl of the fluid's

[†]Co-first author, contributed equally.

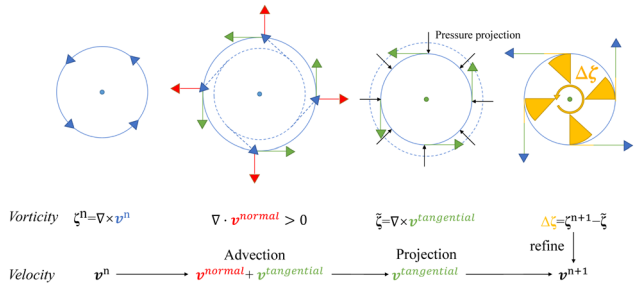


Figure 2: *This diagram shows how the vorticity dissipates in the advection-projection process. Linear velocity at time step n splits into normal (red) and tangential (green) components during the advection procedure. Next, projection eliminates the normal component using the pressure force to keep the flow divergence-free.*

velocity field. Like the divergence error issue mentioned in the DFSPH method [BK15], vorticity dissipation also reduces the realism of simulations. To date, the kinetic energy from the vorticity field could be transformed into positive divergence, causing the loss of surface details and of overall dynamic motion [ZBG15]. State-of-the-art SPH approaches for fluid simulation cannot solve this problem completely.

During the advection-projection process, the advection step maps the original velocity field into a rotational part and a divergent part, after which the pressure projection removes the divergent part, leaving only the rotational part. The angular momentum is therefore lost in the simulation, with the effect becoming worse as the time step size increases; see Fig. 2, the orange part of the diagram. To alleviate this, we use the accurate vorticity field derived from the curl of the Navier-Stokes equations to correct the linear velocity for each particle; see Fig. 2, the green part of the diagram. Moreover, we use a stream function to refine the velocity using a reasonable augmentation of the vorticity field which can restore vivid yet controllable vortices and turbulence effects (as shown in Figs. 1 and 3 among others). Previous related work [XBP*19] looked into correcting the velocity field through the vorticity recovered from the kinematic viscosity by increasing the vorticity field proportionally by the energy dissipated, which is based on the rotational kinetic energy. In contrast, we focus on getting the ideal vorticity field directly from the curl of the Navier-Stokes equations, which is a more physically reasonable model.

Summarizing, the main contributions of this paper are:

- A vivid turbulence-details generation method that recovers numerical dissipation through vorticity field correction;
- A novel vorticity-based constraint and stream function solution for simulating turbulence;
- An orthogonal solver for the SPH fluid framework with turbulence simulation that can be easily integrated into other particle-based methods and fluid solvers.

The remainder of the paper is organized as follows. Section 2 gives an overview of related work. Section 3 discusses the accuracy of numerical calculations in SPH. Our vorticity refinement scheme is presented in Section 4. Section 5 presents and discusses our experimental results. Finally, Section 6 concludes the paper.

2. Related Work

Fluid simulation is a well researched topic in computer graphics. Early works on this topic include [Mon94, FM96, Sta99, MCG03]. For recent overviews, we refer to Bridson's book [Bri15] and the state-of-the-art report of Koschier *et al.* [KBST19]. We further discuss more specific work related to our context, namely

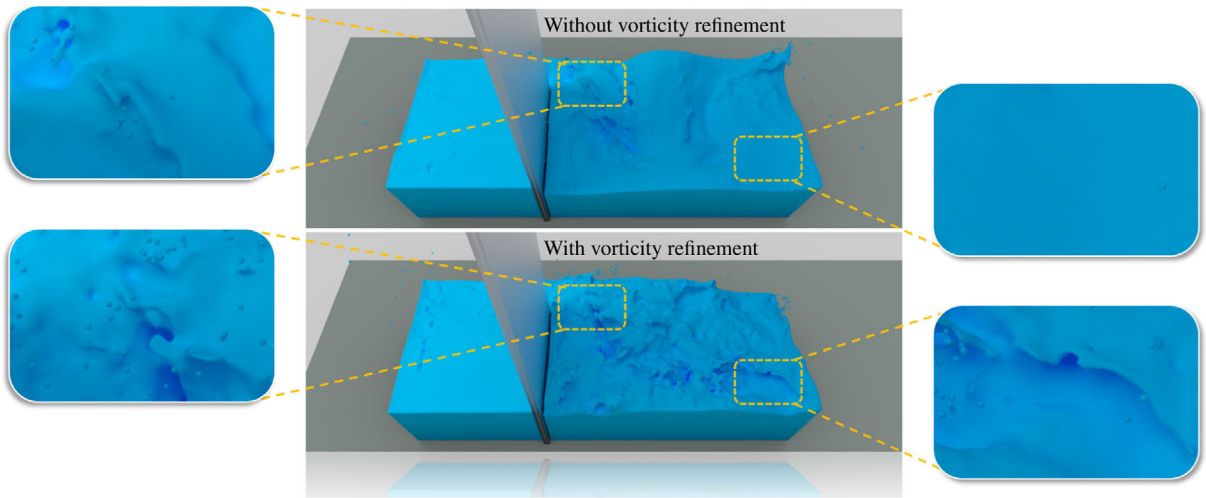
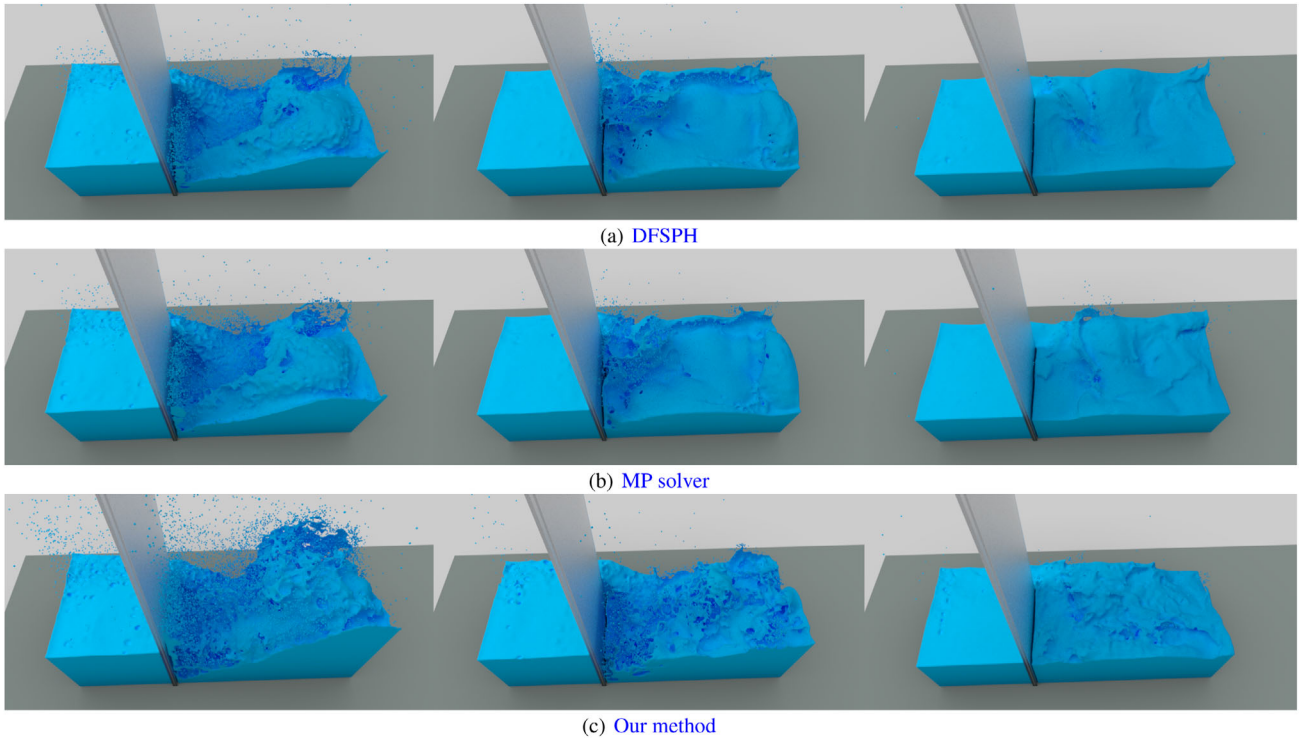


Figure 1: *Our vorticity refinement (VR) solver applied to an DFSPH [BK15] simulation (1.18M particles). In this scene, a breaking dam collides with a board, creating turbulence. Zoom-ins compare the surface under DFSPH without (top) and with (bottom) our VR solver. The result shows that our method better captures turbulence details.*



(a) DFSPH

(b) MP solver

(c) Our method

Figure 3: *A breaking dam collides with a board (1.18M particles). (a) Only few vortex effects are formed using DFSPH. (b) The MP solver enhances the simulation result to some extent. (c) Our method greatly improves the turbulence details, and the surface details are richer than with the MP solver. The MP solver and our solver achieve different styles.*

SPH-based fluid simulation (Section 2.1) and turbulence simulation (Section 2.2).

