Dynamic Importance Monte Carlo SPH Vortical Flows with Lagrangian Samples

Xingyu Ye, Xiaokun Wang, Yanrui Xu, Alexandru C. Telea, Jiří Kosinka, Lihua You, Jian Jun Zhang, Jian Chang

***Abstract*—We present a Lagrangian dynamic importance Monte Carlo method without non-trivial random walks for solving the Velocity-Vorticity Poisson Equation (VVPE) in Smoothed Particle Hydrodynamics (SPH) for vortical flows. Key to our approach is the use of the Kinematic Vorticity Number (KVN) to detect vortex cores and to compute the *KVN-based importance* of each particle when solving the VVPE. We use Adaptive Kernel Density Estimation (AKDE) to extract a probability density distribution from the KVN for the the Monte Carlo calculations. Even though the distribution of the KVN can be non-trivial, AKDE yields a smooth and normalized result which we *dynamically update* at each time step. As we sample actual particles directly, the Lagrangian attributes of particle samples ensure that the continuously evolved KVN-based importance, modeled by the probability density distribution extracted from the KVN by AKDE, can be closely followed. Our approach enables effective vortical flow simulations with significantly reduced computational overhead and comparable quality to the classic Biot-Savart law that in contrast requires expensive global particle querying.**

***Index Terms*—Fluid Simulation, Importance Monte Carlo, SPH, Vortical Flow.**

I. Introduction

Simulating visually rich and complex vortical flows is a hot topic in computer graphics research. Despite recent advances [1], several challenges still remain. Lagrangian methods using Smoothed Particle Hydrodynamics (SPH) show a rapid dissipation of vortical flows stemming from many sources such as numerical damping [2] and coarse sampling [3]. This is typically tackled by adding missing vortical or turbulent energy. For example, the vorticity confinement method enforces vortical flows by amplifying current vorticity [4]. Yet, this can only enforce existing vorticity. To alleviate this, the micropolar fluid method creates turbulent flows via micro-rotations [5]. However, since there exists a conceptual distinction between micro-rotations and vorticity [6], the generated flow details are not necessarily of vortical type.

Vortical flows can be simulated by evolving the vorticity field directly and solving the Velocity-Vorticity Poisson Equation (VVPE) [7]. The Biot-Savart law provides a classic method to solve this Poisson equation, which uses the Green's function to formulate the solution via a convolution integral. Vortex methods numerically evaluate this integral by a discrete summation that queries vorticity tracked by discrete Lagrangian elements *e.g.* particles [8], segments [9], and filaments [10]. Yet, this global summation has a low computational efficiency. Recent Monte Carlo methods address this by efficient point-based Partial Differential Equation (PDE) solvers [11]. These approaches replace the expensive global summation by summing points sampled by random walks including walking-on-spheres [11], [12], walking-on-boundary [13], and walking-on-stars methods [14], [15]. They come with various non-trivial boundary sampling methods to obtain sufficient boundary information for solving PDEs within point-based frameworks.

Although Monte Carlo methods are commonly developed as point-based (grid-free and particle-free) PDE solvers, we focus on the classic SPH framework to enhance vortical flows in more general and non-trivial fluid scenarios than previous work. We propose a dynamic importance Monte Carlo method for solving the VVPE. Since our work is conducted in the context of SPH, particle velocities can be projected to satisfy the free-slip boundary using particle boundaries [16]. Thus, our work can directly apply the Monte Carlo PDE solver with infinite domains [17] and non-trivial random walk techniques can be omitted.

We use the dimensionless Kinematic Vorticity Number (KVN) [18] to compute the importance of each particle when solving the VVPE. The KVN shows how closely a rotation resembles rigid body rotation and works better than vorticity to detect vortex cores, i.e., the rotation center of a vortex. The Monte Carlo method requires a normalized probability density distribution and representative samples. We extract a *dynamically updated probability density distribution* from the KVN through Adaptive Kernel Density Estimation (AKDE) [19]. Even though the KVN distribution can be nontrivial, AKDE ensures a smooth and normalized probability density distribution. Important areas typically evolve with time. We update samples following our KVN-based importance at each time step to ensure this evolution can be closely followed.

In summary, the main contributions of our work are:

- A *Monte Carlo method without random walks* to enhance SPH vortical flows with Lagrangian particle samples closely following the evolution of important areas.
- A *KVN-based importance* that highlights vortex cores and models particle importances for solving the VVPE.
- A *dynamically updated probability density distribution* estimated by AKDE, used to run the Monte Carlo method smoothly and in a normalized fashion.

Xingyu Ye, Xiaokun Wang, and Yanrui Xu are with University of Science and Technology Beijing, Beijing 100083, China. E-mail:{xye@bournemouth.ac.uk, wangxiaokun@ustb.edu.cn, xuyanruiedw@me.com}.

Xingyu Ye, Lihua You, Jian Jun Zhang, and Jian Chang are with National Centre for Computer Animation, Bournemouth University, Bournemouth BH12 5BB, U.K. E-mail:{lyou, jzhang, jchang}@bournemouth.ac.uk.

Alexandru C. Telea is with Utrecht University, Utrecht 3508 TC, Netherlands. E-mail:{a.c.telea@uu.nl}.

Jiří Kosinka is with University of Groningen, Groningen 9712 CP, Netherlands. E-mail:{j.kosinka@rug.nl}

(Corresponding author: Xiaokun Wang, Jian Chang.)

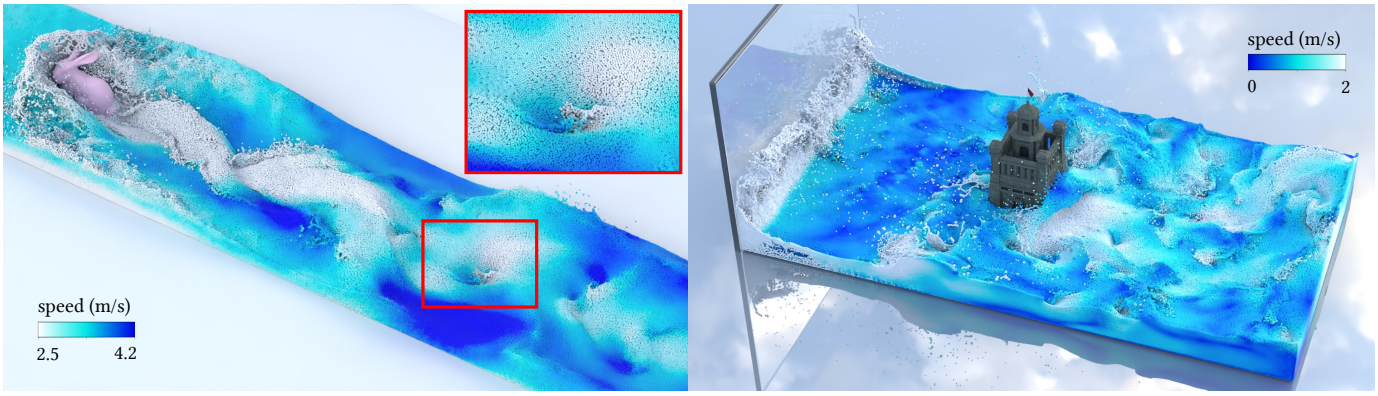


Fig. 1. **Left**: A static Stanford bunny in a high-velocity stream. **Right**: A static tower subjected to periodic waves. Both scenarios are rendered as particles with color-coded speed. These results show our method's ability to handle various vortical flow phenomena and generate realistic vortices.

The remainder of this paper is organized as follows. We start by reviewing related work (Sec. II) and also introduce the vorticity loss of SPH and the solution of VVPE (Sec. III). Next, we detail our dynamic importance Monte Carlo SPH vortical flow approach (Sec. IV). We illustrate our proposal by a series of simulation experiments and comparisons (Sec. V). Finally, we conclude the paper and provide directions for future work (Sec. VI).

II. Related Work

We next review work related to SPH vortical flow simulations and the Monte Carlo method in graphics. For a detailed and broader introduction to fluid simulation and SPH-based fluids, we refer to [1], [20], [21].

Detail-enhanced SPH Flows. Many detail-enhancing schemes for SPH fluids were first developed in the field of Eluerian fluid simulation. To alleviate numerical dissipation in smoke simulation, Fedkiw et al. [4] introduced the vorticity confinement method to the graphics community. Macklin and Müller [22] adapted it to position-based fluids using SPH discretization. To overcome the weakness of non-conservation of momentum and parameter sensitivity, Letine et al. [23] extended this approach to a momentum-conserving form. As the vorticity confinement scheme can only amplify *existing* vorticity, Bender et al. [5] introduced the micropolar model to simulate turbulent flows with SPH methods. This method develops a constitutive model to govern micro-rotation dynamics with applications in heat transfer [24], blood flow [25], and magnetized fluids [26]; and can update velocity using micro-rotation dynamics without explicitly enforcing vorticity. However, the concept of micro-rotations differs from vorticity [6]. Vortical motions constitute a *small* part of the overall produced turbulent motions. Ye et al. [27] introduced the Monte Carlo method to enhance vortical details by uniformly distributed samples. Although this approach effectively simulates vortical flow, the high variance of uniform sampling makes it sensitive to the setting of the sample count; inconsistent vortex patterns can appear for different sample counts. In our work, we use a combined scheme of KVN-based importance sampling and neighbor particles to develop a method that stably enhances vortices, is less sensitive to sample count, and yields coherent vortex patterns for different sample counts.

Vortex Methods. Such methods represent vortices by Lagrangian vortex elements and update vorticity dynamics using the curl of the Navier-Stokes equation (a.k.a. the vorticity equation). Vortex particle methods in this class model vortices using particles [28]. Selle et al. [8] coupled the vortex particle method with Eulerian fluids using the vorticity confinement method to overcome their separate weaknesses. Park et al. [7] simulated smoke by vortex particles with no-through and no-slip boundary handling for vorticity.

In 3D, a vortex center is usually not a point but a curve. The vortex stretching term within the vorticity equation, which is zero in 2D, governs the deformation of this curve. Most Eulerian methods do not directly discretize this term by finite differences due to potential numerical instabilities in gradient computations. The vortex filament method models vortices using Lagrangian filaments [29], [30]. Updating the shape of filaments replaces the finite difference of the vortex stretching term, providing better numerical stability. Yet, handling filament topology brings extra complexities. Zhang and Bridson [31] used a segment-based method in Eulerian approaches. The segment can be easily updated via its two endpoints which favors simplicity. Xiong et al. [9] further introduced explicit Lagrangian segments to simulate vortical flows without classic grid-based and particle-based frameworks.

Another challenge of vortex methods stems from the Biot-Savart law. Converting vorticity to velocity is needed to update the position of Lagrangian vortex elements. This can be done by solving the VVPE. The Biot-Savart law uses the Green's function to formulate a summation as a solution to the Poisson equation. Yet, this summation is a computationally expensive N-body problem with $O(N^2)$ complexity for N fluid particles. Several schemes aim to accelerate this computation. The Fast Multipole Method (FMM) uses dynamically updated multi-scale grids for near and far particles [32], [33]; near particles are handled using dense grids or individually; far particles are transferred to coarse grids (which become coarser with distance) as a whole body. Although the FMM reduces the computational overhead of the Biot-Savart law, dynamically managing multi-scale grids makes its implementation challenging. In the Particle-Particle Particle-Mesh (PPPM) scheme [31], near particles are still treated individually, while far particles are transferred to uniform grids and solved using

the Vortex-in-Cell (VIC) method. Our work also treats near and far particles separately: Near particles affect the kernel-based importance; far particles are modeled by the Monte Carlo method using our *KVN-based importance*.

Monte Carlo Methods in Computer Graphics. Such methods were first used to estimate the outcome of ray tracing [34], [35]. Sawhney and Crane [11] proposed the Monte Carlo geometry processing method – a Monte Carlo point-based partial differential equation (PDE) solver which is highly efficient and easy to implement. Subsequent works extended this method to spatially varying coefficients [36], Neumann boundaries [14], Robin boundaries [15], walk on boundary [13], and differential solvers [37].

Rioux-Lavoie et al. [12] introduced Monte Carlo methods to fluid simulation by solving the VVPE with a point-based Monte Carlo PDE solver. The Poisson equation solution is represented by continuous integration using Green's function. This integration can be efficiently estimated by Monte Carlo methods using a discrete summation. This Monte Carlo PDE solver was next extended to solve the general velocity-based Navier-Stokes equation [17]. While classic grid- or particle-based frameworks solve PDEs by discretizing differential operators using finite difference or SPH approximations, point-based Monte Carlo methods solve PDEs by sampling spatial points. Such samples are usually obtained by random walks, including the walk-on-sphere [11] algorithm and its variants [13], [15]. However, these random-walk methods require non-trivial techniques to obtain sufficient and correctly distributed samples from boundaries with complex geometries. Our approach is developed in the context of the SPH. This allows the Monte Carlo method to be computed on infinite domains without random walks, as boundary conditions are satisfied by the SPH projection.

