The Power Particle-In-Cell Method

ZIYIN QU, University of Pennsylvania and University of California Los Angeles, USA
MINCHEN LI, University of California Los Angeles, USA
FERNANDO DE GOES, Pixar Animation Studios, USA
CHENFANFU JIANG, University of California Los Angeles, USA

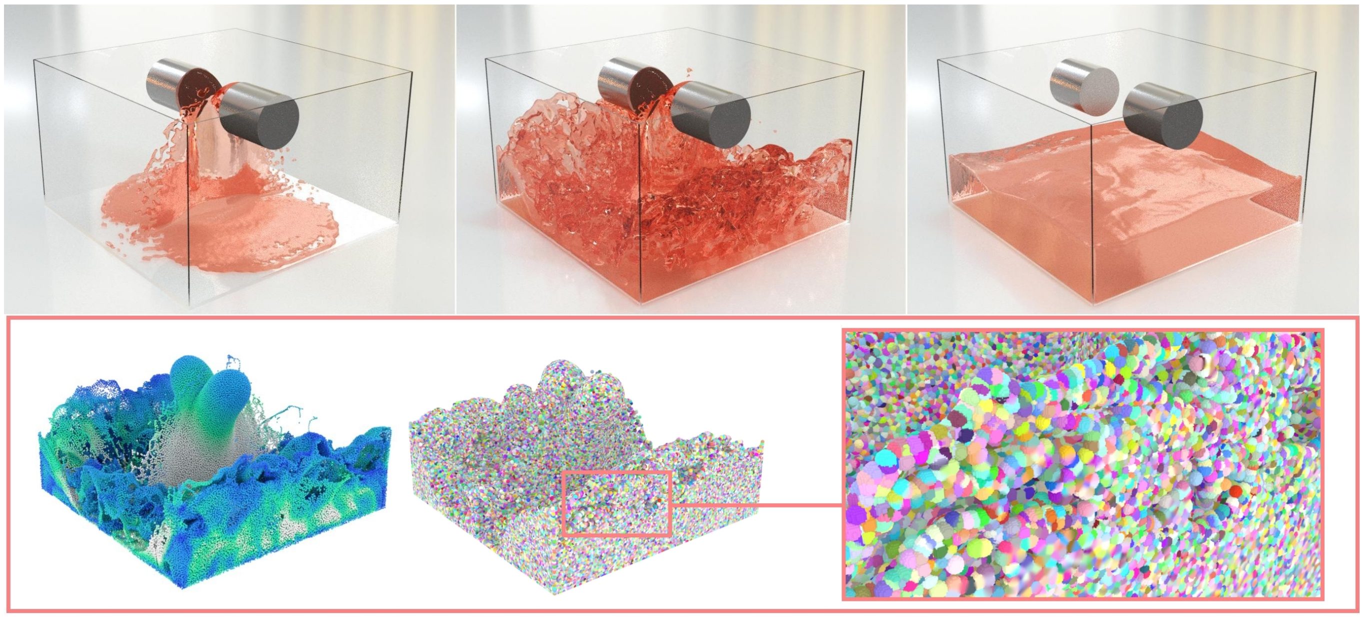
Fig. 1. We introduce the Power Particle-In-Cell method as a new weighting scheme that improves hybrid fluid simulations with evenly spaced particles and volume preservation. The top row shows frames of two liquid jets colliding generated by Power FLIP, our extension of the Fluid Implicit Particle (FLIP) method. In this example, we also render the liquid surfaces using the isocontour constructed directly from our fluid occupancy map. The bottom row displays the particle view of our simulation associated with the top-middle frame using pseudo-colors to indicate the particle velocity magnitude, followed by the visualization of our novel particle-based density kernels that define a regularized representation of Power Particles.

This paper introduces a new weighting scheme for particle-grid transfers that generates hybrid Lagrangian/Eulerian fluid simulations with uniform particle distributions and precise volume control. At its core, our approach reformulates the construction of Power Particles [de Goes et al. 2015] by computing volume-constrained density kernels. We employ these optimized kernels as particle domains within the Generalized Interpolation Material Point method (GIMP) in order to incorporate Power Particles into the Particle-In-Cell framework, hence the name the *Power Particle-In-Cell* method. We address the construction of volume-constrained density kernels as a regularized optimal transportation problem and describe an iterative solver based on localized Gaussian convolutions that leads to a significant performance speedup compared to [de Goes et al. 2015]. We also present novel extensions for handling free surfaces and solid obstacles that bypass the need for cell clipping and ghost particles. We demonstrate the advantages of our transfer weights by improving hybrid schemes for fluid simulation such as the Fluid Implicit Particle (FLIP) method and the Affine Particle-In-Cell (APIC) method with volume preservation and robustness to varying particle-per-cell ratio, while retaining low numerical dissipation, conserving linear and angular momenta, and avoiding particle reseeding or post-process relaxations.

CCS Concepts: • **Computing methodologies → Physical simulation**.

Additional Key Words and Phrases: Incompressible fluid, hybrid fluid solvers, Particle-In-Cell methods, optimal transport.

ACM Reference Format:
Ziyin Qu, Minchen Li, Fernando de Goes, and Chenfanfu Jiang. 2022. The Power Particle-In-Cell Method. *ACM Trans. Graph.* 41, 4, Article 118 (July 2022), 13 pages. https://doi.org/10.1145/3528223.3530066

Authors' addresses: Ziyin Qu, ziyinq@seas.upenn.edu, University of Pennsylvania, University of California Los Angeles, USA; Minchen Li, minchen@math.ucla.edu, University of California Los Angeles, USA; Fernando de Goes, fernando@pixar.com, Pixar Animation Studios, USA; Chenfanfu Jiang, cffjiang@math.ucla.edu, University of California Los Angeles, USA.

1 INTRODUCTION

Numerical methods for fluid simulation continue to be extensively investigated both in computer graphics and scientific computing in order to reproduce the visually rich and intricate dynamics of liquids and smoke. Fluid simulation techniques are generally categorized into Eulerian grid-based and Lagrangian particle-based frameworks [Bridson 2015]. Eulerian approaches are advantageous in discretizing forces but present challenges to compute fluid advection due to numerical dissipation. Conversely, Lagrangian schemes like Smooth Particle Hydrodynamics (SPH) are well-suited for particle advection, but their force computations are often inaccurate aggregating particle attributes through predefined kernels.

To mitigate these opposing issues, hybrid Lagrangian/Eulerian approaches for fluid simulation have received much attention, especially in the visual effects industry, by combining the advection of Lagrangian particles with the force computation produced on Eulerian grids. In these hybrid schemes, transferring physical attributes between particle and grid representations becomes indispensable. The original Particle-In-Cell (PIC) method [Harlow 1962] suffers from severe numerical damping caused by excessive averaging of quantities between the particles and the grid. Later techniques such as the Fluid Implicit Particle (FLIP) method reduce the artificial dissipation by only interpolating the velocity increment from the grid to the particles [Brackbill and Ruppel 1986; Zhu and Bridson 2005]. More recent approaches have further improved energy and momentum preservation by transferring higher-order velocity components [Fu et al. 2017; Jiang et al. 2015]. In addition to fluid dynamics, particle-grid transfers also play a central role in the Material Point Method (MPM) [Jiang et al. 2016; Sulsky et al. 1995] used to simulate elastoplastic [Stomakhin et al. 2014] and viscoplastic phenomena [Yue et al. 2015] as well as granular [Klár et al. 2016; Stomakhin et al. 2013] and codimensional materials [Jiang et al. 2017a].

Despite the popularity of hybrid methods and transfer schemes, controlling the particle distribution and enforcing incompressible flows remain challenging. The archetypal advection-projection splitting between Lagrangian particles and Eulerian grids accumulates noticeable volume change over time that yields particle clumping and artificial voids, thus degrading the simulation accuracy and stability. To alleviate these visual artifacts, existing implementations resort to post-process corrective steps that either push particles apart using spring-like forces [Ando et al. 2012] or adjust the volume error through additional pressure solves [Kugelstadt et al. 2019].

In this work, we derive new weights for particle-grid transfers that improve hybrid fluid simulations with evenly spaced particles and volume preservation. Instead of correcting particles as an afterthought, we propose to compute transfer weights that account for the particle occupancy relative to the grid discretization. With these optimized weights, we are able to enforce incompressibility geometrically and resolve particle distributions together with the advection-projection steps. To this end, we consider Lagrangian particles as fluid parcels with prescribed volumes, whereas the continuum space is discretized as an Eulerian grid with each cell assigned to a maximum volume capacity. We then address the particle-grid transfer as a transportation problem that reallocates volume between the particle and grid representations while conserving the amount of volume on both the particles and the grid cells.

Our approach draws inspiration from the Power Particles method introduced by de Goes et al. [2015], which describes particles as volume-constrained power diagrams that partition the space into well-shaped cells. The Power Particles representation provides the optimal transportation plan we seek for mapping volume from particles to the continuum [Aurenhammer et al. 1998]. Unfortunately, generating volume-constrained power diagrams is computationally expensive requiring repetitive cell clipping and scaling poorly with respect to the simulation resolution. In order to avoid the explicit construction of power diagrams, we adopt a regularization strategy for optimal transportation [Peyré et al. 2019] and then reformulate Power Particles as volume-constrained density kernels that encode the characteristic function for the domain spanned by each particle. In particular, we present a novel extension of this regularized transportation problem that handles free surfaces and solid obstacles free of any meshing or ghost particle seeding. As a result, we obtain a softened version of Power Particles that is more scalable and efficient to generate compared to [de Goes et al. 2015].

Equipped with these volume-constrained kernels, we introduce the *Power Particle-In-Cell* method as a modification of the Particle-In-Cell framework that incorporates the Power Particles properties. We augment particle-grid transfer schemes by computing weights as the correlation between the particle-based density kernels and the grid-based interpolation functions based on the Generalized Interpolation Material Point method [Bardenhagen and Kober 2004; Gao et al. 2017]. The resulting weights can then be retrofitted into existing hybrid fluid solvers (e.g., FLIP and APIC) seamlessly, producing faithful particle distributions with volume control and supporting varying particle-per-cell ratios. We showcase the benefits of our Power Particle-In-Cell method through various simulation scenarios including closed containers, free surfaces, and moving obstacles.

In summary, the technical contributions of our work are:

- A reformulation of Power Particles that replaces the costly construction of power diagrams with a regularized representation using volume-constrained density kernels (§4).
- Novel weights for particle-grid transfer that combine the Power Particles properties with the Particle-In-Cell framework, improving FLIP and APIC solvers with uniform particle distributions and volume preservation (§5).
- An extended transportation objective for optimizing particle kernels that deals with free surface and solid obstacles while avoiding cell clipping and ghost particles (§6).
- Acceleration strategies to compute our new transfer weights efficiently, scaling our method to large fluid simulations (§7).