2.1. Incompressibility and numerical accuracy in SPH fluid simulations

Monaghan simulated free surface flows with SPH [Mon94], which laid the foundation for fluid simulation. Later, Muller *et al.* [MCG03] proposed to simulate fluids using the ideal gas state equation with surface tension and viscosity forces, but without full incompressibility. An improved weakly-compressible SPH (WCSPH) method was proposed by Becker and Teschner [BT07]. The use of the stiff equation of state (EOS) significantly increased realistic effects, but the efficiency of such methods is limited by the size of the used time step. To further enforce incompressibility and improve numerical accuracy, much effort has been invested into implicit pressure solvers. Previous approaches can be categorized as methods that project particle positions onto an incompressible state using iterative EOS solvers, and pressure projection methods [IOS*14], as follows.

He *et al.* [HLL*12] and Solenthaler and Pajarola [SP09] proposed predictive-corrective approaches that iteratively project particle positions onto an incompressible state. This is also done in position-based fluids (PBF) [MM13]. However, PBF avoids accumulating pressure or pressure forces that eventually update the velocity and the position. Ihmsen *et al.* [ICS*14] proposed implicit incompressible SPH (IISPH) following the strategy of pressure projections. Separately, Bender and Koschier [BK15] proposed a method that enforces a low compression (0.01%) and a divergence-free velocity constraint (DFSPH). Among all the variants of the SPH method, the typical advection-projection models are PCISPH [SP09], IISPH [ICS*14] and DFSPH [BK15]. In this paper, we use the DFSPH approach as a baseline for comparisons of computational efficiency and stability.

It is well known that SPH approaches suffer from numerical dissipation problems, especially for coarse discretizations [Mon94, dGWH*15, BKKW18]. Ihmsen *et al.* [IOS*14] pointed out that SPH results in undesired dissipation and high-frequency features are smoothed out. Therefore, avoiding dissipation for turbulence in fluid simulation is needed to improve visual realism.

2.2. Restoring turbulence in fluid simulation

Restoring high-frequency details has been an important challenge in fluid simulation since its very beginning [KTJG08, JSMF*18]. For Eulerian approaches, Stam's scheme [Sta99] first achieved realistic and real-time fluid simulation on consumer-grade graphics hardware. However, the first-order accuracy in both time and space makes this method (and other extensions thereof) suffer from serious numerical dissipation. Kim *et al.* [KLLR05] proposed a higher-order approximation. Jonas *et al.* [ZNT18] proposed an advection-reflection solver for detail-preserving fluid animation

which leads to two orders of magnitude reduction in energy loss. Rahul *et al.* [NZT19] then established a connection between this method and the implicit midpoint time integration scheme, and presented a simple improvement to obtain an advection-reflection scheme with second-order accuracy in time.

Hybrid particle-grid methods were subsequently proposed to further reduce numerical dissipation. Zhu and Bridsons' FLIP method for incompressible flow [ZB05] significantly eliminates the dissipation in advection. Jiang *et al.* [JSS*15] successfully restore most of the rotational motion using a hybrid method.

Although the general simulation methods mentioned above can handle numerical dissipation on a macroscopic level, both Eulerian and Lagrangian approaches face challenges when simulating high-frequency details such as turbulence. Therefore, methods specifically designed for refining turbulent details have emerged. These can be classified into three categories: up-res methods, vorticity confinement methods, and Lagrangian vortex methods [BKKW18], as follows.

Up-res methods add high-frequency details over a coarse discretization. Mercier *et al.* [MBT*15] proposed a post-processing method to apply fine turbulence over particle-based fluid surfaces. High-resolution surface points are seeded after curvature evaluation, and the detailed surface waves are then evolved over coarse particles. Edwards and Bridson [EB14] proposed an adaptive volumetric-mesh method for grid-based fluids by using the adaptive discontinuous Galerkin method. Machine learning methods such as Convolutional Neural Networks (CNNs) [CT17] have been applied in fluid simulation to synthesize high-resolution turbulence on rough simulation results based on a high-resolution source. However, training CNNs is time-consuming and often requires delicate hyperparameter tuning. Overall, up-res methods can typically improve only surface effects.

Vorticity confinement methods aim to find existing vortices and recover their dissipation. A new forcing term is added to increase the velocity of target positions, and to enforce the rotation, of the vortex. Lentine *et al.* [LAF11] improved vorticity confinement to be both energy conserving and momentum conserving. Jang *et al.* [JKB*10] used multi-level vorticity confinement to acquire better results. Macklin and Muller [MM13] presented a simple method to amplify the existing vorticity through accelerating particles using SPH. Overall, vorticity confinement methods provide a simple way for preserving vortices, but are in general unable to create additional turbulence details. Moreover, they are prone to adding excessive energy to the system so that energy conservation is likely to be violated, leading to unstable results.

Lagrangian vortex methods build on the vorticity representation of the Navier-Stokes equations [PK05], which have less numerical dissipation and more divergence retention than vorticity confinement methods. These methods can be applied to particles [WLB*20], curves [AN05], filaments [EWPT17], and even surfaces [WP10]. Yet, boundaries, such as non-rigid obstacles and free surfaces, are difficult to handle. Zhu *et al.* [ZYF10] proposed to simulate vortex details around moving objects using Eulerian grids. Golas *et al.* [GNS*12] also treated boundaries of an Eulerian grid to solve this issue. A disadvantage of these methods is that the velocity field has to be recovered by solving the Biot-Savart integrals or a vector-valued Poisson equation. Recently, Bender *et al.* [BKKW18] introduced the MicroPolar fluid solver (MP solver) for inviscid fluids in order to capture the micro-rotation of fluid particles, achieving impressive visual turbulent features. Wang *et al.* [WLB*20] proposed a turbulence refinement method based on the Rankine vortex model for particle-based simulation. Zhang *et al.* [ZBG15] proposed an Integrated Vorticity of Convective Kinematics (IVOCK) method to restore dissipated energy by measuring vorticity loss in advection. This method can cheaply capture much of the lost details for smoke and fire, but does not work well for liquid simulations. In [ZBG15], only the vorticity dissipated during the advection step is considered. The refined linear velocity in their paper is the velocity after the advection step. This velocity is then further affected by viscosity and the projection step. Viscosity may become another source of vorticity dissipation and the pressure force may introduce vorticity errors into the velocity field after the projection step. Although it maintains an incompressible density field, it is not necessarily divergence-free.

Our method is inspired by the idea of stream functions [ZBG15], extended to Lagrangian fluid simulations. This allows us to efficiently derive velocity refinement from the vorticity field. Recovering turbulence from the curl form of the Navier-Stokes equations has a long history. In 2005, Park and Kim [PK05] gave the governing equations of the vortex method and introduced the concept of the stream function. [ZBG15] and our work, among many other vortex methods, utilize this concept to reduce numerical dissipation during simulation. In our method, we derive the dissipated vorticity during the whole advection-projection step in the SPH approach. This can be easily done with little extra computation overhead. With respect to the concept of the stream function, we carry out the Biot-Savart summation process within smoothing length, which makes it less accurate but more efficient than [ZBG15]; we show this to be sufficient to maintain stability. This is because, theoretically, the refined velocity is the curl of the stream function, and any curl of a vector field is divergence-free. Moreover, we implemented our method using DF-SPH (Divergence-free SPH), which includes an extra divergence-correction solver, thereby eliminating possible errors caused by the summation process. Moreover, we do not need to solve the Biot-Savart integrals or a vector-valued Poisson equation. In contrast to the MP solver [BKKW18], in which the motion equation is obtained from the MicroPolar model and discretized with SPH, we derive the vorticity equation from the curl of the Navier-Stokes equations, and recover velocity from the vorticity field using stream functions. Our results show that our method can not only enhance existing vortices, but also generate turbulence at potential locations of new vortices.

3. SPH discretization for fluid simulation

Traditional Lagrangian-based fluid simulations use the fluid governing equations, the Navier-Stokes equations, to solve for the position and velocity of each fluid particle. The acceleration of the fluid particles is obtained by the combination of pressure $\mathbf{a}_{pres}$, viscous force $\mathbf{a}_{vis}$, and gravity $\mathbf{a}_g$ as

$$\frac{D\boldsymbol{v}}{Dt} = \mathbf{a}_{pres} + \mathbf{a}_{vis} + \mathbf{a}_g = -\frac{1}{\rho}\nabla p + \nu_v \nabla^2 \boldsymbol{v} + \mathbf{g}, \tag{1}$$

where D denotes the material derivative, ρ is the density of the fluid, p represents pressure, $\boldsymbol{v}$ is velocity, ν_v is the kinematic viscosity coefficient, a value that characterizes various fluid types (set to $\nu_v = 0.05$ in our experiments), $\mathbf{g}$ is the gravitational acceleration, and ∇^2 denotes the Laplace operator.

The SPH approach can be used to discretize the Navier-Stokes equations to numerically solve them. The continuous physical values in space can be discretized using a smooth kernel W as in

$$\mathbf{A}(\boldsymbol{x}_i) = \sum_{\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \le h} m(\boldsymbol{x}_j) \frac{\mathbf{A}(\boldsymbol{x}_j)}{\rho(\boldsymbol{x}_j)} W\big(\boldsymbol{x}_i - \boldsymbol{x}_j, h\big) \tag{2}$$

with $\mathbf{A}(\boldsymbol{x}_i)$ being a certain quantity associated with particle i at location $\boldsymbol{x}_i$. This quantity can be interpolated from the values of neighbour particles, indexed by j, within a support radius h. The quantities m and ρ stand for mass and density, respectively. To simplify notation, we next use the shorthand $\mathbf{A}_i$ to denote the quantity $\mathbf{A}$ evaluated at position $\boldsymbol{x}_i$.