Kinematic Vorticity Number (KVN). A well-known tool for analyzing crustal deformations in geology, KVN is a non-dimensional factor that measures the degree of pure rotations, eliminating the influence of angular velocity magnitude [38], [39]. KVN ranges from 0 (pure shear deformation) to ∞ (rigid body rotation). Schielicke et al. [18] used the KVN to estimate vortex core size and circulation. We use the KVN in our solver to evaluate the *importance* of fluid particles in simulating vortical flows. Unlike approaches that estimate KVN via measurements [40], we directly use the theoretical definition of the KVN to compute its value. A dynamically updated probability distribution at each time step based on KVN is applied to select samples.

Kernel Density Estimation (KDE). In statistics, KDE provides an effective way to estimate a smooth probability density function from sample positions by a kernel function [41]. The resulting smoothness is controlled by the kernel bandwidth (akin to the support radius of SPH kernels). Many schemes have been developed for this control, *e.g.*, data-based [42] and linear diffusion processes [43]. George and David [44] studied the effect of variable bandwidths for uni- and multivariate KDE. To improve accuracy at boundaries, Jones [45] discusses several methods for boundary correction. Kerm [19] proposed Adaptive Kernel Density Estimation (AKDE) to adapt to the sparseness of data with variable weights and bandwidths. We use AKDE to estimate a *dynamically updated probability density distribution* representing the KVN-based importance for computations in the Monte Carlo method.

III. Vortical Flow Simulations in SPH

A. Vorticity Loss

SPH typically simulates fluids using a velocity-based form of the Navier-Stokes equation

$$\frac{D\boldsymbol{u}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\boldsymbol{u} + \boldsymbol{f}, \tag{1}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2}$$

where $\frac{D}{Dt} = \frac{\partial}{\partial t} + \boldsymbol{u}\cdot\nabla$ is the material derivative, $\boldsymbol{u}$ is the particle velocity, ρ is the density, ν is the kinematic viscosity, p is the pressure, and $\boldsymbol{f}$ models external body forces. t denotes simulation time. All these fields are defined on $\mathbb{R}^d$ where $d \in \{1, 2, 3\}$ is the simulation dimension. While Eq. (1) can simulate incompressible flows, it may not model nontrivial vortical motions well due to numerical damping and SPH's inherent coarse sampling. By taking the curl of Eq. (1), we obtain the governing equation for the vorticity $\boldsymbol{\omega} = \nabla \times \boldsymbol{u}$ as

$$\frac{D\boldsymbol{\omega}}{Dt} = (\boldsymbol{\omega}\cdot\nabla)\boldsymbol{u} + \nu\nabla^2\boldsymbol{\omega} + \nabla\times\boldsymbol{f}. \tag{3}$$

Eq. (3) gives the time evolution of vorticity and can be used to quantify the vorticity loss at each time step t as follows [46]:

- compute the vorticity $\boldsymbol{\omega}^t = \nabla\times\boldsymbol{u}^t$;
- evolve $\boldsymbol{\omega}^{t+\Delta t}$ from $\boldsymbol{\omega}^t$ using Eq. (3);
- apply non-pressure forces to $\boldsymbol{u}^t$ to compute an intermediate velocity $\bar{\boldsymbol{u}}^t$;
- quantify the loss of vorticity $\boldsymbol{\omega}_L^t = \boldsymbol{\omega}^{t+\Delta t} - \nabla\times\bar{\boldsymbol{u}}^t$ by using the curl of the intermediate velocity $\bar{\boldsymbol{u}}^t$.

These steps can be computed straightforwardly using an SPH formulation. Density can be approximated by

$$\rho_i = \sum_{j\in\mathcal{N}_i} m_j W^h(\boldsymbol{x}_{ij}), \tag{4}$$

where $\mathcal{N}_i$ are the neighbor particles of particle i of mass m_i and $\boldsymbol{x}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$. For the SPH smoothing kernel $W^h : \mathbb{R}^d \to \mathbb{R}$ with support radius h, we use cubic splines [47]

$$W^h(\boldsymbol{r}) = \sigma_d \begin{cases} 6(q^3-q^2)+1 & 0 \le q \le \frac{1}{2} \\ 2(1-q)^3 & \frac{1}{2} < q \le 1 \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $q = \frac{\|\boldsymbol{r}\|}{h}$ and σ_d is a kernel normalization factor depending on the dimension d: $\sigma_1 = \frac{4}{3h}$; $\sigma_2 = \frac{40}{7\pi h^2}$; and $\sigma_3 = \frac{8}{\pi h^3}$, respectively.

Regarding the discretizing of differential operators, the difference formula of SPH has a more accurate discretization than the symmetric formula [21]. Though not fully momentum-conserving, the difference formula is most commonly used for intermediate discretizations. As the whole vorticity-loss computations do not yield ultimate forces applied to fluid

particles directly, we use the difference formula to discretize the curl, gradient, and divergence operators [48] as

$$\nabla \times \boldsymbol{u}^t(\boldsymbol{x}_i) = \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} \boldsymbol{u}^t_{ij} \times \nabla W^h(\boldsymbol{x}_{ij}), \tag{6}$$

$$\nabla \boldsymbol{u}^t(\boldsymbol{x}_i) = \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} \boldsymbol{u}^t_{ij} \otimes \nabla W^h(\boldsymbol{x}_{ij}), \tag{7}$$

$$\nabla \cdot \boldsymbol{u}^t(\boldsymbol{x}_i) = \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} \boldsymbol{u}^t_{ij} \cdot \nabla W^h(\boldsymbol{x}_{ij}), \tag{8}$$

where $\boldsymbol{u}^t_{ij} = \boldsymbol{u}^t(\boldsymbol{x}_i) - \boldsymbol{u}^t(\boldsymbol{x}_j)$ and $\boldsymbol{a} \otimes \boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^T$.

For the Laplacian ∇^2, we use the following formulation

$$\nabla^2 \boldsymbol{u}^t(\boldsymbol{x_i}) = 2(d+2) \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} \frac{\boldsymbol{u}^t_{ij} \cdot \boldsymbol{x}_{ij}}{\|\boldsymbol{x}_{ij}\|^2 + 0.01h^2} \nabla W^h(\boldsymbol{x}_{ij}), \tag{9}$$

where the h^2 term is a regularization factor. This enhances numerical stability by reducing 2^{nd} order derivatives to 1^{st} order ones [49].

We enhance the continuity of the vorticity loss $\boldsymbol{\omega}_L$ by the following smoothing using the SPH kernel

$$\tilde{\boldsymbol{\omega}}^t_L(\boldsymbol{x}_i) = \sum_{j \in \mathcal{N}_i} \frac{m_j}{\rho_j} \boldsymbol{\omega}^t_L(\boldsymbol{x}_j) W^h(\boldsymbol{x}_{ij}). \tag{10}$$

Vortex methods simulate vortical flows by an explicit vorticity field which requires complex solving for vorticity at solid boundaries to enforce no-slip and no-through boundary conditions [7], [9]. Using the vorticity loss provides an implicit scheme allowing us to enhance vortical flows *directly* in the velocity field without solving for boundary vorticity.

B. Solving the VVPE

Recovering a velocity field from a vorticity field requires solving the VVPE

$$-\nabla^2 \boldsymbol{u} = \nabla \times \boldsymbol{\omega}. \tag{11}$$

Using Green's function $G : \mathbb{R}^d \to \mathbb{R}$ for the Laplacian ∇^2, Eq. (11) has a solution with infinite domains as

$$\boldsymbol{u}(\boldsymbol{x}) = \nabla \times [(G * \boldsymbol{\omega})(\boldsymbol{x})] = \int_{\Omega_s} \nabla G(\boldsymbol{x} - \boldsymbol{y}) \times \boldsymbol{\omega}(\boldsymbol{y}) \, \mathrm{d}s_y, \tag{12}$$

where $*$ denotes the convolution product, Ω_s is the simulation domain, $\mathrm{d}s_y$ denotes the differential volume element at position $\boldsymbol{y}$, and $G(\boldsymbol{x}) = \frac{1}{2\pi} \ln \|\boldsymbol{x}\|$ (for $d = 2$) and $\frac{1}{4\pi \|\boldsymbol{x}\|}$ (for $d = 3$), respectively. The vortex particle method discretizes the vorticity field by a set of particles $\boldsymbol{x}_i$ as

$$\boldsymbol{\omega}(\boldsymbol{x}) = \sum_{i \in \mathcal{H}} \boldsymbol{\Gamma}(\boldsymbol{x}_i) \delta(\boldsymbol{x} - \boldsymbol{x}_i), \tag{13}$$

where $\boldsymbol{\Gamma} = \oint_C \boldsymbol{u} \, \mathrm{d}s \approx v\boldsymbol{\omega}$ is the vortex strength or circulation; C is a closed curve (2D) or surface (3D) giving the boundary of a vortex particle; $\mathcal{H}$ is the set of all vortex particle indices; $\delta : \mathbb{R}^d \to \mathbb{R}$ is the Dirac delta function; and v is the vortex particle volume. Using Eq. (13), we can rewrite Eq. (12) as

$$\boldsymbol{u}(\boldsymbol{x}) = \sum_{i \in \mathcal{H}} \nabla G(\boldsymbol{x} - \boldsymbol{x}_i) \times \boldsymbol{\Gamma}(\boldsymbol{x}_i), \tag{14}$$

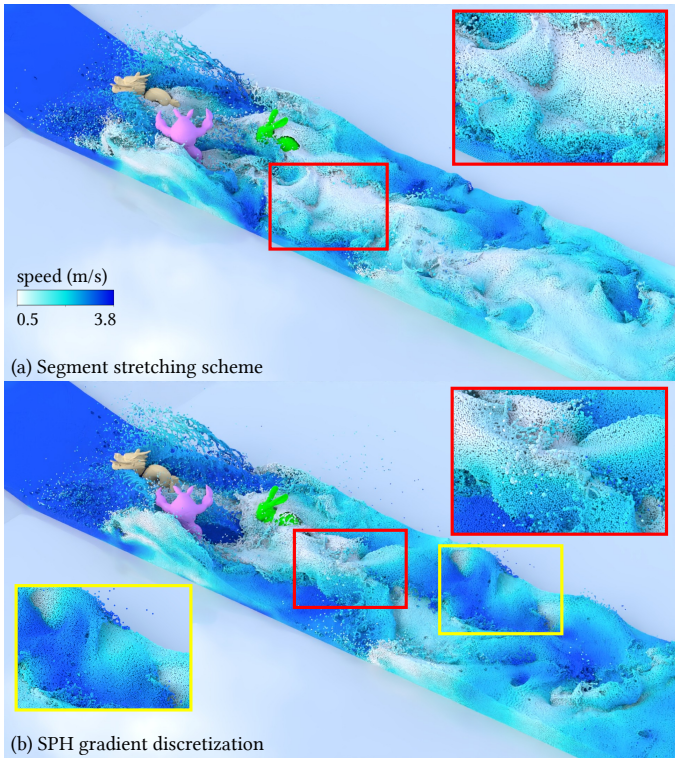


Fig. 2. Comparison of (a) the segment stretching scheme [31] and (b) SPH gradient discretization in a high-velocity stream with multiple static obstacles (green, pink, orange). Particle colors encode speed (white: low speed; blue: high speed). Both methods yield stable vortical flows in our experiments. As the schemes produce comparable results, we use the simpler SPH gradient discretization in our method to compute vortex stretching.

which is known as the Biot-Savart law [28].

To preserve vortex features in SPH flows, we convert the smoothed vorticity loss $\tilde{\boldsymbol{\omega}}_L$ (Eq. (10)) into a velocity loss

$$\boldsymbol{u}_L(\boldsymbol{x}) = v \sum_{i \in \mathcal{H}} \nabla G(\boldsymbol{x} - \boldsymbol{x}_i) \times \tilde{\boldsymbol{\omega}}_L(\boldsymbol{x}_i) \tag{15}$$

and use this loss to enforce vortical motions in the original velocity via

$$\bar{\boldsymbol{u}}(\boldsymbol{x}) = \boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{u}_L(\boldsymbol{x}). \tag{16}$$

The simulation can be unstable due to the singularity of the Green's function G at the zero point. We alleviate this by modifying G with the vortex blob scheme [50], [20] as

$$\tilde{G}(\boldsymbol{x}) = \begin{cases} \frac{1}{2\pi} \ln \left(\|\boldsymbol{x}\| + h \cdot e^{-\|\boldsymbol{x}\|/\theta} \right) & \text{for } d = 2 \\ -\frac{1}{4\pi \left(\|\boldsymbol{x}\| + h \cdot e^{-\|\boldsymbol{x}\|/\theta} \right)} & \text{for } d = 3, \end{cases} \tag{17}$$

where $\theta = 1 \times 10^{-6}$ is a small regularization factor.

