Computer Animation and Virtual Worlds

WILEY

RESEARCH ARTICLE

Neural Fluid Simulator With Hybrid Physical-Visual Constraints

Feilong Du[1,2] | Xiaojuan Ban[1] | Yuhang Xu[1] | Angelos Chatzimparmpas[3] | Yalan Zhang[1,4]

[1]School of Intelligent Science and Technology, University of Science and Technology Beijing, Beijing, China | [2]School of Data Science and Artificial Intelligence, Beijing University of Chinese Medicine, Beijing, China | [3]Department of Information and Computing Sciences, Utrecht University, Utrecht, the Netherlands | [4]Shunde Innovation School, University of Science and Technology Beijing, Beijing, Guangdong, China

Correspondence: Yalan Zhang (zhangyl@ustb.edu.cn)

Received: 19 June 2025 | **Revised:** 13 April 2026 | **Accepted:** 16 April 2026

Keywords: fluid prediction | fluid simulation | hybrid constraints | neural fluid simulator

ABSTRACT

Traditional physics-based fluid simulations typically rely on manual modeling and incremental adjustments to achieve desired effects, which can limit objectivity and generalizability to new scenarios. To address these challenges, we propose a novel neural fluid simulator that integrates visual priors from 2D image sequences with physically constrained continuous convolution. Specifically, we extract and refine point clouds from image sequences, then infer the kinetic properties of the fluid. We introduce an energy-based physical constraint and incorporate it into a continuous convolution solver. By iteratively optimizing these inputs to enforce physical laws—particularly incompressibility—the solver produces accurate fluid motion predictions. Our approach uniquely combines visual data and physical constraints, enhancing the realism and accuracy while providing stronger generalization of fluid simulations.

1 | Introduction

Fluid simulation is an attractive research area in computer graphics, with applications in fields such as computer games and scientific visualization. Traditional physics-based fluid simulations can reproduce flow phenomena but require manual parameter adjustments to achieve desired effects [1], making replication of real-world scenarios challenging. Moreover, they cannot seamlessly integrate with real-world fluid data [2]. Existing deep learning-based fluid simulation methods often rely on purely data-driven models, which learn fluid behaviors from data but lack physical guidance, and frequently struggle to generalize to new or unseen scenarios.

To overcome these limitations, we propose a novel neural fluid simulator that combines visual priors and physical constraints. We use monocular 2D images as input because they are widely available and easily accessible. As shown in Figure 1, we first employ monocular 2D image sequences to generate 3D fluid particle positions [3]. Then, a dynamics-guided inference module computes the velocity field, applies position smoothing, and determines viscosity values of the particles, which serve as inputs to the continuous convolution solver. Next, this solver iteratively optimizes the inputs to ensure adherence to physical laws—especially incompressibility—resulting in highly accurate fluid motion trajectory predictions [4].

The key contribution of this work is a hybrid approach that combines visual data and physical constraints, enhancing the realism and accuracy of fluid simulations, improving computational efficiency over traditional methods, and providing stronger generalization than purely data-driven models, thereby closely aligning with actual phenomena.