The density of a fluid can be derived by simply replacing $\mathbf{A}$ by ρ. In our work, we use the *cubic spline kernel* [Mon85]:

$$W_{ij} = \frac{1}{\pi h^3} \begin{cases} 1 - \frac{2}{3}x^2 + \frac{3}{4}x^3 & 0 \le x \le 1 \\ \frac{1}{4}(2-x)^3 & 1 \le x \le 2 \\ 0 & x \ge 2 \end{cases},$$

where $x = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|/h$ and W_{ij} is a short form of $W(\boldsymbol{x}_i - \boldsymbol{x}_j, h)$. To obtain a better accuracy of the approximation of the divergence of velocity, the gradient and the curl of velocity, we apply the difference form of the SPH discretization as:

$$\nabla \otimes \mathbf{A} = \sum_j \frac{m_j}{\rho_j} \big(\mathbf{A}_j - \mathbf{A}_i\big) \otimes \nabla W_{ij}, \tag{3}$$

which expresses the gradient (∇A), divergence ($\nabla \cdot A$), and curl ($\nabla \times A$, in which case the right hand side is negative) of A. Since the second derivative is often sensitive to particle disorder and sign changes inside the support radius h, we use artificial viscosity to approximate the Laplacian as follows [KBST19]:

$$\nabla^2 \mathbf{A}(\boldsymbol{x}_i) = 2(d+2) \sum_j \frac{m_j}{\rho_j} \frac{\mathbf{A}_{ij} \cdot \boldsymbol{x}_{ij}}{\boldsymbol{x}_{ij}^2 + 0.01h^2} \nabla W_{ij}, \tag{4}$$

where d is the space dimension (in our case equal to 3), $\boldsymbol{x}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$, and $\mathbf{A}_{ij} = \mathbf{A}_i - \mathbf{A}_j$.

Simulating incompressible fluids in DFSPH follows several steps, including advection and projection, and an extra divergence correction step which is applied to keep the velocity field divergence-free. The whole process is summarized in Algorithm 1, where Δt denotes the size of one time step, $\mathbf{a}_{adv} = \mathbf{a}_{vis} + \mathbf{a}_g$, and $\mathbf{a}_{proj}$ and $\mathbf{a}_{correct}$ are the change rate of velocity derived form the implicit pressure field to satisfy the incompressibility and divergence-free conditions accordingly. Further, ρ_0 is the rest density of the fluid, and ρ_{err}, div_{err}, n, and n' are user-specified scalar values as thresholds.

Algorithm 1. Advection-projection with divergence correction

Advectiom process:
  compute $\mathbf{a}_{adv}$
  $\tilde{\boldsymbol{v}} := \boldsymbol{v}^n + \Delta t \mathbf{a}_{adv}$
  $\tilde{\boldsymbol{x}} := \boldsymbol{x}^n + \Delta t \tilde{\boldsymbol{v}}$
  $\tilde{\rho} := positionBasedDensity(\tilde{\boldsymbol{x}})$
Projection process:
  while $(\tilde{\rho} - \rho_0) > \rho_{err}$ || $numberOfIterations < n$
    $p := positionBasedPressure(\tilde{\boldsymbol{x}})$
    $\mathbf{a}_{proj} := pressureBasedForce(p)$
    $\tilde{\boldsymbol{v}} := \tilde{\boldsymbol{v}} + \Delta t \mathbf{a}_{proj}$
    $\tilde{\rho} := positionBasedDensity(\tilde{\boldsymbol{x}} + \Delta t \tilde{\boldsymbol{v}})$
  $\boldsymbol{x}^{n+1} = \tilde{\boldsymbol{x}}$
Divergence correction process:
  while $(\nabla \cdot \tilde{\boldsymbol{v}}) > div_{err}$ || $numberOfIterations < n'$
    $p := velocityBasedPressure(\tilde{\boldsymbol{v}})$
    $\mathbf{a}_{correct} := pressureBasedForce(p)$
    $\tilde{\boldsymbol{v}} := \tilde{\boldsymbol{v}} + \Delta t \mathbf{a}_{correct}$
  $\boldsymbol{v}^{n+1} := \tilde{\boldsymbol{v}}$

4. Vorticity refinement model for turbulence simulation

Our method is closely related to Lagrangian vortex methods, namely it restores the velocity field through vorticity. In our method, besides velocity $\boldsymbol{v}$, each particle has a vector vorticity attribute $\boldsymbol{\zeta}$ defined as

$$\boldsymbol{\zeta} = \nabla \times \boldsymbol{v}. \tag{5}$$

In a particle system, vorticity is a quantity used to describe the rotation of a particle. For the vorticity at the position of particle i, the value can be derived using Equation 3 as:

$$\boldsymbol{\zeta}_i = \boldsymbol{\zeta}(\boldsymbol{v}_i) = \nabla \times \boldsymbol{v}_i = \sum_j \frac{m_j}{\rho_j} (\boldsymbol{v}_i - \boldsymbol{v}_j) \times \nabla W_{ij}. \tag{6}$$

4.1. Vorticity refinement

Similarly to the divergence error issue [BK15], vorticity dissipation can also hinder the performance of a simulation. Recent SPH approaches [ICS*14, BK15] for fluid animation can only correct negative divergence of the velocity field. As a result, the kinetic energy from the vorticity field is still allowed to be transformed into positive divergence during simulation, causing the loss of surface details and overall dynamic motions, effectively violating (the discrete version of) Equation 5.

Given that the numerical dissipation of vorticity occurs between time steps, an ideal non-dissipative rate of change of vorticity is required to know the exact vorticity loss in each projection step. We achieve this through the curl of the Navier-Stokes equation (Equation 1) as:

$$\nabla \times \left(\frac{D\boldsymbol{v}}{Dt}\right) = \frac{D\boldsymbol{\zeta}}{Dt} = \boldsymbol{\zeta} \cdot \nabla \boldsymbol{v} + \nu_v \nabla^2 \boldsymbol{\zeta}, \tag{7}$$

where $\boldsymbol{\zeta} \cdot \nabla \boldsymbol{v}$ is the stretching term, which is vital for physically meaningful turbulence motion evolution. We use Equation 7 to obtain the exact non-dissipative vorticity change of fluid particles between time steps, including boundary treatment [AIA*12].

Note that $\boldsymbol{\zeta} \cdot \nabla \boldsymbol{v}$ in Equation 7 is a vector, which we compute, per coordinate, using the difference form of the SPH approximation (Equation 3) via

$$\nabla v_i^{\{x,y,z\}} = \sum_j \frac{m_j}{\rho_j}\left(v_j^{\{x,y,z\}} - v_i^{\{x,y,z\}}\right)\nabla W_{ij}, \tag{8}$$

where v_i^x is the x component of the velocity of particle with index i, and similarly for y and z. For the particle with index i, the vector $\boldsymbol{\zeta}_i \cdot \nabla \boldsymbol{v}_i$ can be thus derived as

$$\boldsymbol{\zeta}_i \cdot \nabla \boldsymbol{v}_i = \begin{pmatrix} \boldsymbol{\zeta}_i \cdot \nabla v_i^x \\ \boldsymbol{\zeta}_i \cdot \nabla v_i^y \\ \boldsymbol{\zeta}_i \cdot \nabla v_i^z \end{pmatrix}. \tag{9}$$

The Laplacian of $\boldsymbol{\zeta}$ in Eqn. 7 is derived using the artificial approximation analogous to Eqn. 4. Hence, for the particle with index i, $\nu_v \nabla^2 \boldsymbol{\zeta}_i$ can be derived as

$$\nu_v \nabla^2 \boldsymbol{\zeta}_i = 2(d+2)\nu_v \sum_j \frac{m_j}{\rho_j} \frac{\boldsymbol{\zeta}_{ij} \cdot \boldsymbol{x}_{ij}}{\boldsymbol{x}_{ij}{}^2 + 0.01h^2} \nabla W_{ij}. \tag{10}$$

According to Equation 7, the ideal change of the vorticity field with respect to time, i.e. from time t^n to t^{n+1}, is:

$$\boldsymbol{\zeta}^{n+1} = \boldsymbol{\zeta}^n + \Delta t \frac{D\boldsymbol{\zeta}^n}{Dt}, \tag{11}$$

and the dissipative vorticity update is given by:

$$\Delta \boldsymbol{\zeta} = \boldsymbol{\zeta}^{n+1} - \nabla \times \tilde{\boldsymbol{v}}, \tag{12}$$

where $\tilde{\boldsymbol{v}}$ is the (intermediate) velocity, as in the last line of Algorithm 1.

We next explain how we apply the update of Equation 12. Assume that we know the velocity and position of all fluid particles at time t^n, and that the velocity at this time step is non-dissipative. We then get the velocity and position at time t^{n+1} using the DFSPH approach. Next, we compute the vorticity at the current time t^n and the next time t^{n+1}, denoted $\boldsymbol{\zeta}^n$ and $\tilde{\boldsymbol{\zeta}}$, respectively, from the velocity field using Equation 6. By our assumption, $\boldsymbol{\zeta}^n$ is ideal, but $\tilde{\boldsymbol{\zeta}}$ is dissipative due to numerical integration. Thus the ideal vorticity value for a fluid particle at t^{n+1}, denoted $\boldsymbol{\zeta}^{n+1}$, is computed based on $\boldsymbol{\zeta}^n$ and the vorticity equation (Equation 7). Hence, the dissipative vorticity value for this particle in Equation 12 can be converted to $\Delta \boldsymbol{\zeta} = \boldsymbol{\zeta}^{n+1} - \tilde{\boldsymbol{\zeta}}$. The dissipated vorticity is used to refine the velocity using the stream function, as explained next.