2 RELATED WORK

Before detailing our contributions, we review prior work on computational methods for fluid simulation restricting our discussion to hybrid schemes and particle-based strategies. We also describe recent numerical tools for optimal transportation which serve as the bases for our formulation. We refer the reader to [Bridson 2015;

Hu et al. 2019] for an extended overview of fluid solvers used in computational physics and computer graphics.

Hybrid Lagrangian/Eulerian Methods. The family of Particle-In-Cell (PIC) methods [Foster and Metaxas 1996; Harlow 1962] hybridizes the discretization of continuum materials by combining Lagrangian particle and Eulerian grid representations. However, the traditional PIC-style transfer between these two discretizations is highly dissipative and unsuited for turbulent and long-run simulations. The Fluid Implicit Particle (FLIP) method [Brackbill and Ruppel 1986; Zhu and Bridson 2005] was developed to circumvent this dissipation, with various extensions proposed for animating fluids [Ando et al. 2013; Batty et al. 2007; Ferstl et al. 2016; Sato et al. 2018]. Since FLIP introduces noisy velocity fields, a common practice is to blend the FLIP transfer with some amount of the PIC transfer to retrieve numerical stability [Zhu and Bridson 2005]. The Affine Particle-In-Cell (APIC) method was later devised in [Jiang et al. 2015] to improve angular momentum preservation in a PIC-style scheme by augmenting particles with affine velocity terms. Subsequently, the Polynomial Particle-In-Cell method [Fu et al. 2017] utilizes higher-order velocity modes. The Moving Least Squares Material Point Method (MLS-MPM) [Hu et al. 2018] further formalizes these additional velocity components into a weighted least squares optimization. Alternatively, the Generalized Interpolation Material Point method (GIMP) [Bardenhagen and Kober 2004; Gao et al. 2017] presents a different perspective of defining particle-grid transfers by modeling the transfer weights as the convolution between an interpolation function and a particle domain characteristic function. More recently, the work of Nakanishi et al. [2020] employs radial basis functions to provide spatially adaptive weights, while the approach of Fei et al. [2021] reduces numerical dissipation by modifying particle updates. In common, existing transfer schemes are only accurate under the assumption of a nearly uniform particle distribution, making simulations with uneven particles suffer from clumps and voids. In sharp contrast, we introduce new transfer weights that provide precise control over the particle volumes and spacing, thus improving the simulation accuracy and stability.

SPH Methods. Smoothed Particle Hydrodynamics (SPH) is a prevalent approach for Lagrangian simulations of both compressible and incompressible fluids [Ihmsen et al. 2014; Koschier et al. 2020; Monaghan 2005]. The state equation is often applied to resolve weakly compressible SPH with small timesteps [Becker and Teschner 2007]. The PCISPH method [Solenthaler and Pajarola 2009] alleviates this timestep restriction by correcting the density constraint in a predictive-corrective fashion. Similarly, the method of Macklin and Müller [2013] approximates incompressibility constraints iteratively using the Position-based Dynamics framework. Other approaches [Bender and Koschier 2016; Ihmsen et al. 2013] have further improved the overall quality of SPH simulations by alternating pressure solves that impose density and divergence constraints. The work of Reinhardt et al. [2019] attempts to restore the particle density by resizing SPH kernels based on iterative Shepard corrections. Some SPH implementations have also been combined with grid-based fluid simulators [Cornelis et al. 2014; Losasso et al. 2008]. While existing SPH schemes rely heavily on predefined kernels, we propose to optimize density kernels per timestep in order to preserve the volume associated with every Lagrangian particle.

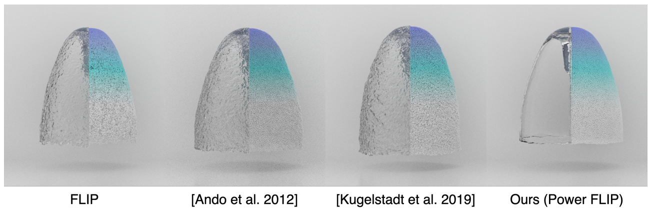

Fig. 2. These images snapshot the simulation of a fountain using four variants of a FLIP solver. The original FLIP method suffers from volume changes and uneven particle distributions leading to noisy surfaces (left). The method of Ando et al. [2012] relaxes particles more uniformly at the cost of artificial volume gain (left-middle). The work of Kugelstadt et al. [2019] preserves the liquid volume, but the reconstructed free surface is still bumpy (middle-right). In contrast, our Power FLIP scheme maintains well-distributed particles and produces smooth surfaces contoured directly from our fluid occupancy estimate (right). Particle velocities are colored using a blue to white ramp.

Position Correction Methods. Hybrid solvers exploit particles as indicators of the fluid occupancy over time. Therefore, accumulating uneven particle distributions leads to unbearable volume changes that cannot be corrected by the pressure forces. For instance, the work of [Ando et al. 2012] proposes weak spring forces to favor uniform particle distances. Similarly, the method of Takahashi and Lin [2019] accounts for changes in particle density using SPH kernels. Grid-based approaches such as [Um et al. 2014] calculate sub-grid corrective forces on particles. In [Kugelstadt et al. 2019], estimates of the fluid density on the Eulerian grid are used to compute extra pressure forces and then displace particles. Instead of post-processing particle volumes and/or positions, our approach modifies the transfer weights directly so that advected particles preserve volume within a uniform distribution.

Power Particles. Departing from previous point-based techniques, the Power Particles method [de Goes et al. 2015] represents Lagrangian particles as moving power diagrams that partition the simulation domain with precise volume control, thus producing fluid simulations with well-distributed particles, accurate pressure forces, and reduced energy dissipation. The work of Mérigot and Mirebeau [2016] showed that the Power Particles formulation is closely related to the discretization of volume-preserving maps defining the dynamics of incompressible and inviscid fluids. A GPU-based implementation was later developed by Zhai et al. [2018] for better computational performance. More recently, Lévy [2022] proposed a modification of Power Particles that handles free surfaces by clipping power diagrams against spheres instead of using ghost particles. Despite the high-quality particle distributions, generating Power Particles is costly due to the repetitive construction of power diagrams and scales poorly in 3D as the number of particles increases. To overcome these limitations, we present a regularized formulation of Power Particles that is significantly faster to compute and compatible with Particle-In-Cell methods. As a result, we obtain fluid simulations that retain the even particle distributions

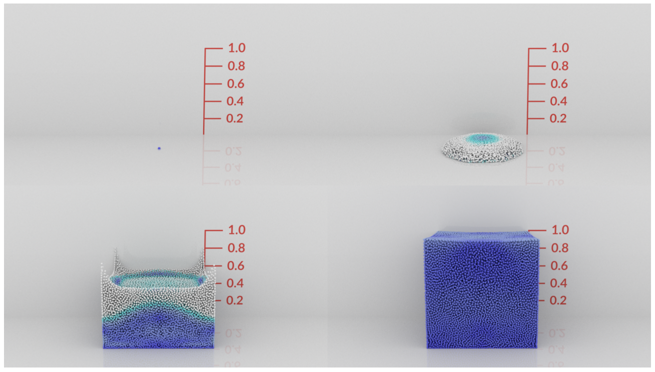

Fig. 3. This example demonstrates that our method is capable of restoring the fluid volume entirely even when particles are initially compressed into a single cell. Notice that our method also produces uniform particle distributions in addition to volume preservation.

and volume preservation of Power Particles, while leveraging the versatility and efficiency of hybrid Lagrangian/Eulerian methods.

Optimal Transportation. Our work builds upon recent advances in numerical methods for optimal transportation [Peyré et al. 2019]. In particular, our approach adopts the entropic regularization presented by Cuturi [2013] that accelerates the construction of optimal transportation plans using localized Gaussian convolutions. The regularization of the unbalanced transportation problem using non-normalized marginals was also described by Chizat et al. [2018]. In graphics, this regularized formulation has been employed to solve transportation optimizations over geometric domains [Solomon et al. 2015] and to generate surface correspondences [Mandad et al. 2017; Solomon et al. 2016]. We exploit these transportation tools to revisit the construction of Power Particles as softened power diagrams encoded by density kernels, similar to the relaxation of the semi-discrete optimal transportation problem discussed by Cuturi and Peyré [2018]. In addition, we propose an extended transportation optimization that incorporates free surfaces and solid obstacles into particle-based density kernels.

3 DEFINITIONS & NOTATIONS

In this work, we consider a simulation domain Ω with volume $|\Omega|$ and boundary $\partial\Omega$. Our formulation involves three distinct discretizations of Ω. We first define Lagrangian particles scattered over Ω represented by a list of n_p points. For the Eulerian computations, we discretize Ω using a uniform grid of n_i cells which we refer to as the simulation grid (a.k.a., *s-grid*). In addition, we introduce an auxiliary transportation grid (a.k.a., *t-grid*) formed by uniform n_j cells used to construct our density kernels through the regularized optimal transportation solve. Hereafter, we make use of subscripts to differentiate quantities assigned to these different discretizations. We reserve the subscripts i for s-grid cells, j for t-grid cells, and p for particles. For instance, we associate each particle p with the position $\mathbf{x}_p$ and a prescribed volume V_p, while we indicate the center of each simulation cell i by the point $\mathbf{x}_i$ and employ the scalar $V_i = |\Omega|/n_i$ to denote its volume capacity. Similarly, each transportation cell j is located at $\mathbf{x}_j$ with volume capacity $V_j = |\Omega|/n_j$. We indicate the velocity at each particle p and s-grid cell i by the vectors $\mathbf{u}_p$ and $\mathbf{u}_i$, respectively. We also denote $\psi^\varepsilon(x) = \exp(x/\varepsilon)$ as a shorthand for the exponential function with parameter $\varepsilon > 0$ and express the Gaussian kernel with squared standard deviation ε as $K_{pj} = \psi^\varepsilon(-\|\mathbf{x}_p - \mathbf{x}_j\|^2)$. Finally, we define the relative entropy function as $\mathcal{R}(x,y) = x\log(x/y) - x$ and the (negated) entropy function as $\mathcal{H}(x) = \mathcal{R}(x,1)$, where x and y are non-negative scalars [Cover and Thomas 2006].

4 REFORMULATING POWER PARTICLES

This section presents the concepts on which our formulation is based. We start by reviewing numerical methods for optimal transportation that generate a mapping between particles and grid cells with precise volume control. We also point the reader to [Peyré et al. 2019] for a detailed introduction to computational optimal transportation. Built upon these techniques, we derive volume-preserving kernel functions that define a partition of unity over the simulation domain. At last, we relate these particle-based density kernels to the construction of Power Particles [de Goes et al. 2015].