C. Discussion of Vortex Stretch

The vortex stretch term $(\boldsymbol{\omega} \cdot \nabla)\boldsymbol{u}$ in Eq. (3) describes the bending or stretching of the vortex central curve in the 3D case; this term is zero in 2D. Most schemes avoid using finite differences when computing this gradient as this can be numerically unstable. Some methods replace particle representation with segments [9] or filaments [10] to directly model the updating result of the vortex stretch term. Yet, handling elements other than particles introduces new complexities.

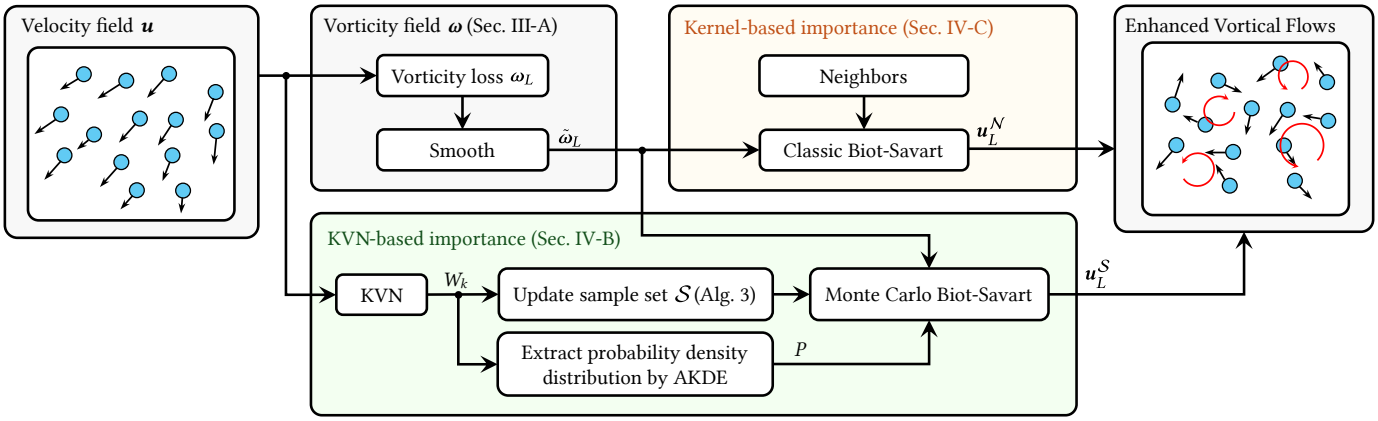


Fig. 3. Workflow of our approach. We evaluate vorticity loss at each time step using two particle groups defined by *kernel-based* importance and *KVN-based* importance. For the former, we sample neighbor particles using the classic Biot-Savart law. For the latter, we use the Monte Carlo Biot-Savart estimator. We update the sample set $\mathcal{S}$ using an importance sampling scheme and use AKDE to compute a probability density distribution from the KVN to get the KVN-based importance. Velocity loss computed by both particle groups contributes to enhancing vortical flows in SPH.

Zhang et al. [31] leverage Kelvin's circulation theorem and vortex segments to compute the vortex stretch by

$$\frac{D\boldsymbol{\omega}}{Dt}(\boldsymbol{x}) \approx \frac{\boldsymbol{\omega}(\boldsymbol{x})}{\|\boldsymbol{x}_a - \boldsymbol{x}_b\|}(\boldsymbol{u}(\boldsymbol{x}_a) - \boldsymbol{u}(\boldsymbol{x}_b)), \tag{18}$$

where $\boldsymbol{x}_a = \boldsymbol{x} + \frac{\|\boldsymbol{x}_a - \boldsymbol{x}_b\|}{2}\frac{\boldsymbol{\omega}(\boldsymbol{x})}{\|\boldsymbol{\omega}(\boldsymbol{x})\|}$ and $\boldsymbol{x}_b = \boldsymbol{x} - \frac{\|\boldsymbol{x}_a - \boldsymbol{x}_b\|}{2}\frac{\boldsymbol{\omega}(\boldsymbol{x})}{\|\boldsymbol{\omega}(\boldsymbol{x})\|}$ are the endpoints of a vortex segment. This scheme can be used flexibly in grid- or particle-based simulation frameworks and has shown good stability in practice [12].

Figure 2 compares this segment-based scheme with the difference form of the SPH gradient approximation in Eq. (7). The scenario simulates a high-velocity stream with multiple static obstacles, color coded by particle speed (white is low, blue is high speed). Both approaches provide effective vortical flow simulations without any observed numerical instability. As such, we approximate vortex stretching using the simpler SPH gradient discretization.

Algorithm 1 Dynamic importance Monte Carlo simulation of vortical flows

Input: time step Δt, current velocity $\boldsymbol{u}^t$
Output: velocity $\boldsymbol{u}^{t+\Delta t}$ at next time step

1: **for all** fluid particles i **do**
2: $\quad\bar{\boldsymbol{u}}^t(\boldsymbol{x}_i) \leftarrow \text{ApplyNonPressureForces}(\Delta t, \boldsymbol{u}^t)$
3: **for all** fluid particles i **do**
4: $\quad\tilde{\boldsymbol{\omega}}_L^t(\boldsymbol{x}_i) \leftarrow \text{QuantifyVorticityLoss}(\Delta t, \boldsymbol{u}^t)$ ▷ Sec. III-A
5: $\quad W_k(\boldsymbol{x}_i) \leftarrow \text{ComputeKVN}(\boldsymbol{u}^t)$ ▷ Eq. (20)
6: **for all** fluid particles i **do**
7: $\quad\hat{f}(\boldsymbol{x}_i) \leftarrow \text{ExtractProbabilityDistribution}(W_k)$ ▷ Eq. (23)
8: $\mathcal{S} \leftarrow \text{UpdateSamples}(W_k)$ ▷ Alg. 3
9: $\boldsymbol{u}_L^t \leftarrow \text{VorticityLossToVelocityLoss}(\tilde{\boldsymbol{\omega}}_L^t, \hat{f}, \mathcal{S})$ ▷ Alg. 2
10: **for all** fluid particles i **do**
11: $\quad$**if** $\|\boldsymbol{u}_L^t(\boldsymbol{x}_i)\| < \|\bar{\boldsymbol{u}}^t(\boldsymbol{x}_i)\|$ **do**
12: $\quad\quad\bar{\boldsymbol{u}}^t(\boldsymbol{x}_i) \leftarrow \bar{\boldsymbol{u}}^t(\boldsymbol{x}_i) + \boldsymbol{u}_L^t(\boldsymbol{x}_i)$
13: **for all** fluid particles i **do**
14: $\quad\boldsymbol{u}^{t+\Delta t}(\boldsymbol{x}_i) \leftarrow \text{PressureProjection}(\Delta t, \bar{\boldsymbol{u}}^t)$

IV. Dynamic Importance Monte Carlo Vortical Flows

In SPH, particles are discrete fluid parcels that record all fluid attributes, *e.g.*, velocity, pressure, and vorticity. Solving the VVPE (Eq. (12)) requires integration over the whole simulation domain, *i.e.*, set $\mathcal{H}$ to all fluid particles and then evaluate Eq. (14) *for each particle*. This has $O(N^2)$ complexity for N particles, which is prohibitive in practice. The Monte Carlo method [12], [27] increases efficiency by replacing the full set $\mathcal{H}$ in Eq. (14) with a smaller set $\mathcal{S}$ as

$$\boldsymbol{u}_L(\boldsymbol{x}) = \frac{v}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\frac{\nabla\tilde{G}(\boldsymbol{x} - \boldsymbol{x}_i) \times \tilde{\boldsymbol{\omega}}_L(\boldsymbol{x}_i)}{P(\boldsymbol{x}_i)}, \tag{19}$$

where P is the discrete probability of samples. Estimation accuracy highly depends on the sample distribution: a *well-representative* sample set yields low variance and accurate estimation of the entire system.

A. Our proposal

Figure 3 shows our entire workflow. We use the Kinematic Vorticity Number (KVN) to estimate the solution of the VVPE by two types of importance. *Kernel-based* importance uses the classic Biot-Savart law for neighbor particles without the need for sampling. *KVN-based importance* is modeled by the *dynamically updated probability density distribution* extracted using AKDE from the KVN. We draw particle samples by a rejection sampling scheme for the Monte Carlo method.

Algorithm 1 lists our full method. We first apply non-pressure forces to obtain the intermediate velocity $\bar{\boldsymbol{u}}$ (line 3). We next evaluate the smoothed vorticity loss $\tilde{\boldsymbol{\omega}}_L$ (line 5) and the KVN (line 6), which we use further to enhance vortical flows. We use KVN-mapped acceptance and rejection probabilities to update the sample set $\mathcal{S}$ (line 9) by importance sampling (Alg. 3). For this, we approximate the probability density distribution modeling the KVN-based importance by Eq. (23) with KVN giving the particle weights (line 8). Next, we convert the smoothed vorticity loss $\tilde{\boldsymbol{\omega}}_L$ to the velocity loss $\boldsymbol{u}_L$ (line 10, see also Alg. 2). We use this velocity loss only if it does not exceed in magnitude the intermediate velocity, *i.e.*, when $\|\boldsymbol{u}_L\| < \|\bar{\boldsymbol{u}}\|$ (lines 12–13). This avoids potential instabilities when the vortex particle volume v is large, as $\boldsymbol{u}_L$ would also be large (see Eq. (19)). Finally, we perform pressure projection to enforce incompressibility (line 15).

Algorithm 2 Vorticity loss to velocity loss conversion

Input: vorticity loss $\tilde{\boldsymbol{\omega}}_L^t$, probabilities $\hat{f}$, sample set $\mathcal{S}$
Output: velocity loss $\boldsymbol{u}_L^t$

for all fluid particles i **do**
  $\mathcal{N}_i \leftarrow$ RecordNeighboringParticles($\boldsymbol{x}_i$)
  $\boldsymbol{u}_L^{\mathcal{N}}(\boldsymbol{x}_i) \leftarrow$ BiotSavart($\mathcal{N}_i$, $\tilde{\boldsymbol{\omega}}_L^t$) ▷ Eq. (26)
for all fluid particles i **do**
  $P(\boldsymbol{x}_i) \leftarrow$ DiscretizeProbablity($\mathcal{N}_i$, $\hat{f}$) ▷ Eq. (24)
for all fluid particles i **do**
  $n_i \leftarrow$ CountNonNeighborSamples($\boldsymbol{x}_i$)
  $\boldsymbol{u}_L^{\mathcal{S}}(\boldsymbol{x}_i) \leftarrow$ MonteCarloBiotSavart($\mathcal{S}$, $\tilde{\boldsymbol{\omega}}_L^t$, P, n_i) ▷ Eq. (25)
for all fluid particles i **do**
  $\boldsymbol{u}_L^t(\boldsymbol{x}_i) \leftarrow \boldsymbol{u}_L^{\mathcal{N}}(\boldsymbol{x}_i) + \boldsymbol{u}_L^{\mathcal{S}}(\boldsymbol{x}_i)$

B. KVN-based importance

Fluid simulations typically use vorticity $\boldsymbol{\omega}$ to evaluate rotation. When approximating the VVPE solution, Rioux-Lavoie et al. [12] found that a sample distribution following the vorticity magnitude yields more accurate results. Vorticity is twice the angular velocity [20], so vorticity magnitude also increases with angular velocity. Separately, when considering particle importance, particles within vortex cores are clearly (more) important in vortical flow simulations. Given the direct relation with angular velocity, if vorticity is used to assess importance, particles within high-angular-velocity vortex cores are more important than low-angular-velocity ones. Particles outside vortex cores can also have larger vorticity and higher importance than those with low-angular-velocity. Yet, given the character of rigid-body-like rotations, particles within vortex cores should be treated equally. Measuring particle importance should thus be *decoupled* from angular velocity and purely depend on the character of rotations.