4.2. Solving velocity via the stream function

Inspired by [ZB14], we express the relationship between the velocity $\boldsymbol{v}$ and the vorticity $\boldsymbol{\zeta}$ using the stream function $\boldsymbol{\psi}$ as:

$$\begin{aligned} \boldsymbol{v} &= \nabla \times \boldsymbol{\psi}, \\ \nabla^2 \boldsymbol{\psi} &= -\boldsymbol{\zeta}. \end{aligned} \tag{13}$$

Green's function provides a semi-analytical solution for the stream function. The derivation from the stream function to linear velocity can be solved using Equation 3. Generalized by the Helmholtz decomposition, the stream function is the vector potential $\boldsymbol{\psi}$ of the velocity field $\boldsymbol{v}$, which can be defined as

$$\boldsymbol{\psi}(\boldsymbol{x}) = \int_{\mathbb{R}^3} \frac{\nabla \times \boldsymbol{v}(\boldsymbol{y})}{4\pi \|\boldsymbol{x} - \boldsymbol{y}\|}, \tag{14}$$

that is, the stream function $\boldsymbol{\psi}$ at position $\boldsymbol{x}$ is computed by integrating the curl of velocity $\boldsymbol{v}$ at position $\boldsymbol{y}$ over the three-dimensional space $\mathbb{R}^3$. Using Equation 5, we next discretize Equation 14 to get the stream function at the local position of particle with index i as:

$$\boldsymbol{\psi}_i = \frac{1}{4\pi} \sum_{\|\mathbf{x}_i - \mathbf{x}_j\| \le h} \frac{\Delta \boldsymbol{\zeta}_j V_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}, \tag{15}$$

where V_j stands for the volume represented by the particle with index j. Ideally, V_j should be infinitely small and all distances $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ should be considered in the summation in Equation 15. However, to limit computational overhead and its adaptability to SPH, we only include neighbouring particles within a smoothing radius h in Equation 15. This is justified by the fact that the influence of neighbour particles shrinks with distance. Although the approximation could potentially induce instability and dissipation, our results show that this improves performance without sacrificing turbulent details, as already observed, e.g. in [MCG03].

With the stream function obtained for each particle, the refined velocity for the particle with index i is derived as

$$\Delta \boldsymbol{v}_i = \sum_j \frac{m_j}{\rho_j}(\boldsymbol{\psi}_i - \boldsymbol{\psi}_j) \times \nabla W_{ij}. \tag{16}$$

To extend the flexibility of our method, we introduce an adjustment parameter $\alpha \in R$, with the default value of 1 representing the ideal vorticity refinement. It controls the amount of turbulence added to every simulation time step. Therefore, the refined linear velocity at t^{n+1} is expressed as

$$\boldsymbol{v}^{n+1} = \tilde{\boldsymbol{v}} + \alpha \Delta \boldsymbol{v}. \tag{17}$$

Since the divergence of the curl of any field is zero, the correction of linear velocity due to vorticity does not cause any further divergence deviations. Hence, our method does not contradict any SPH principles, making it easier to implement into standard Lagrangian approaches. Algorithm 2 summarizes our method, integrated with the DFSPH technique for SPH simulation; see also Fig. 2.

Algorithm 2. Our vorticity refinement (VR) solver

Compute current vorticity field:	$\boldsymbol{\zeta}^n = \nabla \times \boldsymbol{v}^n$	
Advection-projection:	$\boldsymbol{v}^{adv} = advectProject(\boldsymbol{v}^n)$	
Correct divergence field:	$\tilde{\boldsymbol{v}} = correctDivergence(\boldsymbol{v}^{adv})$	
Vorticity through linear field:	$\tilde{\boldsymbol{\zeta}} = \nabla \times \tilde{\boldsymbol{v}}$	
Compute vorticity equation:	$\boldsymbol{\zeta}^{n+1} = \boldsymbol{\zeta}^n + \Delta t \frac{D\boldsymbol{\zeta}^n}{Dt}$	(Eqn. 11)
Dissipation of vorticity:	$\nabla \times \boldsymbol{v}(\boldsymbol{y}) = \boldsymbol{\zeta}^{n+1} - \tilde{\boldsymbol{\zeta}}$	(Eqn. 12)
Compute stream function:	$\boldsymbol{\psi} = \int_{\mathbb{R}^3} \frac{\nabla \times \boldsymbol{v}(\boldsymbol{y})}{4\pi \|\boldsymbol{x} - \boldsymbol{y}\|}$	(Eqn. 14)
Refinement of linear velocity:	$\Delta \boldsymbol{v} = \nabla \times \boldsymbol{\psi}$	(Eqn. 13)
Refine linear velocity:	$\boldsymbol{v}^{n+1} = \tilde{\boldsymbol{v}} + \alpha\, \Delta \boldsymbol{v}$	(Eqn. 17)

5. Results and discussion

We next test our novel vortex refinement (VR) method on several scenes, comparing it with the state-of-the-art micropolar (MP) model and classical SPH approaches.

Both the VR and the MP method are integrated with DFSPH in the following experiments to show the applicability of our method. We used the boundary handling method proposed by Akinci *et al.* [AIA*12]. We implemented the entire framework in C++, with animations rendered by Blender. Our simulation platform is a graphic workstation with an Intel Xeon E5-2687w v4 (15M cache, 3.5 GHz, 12 cores) CPU, 80 GB RAM, and an NVIDIA Quadro P4000 GPU.

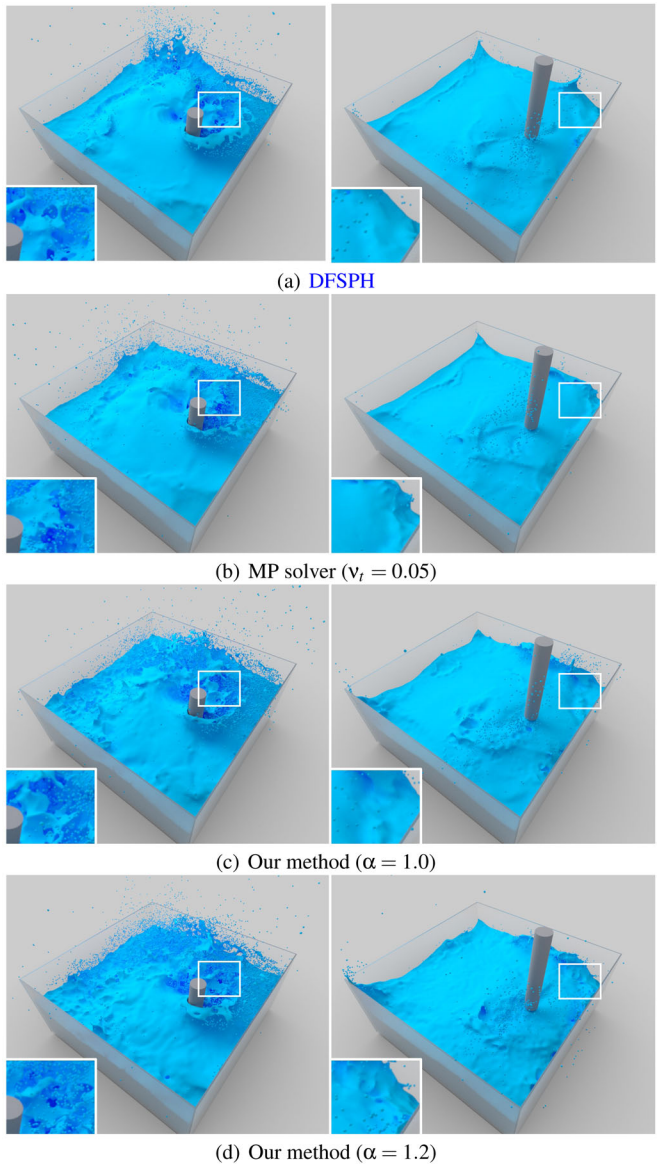
Similarly to the adjustment parameter α in our method, there is a scalar ν_t in the MP method to control it. Based on the mechanism of the MP method [Eri66, BKKW18], ν_t greater than ν_v can potentially violate the second law of thermodynamics. In all experiments below we set $\nu_v = 0.05$. We therefore choose $\nu_t = 0.05$ as a natural refinement for the MP solver, which corresponds to $\alpha = 1$ in our method. However, to explore the stability and performance of the methods, we test α greater than 1 and ν_t greater than 0.05; see Figs. 5 and 9. As stated in [BKKW18], fluids are reasonably stable when $\nu_t \leq 0.4$.

5.1. Effectiveness and comparison

To show the effectiveness of our approach numerically, we executed two breaking-dam experiments, and we executed two other experiments for parameter discussion and energy comparison with other methods, as follows.