Optimal Transportation: In order to map particles to grid cells, we construct a transportation plan encoded by a matrix of size $n_p \times n_j$ that indicates the (non-negative) amount of volume T_{pj} carried from V_p at $\mathbf{x}_p$ and coalesced into V_j at $\mathbf{x}_j$. We start with the assumption that the fluid occupies the entire domain, i.e., $\sum_p V_p = \sum_j V_j$, and discuss the case with free surfaces and solid obstacles later in §6. We define the *optimal* transportation plan as the matrix that minimizes the distance traveled to move volume between particles and grid cells constrained by the volume capacities, which can be written as

$$\min_{\{T_{pj}\}} \sum_{p,j} T_{pj} \|\mathbf{x}_p - \mathbf{x}_j\|^2 \tag{1a}$$

$$\text{s.t.} \begin{cases} \sum_p T_{pj} = V_j & \forall j, \\ \sum_j T_{pj} = V_p & \forall p. \end{cases} \tag{1b}$$

Several linear program solvers and combinatorial schemes have been devised for this optimization problem [Burkard et al. 2009], including specialized methods targeting graphics applications [Balzer et al. 2009; Bonneel et al. 2011]. Unfortunately, these strategies are known to scale poorly relative to the domain size due to the coupled particle-cell unknowns, thus making the generation of transportation plans prohibitive for large particle count and high resolution grids.

Entropic Regularization: To overcome these costly computations, the work of Cuturi [2013] altered Eq. (1a) with a regularization term that favors spread out solutions by penalizing the entropy of the transportation values weighted by a regularization amount ε, i.e.:

$$\min_{\{T_{pj}\}} \sum_{p,j} T_{pj} \|\mathbf{x}_p - \mathbf{x}_j\|^2 + \varepsilon \sum_{p,j} \mathcal{H}(T_{pj}) \tag{2a}$$

$$\text{s.t.} \begin{cases} \sum_p T_{pj} = V_j & \forall j, \\ \sum_j T_{pj} = V_p & \forall p. \end{cases} \tag{2b}$$

Remarkably, this modification makes the objective function strictly convex and Eq. (2) has a unique minimizer concisely expressed as

$$\boxed{T_{pj} = \psi^\varepsilon(r_j + r_p - \|\mathbf{x}_p - \mathbf{x}_j\|^2) = s_j s_p K_{pj},} \tag{3}$$

Algorithm 1 Pseudocode for the Sinkhorn algorithm.

```
1:  Initialize $s_j, s_p = 1 \ \forall j, p$.
2:  Precompute Gaussian Kernels $\{K_{pj}\}$.
3:  while $\max_j |(\sum_p T_{pj})/V_j - 1| > \delta$ do
4:      for $j = 1 \ldots n_j$ do // in parallel
5:          $s_j = V_j/(\sum_p K_{pj} s_p)$
6:      end for
7:      for $p = 1 \ldots n_p$ do // in parallel
8:          $s_p = V_p/(\sum_j K_{pj} s_j)$
9:      end for
10: end while
```

where $\{r\}$ are the Lagrange multipliers associated with the volume constraints in Eq. (2b) and $s = \psi^\varepsilon(r)$ indicates the scaling amount corresponding to each multiplier r. Therefore, we can recast the transportation problem as an optimization that seeks a scaling per particle and grid cell instead of coupled variables, reducing the number of unknowns from $n_p n_j$ to $n_p + n_j$. Note that the resulting volume amounts $\{T_{pj}\}$ are made of scaled Gaussian values $\{K_{pj}\}$, which allow the evaluation of the volume constraints as truncated sums within a window of size proportional to ε.

Sinkhorn Iterations: We optimize these scaling unknowns using the iterative method of [Sinkhorn 1967] summarized in Algorithm 1 that alternates the update of each variable by enforcing its respective volume constraint. Observe that the particles have their volumes restored by the end of every iteration, while the volume constraints over the grid cells dictate the termination criterion up to a numerical tolerance δ set to 0.1 in our experiments. We also note that the denominators in both scaling updates (lines 5 and 8) correspond to Gaussian convolutions, which can be efficiently implemented as matrix-vector multiplications. In addition to scaling linearly in the number of particles and grid cells, this optimization is a first-order method and thus converges at a linear rate [Knight 2008].

The Power Kernel: Equipped with the optimal transportation plan, we can now construct a density kernel $\chi_p^\varepsilon : \Omega \longrightarrow [0, 1]$ that defines a weighting function for each particle p via

$$\chi_p^\varepsilon(\mathbf{x}) = \frac{\psi^\varepsilon(r_p - \|\mathbf{x}_p - \mathbf{x}\|^2)}{\sum_q \psi^\varepsilon(r_q - \|\mathbf{x}_q - \mathbf{x}\|^2)}. \tag{4}$$

Since this kernel resembles a normalized radial basis function but using power distances [Aurenhammer 1987], we name it the *power kernel.* Notice that the kernel evaluated at any grid cell j is equivalent to $\chi_p^\varepsilon(\mathbf{x}_j) = T_{pj}/\sum_q T_{qj} = T_{pj}/V_j$. These kernels also form a partition of unity, i.e., $\sum_p \chi_p^\varepsilon(\mathbf{x}) = 1$ for any $\mathbf{x} \in \Omega$. Importantly, the integration of each kernel reproduces the volume of its respective particle:

$$\int_\Omega \chi_p^\varepsilon(\mathbf{x}) \, d\mathbf{x} \approx \textstyle\sum_j \chi_p^\varepsilon(\mathbf{x}_j) V_j = \sum_j T_{pj} = V_p, \tag{5}$$

where we use the t-grid cells as quadrature points. Additionally, we can compute the centroid $\mathbf{c}_p$ associated with each density kernel as

$$\mathbf{c}_p = \frac{1}{V_p} \int_\Omega \chi_p^\varepsilon(\mathbf{x}) \, \mathbf{x} \, d\mathbf{x} \approx \frac{1}{V_p} \textstyle\sum_j T_{pj} \mathbf{x}_j. \tag{6}$$

It is worth noting that the centroid $\mathbf{c}_p$ always lies inside the span of its power kernel χ_p^ε, but the particle position $\mathbf{x}_p$ may fall outside.

Power Diagrams: As discussed by Cuturi and Peyré [2018], Eq. (4) corresponds to a *softmin* function between power distances that returns the probability of the query point $\mathbf{x}$ to be assigned to particle p. In particular, as the entropic regularization diminishes towards zero, the power kernel of particle p reduces to

$$\lim_{\varepsilon \to 0} \chi_p^\varepsilon(\mathbf{x}) = \begin{cases} 1, & \|\mathbf{x}_p - \mathbf{x}\|^2 - r_p \leq \|\mathbf{x}_q - \mathbf{x}\|^2 - r_q \ \forall q \neq p, \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

which is equivalent to the definition of power diagrams with weighted points $\{(\mathbf{x}_p, r_p)\}$ [Aurenhammer 1987]. Therefore, we can use power kernels as indicator functions representing softened power cells. Figure 4 illustrates the blurred diagrams generated by power kernels with different regularization amounts. Moreover, we can interpret Algorithm 1 as a regularized alternative to the semi-discrete transportation problem [Aurenhammer et al. 1998] that replaces cell clipping with localized convolutions.

Power Particles: Based on these observations, we can revisit the formulation of Power Particles [de Goes et al. 2015], which is a Lagrangian method for fluid simulation described by a time series of moving power diagrams. Using power kernels, we can approach Power Particles as a variant of the SPH method [Monaghan 2005] that optimizes the density kernels at every timestep in order to ensure volume preservation and evenly spaced particles. The discrete operators in [de Goes et al. 2015, Section 2.3] can also be reproduced by computing the spatial derivatives of Eq. (5), thus providing accurate pressure forces. In addition to improving the Lagrangian discretization, we can further exploit the construction of power kernels to adapt the Power Particles representation into Particle-In-Cell methods, as we discuss next.

5 POWER PARTICLE-IN-CELL

We now present our novel transfer weights that combine the density kernels encoding Power Particles into the Particle-In-Cell framework. Additionally, we describe how to integrate these new transfer weights into PIC/FLIP/APIC fluid solvers. Other hybrid schemes such as PolyPIC [Fu et al. 2017] and ASFLIP [Fei et al. 2021] can also be extended in a similar way. Algorithm 2 summarizes the pseudocode for our modified fluid simulator.

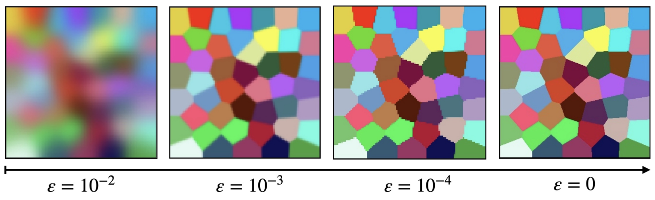

Fig. 4. Our new particle representation based on optimized density kernels converges to volume-constrained power diagrams as the regularization amount ε decreases. We visualize the density kernels over a grid by blending particle colors weighted by the transportation contribution between each particle and grid cell. Notice how the particle domains overlap and the colors get more blurry as ε increases.

Algorithm 2 Pseudocode for a single timestep of our method.

```
1: Compute transportation plan $\{T_{pj}\}$ via Algorithm 1.
2: Compute power weights $\{w_{pi}\}$ via Eq. (9).
3: Particle-to-grid mass transfer via Eq. (11).
4: Particle-to-grid velocity transfer:
   a: Power PIC via Eq. (12) or
   b: Power FLIP via Eq. (12) or
   c: Power APIC via Eq. (13).
5: Apply external forces onto grid velocity.
6: Compute grid-based pressure projection.
7: Grid-to-particle velocity transfer:
   a: Power PIC via Eq. (14) or
   b: Power FLIP via Eq. (16) or
   c: Power APIC via Eqs. (14) and (15).
8: Advect particles via Eq. (17).
```

Simulation vs. Transportation Grid: In our formulation, we consider the simulation grid (s-grid) used by the Eulerian computations to be decoupled from the transportation grid (t-grid). By adopting these two grid representations, our transfer weights can account for arbitrary layouts of the s-grid. For instance, we can utilize a s-grid resolution coarser than the t-grid, thus balancing computational cost and visual quality. We can also adapt the s-grid to various Eulerian discretizations such as collocated and staggered methods. Choosing a s-grid layout, we assign every cell i an interpolation basis function $N_i(\mathbf{x})$ so that, for every point $\mathbf{x} \in \Omega$, we have

$$\begin{cases} N_i(\mathbf{x}) \geq 0, \\ \sum_i N_i(\mathbf{x}) = 1, \\ \sum_i N_i(\mathbf{x})\mathbf{x}_i = \mathbf{x}. \end{cases} \tag{8}$$

In our experiments, we set these grid-based functions to piecewise linear interpolants. We also make use of $N_{ij} \equiv N_i(\mathbf{x}_j)$ as a shorthand to indicate the evaluation of N_i at each t-grid point $\mathbf{x}_j$.