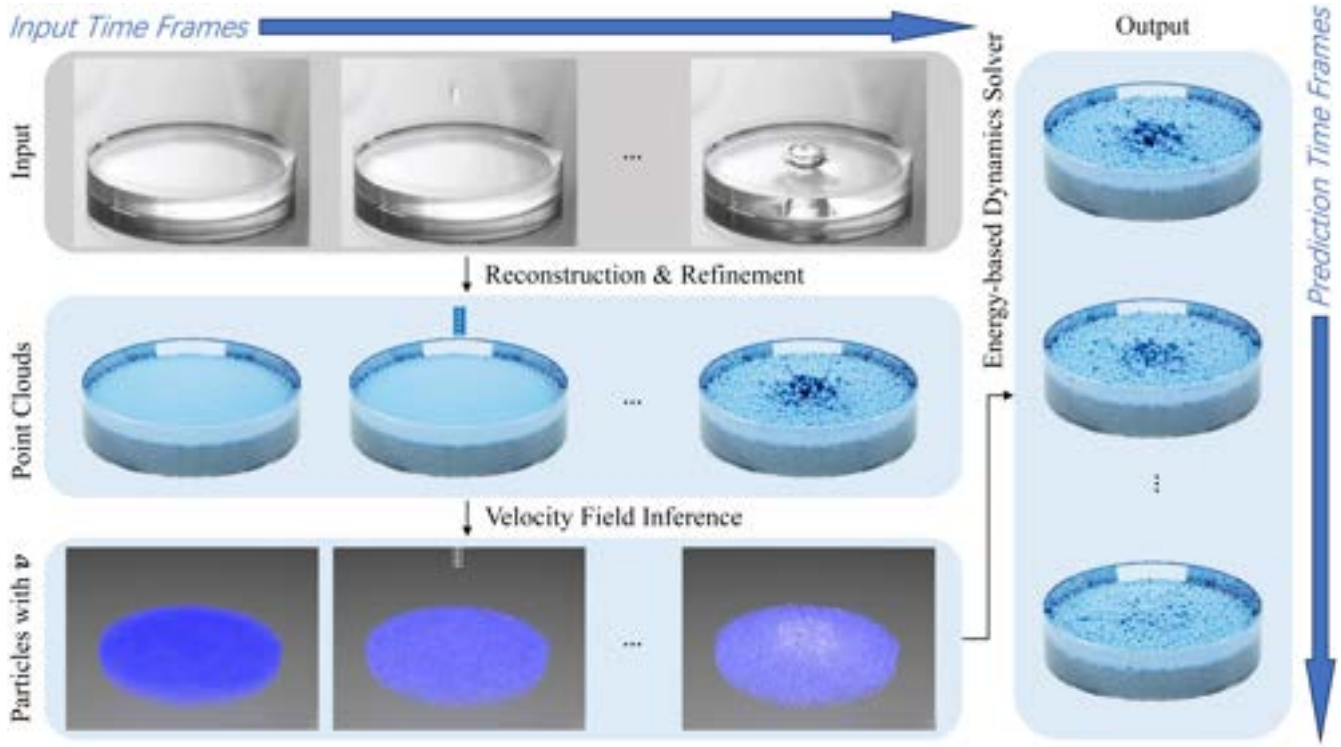

FIGURE 1 | Overview of the proposed neural fluid simulator. The process begins with reconstructing and refining point clouds from a monocular 2D image sequence. Next, a velocity field is inferred from the point cloud sequence to provide particles with kinetic properties. Finally, an energy-based physical constraint dynamics solver predicts future particle states to achieve the fluid simulation.

2 | Related Work

2.1 | Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) provides the mathematical and numerical foundations for simulating viscous incompressible flow. Foundational projection and fractional-step methods for the incompressible Navier–Stokes equations [5] decouple velocity and pressure updates by solving a pressure Poisson equation that enforces divergence-free velocity fields. Modern CFD practice—as systematized in classic texts [6]—covers discretizations on structured and unstructured grids, high-resolution advection schemes, multigrid and Krylov solvers for elliptic subproblems, and turbulence modeling. These techniques remain the baseline in accuracy and robustness. They have been widely used across thermal and hydrodynamic studies [2, 7] and augmented with sparse data assimilation or vision-based estimation to improve fidelity [8, 9]. However, the high computational cost required to resolve complex geometries and transient or turbulent flows motivates research into accelerated and surrogate approaches.

2.2 | Particle-Based Fluid Simulation

Because our approach is particle-based, we review relevant particle methods. Beyond the widely used smoothed particle hydrodynamics (SPH), particle techniques have addressed interactions with complex media that closely relate to our setting. Bayraktar et al. proposed a particle framework to simulate and visualize flows through porous media [10] and introduced a GPU-accelerated neighbor-search algorithm that scales to large particle sets [11]. Güdükbay et al. presented a two-way coupled particle method for fluid–textile interaction that accounts for capillarity and surface tension in knitwear [12]. These works established core components, efficient neighbor queries, porous/elastic coupling, and surface-tension handling that are equally central to modern learned particle simulators.

2.3 | Fluid Simulation With Machine Learning

Machine learning (ML) has been increasingly applied to fluid simulation to accelerate computation and to learn closure or surrogate models. Learning-based simulators have shown strong performance for both Lagrangian and Eulerian discretizations. Graph Network-based Simulators (GNS) learn particle interactions via message passing and achieve accurate long-horizon rollouts across fluids and multi-physics settings [13]. Ummenhofer et al. operate directly on moving particle sets with continuous convolutions for scalable Lagrangian fluid simulation [14]. On meshes, MeshGraphNets learn dynamics on unstructured grids with resolution adaptivity [15], while operator-learning methods such as the Fourier Neural Operator (FNO) provide resolution- and domain-generalization for PDE surrogates, including incompressible flow [16]. In parallel, ML has been used to improve classical workflows and application fidelity, for example, accelerating timestepping [17], optimizing mesh/solver settings [2], learning visual or physical priors from observational data [3, 18–20], and enforcing conservation properties in particle-based dynamics [4]. Recent domain studies further illustrate breadth across nanofluid transport,

plasticity-aware learning for coupled materials, and interactions with thin structures [7, 21–23].