The KVN is a dimensionless measure of how close a rotation resembles a rigid body rotation. It can be defined as the ratio of the magnitudes of the rotation tensor $\boldsymbol{\Omega}$ and strain tensor $\boldsymbol{S}$ [18], [38], that is

$$W_k = \frac{\|\boldsymbol{\Omega}\|}{\|\boldsymbol{S}\|} = \frac{\|\boldsymbol{\omega}\|}{2\|\boldsymbol{S}\|}, \tag{20}$$

where $\boldsymbol{\Omega} = \frac{1}{2}(\nabla\boldsymbol{u} - \nabla\boldsymbol{u}^T)$, $\boldsymbol{S} = \frac{1}{2}(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)$, and $\|\cdot\|$ is the vector or tensor 2-norm. W_k determines three cases: Rotation rate exceeds strain rate ($W_k > 1$); pure shear motion ($W_k = 1$); and strain rate exceeds rotation rate ($W_k < 1$).

KVN measures pure rotations based on the normalized strain rate, decoupled from angular velocity magnitude. Since only the *kind* of rotation is measured, two vortex cores with different vorticities or angular velocities can have equal W_k and importance. Our experiments showed that the KVN can more effectively highlight vortex cores than vorticity alone (see further Sec. V and Fig. 12).

The Monte Carlo method requires a probability density distribution representing our KVN-based importance to solve the VVPE. We consider for this the Kernel Density Estimation (KDE) [51], defined for a sample set $\mathcal{S}$ as

$$\hat{f}(\boldsymbol{x}_i) = \frac{1}{|\mathcal{S}|\eta^d} \sum_{j\in\mathcal{S}} K^\eta(\boldsymbol{x}_{ij}), \tag{21}$$

where d is the spatial dimension; and $K^\eta : \mathbb{R}^d \to \mathbb{R}$ is a kernel function with bandwitdth η. For K^η, we use the dyadic product of 1D Epanechnikov kernels. In $d = 3$ dimensions, $K^\eta(\boldsymbol{x}) = K(q_x)K(q_y)K(q_z)$, where $\boldsymbol{x} = (x, y, z)$, $q_x = \frac{x}{\eta}$, $q_y = \frac{y}{\eta}$, $q_z = \frac{z}{\eta}$, and

$$K(q) = \begin{cases} \frac{3}{4}(1-q^2) & \text{for } |q| \le 1, \\ 0 & \text{for } |q| > 1. \end{cases} \tag{22}$$

The Epanechnikov kernel provides good performance in minimizing the error between estimation and benchmark values [52]. Given its compact support, Eq. (21) can be easily and efficiently implemented in typical SPH frameworks. We further speed up the computation of $\hat{f}$ by setting η to the support radius h of SPH. However, h is typically quite small, as it only needs to model how a particle moves under the influence of its *neighbors*. This yields fewer samples than KDE's typical bandwidth, which can result in a highly jittering probability density distribution. We address this by using AKDE instead of KDE. This allows us to set *weights* to fluid particles and estimate the probability density function by querying weighted fluid particles directly [19] as

$$\hat{f}(\boldsymbol{x}_i) = \frac{1}{\eta^d \sum_{j\in\mathcal{F}} W_k(\boldsymbol{x}_j)} \sum_{j\in\mathcal{N}_i} W_k(\boldsymbol{x}_j) K^\eta(\boldsymbol{x}_{ij}), \tag{23}$$

where particle weights are set to W_k, the KVN-based importance; and $\mathcal{F}$ is the set of all fluid particles. Enforcing incompressibility at each time step keeps a relatively uniform fluid particle distribution. In turn, this ensures that all particles i have a stable number of neighbors being used when computing Eq. (23), which removes the jittering problem.

We next compute the normalized discrete probability with neighbor particles excluded as

$$P(\boldsymbol{x}_i) = \frac{\hat{f}(\boldsymbol{x}_i)}{\sum_{j\in\mathcal{F}\setminus\mathcal{N}_i} \hat{f}(\boldsymbol{x}_j)}. \tag{24}$$

Normalization ensures that all discrete probabilities sum to one over all particles while accounting for each particle's relative KVN-based importance as given by AKDE. Importantly, note that the neighbors $\mathcal{N}_i$ of particle i are considered separately in the kernel-based importance (see next Sec. IV-C). As such, some particles in $\mathcal{S}$ could already be considered by the kernel-based importance. We remove the influence of this overlap by ignoring particle samples in $\mathcal{N}_i$ when we evaluate Eqn. (24). The Monte Carlo Biot-Savart estimator based on non-neighbor samples thus becomes

$$\boldsymbol{u}_L^{\mathcal{S}}(\boldsymbol{x}_i) = \frac{v}{|\mathcal{S}\setminus\mathcal{N}_i|} \sum_{j\in\mathcal{S}\setminus\mathcal{N}_i} \frac{\nabla\tilde{G}(\boldsymbol{x}_{ij}) \times \tilde{\boldsymbol{\omega}}_L(\boldsymbol{x}_j)}{P(\boldsymbol{x}_j)}. \tag{25}$$

C. Drawing Lagrangian Particle Samples

Sample-drawing approaches in the literature [11], [13], [15] provide *points* rather than actual Lagrangian *particles* as sampling results. In the context of vortical flow simulations, the loss of the Lagrangian attribute could smooth out the advection of vortices. Points can be converted to actual particles using nearest-neighbor or other schemes. Still, the density of point

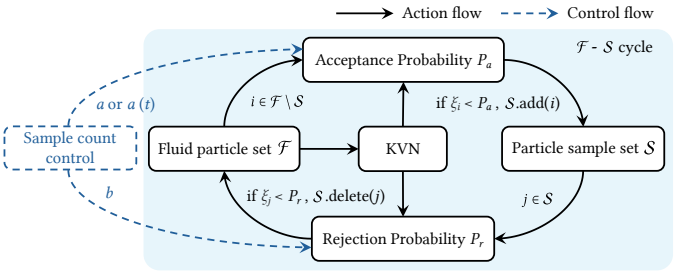


Fig. 4. Dynamic sampling workflow. We map the KVN to acceptance (P_a) and rejection (P_r) probabilities. P_a determines the addition of new members to the sample set $\mathcal{S}$. P_r is used to delete unneeded members from $\mathcal{S}$.

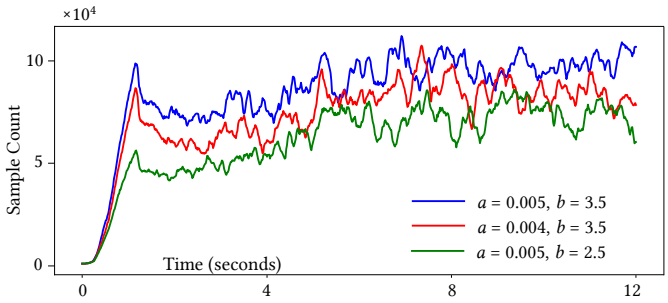


Fig. 5. Sample count in the Von Kármán vortex street scenario shown in Fig. 7 for different hyperparameter values a and b for P_a and P_r. Increasing a increases P_a, while increasing b reduces P_r, both yielding more samples.

samples could far exceed that of actual particles, as their positions are unaware of the particle-based discretization in our SPH framework. Point-particle converting schemes could lead to *repeated sampling* of the same particles and a waste of computational resources.

To solve the VVPE, we must consider two aspects. First, we have the kernel-based importance given by the Biot-Savart kernel ∇G, which attaches more importance to nearby particles and less importance to distant ones. This adapts perfectly to the SPH framework. We can naively consider all neighbor particles within the SPH support radius and recover the velocity loss by the classic Biot-Savart law as

$$\boldsymbol{u}_L^{\mathcal{N}}(\boldsymbol{x}_i) = v \sum_{j \in \mathcal{N}_i} \nabla \tilde{G}(\boldsymbol{x}_{ij}) \times \tilde{\boldsymbol{\omega}}_L(\boldsymbol{x}_j). \tag{26}$$

Yet, particles outside $\mathcal{N}_i$ contribute to the overall shape of vortices [31] so we cannot simulate visually authentic vortical flows with neighbor particles alone.

Algorithm 3 Update samples in $\mathcal{S}$

Input: KVN values W_k, current sample set $\mathcal{S}$
Output: updated sample set $\mathcal{S}$

for all fluid particles i **do**
  $\xi \leftarrow$ GetRandomNumber() $(0 \le \xi < 1)$
  if $i \notin \mathcal{S}$ **do**
    $P_a \leftarrow$ GetAcceptanceProbability($W_k(\boldsymbol{x}_i)$) ▷ Eq. (27)
    if $\xi < P_a$ **do**
      Add i to $\mathcal{S}$
  else if $i \in \mathcal{S}$ **do**
    $P_r \leftarrow$ GetRejectionProbability($W_k(\boldsymbol{x}_i)$) ▷ Eq. (28)
    if $\xi < P_r$ **do**
      Delete i from $\mathcal{S}$

The second aspect determining the importance of particles is their contribution to *vortical flows*. We should obtain particles within vortex cores based on their KVN values. A simple idea is to directly sample all particles with $W_k > 1$ (for which rotation rate exceeds strain rate, see Sec. IV-B). Yet, this hard cut-off that ignores particles with $W_k \le 1$ can disturb the continuous distribution of vortices and produce visual artifacts.

A better option is to sample such particles using *acceptance* and *rejection* probabilities $P_a : \mathbb{R}^d \to [0,1]$ and $P_r : \mathbb{R}^d \to [0,1]$. At each time step, we generate a random number $\xi \in [0,1]$ for each particle and mark the particle as a sample if $P_a > \xi$. To keep continuity of vortical effects, we keep samples at the current time step in the next time step unless $P_r > \xi$. Each particle is queried only once per time step, which is efficient.

We could set P_a to the discrete probability P (Eq. (24)). But, as we query each particle only once per time step, we lose control over the created sample count. A better option is to draw samples following the KVN-based importance, *i.e.*, distribute them *densely* in areas where W_k is large and *sparsely* where W_k is small. We propose next such a scheme with the acceptance probability

$$P_a(\boldsymbol{x}) = 1 - \frac{1}{[W_k(\boldsymbol{x}) + 1]^a}, \tag{27}$$

and the rejection probability

$$P_r(\boldsymbol{x}) = e^{-bW_k(\boldsymbol{x})}, \tag{28}$$

where the $a > 0$ and $b > 0$ hyperparameters control the sample creation and deletion rates respectively. Alg. 3 and Fig. 4 show this procedure for dynamically sampling particles.

Fig. 5 shows how the sample count $|\mathcal{S}|$ changes with the values a and b (Eqs. (27) and (28)) for the Von Kármán vortex street scenario in Fig. 7. The sample count increases with larger a (since this increases acceptance P_a) or b (since this decreases rejection P_r).

Our sampling scheme is an *accumulation* process. For scenarios with open boundaries (Fig. 2), particles near an outflow side stay longer in the domain and have a larger chance to be used as samples. This can lead to dense samples near outflows and sparse samples near inflows. We alleviate this by setting a to a *time-dependent* form for each particle i as

$$a_i = 0.018e^{-ct_i} + 0.004, \tag{29}$$

where t_i is the time that particle i stays in the simulation domain; and $c > 0$ is a decay rate. This makes a decay over time and converges to a static value for the acceptance probability P_a, allowing relatively higher acceptance probabilities for particles near inflows.

V. Results

We next showcase our method by several experiments. All simulations were implemented using the *Taichi* graphics programming language [53]. Computations were performed on a desktop PC equipped with an NVIDIA Quadro RTX 4000 GPU, a 3GHz 8-core Intel Core i7-9700 processor, and 32GB memory. For rendering, we use the *Karma* renderer in

TABLE I
PARAMETER VALUES, AVERAGE SAMPLE AMOUNTS, AND AVERAGE COMPUTATIONAL TIME PER FRAME FOR ALL EXPERIMENTS.