The Power Weights: We construct our weighting scheme based on the GIMP method [Bardenhagen and Kober 2004; Gao et al. 2017], which defines weights by computing the correlation between particle-based characteristic functions and grid-based interpolants:

$$\boxed{w_{pi} = \frac{1}{V_p}\int_{\Omega} \chi_p^{\varepsilon}(\mathbf{x}) N_i(\mathbf{x})\, d\mathbf{x} \approx \frac{1}{V_p}\sum_j T_{pj} N_{ij}.} \tag{9}$$

Since the resulting weights are derived from power kernels, we refer to them as *power weights.* Note that we use t-grid cells as a cubature approximation of Eq. (9), which can be efficiently evaluated by a truncated sum that involves only a bounded window of cells around each s-grid element thanks to the localized support of both particle and grid kernels. These power weights also inherit the high-order continuity of the scaled Gaussian functions defining the power kernels. Importantly, we exploit the volume constraints imposed by the particle-based density kernels to further show that:

$$\begin{cases} w_{pi} \geq 0, \\ \sum_i w_{pi} = 1, \\ \sum_i w_{pi}\mathbf{x}_i = \mathbf{c}_p. \end{cases} \tag{10}$$

Therefore, the power weights form a partition of unity but are not interpolants, returning instead the centroid $\mathbf{c}_p$ of the domain covered by each particle kernel.

Mass Transfer: As typical in hybrid solvers, we associate every Lagrangian particle p with a time-independent mass attribute m_p. We define this particle mass as $m_p = \rho V_p$ that multiplies the prescribed particle volume V_p by the constant fluid density ρ. The amount of mass in each s-grid cell i is then expressed as:

$$m_i = \textstyle\sum_p w_{pi} m_p. \tag{11}$$

Particle-to-Grid Transfer: The power weights can be easily integrated into the transfer of velocity fields from particles to s-grid cells at any given simulation timestep. The simplistic transfer used in PIC and FLIP methods [Brackbill and Ruppel 1986; Harlow 1962; Zhu and Bridson 2005] is written in terms of power weights as

$$m_i \mathbf{u}_i = \textstyle\sum_p w_{pi} m_p \mathbf{u}_p. \tag{12}$$

We can also adapt the particle-to-grid transfer from the APIC method [Jiang et al. 2015] to use power weights, leading to

$$m_i \mathbf{u}_i = \textstyle\sum_p w_{pi} m_p \left(\mathbf{u}_p + \mathbf{A}_p(\mathbf{x}_i - \mathbf{c}_p)\right), \tag{13}$$

with the matrix $\mathbf{A}_p$ indicating the linear component of the particle velocity, which we update in the subsequent grid-to-particle transfer step. Note that Eq. (13) employs $\mathbf{c}_p$ instead of $\mathbf{x}_p$, which is necessary to ensure linear and angular momenta conservation for our Power APIC transfer. These properties can be verified following a derivation similar to [Jiang et al. 2017b] that makes use of Eq. (10).

Eulerian Update: Given the fluid mass and velocity transferred from particles to the s-grid, we can proceed with the Eulerian simulation by adding any external forces and computing the pressure projection that makes the grid velocity divergence-free. We indicate the velocity at each s-grid cell i updated by the Eulerian steps as $\widetilde{\mathbf{u}}_i$.

Grid-to-Particle Transfer: To map the updated grid velocity back to particles, we modify the PIC-style transfer with power weights:

$$\mathbf{u}_p = \textstyle\sum_i w_{pi} \widetilde{\mathbf{u}}_i. \tag{14}$$

In the case of the APIC method, we additionally update the particle matrix $\mathbf{A}_p = \mathbf{B}_p(\mathbf{D}_p)^{-1}$ with the matrices $\mathbf{B}_p$ and $\mathbf{D}_p$ defined by

$$\begin{cases} \mathbf{D}_p = \sum_i w_{pi}(\mathbf{x}_i - \mathbf{c}_p)(\mathbf{x}_i - \mathbf{c}_p)^t, \\ \mathbf{B}_p = \sum_i w_{pi}\widetilde{\mathbf{u}}_i(\mathbf{x}_i - \mathbf{c}_p)^t. \end{cases} \tag{15}$$

Once again, observe that our power version of the APIC method uses particle centroids $\mathbf{c}_p$ instead of particle positions $\mathbf{x}_p$. Alternatively, we can devise a FLIP-style transfer that only interpolates the grid velocity increments, leading to

$$\mathbf{u}_p^{k+1} = \mathbf{u}_p^k + \textstyle\sum_i w_{pi}^k \left(\widetilde{\mathbf{u}}_i^k - \mathbf{u}_i^k\right), \tag{16}$$

where we use superscripts to differentiate the simulation timesteps.

Particle Advection: Finally, we approach the particle advection by deforming the domain covered by each particle based on the updated grid velocity. The particle advection can thus be computed with our power weights through

$$\mathbf{x}_p = \textstyle\sum_i w_{pi}\left(\mathbf{x}_i + \Delta t\, \widetilde{\mathbf{u}}_i\right) = \mathbf{c}_p + \Delta t\, \mathbf{u}_p, \tag{17}$$

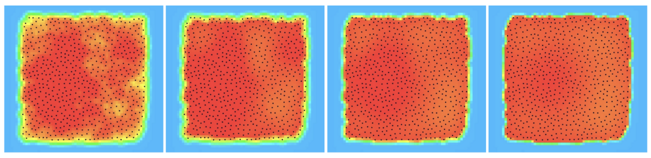

Fig. 5. We visualize the fluid occupancy generated by our transportation optimization with free surfaces for varying sharpness values η. The fluid occupancy is shown by a color ramp from blue to red. In the case of no baseline (left), we obtain non-uniform grid volumes inside the fluid with a smeared fluid-air interface. By increasing η, the transition near the free surface sharpens and the fluid interior becomes nearly constant.

where Δt indicates the time increment between consecutive timesteps. As a result, our particle advection favors the formation of evenly spaced distributions similar to [de Goes et al. 2015].

6 FREE SURFACES AND SOLID OBSTACLES

So far we have described how to construct the Power Particle-In-Cell method based on the assumption that the fluid occupies the entire domain. We present next extensions to our formulation that enable fluid simulations with free surfaces and solid obstacles.

Slack Air Variables: To represent free surfaces, we introduce an extra variable, denoted by a_j, that measures the volume of air contained in each cell j of the t-grid. These air volumes can be interpreted as slack variables necessary to complement the fluid volume gathered from the Lagrangian particles by the transportation plan $\{T_{pj}\}$ in order to fulfill the volume capacity within the grid cells. Using these air variables, we can revisit Eq. (2b) and expand the volume constraint of each grid cell as $V_j = \sum_p T_{pj} + a_j$. We can also verify that $\sum_p V_p = \sum_{p,j} T_{pj} = \sum_j (V_j - a_j) \leq \sum_j V_j = |\Omega|$, thus confirming that the fluid occupies no more than the domain volume.

Air Volume Baseline: Free surfaces are often estimated as the zero levelset of a signed distance function approximated based on the distribution of the Lagrangian particles [Bridson 2015]. The usual convention is to define the signed distance function $\phi : \Omega \rightarrow \mathbb{R}$ so that $\phi(\mathbf{x}) < 0$ for every point $\mathbf{x}$ inside the fluid. Although particle-based estimates tend to be loose near the liquid-air interface, they can reliably detect if points far away from the free surface are inside or outside the fluid. Based on this observation, we propose to convert any given signed distance approximation to an air volume baseline that guides the formation of the free surface in our transportation optimization. We start by generating a signed distance function ϕ by splatting the particles as spheres using [Museth et al. 2013]. We then construct a narrow band around the zero levelset in ϕ of thickness τ and compute an indicator function $\zeta : \Omega \rightarrow [0, 1]$ of the form

$$\zeta(\mathbf{x}) = \begin{cases} 1 & \text{if } \phi(\mathbf{x}) \geq \tau, \\ 0 & \text{if } \phi(\mathbf{x}) \leq -\tau, \\ \left(\dfrac{\phi(\mathbf{x}) + \tau}{2\tau}\right)^{\eta} & \text{otherwise,} \end{cases} \tag{18}$$

where η is a user parameter that controls the sharpness of the ramp smearing across the narrow band. Note that points outside the narrow band but inside the fluid are clamped to zero indicating the absence of air, while points outside the fluid away from the narrow band are set to one suggesting the presence of air. In our experiments, we set η to one and τ to the radius used for building the signed distance function ϕ. With this indicator function, we assign the air volume baseline for each cell j in the t-grid to $z_j = \zeta(\mathbf{x}_j) V_j$.

Transportation Plan: Our next step is to augment the transportation optimization in order to account for air volumes while shaping the free surface similar to the signed distance approximation. To this end, we include in our objective function an extra regularization term that compares the slack air variables with the air volume baselines by measuring their relative entropy $\mathcal{R}(a_j, z_j)$. Combining this extended transportation cost with the volume constraints, we solve for an optimal transportation plan with free surfaces via

$$\min_{\{T_{pj}, a_j\}} \sum_{p,j} T_{pj} \|\mathbf{x}_p - \mathbf{x}_j\|^2 + \varepsilon \sum_{p,j} \mathcal{H}(T_{pj}) + \varepsilon \sum_j \mathcal{R}(a_j, z_j) \tag{19a}$$

$$\text{s.t.} \begin{cases} \sum_p T_{pj} + a_j = V_j & \forall j, \\ \sum_j T_{pj} = V_p & \forall p. \end{cases} \tag{19b}$$

As detailed in the Appendix A, both the transportation plan and the air volume unknowns can be concisely expressed in terms of scaling variables for the particles and grid cells, yielding

$$\boxed{\begin{cases} T_{pj} = s_j s_p K_{pj}, \\ a_j = s_j z_j. \end{cases}} \tag{20}$$

Sinkhorn Iterations: By substituting Eq. (20) into Eq. (19b), we can update the scaling amounts for our free surface optimization using a simple modification of lines 5 and 8 in Algorithm 1:

$$\begin{cases} s_j = V_j / \left(\sum_p K_{pj} s_p + z_j\right) \forall j, \\ s_p = V_p / \left(\sum_j K_{pj} s_j\right) \quad \forall p. \end{cases} \tag{21}$$

Note that, despite the additional unknowns to represent the air volumes, the number of scaling variables are kept the same and our iterative solver retains its linear convergence.