Breaking dam with a board. In Figs. 1 and 3, a board collides with a breaking dam which only allows fluid to go through the so-created gap. Figure 3 shows the results with 1.18M particles. Only few vortex effects can be seen using DFSPH. Water flushes through the gap and dissipates quickly without clear turbulence effects. Compared to DFSPH, our solver generates several realistic vortices around the board and corners. The MP solver also improves the visual result, but not as obviously as our method. Since our method refines particle velocity based on the vorticity field, vortices are naturally preserved and turbulence is generated from the dissipated energy in a realistic way. In Fig. 4, the vorticity magnitude of all particles is visualized. The comparison shows that both our method and the MP method yield higher energy values than DFSPH. The MP solver adds energy in a natural way, while our method recovers energy from numerical dissipation more effectively and is thus able to simulate more details.

Breaking dam with three obstacles. As shown in Fig. 11, a breaking dam scenario with static obstacles was tested using 457K fluid particles. The fluid flows in from the left and hits the wall on the right. Several waves are generated in the process, which then come back and interact with three rigid bodies. Desirable turbulence can be observed over the surface. We compared our method with the DFSPH and MP solvers. In DFSPH, the fluid seems to go around the pillars and forms splashes, but scarcely any complex turbulence effects. In contrast, our solver creates small-scale vortices instead of just the fluid smoothly flowing around the pillars. Since these small vortices cannot sustain a self-spinning state, they quickly break down into turbulence. Compared to the MP solver in this scene, our method seems to generate more turbulent details but smaller vortices. The MP solver and our method can achieve different visual effects.



(a) DFSPH

(b) MP solver ($\nu_t = 0.05$)

(c) Our method ($\alpha = 1.0$)

(d) Our method ($\alpha = 1.2$)

Figure 5: *Comparison of DFSPH, the MP solver ($\nu_t = 0.05$), and our method ($\alpha = 1.0$ and $\alpha = 1.2$) for a simulation of a stick (rod) mixing water. Besides turbulence enhancement, our method stands out in keeping the flow trail visible and maintaining the stability of the surface. As visible, the MP solver and our solver can obtain different enhancement results.*

Energy comparison. An energy comparison of a breaking dam experiment (see Fig. 8) is shown in Fig. 7. The left plot shows the energy comparison, while the right plot shows the energy increase ratio relative to DFSPH. When $t \in [2, 8]$, the fluid keeps flowing and forming turbulence. If the energy is larger than that of DFSPH,

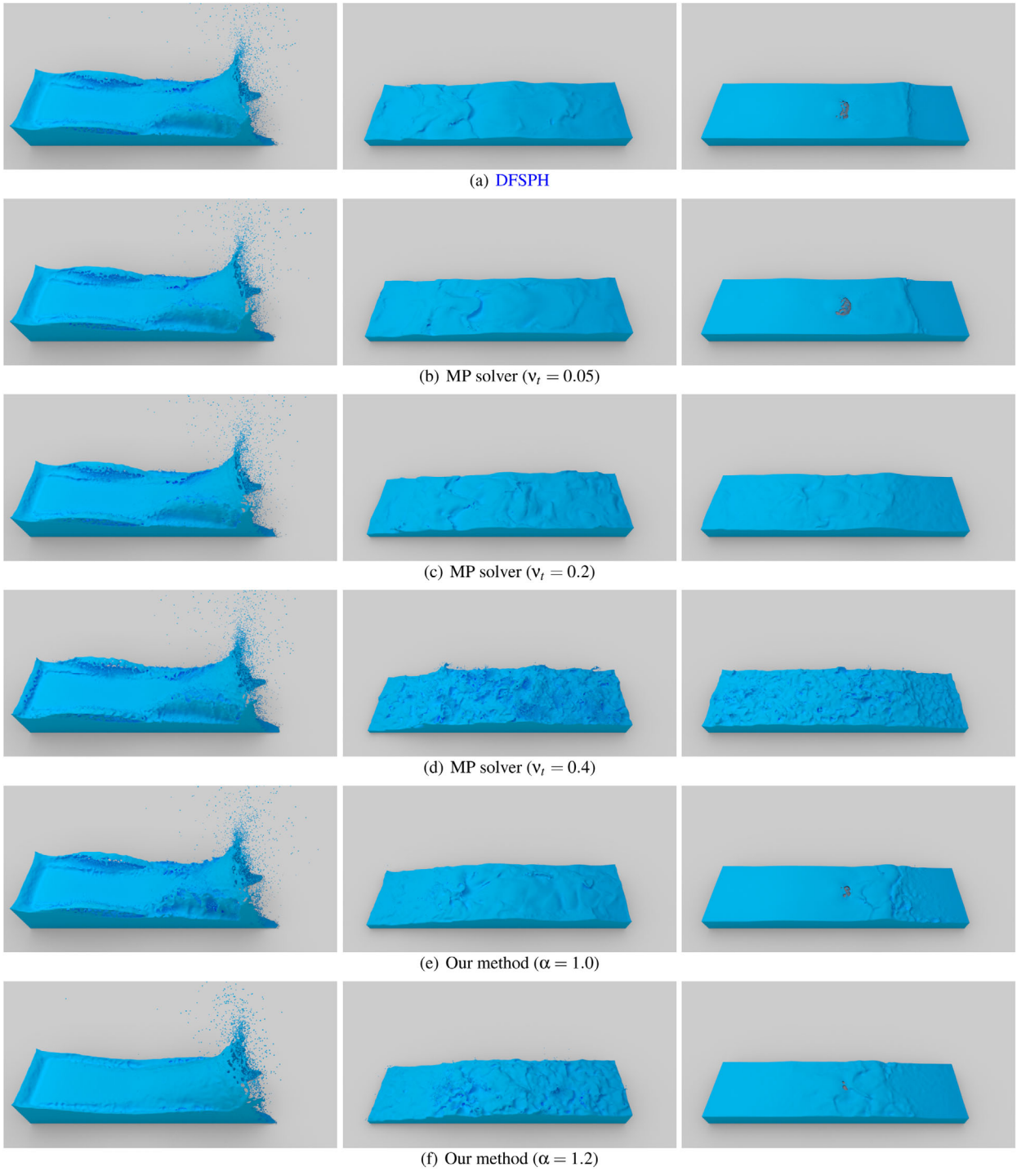


Figure 9: *Comparison of DFSPH, the MP solver with* $\nu_t = 0.05, 0.2, 0.3, 0.4$*, and our method with* $\alpha = 1.0, 1.2$ *in the breaking dam scenario with a static spherical obstacle placed to the right.*

then energy is recovered (or added) successfully. After the water surface calms down (after about 10s), the scene should contain only potential energy (no kinetic energy). The energy of the traditional DFPSH method can be used as a benchmark: If a method generates, at this time point, more energy than DFSPH, then this method is considered to create *additional* energy. In this comparison experiment, the energy values after 10s for both the MP solver with $\nu_t = 0.05, 0.1$ and for our method with $\alpha = 1.0, 1.1, 1.2$ are very close to the DFSPH values. Our method with $\alpha = 1.3$ and the MP method with $\nu_t = 0.2, 0.3, 0.4$ have higher energy than DFSPH. In

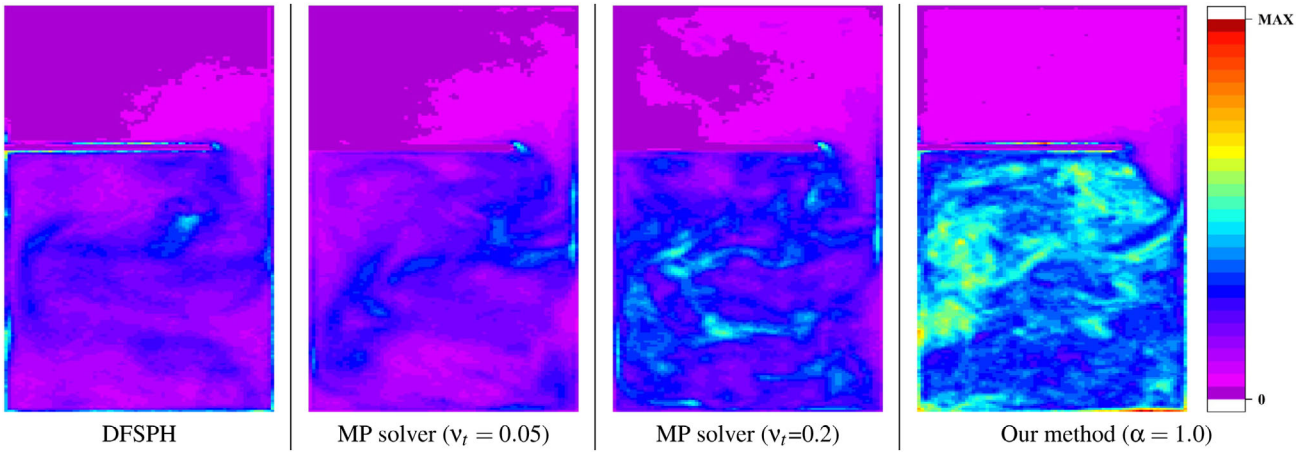


Figure 4: *Comparison of vorticity in a 2D scene extracted from the 3D scene in Fig. 3. This visualization corresponds to the second column in Fig. 3. Colour shows the vorticity magnitude of the particles, thereby allowing one to compare the vorticity of DFSPH, the MP solver, and our VR method.*