Scenario	Domain size [m]	Avg. part. count	Particle radius [mm]	Time step [ms]	Method	Coefficients	Avg. samp. count	Avg. comp. time / frame [s]
Fig. 2	$11 \times 1.6 \times 2$	2.7 M	6	2	Segment stretching	$\beta = 0.7$, $b = 2.5$, $a = 0.018e^{-12t} + 0.004$	84 k	7.31
					SPH gradient discretization	$\beta = 1.0$, $b = 2.5$, $a = 0.018e^{-12t} + 0.004$	71 k	6.34
Fig. 6	$3.5 \times 2.0 \times 3.5$	2.4 M	6	2	**Ours**	$\beta = 0.7$, 1.0, 1.2, $a = 0.018e^{-12t} + 0.004$, $b = 1.5$	44 k	5.50
Fig. 7	4×1	130 k	3	1	Divergence-free SPH	-	-	0.10
					Uniform Monte Carlo	$\beta = 0.1$	10 k	0.12
					Ours	$\beta = 0.1$, $b = 3.5$, $a = 0.018e^{-5t} + 0.004$	10 k	0.12
Fig. 11	1×1	14 k	4.5	1	Vorticity-based sampling	$\beta = 0.2$, $b = 1$, $a = 0.001$	6.2 k	0.023
					Ours	$\beta = 0.2$, $b = 4.5$, $a = 0.005$	6.2 k	0.023
		126 k	1.5	0.2	Divergence-free SPH	-	-	0.008
					Vorticity-based sampling	$\beta = 0.2$, $b = 1$, $a = 0.0002$	61.9 k	0.165
Fig. 13	1×1	14 k	4.5	1	Divergence-free SPH	-	-	0.008
					Vorticity confinement	$\epsilon = 0.0002$	-	0.012
					Micropolar fluids	$\nu_t = 0.004$	-	0.012
					Ours	$\beta = 0.08$, $b = 2.5$, $a = 0.005$	2 k	0.019
Fig. 14	$9 \times 1.6 \times 2$	2.7 M	6	2	Divergence-free SPH	-	-	1.06
					Classic Biot-Savart	$\beta = 2.0$	-	158.44
					Ours	$\beta = 1.3$, $b = 2.0$, $a = 0.018e^{-6t} + 0.004$	47 k	4.79
Fig. 15	$3.5 \times 2 \times 3.5$	2.4 M	6	2	Vorticity confinement	$\epsilon = 0.0001$	-	1.68
					Micropolar fluids	$\nu_t = 0.1$	-	1.41
					Ours	$\beta = 1.0$, $b = 1.5$ $a = 0.018e^{-12t} + 0.004$	40 k	4.93
Fig. 16	$3.8 \times 5.0 \times 3.8$	2.5 M	6	1	**Ours**	$\beta = 0.7$, $b = 1.5$, $a = 0.018e^{-12t} + 0.004$	38 k	3.52
Fig. 17	$6 \times 4 \times 6$	2.5 M	8	2	**Ours**	$\beta = 1.5$, $b = 1.5$, $a = 0.018e^{-12t} + 0.004$	43 k	3.66
Fig. 18	$6 \times 2 \times 3$	2.6 M	6	1	**Ours**	$\beta = 1.0$, $b = 1.5$, $a = 0.018e^{-12t} + 0.004$	40 k	3.92

Houdini 20.0. We encourage readers to watch our results in the accompanying video.

For pressure projection, we use the divergence-free SPH [54] due to its good performance in maintaining incompressibility. We use a particle-based scheme to enforce solid boundaries in our experiments [16]. Table I gives parameter values and performance metrics for each experiment. For brevity, we next focus on important observations in each experiment, and refer to Tab. I for specific settings.

Regulating vortex strength. Practical 3D content creation requires the controllability of visual effects. In our approach, we can control the overall vortex profile by adjusting the volume v of vortex particles (Eq. (25)). Given some baseline value v_f, let next $v = \beta v_f$, where β is a scaling factor. To evaluate the effect of β, we consider a dynamic scenario where four propellers rotate in water with an angular velocity of $0.5\pi\,\mathrm{rad/s}$ (Fig. 6). Rotation directions are opposite for any two adjacent propellers. This scenario is rendered as (left) particles with color-coded speed and (right) translucent water. This experiment clearly shows that increasing β enhances vortical flows in the simulations – so this parameter offers a simple but effective way to control the overall vortex motions.

2D Von Kármán vortex street. We simulate this classical scenario using a $0.1\,\mathrm{m}$ disk with fluid particles initialized at a velocity of $3\,\mathrm{m/s}$. We compare the divergence-free SPH [54], uniform Monte Carlo [27], and our method. Figure 7 shows results color-coded by vorticity (top row) and by speed (middle row). Our approach produces more distinct vortex cores with rotational motions that are more effectively enforced than in the compared methods. To illustrate the advantage of our approach in sample distribution, the images in Fig. 7 (bottom row) show the fluid particles colored by the adaptive kernel density computed using Eq. (23). The results show that our approach effectively identifies vortex core areas which have high KVN. Our approach performs importance sampling in these areas using Lagrangian particle samples that closely follow the advection of vortex cores. As our method densely samples vortex cores, vortex energy is 'gathered' to enhance vortical motions. In contrast, uniform sampling leads to a visible loss of vortex energy.

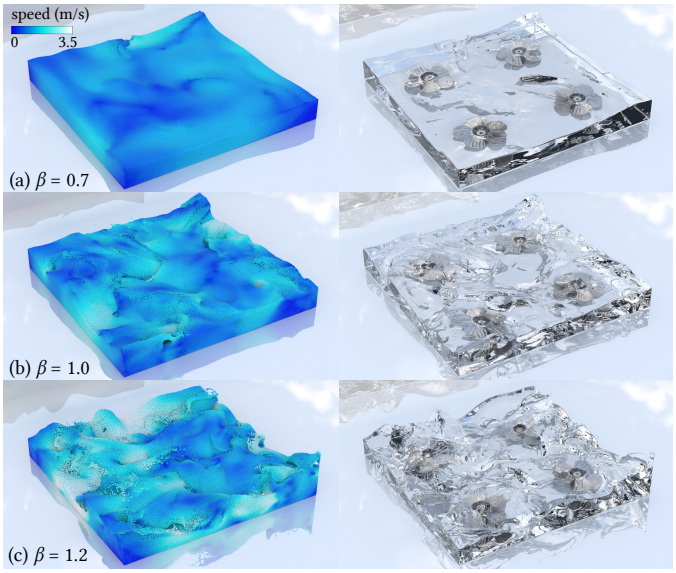


Fig. 6. Setting the scaling factor β using a scenario with four propellers rotating in water with an angular velocity of $0.5\pi\,\mathrm{rad/s}$. Adjacent propellers rotate in opposite directions. Rendering uses particles color-coded by speed (left; dark blue is low, light blue is high, speed) and water (right). We observe that increasing β produces more pronounced vortices.

2D Von Kármán vortex street with two cubes. We conduct another Von Kármán vortex street experiment with two cubes as obstacles (Fig. 8) to highlight the difference between querying only neighbor particles and querying samples outside the neighbor scope. Images (a), (b), and the average vorticity norm plotted in (d) show that querying neighbor particles yields slightly enforced vorticity as the Green's function G is large for closer particles. Yet, the vortex enforcing result is not visually prominent enough for this scheme to work on its own. Image (c) and the vorticity norm plot in (d) show that the enforced vorticity is mainly obtained from samples *outside* the neighbor scope.

2D rotating panel. To further show the advantage of our approach *vs* previous uniform sampling methods [27], we conduct a series of experiments using a rotating panel with 1.5π rad/s (Fig. 9). For uniform sampling, vortices are randomly distributed within the domain when using $N_s = 400$ samples (a); for $N_s = 3000$ samples, almost no vortices are produced (b). In contrast, our dynamic KVN-based importance sampling scheme creates coherent vortex patterns for both sample counts (c,d) – that is, we see a central vortex and two smaller ones around it, and no random vortex distributions.

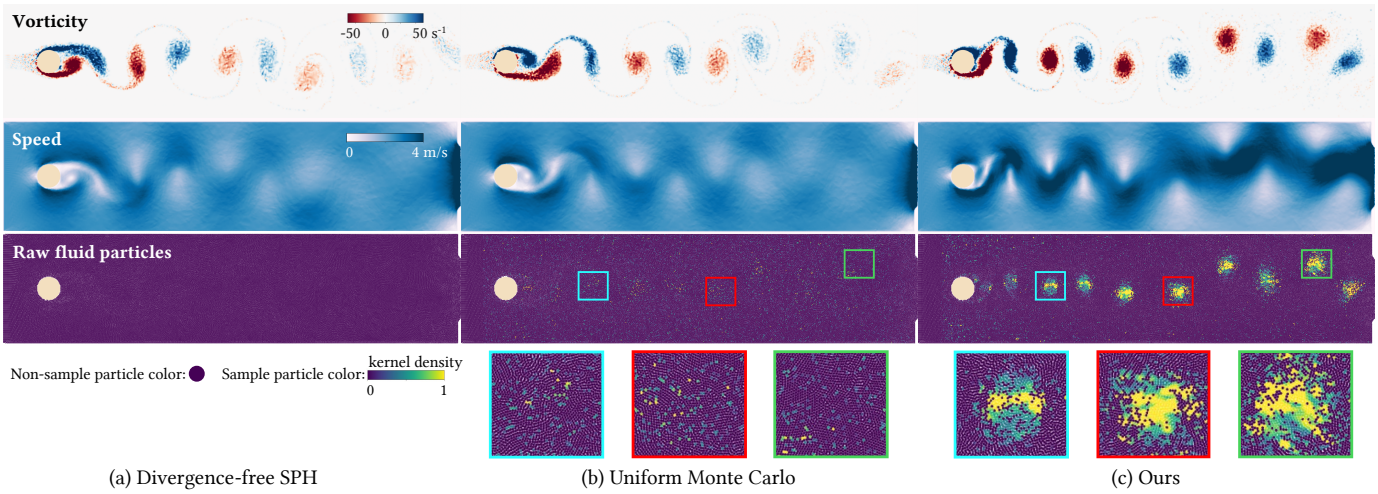
Figure 10(a) shows the average vorticity norm for different sample counts. We see that uniform sampling largely enforces vorticity when $N_s < 1000$. However, the vorticity decays rapidly with N_s increasing. When N_s is small, the high vorticity is attributed to the high variance of the uniform sampling. Samples are randomly distributed within the domain and each sample is given a too large weight in the Monte Carlo estimation. Thus, vortices with high angular velocity are randomly produced as we saw in Fig. 9(a). We also noticed that this can lead to numerical instability. Moreover, since samples are equally weighted, when N_s increases, the vorticity enforcing effect is *dissipated* by evenly distributed samples without any importance, leading to the result shown in Fig. 9(b). In contrast, our importance-based sampling allocates more samples to high KVN areas. Figure 10(a) shows a more stable and reliable vorticity-enforcing performance of our approach among multiple sample counts, which aligns with the results we saw in Fig. 9(c,d).

Figure 10(b) shows the average computation time per frame of uniform sampling *vs* our method. Our approach is slightly slower than uniform sampling for $N_s > 500$ samples since we need extra sampling and probability computations for our KVN-based importance. Interestingly, uniform sampling is quite slow for sample counts $N_s < 1000$. Indeed, as discussed, small N_s usually leads to high variance and introduces numerical instability. Hence, more pressure projection iterations are needed, which leads to an overall larger computing time. Our approach removes this issue since importance sampling effectively reduces variance.

2D lid driven cavity. We simulate a 2D $1 \times 1\,\mathrm{m}$ box with the top lid driven with $1\,\mathrm{m/s}$. Figure 11 compares divergence-free SPH, the vorticity-based sampling scheme, and our approach by showing the resulting flows with velocity-colored streamlines. Divergence-free SPH with particle count $N = 126\,\mathrm{k}$ (image a) and the vorticity-based sampling scheme with $N = 14\,\mathrm{k}$ (image b) create only one central vortex. The vorticity-based scheme with $N = 14\,\mathrm{k}$ particles can only enforce *existing* vortices. Increasing N to $126\,\mathrm{k}$ particles allows this scheme to create two extra vortices (image c). Our approach shows a similar vortex pattern with image (c) when using only $14\mathrm{k}$ particles (image d). These differences relate to the explained discrepancy between vorticity and the KVN (Sec.IV-B). As vorticity directly relates to angular velocity, the vorticity-based scheme focuses on particles that *already* have a high angular velocity, *i.e.*, part of the large vortex in image (b). This is similar to the vorticity confinement scheme and limits the ability to generate new vortices. In contrast, our KVN-based scheme uses rotation quality as an importance measure. The small vortex cores (low angular velocity, akin to rigid body rotations, and easily dissipated) are densely sampled and further enforced by our approach.