Surfacing: Equipped with air volumes, we now define the fluid occupancy within each cell j of the transportation grid as $\mu_j = 1 - a_j / V_j$. If necessary, we can aggregate the fluid occupancy over the simulation grid via $\mu_i = \sum_j N_{ij} \mu_j$. The free surface is then identified as a smeared interface spanned by every grid cell with a partial fluid occupancy, similar to interface tracking methods (see, e.g., [Mullen et al. 2007]). Therefore, we can surface the liquid-air interface into a

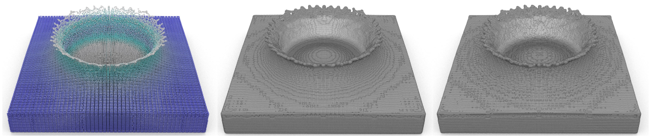

Fig. 6. Our method provides reliable and volume-preserving surfacing through isocontours of the fluid occupancy defined by our regularized transportation plan. We also compare our isocontour versus the result of converting particles to VDB to mesh, where we construct VDB from particles using radii estimated from the prescribed particle volumes.

mesh by directly contouring the fluid occupancy through an isolevel between zero and one. Figure 6 shows our surfacing results extracted by the dual contouring method available in [Museth et al. 2013] using the levelset of value 1/2. Additionally, Figure 5 compares the fluid occupancy produced by our solver without the relative entropy term and varying the sharpness parameter η. Observe that our optimization conserves the fluid volume by correcting the baseline values with smaller η favoring more smeared fluid-air interfaces.

Obstacle Masks: In addition to free surfaces, we can represent solid obstacles using a masking attribute discretized by the value $0 \le b_j \le 1$ for every cell in the t-grid that quantifies the fraction of cell volume occluded by the obstacles. To incorporate these obstacle masks in our optimization, we simply update the volume capacity $V_j = (1-b_j)|\Omega|/n_j$ for every grid cell j. We can then proceed with the optimization of the regularized transportation plan described in Eq. (19). Note that the prescribed volumes for the Lagrangian particles must also be updated so that $\sum_p V_p \le \sum_j V_j$, where we use the inequality to indicate the presence of free surfaces.

Pruning Variables: We can further exploit the discretization of free surfaces and solid obstacles to speed up our optimization by culling out the scaling variables for some of the t-grid cells. We first pin down $s_j = 0$ for any cell j with $V_j = 0$, since they are completely blocked by solid obstacles. We also detect every cell j with no particle contribution (i.e., $\max_p K_{pj} = 0$), and then overwrite these cells with the baseline $z_j = V_j$ and the scaling $s_j = 1$ so that $a_j = V_j$ is enforced. As a result, we can compute the Sinkhorn iterations in Eq. (21) by only traversing the "active" cells that have a non-zero volume capacity and interact with at least one of the Lagrangian particles.

Discussion: Once the optimal transportation plan is found, we can construct power kernels using Eq. (4) and compute transfer weights with Eq. (9) as previously described with no modification. Note that the approximation $d\mathbf{x} \approx V_j$ used in §5 is now replaced with $d\mathbf{x} \approx \mu_j V_j$, but the derivation steps and the final expressions for our Power Particle-In-Cell method are exactly the same. Finally, we point out that our approach for free surface and solid obstacles adds negligible computational cost to the optimization of transportation plans, in sharp contrast to the construction of power diagrams with ghost particles used by de Goes et al. [2015] and the elaborated clipping scheme against spheres proposed by Lévy [2022].

7 IMPLEMENTATION DETAILS

In this section, we discuss some practical considerations developed to accelerate the computations of the Power Particle-In-Cell method. Our implementation of hybrid Lagrangian/Eulerian simulators follows the variational framework proposed by Batty et al. [2007]. We make use of Eigen [Guennebaud et al. 2010] as our linear algebra library, leveraging the sparse matrix-vector multiplication (SpMV) in our Sinkhorn routine. The AMGCL library [Demidov 2019] is also used for solving the pressure projection linear system. Lastly, we employ OpenVDB [Museth et al. 2013] as the sparse data structure representing both the simulation and transportation grids.

Sparse Data Structure: The Sinkhorn method in Algorithm 1 involves multiple iterations of matrix-vector multiplications that update the scaling unknowns until convergence. Therefore, a performant implementation of matrix-vector multiplication is essential. Since the Gaussian kernels $\{K_{pj}\}$ decay exponentially with the distance between particles and t-grid cells, most of the kernel values are close to zero. If not carefully handled, these small entries can cause numerical blowups due to limited floating-point dynamic range. One can resolve this numerical instability by implementing the Sinkhorn iterations in log-domain (see, e.g., [Mandad et al. 2017, Section 3.3]). However, it is still unaffordable to manage the Gaussian kernels as a dense matrix because of high memory usage and costly matrix-vector computations. We instead construct $\{K_{pj}\}$ as a sparse matrix, which accelerates computations via SpMV while improving numerical stability by avoiding small kernel values. Towards this goal, we assemble the Gaussian kernel sparse matrix by cutting off its coefficients smaller than three standard deviations:

$$K_{pj} = \begin{cases} \psi^{\varepsilon}(-\|\mathbf{x}_p - \mathbf{x}_j\|^2) & \text{if } \|\mathbf{x}_p - \mathbf{x}_j\| \le 3\sqrt{\varepsilon}, \\ 0 & \text{otherwise.} \end{cases} \quad (22)$$

With this threshold, we have observed a reduction by an order of magnitude on both the memory footprint and computational cost of the Sinkhorn algorithm, thus making our regularized transportation optimization practical even for large-scale simulations.

Regularization Amount: As pointed out by Cuturi [2013], the Sinkhorn algorithm converges faster as the regularization amount ε increases. However, larger values of ε end up creating a denser matrix for the Gaussian kernel and then the overall performance of each Sinkhorn iteration worsens due to the slower SpMV computations. Moreover, increasing ε spreads out and blends the impact of nearby particles, resulting in a more viscous fluid dynamics. On the other hand, smaller ε may prune the kernel values too much, making the influence range of each particle too short to reach any other

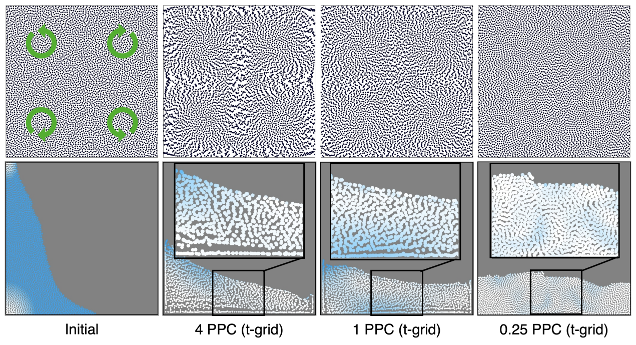

Fig. 7. We run Taylor-Vortex and dram break simulations in 2D to study the effect of different particle-per-cell (PPC) ratios for the transportation grid (t-grid). We used the Power FLIP method for these examples with a resolution for the simulation grid (s-grid) fixed at 35×35 in the first row and 50×50 in the second row. From left to right, the resolution of the t-grid is set to 35×35, 70×70, 140×140 in the first row, and 50×50, 100×100, 200×200 in the second row, which correspond to PPC ratios of 4, 1, and 0.25, respectively. Observe that decreasing PPC in the t-grid improves the particle distribution.

particle. Therefore, we must choose ε large enough for multiple particles to interact and small enough to produce sparse matrices in order to achieve fast Sinkhorn runtime. In our experiments, we set ε to be proportional to the t-grid resolution using $\varepsilon = 2(\Delta x)^2$, where Δx is the width of the t-grid cells. Combined with the cut-off threshold in Eq. (22), each particle contributes to our sparse matrix with an average of six t-grid cells for every dimension in a total of 36 cells in 2D and 216 cells in 3D. Additionally, given a desirable number of particles, we suggest setting the t-grid width Δx so that the particle-per-cell ratio (PPC) is between 0.25 and 1. With this small PPC in the t-grid, multiple grid cells are allocated to discretize the optimized transportation plans, thus offering a more accurate approximation of the particle density kernels. Figure 7 compares the particle distributions produced by our Power FLIP solver using different PPC ratios for the t-grid. Although our results remain stable, particles are more uneven as we increase the t-grid PPC. It is also worth noticing that the PPC for the s-grid is set the same as in prior hybrid solvers, detached from the t-grid as discussed in §5.

Assembling Pressure Projection: Even though our method can be incorporated into fluid simulation frameworks like Batty et al. [2007], there are still some subtleties we need to take care of. Conventionally, particles are used as markers to indicate the existence of fluid within the Eulerian grid, and a subsequent signed distance function is constructed from particles to identify the pressure degrees of freedom. However, signed distance functions built from particles may not represent the fluid occupancy accurately, often causing volume oscillations around the free surface. In contrast, our free surface handling described in §6 provides precise fluid occupancy values μ_i per grid cell i directly derived from our volume-constrained transportation plan. Therefore, we can identify the degrees of freedom in our pressure projection by simply verifying if the fluid volume in each s-grid cell i is more than half of its capacity, i.e., $\mu_i \geq 1/2$, otherwise we mark the cell as air or solid. These occupancy estimates can be further used to extend our solver with second-order accurate discretizations via the ghost method [Gibou et al. 2002].

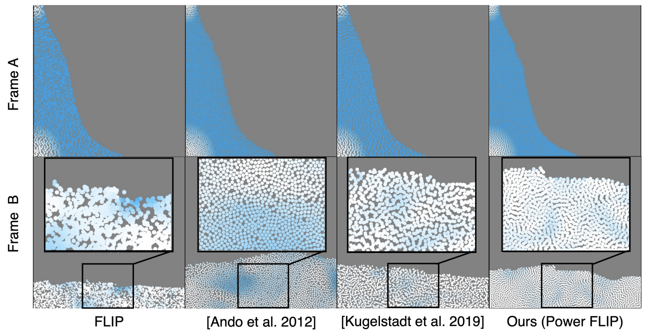

Fig. 8. We compare the results of a dam break simulation in 2D using FLIP (left), FLIP combined with spring forces [Ando et al. 2012] (left-middle), a modified FLIP with the implicit density projection [Kugelstadt et al. 2019] (middle-right), and our Power FLIP method (right). The top row shows the particle view in an early simulation frame versus a particle closeup at a later frame in the bottom row. Pseudo-colors encode the magnitude of the particle velocities. Notice that our weighting scheme improves the quality of the particle distributions, while still preserving the fluid volume as indicated by the fluid height in the bottom row.

8 RESULTS AND DISCUSSION

We provide next a series of benchmark tests in 2D and 3D that showcase the capabilities of our method. We point the reader to the accompanying video for complete simulation clips. Table 1 summarizes the simulation configurations and their performances.