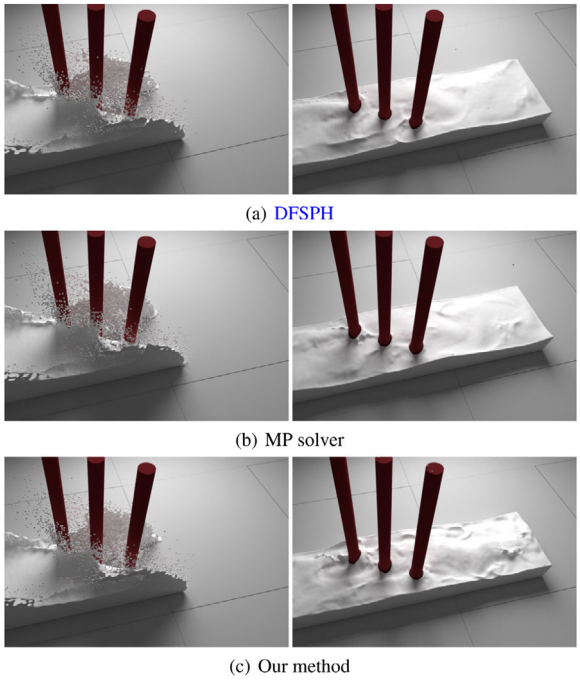


Figure 11: *A comparison of DFSPH, the MP solver and our method in a breaking dam scene with 3 static pillars as obstacles. The water hits the cylinders and the right wall and bounces back, forming turbulence in the process. Compared to DFSPH, our solver gives rise to tiny vortices instead of the water simply going around the pillars. Compared to the MP solver, our method seems to provide more turbulent details but smaller vortices in this scene.*

some applications, in order to enhance the visual effect, one can use such larger parameter values. However, this can very likely cause excessive chaos and even instability such as unnatural turbulence similar to boiling. Hence, we recommend to use our method with

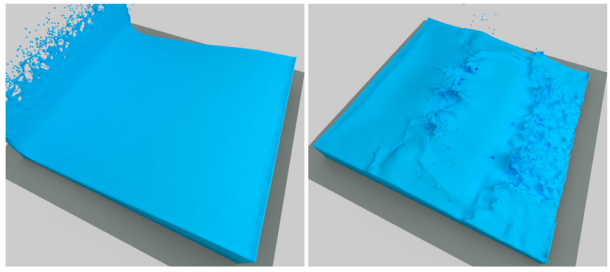


Figure 8: *A breaking dam scenario for the evaluation of energy changes (461K fluid particles); see Fig. 7.*

$\alpha = 1.0$ to ensure the energy is always in line with the underlying physics.

Breaking dam with a hemisphere: parameter influence. In this experiment (see Fig. 9) we flush a hemisphere obstacle with a fixed volume of fluid. This means only limited kinetic energy is involved in this scenario (from gravitational potential energy). We simulated the flow using DFSPH, our method with $\alpha = 1$ and $\alpha = 1.2$, and the MP solver with $\nu_t = 0.05$, $\nu_t = 0.2$ and $\nu_t = 0.4$. When comparing the DFSPH approach with our method with $\alpha = 1$ and with MP with $\nu_t = 0.05$, both methods are able to increase the turbulence performance, but our result is more pronounced than the MP one. To obtain more obvious turbulence effects, we increase the turbulence control parameters in the two methods, which means that more energy is added to the simulation. The renderings show that our method with $\alpha = 1.2$ yields more turbulence and the result is better than that of the MP solver with $\nu_t = 0.2$. To keep our method in line with the underlying physics, as explained for the earlier example, we do not use higher parameter values. The MP solver adds more turbulence in this scene. The obtained results are visually more salient for large parameter values, *e.g.* $\nu_t = 0.2$. However, ν_t cannot be increased indefinitely. For example, if we set $\nu_t = 0.4$ (Fig. 9, last row), the fluid does not calm down, which is unnatural. The detailed energy

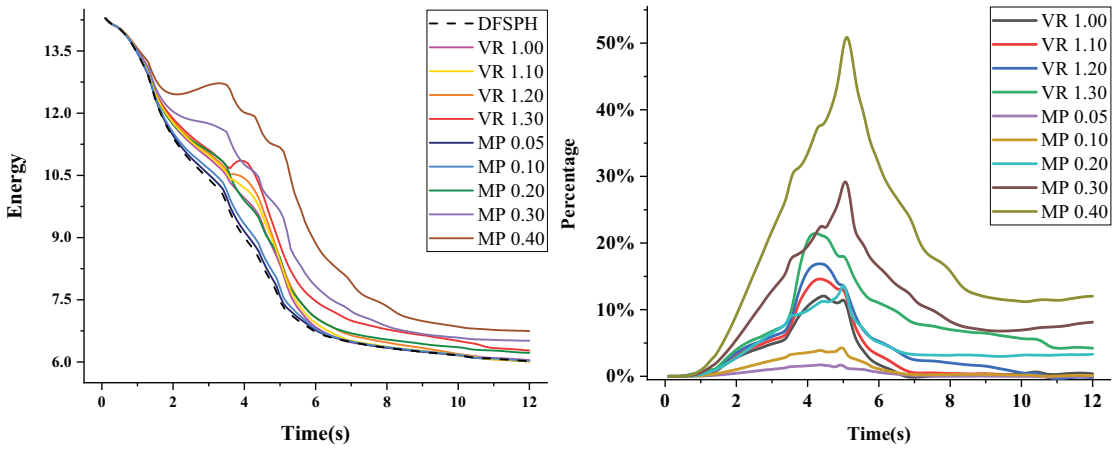


Figure 7: *Comparison of energy changes for different methods using different parameter values in the breaking dam scene shown in Fig. 8. Left: direct energy comparison. Right: energy increase ratio relative to DFSPH. When the fluid is flowing ($t \in [2, 8]$), our method and the MP solver are able to add energy to the scene and enhance the visual effect. However, the MP solver with $\nu_t = 0.2, 0.3, 0.4$ and our method with $\alpha = 1.3$ do not converge after $t = 8$ due to the excessive energy added.*

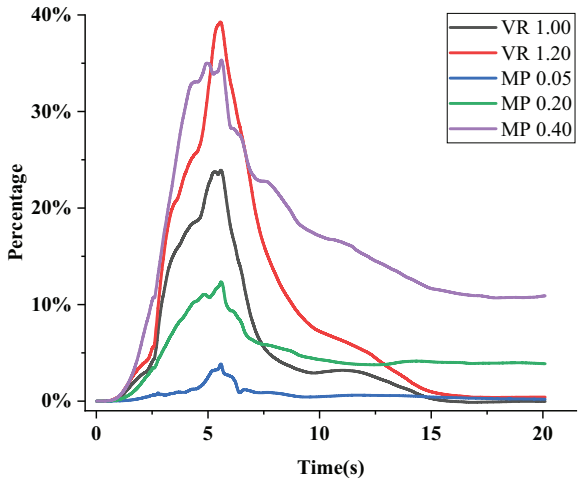


Figure 10: *Similar to Fig. 8, we compared the energy increase ratio of different methods to the DFSPH method using the scene shown in Fig. 9. We see that all methods produce more energy than DFSPH when the fluid is flowing ($t \in [0, 15]$). Note that the MP solver with $\nu_t = 0.2, 0.4$ does not converge after $t = 15$.*

comparison is shown in Fig. 10. Our method can be applied to scenes that are more sensitive to physics laws, such as adding more details to a relatively stably-flowing scene. In contrast, the MP method can be used in scenes where one wants to create a stronger visual impact, such as collapses or violent shocks.

Overall, this experiment shows that the MP solver and our solver can achieve different turbulence effects. Our method achieves better turbulence results without adding energy sources. In contrast, the MP solver can add small vortices, but when increasing its parameter values, energy sources will pop up and prevent the fluid from calming down.

5.2. Quality

To further demonstrate the turbulence quality of our method, we simulated several complex scenarios with dynamic boundary conditions and compared them with the MP solver.

Spinning propeller. A propeller is slowly submerged into water, after which it starts spinning at 3 radians per second. Fig. 12 shows the results of this simulation using 1.29M fluid particles for DFSPH, MP, and VR (our method). Observe that neither the complex flow nor strong turbulence effects are produced and preserved using DFSPH. Both our method and the MP method enhance the visual effect. In contrast to the MP method, our method adds energy in a physically reasonable way (no turbulence in front of the propeller) and creates vivid turbulent details over the free fluid surface. The key areas are zoomed in on. Also, a vortex is observed with our method after the propeller has stopped spinning (see also the supplementary video).

Boat-sinking. In this scenario, a boat and two columns interact with a breaking dam. Figure 6 shows the results using 1.7M fluid particles. The potential energy of the fluid transforms into the kinetic energy of the fluid particles and the boat. The water is first violently displaced when it hits the column and the boat, and next gradually calms down as time goes by, finally reaching a stable state. We see that the DFSPH method produces relatively weakly turbulent details, which get lost quickly due to numerical dissipation. In contrast, our method and MP server shows more natural dynamics with realistic turbulent effects on the fluid surface. The fluid gradually calms down as time goes on. Our method and the MP method achieve different styles.