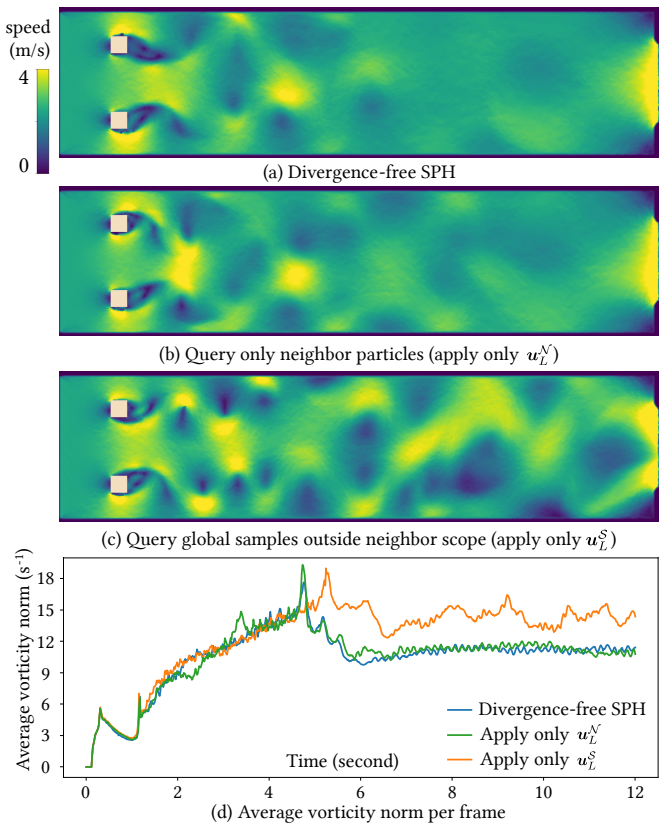
Fig. 12 shows the difference between vorticity $\boldsymbol{\omega}$ and the KVN. Comparing the streamline plot (a) with the corresponding vorticity magnitude (b), we see that vortex cores, *i.e.*, areas where streamlines quickly curl, like the central area in (a), do not necessarily have high vorticity magnitude. Conversely, areas with high vorticity magnitude (bright yellow in (b)) do not always correspond to vortex cores in (a). So, if particle samples were created based on *vorticity* alone, the resulting simulations would not do well in generating new vortices. In contrast, we see that high-KVN values (bright areas in (c)) match well the vortex core areas in (a), so KVN is more indicative of vortex cores than vorticity. In (d), we show the sample particles colored by the adaptive kernel density $\hat{f}$ and non-sample particles colored dark blue. We see that our approach successfully uses KVN-mapped probabilities to produce Lagrangian particle samples that adequately cover vortex core areas while allocating fewer samples to other areas.

2D vortex interaction. Dissipation in SPH typically



(a) Divergence-free SPH (b) Uniform Monte Carlo (c) Ours

Fig. 7. Two-dimensional Von Kármán vortex streets. (a) Divergence-free SPH, (b) uniform Monte Carlo, and (c) our method. **Top row**: Color-coded vorticity. **Middle row**: Color-coded speed. Our approach produces more vortex cores and enhances the rotation of vortices. **Bottom row**: Raw fluid particles with sample particles colored by the adaptive kernel density (Eq. (23)) and non-sample particles dark blue. Our approach achieves a higher concentration of samples within vortex cores. These Lagrangian samples closely follow the advection of vortex cores.



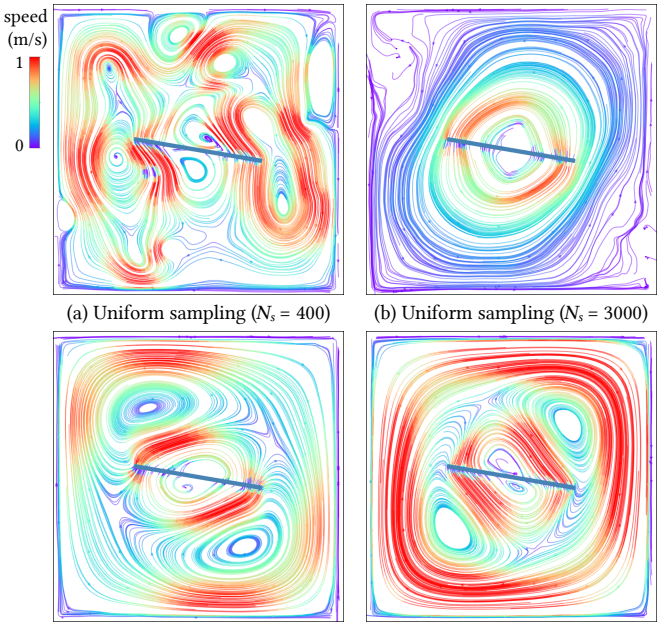
(a) Divergence-free SPH

(b) Query only neighbor particles (apply only $u_L^{\mathcal{N}}$)

(c) Query global samples outside neighbor scope (apply only $u_L^{\mathcal{S}}$)

(d) Average vorticity norm per frame

Fig. 8. Two-dimensional Von Kármán vortex street with double cubes comparing (a) divergence-free SPH, (b) our scheme using only $u_L^{\mathcal{N}}$, and (c) our scheme using only $u_L^{\mathcal{S}}$. Vorticity norm plots over time are shown in (d). The vorticity enhancing effect mainly sources from global samples. Using only $u_L^{\mathcal{N}}$ shows slightly more vorticity than DFSPH but far under what one would arguably desire to reach in practice.

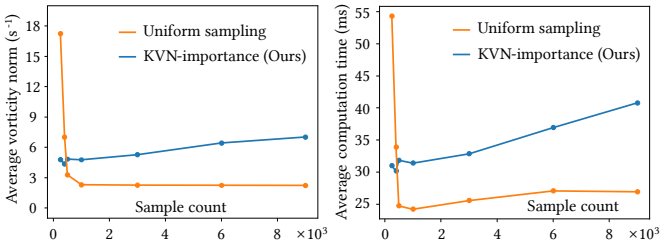


(a) Uniform sampling (N_s = 400) (b) Uniform sampling (N_s = 3000)

(c) KVN-importance (Ours, N_s = 400) (d) KVN-importance (Ours, N_s = 3000)

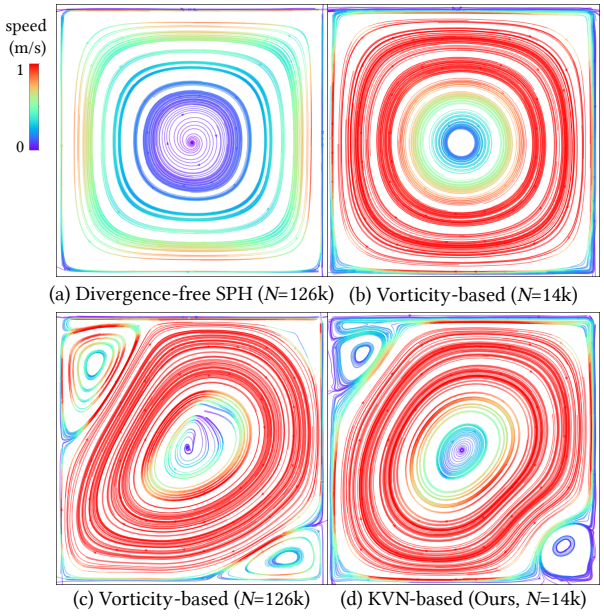
Fig. 9. Two-dimensional rotating panel with an angular velocity of 1.5π rad/s using 30k particles. We compare our scheme with the uniform sampling scheme [27] using different sample count. For the uniform sampling, vortices are randomly produced without any regularity when sample count $N_s = 400$, while there is almost no enforced vortex when $N_s = 3000$. Our KVN-importance sampling maintains coherent enforced vortex patterns regardless of $N_s = 400$ or $N_s = 3000$.

smooths out most vortical flows and kinetic energy. The 2D vortex interaction scenario in Fig. 13 compares the kinetic energy preservation of divergence-free SPH (DF), vorticity confinement (VC) [22], micropolar fluids (MF) [5], and our approach. Four vortices are initialized with an angular velocity of $2\,\mathrm{rad/s}$ in different directions within a $1 \times 1\,\mathrm{m}$ box as shown in image (a). Since all these methods except divergence-free SPH use hyperparameters to control their effects, for fair comparison, we tune these parameters as high as possible



(a) Average vorticity norm (b) Average computation time per frame

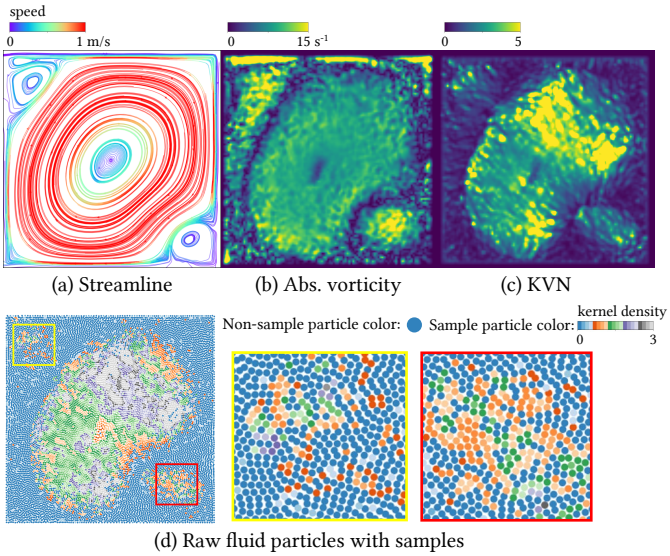
Fig. 10. (a) Average vorticity norm with increasing sample count and (b) average computation time per frame with increasing sample count using the scenario shown in Fig. 9 with 30k fluid particles. Our dynamic KVN-importance sampling scheme shows a more stable vorticity enhancing performance with lower sample count sensitivity.



(a) Divergence-free SPH (N=126k) (b) Vorticity-based (N=14k)

(c) Vorticity-based (N=126k) (d) KVN-based (Ours, N=14k)

Fig. 11. Lid driven cavity visualized by streamlines. (a) Divergence-free SPH yields one central vortex using 126 k particles. (b) The vorticity-based sampling scheme using 14 k particles produces similar results, albeit with an reinforced vortex. (c) Increasing the particle count to 126 k enables vorticity-based sampling to create additional vortices. (d) Our approach produces similar results to (c) with only 14 k particles.



(a) Streamline (b) Abs. vorticity (c) KVN

(d) Raw fluid particles with samples

Fig. 12. Comparison of vorticity $\boldsymbol{\omega}$ and the KVN. (a) Stream. (b) Vorticity magnitude. Note that vortex cores do not always have high vorticities, and high vorticities can exist outside vortex cores. (c) The KVN effectively highlights vortex cores. (d) Raw fluid particles. The adaptive kernel density $\hat{f}$ is color-coded on particle samples, while non-sample fluid particles keep the default color. Our importance sampling scheme draws Lagrangian particle samples from vortex cores with high probabilities.

while ensuring that there is no energy increase. Image (b) shows that our approach better preserves energy compared to DF, VC, and MF. Interestingly, energy in our approach declines relatively quickly for about $4\,\mathrm{s}$, after which the decay rate slows. We analyze the reason for this using the sample count graph in image (c). The initial profile of the simulation before $4\,\mathrm{s}$ consists of four vortices. This represents a vortex merging process, leading to rapid changes in sample count and relatively weak vortex enforcement. After the vortices merge into a stable structure at about $4\,\mathrm{s}$, samples successfully converge to the stable and important areas, resulting in a relatively lower energy decay rate.

Stanford bunny with high-velocity streams. We compare our sampling scheme with the computation of the classic Biot-Savart law by naively querying all fluid particles (Eq. (14)). Figure 14 shows the experiment designed for this which involves a Stanford bunny hit by high-velocity streams. Particle color encodes speed (white is low, blue is high speed). Flows are initialized with a $2\,\mathrm{m/s}$ velocity on a plane tilted at an angle of 5 degrees. In image (a) and also in Figure 1 (left), computed with standard incompressible SPH, we find barely visible vortical flows behind the bunny due to the method's dissipative character. (b) The classic Biot-Savart formulation that uses all particles recovers the vorticity loss into the velocity field, so vortical flows are preserved well. However, this has a very high computational overhead – see Tab. I. (c) Our approach effectively reduces this overhead to an acceptable level. While vortex patterns are not completely identical to (b), the large performance gain of our approach is worth this trade-off.

Rotating glass panel. Figure 15 compares the vorticity confinement method, micropolar fluids, and our approach in a 3D scenario where a square glass panel rotates in water with an angular velocity of $0.5\pi\,\mathrm{rad/s}$. Particle color encodes speed (white is high, blue is low speed). The vorticity confinement method shows little vortical flow; the micropolar fluid method generates fine ripple motion artifacts. Compared to these approaches, our method yields significant vortical flows.

Dam breaking. Figure 16 shows the simulation of a dam breaking scenario which releases three water blocks in an area containing multiple static obstacles. The top half of the figure shows particles color-coded by speed (top); the bottom half shows water color-coded on vorticity norm. Realistic vortices can be seen on the water surface around the obstacles.

Rotating vessel. Figure 17 shows a vessel rotating in water with an angular velocity of $4\,\mathrm{rad/s}$. Particles are color-coded on speed (white is high, blue is low speed). Our method creates visually authentic wake flows behind the vessel.