Parameters: The setup for our experiments is based on four input values: the simulation timestep, the particle count, the t-grid particle-per-cell ratio, and the resolution ratio between s-grid and t-grid. The cell size Δx for both the s-grid and the t-grid as well as the regularization amount ε are then derived using these parameters, as previously detailed in §7. Our tests also used a tolerance of $\delta = 0.1$ in Algorithm 1 in order to control the volume residual of the t-grid cells, while enforcing the volume constraints on particles. This implies that every particle retains the exact amount of volume at every timestep, but the volume of each individual t-grid cell may differ from its capacity. Since we use more t-grid cells than particles, measuring the deviation of cell volumes is more strict than the particle-based termination criterion used by de Goes et al. [2015].

2D Dam Break: In Figure 8, we visualize the differences between FLIP, FLIP relaxed with [Ando et al. 2012] , FLIP combined with [Kugelstadt et al. 2019], and our Power FLIP method for a 2D dam break simulation. This example used $10k$ particles, a s-grid resolution of 50×50, a t-grid resolution of 200×200, and timestep of $\Delta t = 0.01s$. FLIP without any post-process particle relaxations suffers from significant volume loss and particle clumping (first column). The method of Ando et al. [2012] favors more uniform particles but it artificially inflates the fluid volume since only particle clustering is penalized (second column). By adding the volume correction proposed by Kugelstadt et al. [2019], FLIP is able to restore its volume, however, uneven particle distribution persists because the particle displacements can only resolve the volume changes up to the grid resolution (third column). In contrast, our power weights combined with FLIP preserve particle volumes over time while maintaining evenly spaced particle distributions (fourth column).

Non-uniform Sampling: Unlike traditional hybrid fluid solvers where particles serve purely as markers, our method accounts for

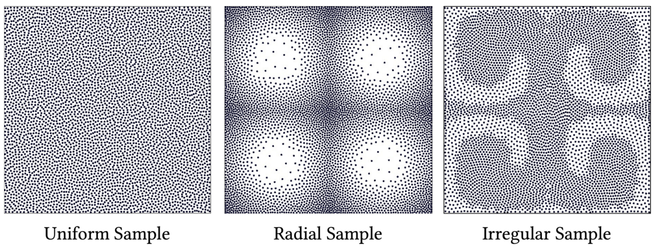

Fig. 9. Our method enables hybrid solvers to handle non-uniform particle distributions with varying particle-per-cell ratios. Here, we compare 2D Taylor-Vortex simulations using particles with equal volumes (left) versus two configurations of spatially varying volumes (middle and right).

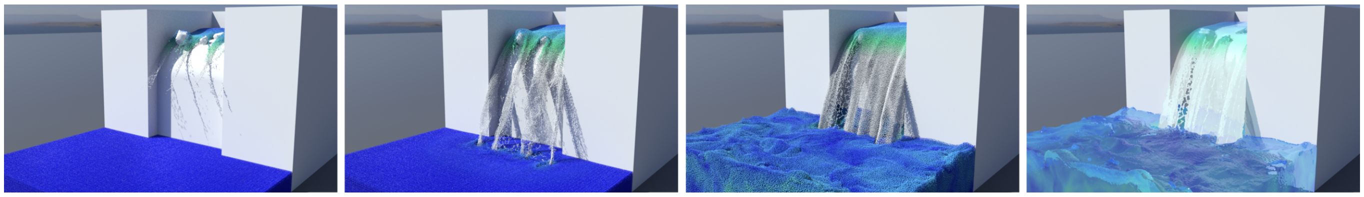

Fig. 10. This example shows a large-scale simulation of a waterfall with 3.5 million particles computed using our Power FLIP method. The first three images display the particle view at different simulation frames with color-coded velocities. In the last image, we include the surface rendering produced by the isocontour of our volume-preserving fluid occupancy.

the actual volume occupied by particles. This allows us to initialize particles with different volumes and generate adaptive transfer kernels through our transportation optimization, thus producing well-spaced yet non-uniform particle distributions as shown in Figure 9. We point out that prior correction schemes like [Ando et al. 2012; Kugelstadt et al. 2019] expect nearly constant particle volumes and, therefore, they cannot handle non-uniform particle distances. Moreover, the transfer kernel in hybrid methods is predefined using at least 4 PPC in 2D and 8 PPC in 3D in order to avoid artificial vacuum and incorrect dynamics. Instead, our optimized kernels work robustly even when grid cells are not occupied by any particles.

Energy Preservation: In hybrid methods, energy loss usually happens during pressure projection and particle-grid transfer. To study the energy decay of our method compared to prior work, we conduct a 2D Taylor-Green vortex simulation with $5k$ particles as shown in Figure 11. We used a simulation grid with resolution of 25×25 for our Power APIC and also for the original APIC method with linear and quadratic kernels. We also experimented with a version of Power APIC using a 100×100 simulation grid resolution, thus matching the transportation grid resolution. Power APIC has slightly worse energy conservation than APIC with linear kernel, but better behavior than APIC with quadratic kernel. This is not surprising since our power kernel is of higher-order and has wider transfer stencil than the simplistic linear kernel. In fact, transfer schemes with higher-order kernels have been previously reported to damp out more energy [Ding et al. 2020]. On the other hand, the Power Particles method [de Goes et al. 2015] shows a superior capability of preserving energy. We can improve the energy loss of our Power APIC scheme to the same level as Power Particles by simply increasing the simulation grid resolution. We note that APIC with linear and quadratic kernels can further improve the energy rate by employing higher resolution grids, but it requires a significant increase in particle count. In contrast, our Power APIC method can provide an energy behavior similar to the high-resolution APIC using a reduced number of particles ($10k$ vs. $80k$).

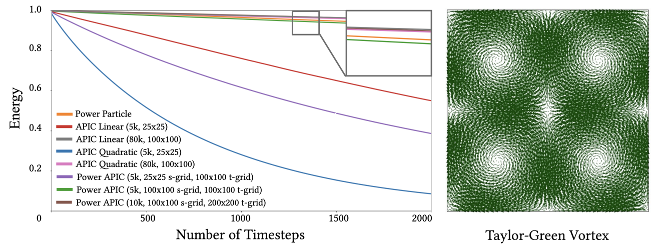

Fig. 11. This plot compares the kinetic energy preservation over time produced by Power Particles [de Goes et al. 2015], APIC [Jiang et al. 2015], and our method (Power APIC) through a 2D Taylor-Green vortex simulation. Under the same simulation resolution, Power APIC shows an energy decay between APIC with linear kernel and APIC with quadratic kernel. By increasing the simulation grid resolution, our results lead to energy conservation similar to the original Power Particles. The energy rate can be further improved by adding more particles to the fluid simulation, with our approach requiring only one eighth of the particle count used by APIC schemes to produce competitive behavior (see close-up).

Additional Examples: We performed multiple simulation scenarios in order to evaluate the correctness and effectiveness of our method. Figure 13 demonstrates that our approach is capable of handling complex solid boundaries, while yielding results qualitatively similar to [de Goes et al. 2015]. To test the stability of our method, we computed a stress example that initializes fluid particles by compressing them into a single simulation cell. Our solver is able to fully restore fluid volume as shown in Figure 3. We also ran comparisons of our method against FLIP and APIC combined with [Ando et al. 2012] and [Kugelstadt et al. 2019]. Figures 2 and 12 show the results for two free surface simulations. In both cases, the work of Kugelstadt et al. [2019] provides improved volume conservation but it is unable to correct the sub-grid non-uniform particle distributions, which cause the breakage of the fluid thin sheet and noisy free surfaces. The method of Ando et al. [2012] produces more relaxed particles at the cost of an inflated fluid volume. Conversely, our method resolves the particle distributions while preserving the fluid volume and producing smooth free surfaces. We note that our comparisons did not insert neither delete particles from the simulation. Finally, we illustrate in Figure 10 a large-scale waterfall simulation with 3.5 million particles produced by our technique.

Performance: In addition to reproducing results qualitatively similar to [de Goes et al. 2015], our method achieves an averaged speedup factor greater than two compared to the original Power Particles runtimes, as reported in Table 1. For a more detailed analysis, we also broke down the timing spent by a single timestep for a 3D splash simulation of 1M particles (similar to the left column in Figure 13). In this test, we used the Power FLIP transfer scheme and set the s-grid to be twice coarser than the t-grid. Our method takes an average of 6.5 seconds per timestep with 46% of time spent solving the Sinkhorn iterations, 29% for pressure projection, 11% for particle-grid transfer, 7% for particle advection, and 7% for building the air volume baselines from particles. The implementation of de Goes et al. [2015] takes instead 21 seconds per timestep, where 51% is used for constructing the power diagram, 24% for solving pressure, 14% for seeding air particles, and 11% for the weight update. Our performance gain comes mostly from three parts: our method replaces the power diagram construction with the more efficient Sinkhorn algorithm, it also makes air particle seeding no longer needed, and our pressure system is of a smaller size because we can use a coarse simulation grid decoupled from the particle resolution.

Limitations: Although our method provides better particle distributions and volume preservation than existing hybrid fluid solvers, these improvements come at an additional computational cost. As described in Algorithm 2, the Power Particle-In-Cell method introduces two new steps, namely the transportation optimization and the power weights assembly in lines 1 and 2, respectively. In our tests with the FLIP-style transfer and the s-grid twice coarser than the t-grid, these extra routines led to an increase in the total time spent per timestep of nearly 50%. To mitigate this overhead, we have considered changing the cut-off threshold in Eq. (22) to two standard deviations, shrinking the particle stencils to four grid cells per dimension. By doing so, the overhead added by our method reduces to 25% of the total cost and then the Eulerian-based pressure projection becomes the bottleneck. However, these smaller Gaussian stencils are effectively limiting the influence range for each particle and, therefore, the resulting simulations trade visual quality by improved performance. Finally, we note that further speedup can be achieved by our method through GPU due to the parallel-friendly nature of our computations.

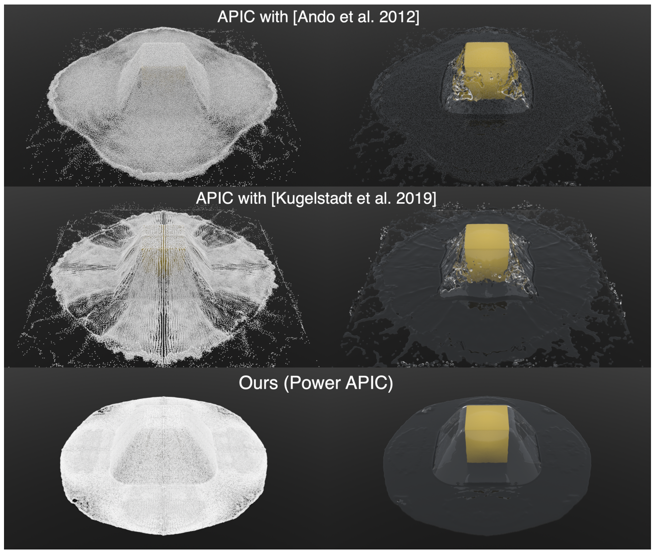
APIC with [Ando et al. 2012]
APIC with [Kugelstadt et al. 2019]
Ours (Power APIC)

Fig. 12. In this example, we drop a liquid sphere onto a box using three variants of the APIC method. Compared with APIC using [Ando et al. 2012] and [Kugelstadt et al. 2019], our method generates a smooth fluid thin sheet with evenly spaced particles and volume preservation.