Stirring water. In Fig. 5, a cylindrical stick was inserted into a tank of water, and stirred at a uniform speed for several seconds. The water splashed around due to the quick movement of the stick. Observe that the trace left on the surface lasts longer in our method than with the MP method, which is a critical point for boat-sailing animation scenarios. After the stirring process, the stick is pulled

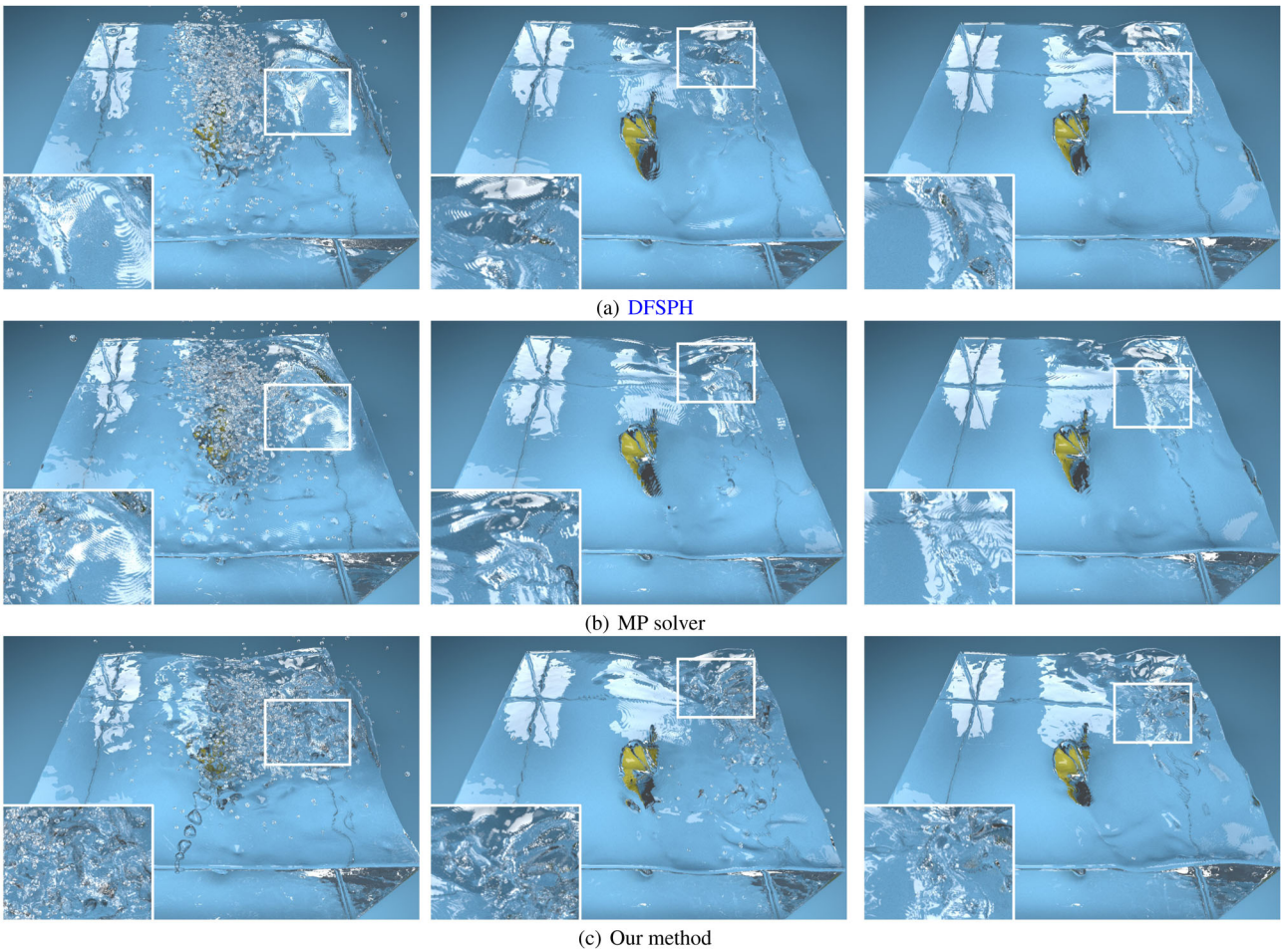


Figure 12: *A propeller interacts with 1.29M fluid particles using DFSPH, the MP solver, and our method. DFSPH (top row) is not able to produce a complex flow. Under stable conditions, the MP solver (middle row) can only add limited turbulence effects to DFSPH, while our VR method is able to generate realistic vortices (bottom row). The improved turbulence performance can be seen clearly in the insets.*

out of the fluid, and the water starts to calm down. The DFSPH approach calms the fluid down quickly due to numerical dissipation. The surface details are clearer and sharper in our method. Also, we notice a disturbance wave in the MP method, caused by the fact that ν_t exceeds the kinematic viscosity.

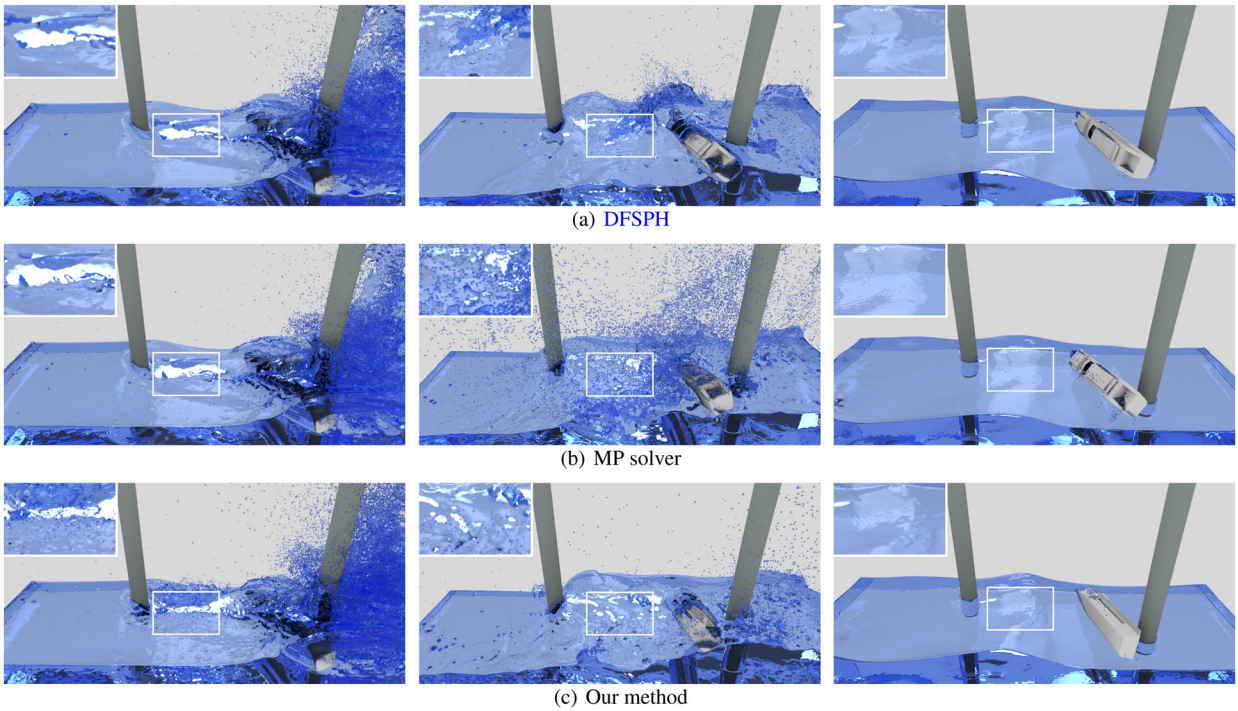
The above three scenarios show that our method can keep stability when dealing with extreme conditions like strong collisions, while physically preserving energy. Moreover, in the accompanying video it can be seen that our method not only amplifies existing vortices but also generates new ones.

Computational overhead. The computational overhead of our method is negligible compared to the whole SPH simulation procedure. Table 1 shows the computing times for DFSPH, the MP solver, and our method for different simulation scenes. The different computation times are explained as follows. Compared to DFSPH, both turbulence methods (MP and ours) need to compute the vorticity field, i.e., solve for the Laplacian $\nabla^2\boldsymbol{\zeta}$. Further, $\nabla\times\boldsymbol{\zeta}$ (in the MP solver) and $\nabla\boldsymbol{v}$ (in our method) also need to be solved for. The difference is that our method needs to compute $\boldsymbol{\psi}(\boldsymbol{\zeta})$ and $\nabla\times\boldsymbol{\psi}$ to get the refined velocity, but as Table 1 shows, the extra computational effort is negligible.

6. Conclusion and discussion

We have presented a particle-based turbulence refinement method that recovers lost velocity from the difference between the theoretical and the actual vorticity value. Our method can not only increase existing vortices significantly by recovering numerical dissipation, but also generates new turbulence at potentially different locations. The turbulence-enhancement parameter of our method has a theoretically optimal value $\alpha = 1$ that can increase turbulence without adding too much energy. At the same time, one can easily adjust this parameter to achieve different turbulence levels for different simulation effects.