Periodic waves crashing against a tower. Figure 1 (right) and Figure 18 show a more dynamic scenario with waves periodically crashing against a static tower. The square glass

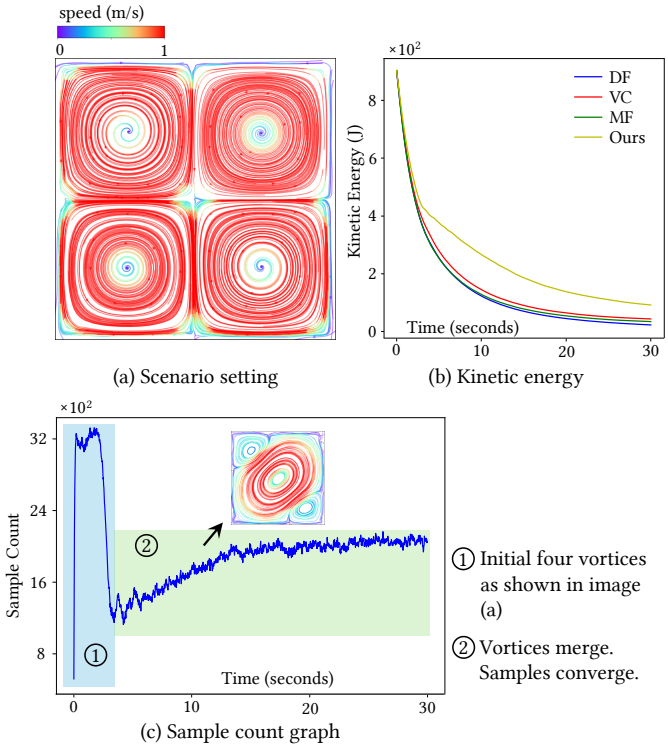


Fig. 13. 2D vortex interaction scenario: Comparison of kinetic energy among divergence-free SPH (DF), vorticity confinement (VC), micropolar fluids (MF), and our approach. (a) Initial scenario configuration. (b) Kinetic energy over time: our approach reduces kinetic energy decay better, with a notable slowdown around 4 s. (c) The sample count over time illustrates the convergence of samples after the vortex merging phase at about 4 s, corresponding to slower energy decay.

panel on the left pushes the water periodically to create waves. Results are rendered as particles with color-coded speed (top half of figure) and water color-coded by vorticity norm (bottom half of figure). We observe realistic vortical flows with many whirlpools to the right of the tower, which confirms the effectiveness of our method in enhancing vortical flows.

Performance. Table I presents computational time and other relevant metrics for all studied scenarios. Compared to the classic Biot-Savart law that uses all particles (experiment in Fig. 14), our approach has an average computation time of 4.79 s per frame, while the classic Biot-Savart law requires 158.44 s. This shows that our dynamic importance Monte Carlo approach significantly enhances efficiency – achieving roughly a 33-fold speedup. Compared to divergence-free SPH, micropolar fluids, and vorticity confinement, our approach is slightly less efficient. Indeed, our approach queries for particle samples (Sec. IV-C) whose total count $|\mathcal{S}|$ can be larger than the total number of neighbor particles. Since most SPH-based methods only query neighbor particles, our cost can be higher. In return, our method offers a substantial improvement in vortical flow preservation while maintaining a reasonable computational overhead. This balance between performance and visual quality positions our approach as a viable option for applications where high-fidelity vortical flows are crucial.

VI. Conclusion and Future Work

We have presented a dynamic importance Monte Carlo method to simulate SPH-based vortical flows by solving the Velocity-Vorticity Poisson Equation (VVPE). Unlike most point-based Monte Carlo PDE solvers, our method is Lagrangian with particle samples closely following the advection of areas which are important for vortex dynamics. We use the dimensionless Kinematic Vorticity Number (KVN) to measure the pure rotation and the importance of each fluid particle in vortex cores. To represent the KVN-based importance, we applied Adaptive Kernel Density Estimation (AKDE) to extract a smooth and dynamically updated probability density distribution for use in the Monte Carlo estimator. We use KVN-mapped acceptance and rejection probabilities to regulate the sample count within the desired scale while maintaining a particle sample distribution similar to that estimated by AKDE.

Our experiments showed that our approach produces convincing vortical flows in SPH fluids with acceptable computational overhead. Compared to the classic Biot-Savart law which naively queries all fluid particles, our method is over 30 times faster while yielding similar results. Our KVN-based importance also showed to be better in tracking vortex cores than vorticity alone. Moreover, compared with the uniform sampling scheme, our importance sampling is more stable and effective in enforcing vortical flows. Separately, we showed that our method produces more vortical flow when compared to techniques such as divergence-free SPH, uniform Monte Carlo, vorticity sampling, vorticity confinement, and micropolar fluids, while having only a small extra computational cost.

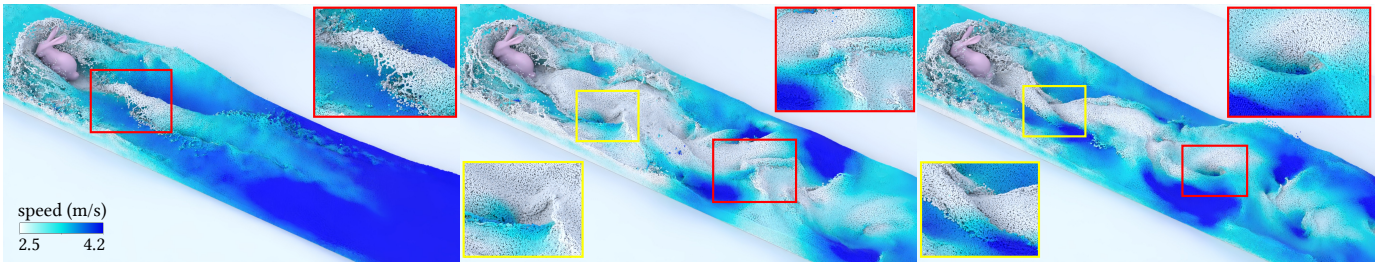
Despite the effectiveness of our scheme in preserving vortices and reducing the computational overhead of the Biot-Savart law, some limitations exist. Apart from the higher cost as compared to other SPH-based turbulent or vortical detail-enhancing methods, while vortices represented by Lagrangian particles perfectly adapt to the SPH framework, we lack a special mechanism to preserve filament-shaped vortex cores in 3D, potentially leading to a loss of vortical flows. In our future work, we aim to extend our approach to preserve such vortex filaments with further improved computational efficiency.

Acknowledgments

We would like to thank the *Taichi* graphics community for the kind support in coding and high-performance computation. We also acknowledge the use of Houdini software courtesy of SideFX. This research was supported by National Key Research and Development Program of China (No. 2022ZD0118001), National Natural Science Foundation of China (Nos. 62376025, 62332017, U22A2022), Guangdong Basic and Applied Basic Research Foundation (No. 2023A1515030177), China Scholarship Council (202206320039), and was partially funded by Horizon 2020-Marie Skłodowska-Curie Action-Individual Fellowships (No. 895941).

References

[1] X. Wang, Y. Xu, S. Liu, B. Ren, J. Kosinka, A. C. Telea, J. Wang, C. Song, J. Chang, C. Li, J. J. Zhang, and X. Ban, "Physics-based fluid simulation in computer graphics: Survey, research trends, and challenges," *Comput. Vis. Media*, pp. 1–56, Apr. 2024.



(a) Standard incompressible SPH (b) Classic Biot-Savart (c) Ours

Fig. 14. Stanford bunny hit by high-velocity streams. Particles are color-coded by speed. (a) Standard incompressible SPH (divergence-free SPH) produces barely visible vortices. (b) Classic Biot-Savart that naively queries all fluid particles yields prominent vortices behind the bunny. (c) Our approach also yields strong vortical flows, yet requires far less computational overhead compared to the classic scheme, as shown in Tab. I.

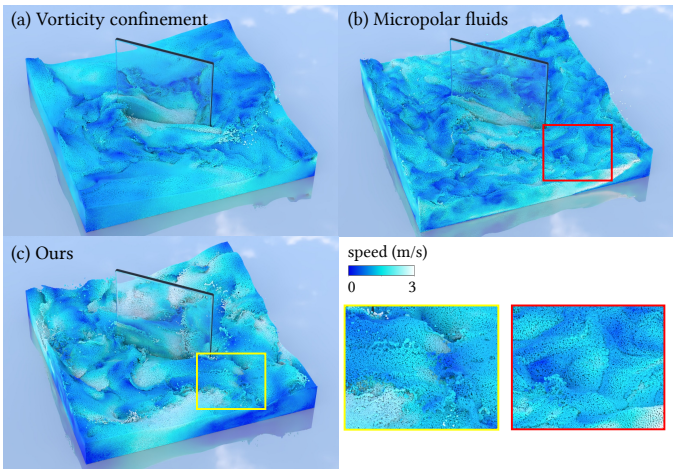


Fig. 15. Rotating glass panel with an angular velocity of 0.5π rad/s. Particles are colored by speed (dark blue: low, white: high). We compare the vortical effect of several methods. (a) The vorticity confinement method generates only little vortical flows. (b) The micropolar fluids yields obvious ripple artifacts. (c) Our approach produces strong vortical flows.

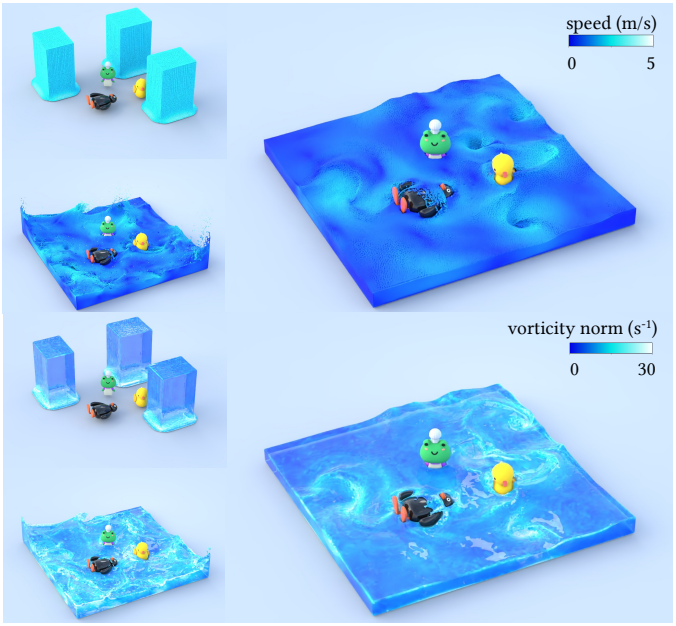


Fig. 16. Dam breaking scenario that releases three water blocks among multiple static solid objects. Top half: Particles color coded by speed. Bottom half: Water color coded by vorticity norm (bottom). Our approach is able to produce significant vortices.

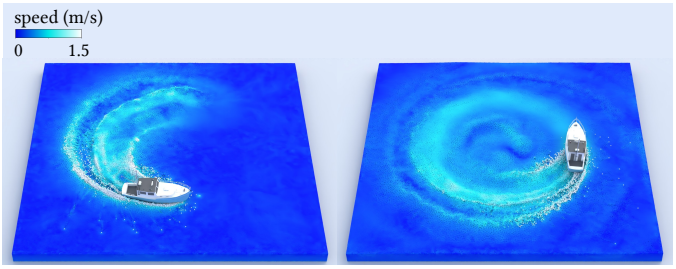


Fig. 17. Vessel rotating on water with an angular velocity of $4\,\mathrm{rad/s}$. Particles are color-coded by speed (blue is low, white is high). Realistic wake flows are produced behind the vessel.

[2] 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, jul 2015.
[3] 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.
[4] R. Fedkiw, J. Stam, and H. W. Jensen, "Visual simulation of smoke," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 15–22.
[5] J. Bender, D. Koschier, T. Kugelstadt, and M. Weiler, "Turbulent micropolar SPH fluids with foam," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 6, pp. 2284–2295, 2019.
[6] J. Chen, C. C. Liang, and J. D. Lee, "Theory and simulation of micropolar fluid dynamics," *Proc. Inst. Mech. Eng. N*, vol. 224, no. 1-2, pp. 31–39, 2010.
[7] S. I. Park and M. J. Kim, "Vortex fluid for gaseous phenomena," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 261–270.
[8] A. Selle, N. Rasmussen, and R. Fedkiw, "A vortex particle method for smoke, water and explosions," in *ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 910–914.
[9] S. Xiong, R. Tao, Y. Zhang, F. Feng, and B. Zhu, "Incompressible flow simulation on vortex segment clouds," *ACM Trans. Graph.*, vol. 40, no. 4, jul 2021.
[10] S. Weißmann and U. Pinkall, "Filament-based smoke with vortex shedding and variational reconnection," in *ACM SIGGRAPH 2010 Papers*, ser. SIGGRAPH '10. New York, NY, USA: Association for Computing Machinery, 2010.
[11] R. Sawhney and K. Crane, "Monte carlo geometry processing: a grid-free approach to pde-based methods on volumetric domains," *ACM Trans. Graph.*, vol. 39, no. 4, aug 2020.
[12] D. Rioux-Lavoie, R. Sugimoto, T. Özdemir, N. H. Shimada, C. Batty, D. Nowrouzezahrai, and T. Hachisuka, "A monte carlo method for fluid simulation," *ACM Trans. Graph.*, vol. 41, no. 6, nov 2022.
[13] R. Sugimoto, T. Chen, Y. Jiang, C. Batty, and T. Hachisuka, "A practical walk-on-boundary method for boundary value problems," *ACM Trans. Graph.*, vol. 42, no. 4, jul 2023.