Table 1. This table reports statistics for our experiments, including the simulation configuration and the averaged time spent per frame. The value Δt indicates the simulation timestep, while Δx_s and Δx_t denote the grid width for the s-grid and t-grid, respectively. We also provide the performance speedup of our method versus the original Power Particles implementation. All timings were measured on Intel Core i7-9700 with 8 cores.

Example	Δt	Δx_t	Δx_s	ε	n_p	**sec/step**	**speedup**
Figure 1	0.04	0.005	0.005	5×10^{-5}	1.25M	10.2	-
Figure 2	0.01	0.005	0.01	5×10^{-5}	1M	8.7	-
Figure 3	0.1	0.08	0.16	1.28×10^{-2}	100k	2.4	-
Figure 6	0.01	0.01	0.01	2×10^{-4}	463k	5.6	-
Figure 10	0.4	0.3	0.3	1.8×10^{-1}	3.5M	27.3	-
Figure 12	0.01	0.01	0.02	2×10^{-4}	267k	1.9	-
Figure 13a	0.04	0.08	0.16	1.28×10^{-2}	600k	3.2	2.3x
Figure 13b	0.01	0.1	0.2	2×10^{-2}	64k	1.2	1.6x
Figure 13c	0.04	0.008	0.008	1.28×10^{-4}	616k	6.5	2.5x
Figure 13d	0.04	0.05	0.1	5×10^{-3}	100k	1.2	2.6x

9 CONCLUSION AND FUTURE WORK

In this paper, we introduced the Power Particle-In-Cell method as a new weighting strategy for fluid simulation that provides the high-quality particle distributions of the Power Particles method combined with the computational efficiency of hybrid Lagrangian/Eulerian solvers. Our power weights can be incorporated into existing transfer schemes (e.g., PIC, FLIP, APIC) with minor implementation modifications and at the cost of an iterative Sinkhorn solve in the beginning of every timestep. The resulting simulations produce faithful incompressible fluid flows with low energy dissipation and evenly spaced particles. In particular, we demonstrated through a series of experiments that our approach is robust to varying particle-per-cell ratio and supports non-uniform particle distributions free of any corrective relaxations. In this process, we also developed a new algorithm to construct Power Particles via the optimization of regularized transportation plans that handles free surfaces and solid obstacles faster than [de Goes et al. 2015].

We believe this work opens up different possibilities for future work. First, we are interested in extending our power weights to graded Eulerian grids similar to [Gao et al. 2017], which can further improve the scalability of our method. We are also investigating adaptive fluid simulations by dynamically updating the particle volumes and then accounting for these volume changes as part of our transportation optimization. Combining our scheme with surface tension is another direction we wish to pursue. Since our power weights are differentiable, the Power Particle-In-Cell method is also well-suited to improve the Material Point Method [Jiang et al. 2016] with uniform particle distributions. Finally, our optimization of fluid occupancy described in Eq. (19) offers a new perspective for stylizing fluid simulations [Kim et al. 2019, 2020; Sato et al. 2021], in which guide exemplars can be represented as baseline volumes.

ACKNOWLEDGMENTS

This work is partially supported by NSF 2153851, 2153863, 2023780 and DOE ORNL 4000171342.

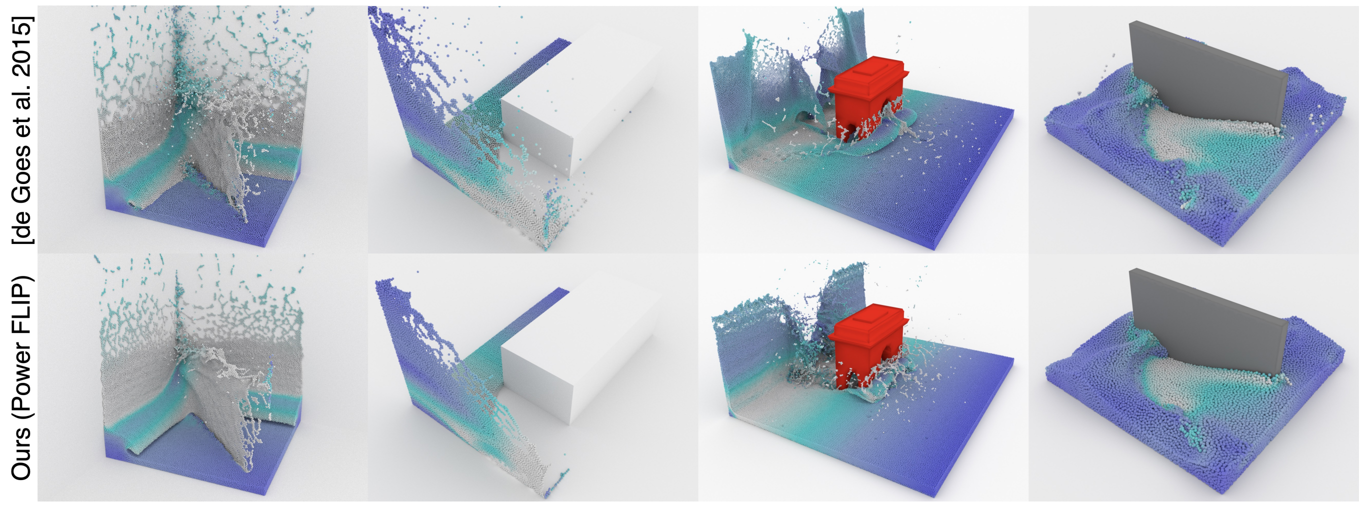

Fig. 13. We conduct several benchmark simulations to compare the Power Particle-In-Cell method against the original implementation of Power Particles [de Goes et al. 2015]. Our method provides results qualitatively similar to Power Particles while achieving a significant performance speedup.

REFERENCES

Ryoichi Ando, Nils Thürey, and Reiji Tsuruno. 2012. Preserving Fluid Sheets with Adaptively Sampled Anisotropic Particles. *IEEE Transactions on Visualization and Computer Graphics* 18, 8 (2012), 1202–1214.

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2013. Highly Adaptive Liquid Simulations on Tetrahedral Meshes. *ACM Transactions on Graphics* 32, 4, Article 103 (2013), 10 pages.

Franz Aurenhammer. 1987. Power Diagrams: Properties, Algorithms and Applications. *SIAM J. Comput.* 16, 1 (1987), 78–96.

Franz Aurenhammer, Friedrich Hoffmann, and Boris Aronov. 1998. Minkowski-type theorems and least-squares clustering. *Algorithmica* 20, 1 (1998), 61–76.

Michael Balzer, Thomas Schlömer, and Oliver Deussen. 2009. Capacity-Constrained Point Distributions: A Variant of Lloyd's Method. In *ACM SIGGRAPH.* Article 86, 8 pages.

Scott G. Bardenhagen and Edward M. Kober. 2004. The generalized interpolation material point method. *Computer Modeling in Engineering and Sciences* 5, 6 (2004), 477–496.

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics* 26, 3 (2007), 100–es.

Markus Becker and Matthias Teschner. 2007. Weakly compressible SPH for free surface flows. In *Symposium on Computer Animation.* 209–217.

Jan Bender and Dan Koschier. 2016. Divergence-free SPH for incompressible and viscous fluids. *IEEE Transactions on Visualization and Computer Graphics* 23, 3 (2016), 1193–1206.

Nicolas Bonneel, Michiel van de Panne, Sylvain Paris, and Wolfgang Heidrich. 2011. Displacement Interpolation Using Lagrangian Mass Transport. *ACM Transactions on Graphics* 30, 6 (2011), 1–12.

Jeremiah U Brackbill and Hans M Ruppel. 1986. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys.* 65, 2 (1986), 314–343.

Robert Bridson. 2015. *Fluid simulation for computer graphics.* CRC press.

Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. 2009. *Assignment Problems.* Society for Industrial and Applied Mathematics.

Lenaïc Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. 2018. Scaling algorithms for unbalanced optimal transport problems. *Math. Comput.* 87, 314 (2018), 2563–2609.

Jens Cornelis, Markus Ihmsen, Andreas Peer, and Matthias Teschner. 2014. IISPH-FLIP for incompressible fluids. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 255–262.

Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing).* Wiley-Interscience, USA.

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems* 26 (2013), 2292–2300.

Marco Cuturi and Gabriel Peyré. 2018. Semidual Regularized Optimal Transport. *SIAM Rev.* 60, 4 (2018), 941–965.

Fernando de Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun. 2015. Power Particles: an incompressible fluid solver based on power diagrams. *ACM Transactions on Graphics* 34, 4 (2015), 50–1.

Denis Demidov. 2019. AMGCL: An efficient, flexible, and extensible algebraic multigrid implementation. *Lobachevskii Journal of Mathematics* 40, 5 (2019), 535–546.

Ounan Ding, Tamar Shinar, and Craig Schroeder. 2020. Affine particle in cell method for MAC grids and fluid simulation. *J. Comput. Phys.* 408 (2020), 109311.

Yun Fei, Qi Guo, Rundong Wu, Li Huang, and Ming Gao. 2021. Revisiting integration in the material point method: a scheme for easier separation and less dissipation. *ACM Transactions on Graphics* 40, 4 (2021), 1–16.

Florian Ferstl, Ryoichi Ando, Chris Wojtan, Rüdiger Westermann, and Nils Thuerey. 2016. Narrow Band FLIP for Liquid Simulations. *Computer Graphics Forum* 35, 2 (2016), 225–232.

Nick Foster and Dimitri Metaxas. 1996. Realistic animation of liquids. *Graphical models and image processing* 58, 5 (1996), 471–483.

Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A polynomial particle-in-cell method. *ACM Transactions on Graphics* 36, 6 (2017), 1–12.

Ming Gao, Andre Pradhana Tampubolon, Chenfanfu Jiang, and Eftychios Sifakis. 2017. An adaptive generalized interpolation material point method for simulating elastoplastic materials. *ACM Transactions on Graphics* 36, 6 (2017), 1–12.

Frederic Gibou, Ronald P Fedkiw, Li-Tien Cheng, and Myungjoo Kang. 2002. A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. *J. Comput. Phys.* 176, 1 (2002), 205–227.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Francis H Harlow. 1962. *The particle-in-cell method for numerical solution of problems in fluid dynamics.* Technical Report. Los Alamos Scientific Lab., N. Mex.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics* 37, 4 (2018), 1–14.