Experimental results show that, compared to the classical and micropolar SPH methods, our method is able to enhance turbulent effects more visibly. Furthermore, our method guarantees energy conservation, even when using a large particle radius and/or a large time



(a) DFSPH

(b) MP solver

(c) Our method

Figure 6: *A breaking dam collides with a ship and 2 static pillars (1.70M particles). The key area is marked in white and zoomed in on. Our method and the MP solver have more details than the DFSPH solution. In the second column, the MP solver becomes unstable and some particles explode, while our method enhances the turbulence effect in a more stable and realistic way.*

Table 1: *Total time comparisons of three methods: DFSPH, MP, and our method (VR) over five simulations. Δt, in milliseconds, is the time step used in the experiments, and the total computation times, in minutes, include the costs of the density solver and the divergence-free solver in DFSPH.*

Experiment	Fig.	Particles	Δt (ms)	Steps	DFSPH (m)	MP (m)	VR (m)
Board	3	1.18M	2.4	9542	2401.2	2565.9	2565.1
Stirring	5	1.39M	2.4	9542	2399.9	2864.4	2693.4
Sphere	9	899.8K	2.4	8375	1657.4	1715.2	1827.2
Pillars	11	457K	3	6667	156.4	188.3	218.5
Propeller	12	1.29M	3	7334	1782.1	2309.1	2338.9

step. This means that our method is still robust even under extreme simulation conditions and can handle complex large-scale scenes, as demonstrated in our simulation scenarios.

Numerical dissipation is difficult to fully correct in SPH methods. Our method can simulate typical turbulent scenes efficiently and is relatively stable even for scenarios with highly turbulent flow. At the same time, we should note that some vorticity is lost in such cases. While this small amount of loss does not affect the general visual quality, decreasing it is an open topic for future research, which can be expected to lead to even more realistic fluid simulations.

In the future, we aim to investigate merging our method with microstructural models, since these models show great potential for rough simulation conditions and also have a close relationship with viscosity. Improving computation accuracy is another potential future research direction. Finally, increasing the computational scalability of our method by, e.g. efficient and effective parallelization is attractive for making our method directly applicable to complex real-world and/or interactive simulations.

References

[AIA*12] Akinci N., Ihmsen M., Akinci G., Solenthaler B., Teschner M.: Versatile rigid-fluid coupling for incompressible sph. *ACM Transactions on Graphics (TOG) 31*, *4* (2012), 62.

[AN05] Angelidis A., Neyret F.: Simulation of smoke based on vortex filament primitives. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), ACM, pp. 87–96.

[BK15] Bender J., Koschier D.: Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2015), ACM, pp. 147–155.

[BKKW18] Bender J., Koschier D., Kugelstadt T., Weiler M.: Turbulent micropolar sph fluids with foam. *IEEE Transacttions on Visualization and Computer Graphics* (2018).

[Bri15] Bridson R.: *Fluid Simulation for Computer Graphics*. AK Peters/CRC Press, 2015.

[BT07] Becker M., Teschner M.: Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (2007), Eurographics Association, pp. 209–217.

[CT17] Chu M., Thuerey N.: Data-driven synthesis of smoke flows with cnn-based feature descriptors. *ACM Transactions on Graphics (TOG) 36*, *4* (2017), 69.

[dGWH*15] de Goes F., Wallez C., Huang J., Pavlov D., Desbrun M.: Power particles: an incompressible fluid solver based on power diagrams. *ACM Transactions on Graphics 34*, *4* (2015), 50–1.

[EB14] Edwards E., Bridson R.: Detailed water with coarse grids: combining surface meshes and adaptive discontinuous galerkin. *ACM Transactions on Graphics (TOG) 33*, *4* (2014), 136.

[Eri66] Eringen A. C.: Theory of micropolar fluids. *Journal of Mathematics and Mechanics* (1966), 1–18.

[EWPT17] Eberhardt S., Weissmann S., Pinkall U., Thuerey N.: Hierarchical vorticity skeletons. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2017), ACM, p. 6.

[FGG*17] Fu C., Guo Q., Gast T., Jiang C., Teran J.: A polynomial particle-in-cell method. *ACM Transactions on Graphics 36*, *6* (Nov. 2017), 222:1–222:12.

[FM96] Foster N., Metaxas D.: Realistic animation of liquids. *Graphical Models and Image Processing 58*, *5* (1996), 471–483.

[FSJ01] Fedkiw R., Stam J., Jensen H. W.: Visual simulation of smoke. In *Proceedings of the 28th Conference on Computer Graphics and Interactive Techniques* (2001), ACM, pp. 15–22.

[GNS*12] Golas A., Narain R., Sewall J., Krajcevski P., Dubey P., Lin M.: Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Transactions on Graphics (TOG) 31*, *6* (2012), 148.

[HLL*12] He X., Liu N., Li S., Wang H., Wang G.: Local poisson sph for viscous incompressible fluids. In *Computer Graphics Forum* (2012), vol. *31*, Wiley Online Library, pp. 1948–1958.

[ICS*14] Ihmsen M., Cornelis J., Solenthaler B., Horvath C., Teschner M.: Implicit incompressible sph. *IEEE Transactions on Visualization and Computer Graphics 20*, *3* (2014), 426–435.

[IOS*14] Ihmsen M., Orthmann J., Solenthaler B., Kolb A., Teschner M.: SPH fluids in computer graphics. Eurographics 2014 - State of the Art Reports, The Eurographics Association, 2014.

[JKB*10] Jang T., Kim H., Bae J., Seo J., Noh J.: Multilevel vorticity confinement for water turbulence simulation. *The Visual Computer 26*, 6–8 (2010), 873–881.

[JSMF*18] Jeschke S., Skřivan T., Müller-Fischer M., Chentanez N., Macklin M., Wojtan C.: Water surface wavelets. *ACM Transactions on Graphics 37*, *4* (July 2018), 94:1–94:13.

[JSS*15] Jiang C., Schroeder C., Selle A., Teran J., Stomakhin A.: The affine particle-in-cell method. *ACM Transactions on Graphics (TOG) 34*, *4* (2015), 51.

[JST17] Jiang C., Schroeder C., Teran J.: An angular momentum conserving affine-particle-in-cell method. *Journal of Computational Physics 338* (2017), 137–164.

[KBST19] Koschier, D., Bender, J., Solenthaler, B., Teschner, M.: Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids. Eurographics 2019 - Tutorials, The Eurographics Association, 2019.

[KLLR05] Kim B., Liu Y., Llamas I., Rossignac J. R.: *Flowfixer: Using bfecc for Fluid Simulation*. Tech. rep., Georgia Institute of Technology, 2005.

[KTJG08] Kim, T., Thürey, N., James, D., Gross, M.: Wavelet turbulence for fluid simulation. *ACM Transactions on Graphics (TOG)* (2008), vol. *27*, ACM, p. 50.

[LAF11] Lentine M., Aanjaneya M., Fedkiw R.: Mass and momentum conservation for fluid simulation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), ACM, pp. 91–100.

[MBT*15] Mercier O., Beauchemin C., Thuerey N., Kim T., Nowrouzezahrai D.: Surface turbulence for particle-based liquid simulations. *ACM Transactions on Graphics (TOG) 34*, *6* (2015), 202.

[MCG03] Müller M., Charypar D., Gross M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), Eurographics Association, pp. 154–159.

[MM13] Macklin M., Müller M.: Position based fluids. *ACM Transactions on Graphics (TOG) 32*, *4* (2013), 104.

[Mon85] Monaghan J.: Particle methods for hydrodynamics. *Computer Physics Reports 3*, *2* (1985), 71–124.

[Mon94] Monaghan J. J.: Simulating free surface flows with sph. *Journal of Computational Physics 110*, *2* (1994), 399–406.

[NZT19] Narain R., Zehnder J., Thomaszewski B.: A second-order advection-reflection solver. *Proceedings of the ACM on*

Computer Graphics and Interactive Techniques 2, *2* (2019), 1–14.

[PK05] Park S. I., Kim M. J.: Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), ACM, pp. 261–270.

[SP09] Solenthaler B., Pajarola R.: Predictive-corrective incompressible sph. In *ACM Transactions on Graphics (TOG)* (2009), vol. *28*, ACM, p. 40.

[Sta99] Stam J.: Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128.

[WLB*20] Wang X., Liu S., Ban X., Xu Y., Zhou J., Kosinka J.: Robust turbulence simulation for particle-based fluids using the rankine vortex model. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)* (2020), IEEE, pp. 657–658.

[WP10] Weissmann S., Pinkall U.: Filament-based smoke with vortex shedding and variational reconnection. In *ACM Transactions on Graphics (TOG)* (2010), vol. 29, ACM, p. 115.

[XBP*19] Xu Y., Ban X., Peng Y., Wang X., Liu S., Zhou J.: Turbulence enhancement for sph fluids visualization. In *International Conference on Cooperative Design, Visualization and Engineering* (2019), Springer, pp. 254–260.

[ZB05] Zhu Y., Bridson R.: Animating sand as a fluid. *ACM Transactions on Graphics (TOG) 24*, *3* (2005), 965–972.

[ZB14] Zhang X., Bridson R.: A pppm fast summation method for fluids and beyond. *ACM Transactions on Graphics (TOG) 33*, *6* (2014), 206.

[ZBG15] Zhang X., Bridson R., Greif C.: Restoring the missing vorticity in advection-projection fluid solvers. *ACM Transactions on Graphics (TOG) 34*, *4* (2015), 52.

[ZNT18] Zehnder J., Narain R., Thomaszewski B.: An advection-reflection solver for detail-preserving fluid simulation. *ACM Transactions on Graphics (TOG) 37*, *4* (2018), 1–8.

[ZYF10] Zhu B., Yang X., Fan Y.: Creating and preserving vortical details in sph fluid. In *Computer Graphics Forum* (2010), vol. *29*, Wiley Online Library, pp. 2207–2214.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Data video S1

Data video S2

Data video S3