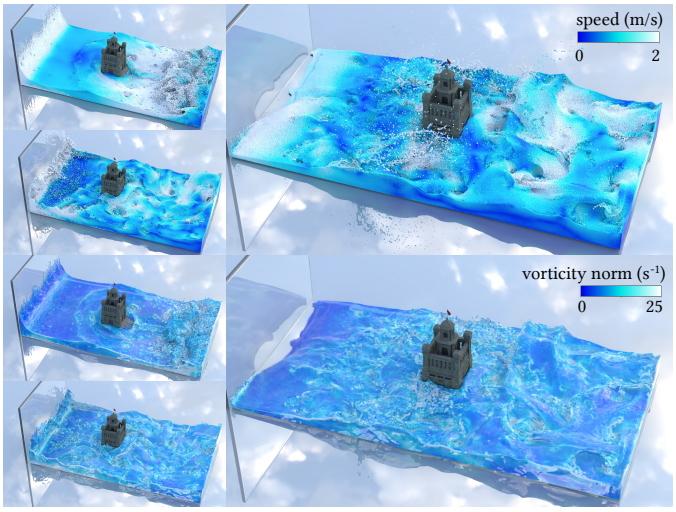


Fig. 18. Periodic waves crash against a static tower rendered as (top) particles with color-coded speed and (bottom) water with color-coded vorticity norm. The square glass panel at the left side pushes water periodically to produce waves. Many realistic whirlpools appear to the right of the tower.

[14] R. Sawhney, B. Miller, I. Gkioulekas, and K. Crane, "Walk on stars: A grid-free monte carlo method for pdes with neumann boundary conditions," *ACM Trans. Graph.*, vol. 42, no. 4, jul 2023.

[15] B. Miller, R. Sawhney, K. Crane, and I. Gkioulekas, "Walkin' robin: Walk on stars with robin boundary conditions," *ACM Trans. Graph.*, vol. 43, no. 4, Jul. 2024.

[16] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, and M. Teschner, "Versatile rigid-fluid coupling for incompressible SPH," *ACM Trans. Graph.*, vol. 31, no. 4, jul 2012.

[17] R. Sugimoto, C. Batty, and T. Hachisuka, "Velocity-based monte carlo fluids," in *ACM SIGGRAPH 2024 Conference Papers*, ser. SIGGRAPH '24. New York, NY, USA: Association for Computing Machinery, 2024.

[18] L. Schielicke, P. Névir, and U. Ulbrich, "Kinematic vorticity number – a tool for estimating vortex sizes and circulations," *Tellus A*, vol. 68, no. 1, p. 29464, 2016.

[19] P. V. Kerm, "Adaptive kernel density estimation," *Stata J.*, vol. 3, no. 2, pp. 148–156, 2003.

[20] R. Bridson, *Fluid Simulation for Computer Graphics*, 2nd ed. Natick: CRC Press, 2015.

[21] D. Koschier, J. Bender, B. Solenthaler, and M. Teschner, "Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids," in *Eurographics 2019 - Tutorials*, W. Jakob and E. Puppo, Eds. The Eurographics Association, 2019.

[22] M. Macklin and M. Müller, "Position based fluids," *ACM Trans. Graph.*, vol. 32, no. 4, jul 2013.

[23] M. Lentine, M. Aanjaneya, and R. Fedkiw, "Mass and momentum conservation for fluid simulation," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 91–100.

[24] P. Pasha, S. Mirzaei, and M. Zarinfar, "Application of numerical methods in micropolar fluid flow and heat transfer in permeable plates," *Alex. Eng. J.*, vol. 61, no. 4, pp. 2663–2672, 2022.

[25] E. Karvelas, G. Sofiadis, T. Papathanasiou, and I. Sarris, "Effect of micropolar fluid properties on the blood flow in a human carotid model," *Fluids*, vol. 5, no. 3, 2020.

[26] N. Abbas, S. Nadeem, and M. N. Khan, "Numerical analysis of unsteady magnetized micropolar fluid flow over a curved surface," *J. Therm. Anal. Calorim.*, vol. 147, no. 11, pp. 6449–6459, 2022.

[27] X. Ye, X. Wang, Y. Xu, J. Kosinka, A. C. Telea, L. You, J. J. Zhang, and J. Chang, "Monte carlo vortical smoothed particle hydrodynamics for simulating turbulent flows," *Comput. Graph. Forum*, vol. 43, no. 2, p. e15024, 2024.

[28] G. S. Winckelmans and A. Leonard, "Contributions to vortex particle methods for the computation of three-dimensional incompressible unsteady flows," *J. Comput. Phys.*, vol. 109, no. 2, p. 247–273, dec 1993.

[29] A. Angelidis and F. Neyret, "Simulation of smoke based on vortex filament primitives," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 87–96.

[30] A. Barnat and N. S. Pollard, "Smoke sheets for graph-structured vortex filaments," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '12. Goslar, DEU: Eurographics Association, 2012, p. 77–86.

[31] X. Zhang and R. Bridson, "A pppm fast summation method for fluids and beyond," *ACM Trans. Graph.*, vol. 33, no. 6, nov 2014.

[32] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *J. Comput. Phys.*, vol. 73, no. 2, pp. 325–348, 1987.

[33] A. Angelidis, "Multi-scale vorticle fluids," *ACM Trans. Graph.*, vol. 36, no. 4, jul 2017.

[34] R. L. Cook, "Stochastic sampling in computer graphics," *ACM Trans. Graph.*, vol. 5, no. 1, p. 51–72, jan 1986.

[35] J. T. Kajiya, "The rendering equation," *SIGGRAPH Comput. Graph.*, vol. 20, no. 4, p. 143–150, aug 1986.

[36] R. Sawhney, D. Seyb, W. Jarosz, and K. Crane, "Grid-free monte carlo for pdes with spatially varying coefficients," *ACM Trans. Graph.*, vol. 41, no. 4, jul 2022.

[37] Z. Yu, L. Wu, Z. Zhou, and S. Zhao, "A differential monte carlo solver for the poisson equation," in *ACM SIGGRAPH 2024 Conference Papers*, ser. SIGGRAPH '24. New York, NY, USA: Association for Computing Machinery, 2024.

[38] C. Truesdell, "Two measures of vorticity," *J. Ration. Mech. Anal.*, vol. 2, pp. 173–217, 1953. [Online]. Available: http://www.jstor.org/stable/24900328

[39] W. D. Means, H. B. Edward, G. S. Lister, and P. F. Williams, "Vorticity and non-coaxiality in progressive deformations," *J. Struct. Geol.*, vol. 2, no. 3, pp. 371–378, 1980.

[40] R. Graziani, K. P. Larson, R. D. Law, M.-A. Vanier, and J. R. Thigpen, "A refined approach for quantitative kinematic vorticity number estimation using microstructures," *J. Struct. Geol.*, vol. 153, p. 104459, 2021.

[41] S. Węglarczyk, "Kernel density estimation and its application," *ITM Web Conf.*, vol. 23, p. 00037, 2018.

[42] S. J. Sheather and C. M. Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *J. R. Stat. Soc. B*, vol. 53, no. 3, pp. 683–690, 1991.

[43] Z. I. Botev, J. F. Grotowski, and D. P. Kroese, "Kernel density estimation via diffusion," *Ann. Stat.*, vol. 38, no. 5, pp. 2916 – 2957, 2010.

[44] G. R. Terrell and D. W. Scott, "Variable kernel density estimation," *Ann. Stat.*, vol. 20, no. 3, pp. 1236–1265, 1992.

[45] C. M. Jones, "Simple boundary correction for kernel density estimation," *Stat. Comput.*, vol. 3, pp. 135–146, 1993.

[46] X. Zhang, R. Bridson, and C. Greif, "Restoring the missing vorticity in advection-projection fluid solvers," *ACM Trans. Graph.*, vol. 34, no. 4, jul 2015.

[47] J. J. Monaghan and J. C. Lattanzio, "A refined particle method for astrophysical problems," *A&A*, vol. 149, no. 1, pp. 135–143, Aug. 1985.

[48] D. J. Price, "Smoothed particle hydrodynamics and magnetohydrodynamics," *J. Comput. Phys.*, vol. 231, no. 3, pp. 759–794, 2012, special Issue: Computational Plasma Physics.

[49] J. J. Monaghan, "Smoothed particle hydrodynamics," *Rep. Prog. Phys.*, vol. 68, no. 8, p. 1703, jul 2005.

[50] A. J. Chorin and P. S. Bernard, "Discretization of a vortex sheet, with an example of roll-up," *J. Comput. Phys.*, vol. 13, no. 3, pp. 423–429, 1973.

[51] A. Eckert-Gallup and N. Martin, "Kernel density estimation (kde) with adaptive bandwidth selection for environmental contours of extreme sea states," in *OCEANS 2016 MTS/IEEE Monterey*, 2016, pp. 1–5.

[52] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[53] Y. Hu, T.-M. Li, L. Anderson, J. Ragan-Kelley, and F. Durand, "Taichi: a language for high-performance computation on spatially sparse data structures," *ACM Trans. Graph.*, vol. 38, no. 6, nov 2019.

[54] J. Bender and D. Koschier, "Divergence-free SPH for incompressible and viscous fluids," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 3, pp. 1193–1206, 2017.

Xingyu Ye is a PhD candidate at National Centre for Computer Animation, Bournemouth University, United Kingdom. He received his MEng degree in vehicle engineering from Zhejiang University in 2022, and BEng degree in vehicle engineering from Chongqing University in 2019. His research interests include physics-based fluid simulation and neural fluid simulation.

Xiaokun Wang is an associate professor in Intelligence Science and Technology, University of Science and Technology Beijing, China. He received a PhD degree in Computer Science and Technology from the University of Science and Technology Beijing in 2017. From 2021 to 2023, he worked at the National Centre for Computer Animation at Bournemouth University supported by the EU Horizon 2020 Marie Curie Individual Fellowship. His research interests include computer graphics, virtual reality and human-computer interaction.

Yanrui Xu is a postdoctoral researcher at Tsinghua University. He received his PhD in Computer Science and Technology from the University of Science and Technology Beijing in 2025. His research interests focus on physics-based fluid simulation.

Alexandru C. Telea received his PhD (2000) in Computer Science from the Eindhoven University of Technology. He was assistant professor in visualization and computer graphics at the same university (until 2007) and then full professor of visualization at the University of Groningen. Since 2019 he is full professor of visual data analytics at Utrecht University. His interests include highdimensional visualization, visual analytics, and image-based information visualization.

Jiří Kosinka received the PhD and MSc degrees in mathematics and physics from Charles University in Prague, Czech Republic, in 2006 and 2002, respectively. He is currently an Associate (Adjunct) Professor with the Bernoulli Institute, University of Groningen, the Netherlands. His research interests include geometric modeling, computer-aided design, computer graphics, and image processing.

Lihua You is a professor at the National Center for Computer Animation, Bournemouth University, UK. He received his BSc degree from Yanshan University, MSc degree and PhD degree from Chongqing University, China, and another PhD degree from Bournemouth University, UK. Before joining Bournemouth University, UK, he was a lecturer, an associate professor, and a professor of Chongqing University, China. He was an academic visiting scholar at Faculty of Engineering, Imperial College London, UK from 1995 to 1996.

Jian jun Zhang is professor of Computer Animation at the National Centre for Computer Animation, Bournemouth University where he leads the National Research Centre. His research focuses on a number of topics relating to 3D computer animation, virtual human modelling and simulation, and physics-based animation. He is also interested in virtual reality, medical visualisation and simulation, and conversational AI.

Jian Chang is a professor in National Centre Computer Animation at Bournemouth University, United Kindom. He received his PhD degree in Computer Graphics from Bournemouth University, in 2007. His research interests include physics based modelling (deformation & fluid), motion synthesis, virtual reality (surgery simulation), and novel HCI (eye tracking, gesture control and haptic).