Yuanming Hu, Xinxin Zhang, Ming Gao, and Chenfanfu Jiang. 2019. On hybrid lagrangian-eulerian simulation methods: practical notes and high-performance aspects. In *ACM SIGGRAPH Courses.* 16.

Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. 2013. Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics* 20, 3 (2013), 426–435.

Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. 2014. SPH fluids in computer graphics. In *EUROGRAPHICS 2014/S. LEFEBVRE AND M. SPAGNUOLO.* Citeseer.

Chenfanfu Jiang, Theodore Gast, and Joseph Teran. 2017a. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Transactions on Graphics* 36, 4 (2017), 1–14.

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Transactions on Graphics* 34, 4 (2015), 1–10.

Chenfanfu Jiang, Craig Schroeder, and Joseph Teran. 2017b. An angular momentum conserving affine-particle-in-cell method. *J. Comput. Phys.* 338 (2017), 137–164.

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. 2016. The material point method for simulating continuum materials. In *ACM SIGGRAPH Courses*. 1–52.

Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. 2019. Transport-Based Neural Style Transfer for Smoke Simulations. *ACM Transactions on Graphics* 38, 6, Article 188 (2019), 11 pages.

Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. 2020. Lagrangian Neural Style Transfer for Fluids. *ACM Transactions on Graphics* 39, 4, Article 52 (2020), 10 pages.

Gergely Klár, Theodore Gast, Andre Pradhana, Chuyuan Fu, Craig Schroeder, Chenfanfu Jiang, and Joseph Teran. 2016. Drucker-prager elastoplasticity for sand animation. *ACM Transactions on Graphics* 35, 4 (2016), 1–12.

Philip A. Knight. 2008. The Sinkhorn–Knopp Algorithm: Convergence and Applications. *SIAM J. Matrix Anal. Appl.* 30, 1 (2008), 261–275.

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. 2020. Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids. *arXiv preprint arXiv:2009.06944* (2020).

Tassilo Kugelstadt, Andreas Longva, Nils Thurey, and Jan Bender. 2019. Implicit density projection for volume conserving liquids. *IEEE Transactions on Visualization and Computer Graphics* (2019).

Frank Losasso, Jerry Talton, Nipun Kwatra, and Ronald Fedkiw. 2008. Two-way coupled SPH and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics* 14, 4 (2008), 797–804.

Bruno Lévy. 2022. Partial optimal transport for a constant-volume Lagrangian mesh with free boundaries. *J. Comput. Phys.* 451 (2022), 110838.

Miles Macklin and Matthias Müller. 2013. Position based fluids. *ACM Transactions on Graphics* 32, 4 (2013), 1–12.

Manish Mandad, David Cohen-Steiner, Leif Kobbelt, Pierre Alliez, and Mathieu Desbrun. 2017. Variance-Minimizing Transport Plans for Inter-Surface Mapping. *ACM Transactions on Graphics* 36, 4, Article 39 (2017), 14 pages.

Quentin Mérigot and Jean-Marie Mirebeau. 2016. Minimal Geodesics Along Volume-Preserving Maps Through Semi-discrete Optimal Transport. *SIAM J. Numer. Anal.* 54, 6 (2016), 3465–3492.

Joseph J. Monaghan. 2005. Smoothed particle hydrodynamics. *Reports on Progress in Physics* 68, 8 (2005), 1703–1759.

Patrick Mullen, Alexander McKenzie, Yiying Tong, and Mathieu Desbrun. 2007. A Variational Approach to Eulerian Geometry Processing. In *ACM SIGGRAPH*.

Ken Museth, Jeff Lait, John Johanson, Jeff Budsberg, Ron Henderson, Mihai Alden, Peter Cucka, David Hill, and Andrew Pearce. 2013. OpenVDB: an open-source data structure and toolkit for high-resolution volumes. In *ACM SIGGRAPH Courses*. 1–1.

Rafael Nakanishi, Filipe Nascimento, Rafael Campos, Paulo Pagliosa, and Afonso Paiva. 2020. RBF liquids: an adaptive PIC solver using RBF-FD. *ACM Transactions on Graphics* 39, 6 (2020), 1–13.

Gabriel Peyré, Marco Cuturi, et al. 2019. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning* 11, 5-6 (2019), 355–607.

Stefan Reinhardt, Tim Krake, Bernhard Eberhardt, and Daniel Weiskopf. 2019. Consistent shepard interpolation for SPH-based fluid animation. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–11.

Syuhei Sato, Yoshinori Dobashi, and Theodore Kim. 2021. Stream-Guided Smoke Simulations. *ACM Transactions on Graphics* 40, 4, Article 161 (2021), 7 pages.

Takahiro Sato, Chris Wojtan, Nils Thuerey, Takeo Igarashi, and Ryoichi Ando. 2018. Extended Narrow Band FLIP for Liquid Simulations. *Computer Graphics Forum* (2018). https://doi.org/10.1111/cgf.13351

Richard Sinkhorn. 1967. Diagonal Equivalence to Matrices with Prescribed Row and Column Sums. *The American Mathematical Monthly* 74, 4 (1967), 402–405.

Barbara Solenthaler and Renato Pajarola. 2009. Predictive-corrective incompressible SPH. In *ACM SIGGRAPH*. 1–6.

Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. 2015. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics* 34, 4 (2015), 1–11.

Justin Solomon, Gabriel Peyré, Vladimir G. Kim, and Suvrit Sra. 2016. Entropic Metric Alignment for Correspondence Problems. *ACM Transactions on Graphics* 35, 4, Article 72 (2016), 13 pages.

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A material point method for snow simulation. *ACM Transactions on Graphics* 32, 4 (2013), 1–10.

Alexey Stomakhin, Craig Schroeder, Chenfanfu Jiang, Lawrence Chai, Joseph Teran, and Andrew Selle. 2014. Augmented MPM for phase-change and varied materials. *ACM Transactions on Graphics* 33, 4 (2014), 1–11.

Deborah Sulsky, Shi-Jian Zhou, and Howard L Schreyer. 1995. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications* 87, 1-2 (1995), 236–252.

Tetsuya Takahashi and Ming C Lin. 2019. A Geometrically Consistent Viscous Fluid Solver with Two-Way Fluid-Solid Coupling. In *Computer Graphics Forum*, Vol. 38. 49–58.

Kiwon Um, Seungho Baek, and JungHyun Han. 2014. Advanced hybrid particle-grid method with sub-grid particle correction. In *Computer Graphics Forum*, Vol. 33. 209–218.

Yonghao Yue, Breannan Smith, Christopher Batty, Changxi Zheng, and Eitan Grinspun. 2015. Continuum foam: A material point method for shear-dependent flows. *ACM Transactions on Graphics* 34, 5 (2015), 1–20.

Xiao Zhai, Fei Hou, Hong Qin, and Aimin Hao. 2018. Fluid simulation with adaptive staggered power particles on gpus. *IEEE Transactions on Visualization and Computer Graphics* 26, 6 (2018), 2234–2246.

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Transactions on Graphics* 24, 3 (2005), 965–972.

A DERIVATION

In this appendix, we summarize the derivation steps showing that Eq. (20) is the solution of our regularized transportation problem given by Eq. (19). We start by writing the Lagrangian function associated with Eq. (19), which introduces Lagrange multipliers r for the particles and transportation grid cells, producing:

$$\begin{aligned}\mathcal{L}(T,a,r) &= \textstyle\sum_{p,j} T_{pj}\|\mathbf{x}_p-\mathbf{x}_j\|^2 + \varepsilon\sum_{p,j}\mathcal{H}(T_{pj}) + \varepsilon\sum_{p,j}\mathcal{R}(a_j,z_j)\\ &\quad - \textstyle\sum_j r_j\left(\sum_p T_{pj}+a_j-V_j\right) - \sum_p r_p\left(\sum_j T_{pj}-V_p\right).\end{aligned}$$

One can easily deduce that the partial derivatives of the Lagrangian function $\mathcal{L}$ with respect to (T,a,r) are:

$$\begin{cases}\partial_{T_{pj}}\mathcal{L} = \|\mathbf{x}_p-\mathbf{x}_j\|^2 + \varepsilon\log T_{pj} - r_j - r_p,\\ \partial_{a_j}\mathcal{L} = \varepsilon\left(\log a_j - \log z_j\right) - r_j,\\ \partial_{r_j}\mathcal{L} = V_j - a_j - \sum_p T_{pj},\\ \partial_{r_p}\mathcal{L} = V_p - \sum_j T_{pj}.\end{cases}$$

By equating the gradient of the Lagrangian function $\mathcal{L}$ to zero, we obtain the optimality conditions of Eq. (19), which imply that:

$$\begin{cases}\partial_{T_{pj}}\mathcal{L}=0 \implies T_{pj} = \psi^{\varepsilon}(r_j+r_p-\|\mathbf{x}_p-\mathbf{x}_j\|^2),\\ \partial_{a_j}\mathcal{L}=0 \implies a_j = \psi^{\varepsilon}(r_j)z_j,\\ \partial_{r_j}\mathcal{L}=0 \implies V_j = a_j + \sum_p T_{pj},\\ \partial_{r_p}\mathcal{L}=0 \implies V_p = \sum_j T_{pj}.\end{cases}$$

The two top terms confirm that Eq. (20) provides the solution of Eq. (19), while the two bottom terms reproduce the volume constraints.

Next, we show that the constrained optimization in Eq. (19) is equivalent to an unconstrained convex minimization $\min_r \mathcal{F}(r)$, where $\mathcal{F}$ is a dual function in terms of the Lagrange multipliers r. We define this dual function as:

$$\begin{aligned}\mathcal{F}(r) &= \varepsilon\left[\textstyle\sum_{p,j}\psi^{\varepsilon}(r_j+r_p)K_{pj} + \sum_j \psi^{\varepsilon}(r_j)z_j\right]\\ &\quad - \left[\textstyle\sum_j r_jV_j + \sum_p r_pV_p\right].\end{aligned}$$

Note that $\mathcal{F}$ is convex with respect to r because $\psi^{\varepsilon}(r)$ is convex and the remaining terms are linear in r, while the constants (K,z,V) are non-negative. Since the optimization of $\mathcal{F}$ is over any configuration of Lagrange multipliers r, we obtain a unconstrained convex minimization over an Euclidean space, which ensures that a solution exists and is unique. Finally, we confirm that the derivatives of $\mathcal{F}$ reproduce the volume constraints:

$$\begin{cases}\partial_{r_j}\mathcal{F} = \sum_p \psi^{\varepsilon}(r_j+r_p)K_{pj} + \psi^{\varepsilon}(r_j)z_j - V_j = \sum_p T_{pj} + a_j - V_j,\\ \partial_{r_p}\mathcal{F} = \sum_j \psi^{\varepsilon}(r_j+r_p)K_{pj} - V_p = \sum_j T_{pj} - V_p.\end{cases}$$