2.4 | Hybrid and Physics-Aware Approaches

Hybrid approaches combine numerical solvers with learned components to improve stability and efficiency while enforcing physical constraints. Physics-informed learning introduces PDE residuals and boundary conditions during training to maintain consistency with governing equations [24]. Complementary solver-in-the-loop strategies learn corrections to pressure projection, advection, or subgrid closures to accelerate classical pipelines while retaining their inductive biases [15]. Recent works embed global or local physical priors within neural reconstructions or simulators, including Laplacian projection priors for smoke [25], material-point-network hybrids for fluid–solid interactions [26], and hybrid neural fluid fields inferred from videos [27]. Other graph-based or neural-vortex formulations integrate physical structure directly into learned operators [28, 29]. Physics-informed neural networks and related constrained-learning formulations have also been explored for diverse coupled materials [21, 22]. However, relatively few efforts combine hybrid simulation with real-world visual data at scale. Our method addresses this gap by integrating visual priors from 2D image sequences with physically constrained continuous convolutions to refine and infer 3D fluid dynamics.

3 | Methodology

The proposed neural fluid simulator integrates visual priors and physical constraints, as illustrated in Figure 1. Our method can reconstruct an initial fluid state from a video sequence and then predict the fluid's evolution after the video ends.

The overall workflow is as follows: first, we employ a monocular view point cloud reconstruction method [30] to obtain 3D particle positions, where a position refinement module is introduced to reduce outliers caused by mono-view reconstruction. Then, we design a dynamics-guided inference module to compute the fluid velocity field. Finally, by iteratively minimizing energy and enforcing incompressibility constraints, the neural fluid simulator generates fluid motion trajectories that adhere to physical laws.

3.1 | Kinetic Estimation and Optimization

To reconstruct the scene, we first distinguish between the dynamic fluid and the static background boundaries. We employ a segmentation model [31] on the initial video frames to separate the foreground fluid from the background. Subsequently, we reconstruct both components into point clouds. The foreground yields a sequence of dynamic fluid particles, while the background points from multiple frames are aggregated to form a single, static point cloud representing the fixed boundaries.

We then adopt the method from [30] to generate the initial 3D point clouds of the fluid from the segmented 2D images. Next, we apply the Moving Least Squares (MLS) algorithm on local neighborhoods of the coarse fluid point cloud to refine and enhance its quality. The core principle behind MLS is to perform local surface fitting for each point and its neighbors. The local surface is obtained by minimizing a weighted least squares error, where the weights reflect the distances of neighboring points to the target point, with closer points contributing more to the fitting.

For a point p in the point cloud with neighboring points $\{q_j\}_{j=1}^{k}$, where k is the number of neighbors, the MLS method seeks a local surface f_p that minimizes the weighted squared differences $S(f_p)$:

$$S(f_p) = \sum_{j=1}^{k} w_j \cdot \| f_p(q_j - p) - (q_j - p) \|^2 \tag{1}$$

where w_j is a distance-based weight function, chosen as a Gaussian $w_j = \exp\left(-\frac{\|q_j - p\|^2}{2\sigma^2}\right)$, and σ controls the size of the Gaussian neighborhood. We use high-order polynomials to fit the local surfaces f_p.

In fluid simulations, particle positions alone are insufficient. Physical attributes such as velocity are required but not provided by the reconstruction. The physical velocity inference module employs a shortest distance matching algorithm to compute the fluid velocities.

Given two point clouds $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, \ldots, q_m\}$ from consecutive frames, we find the nearest point q_j in Q for each point p_i in P to calculate their displacement. For each point $p_i \in P$, we calculate:

$$q_{\min} = \arg\min_{q_j \in Q} d(p_i, q_j) \tag{2}$$

where $d(p_i, q_j) = \| p_i - q_j \|_2$.

Since multiple points in P might share the same nearest neighbor in Q (and $n = m$), some points could remain unmatched, which is physically implausible. To resolve this, we apply the Hungarian algorithm to find an optimal matching in a weighted bipartite graph. This assignment problem can be solved in polynomial time to determine the matching with the lowest cost.

Once the nearest neighbors $q_{\min}$ are determined for all points in P, the displacement vectors $\Delta p_i = q_{\min} - p_i$ are computed. Assuming a fixed timestep Δt, the velocity v_i for each particle is obtained by $v_i = \frac{q_{\min} - p_i}{\Delta t}$. During the initial phase of fluid motion, velocity is influenced only by initial velocity and gravity, so all particles should share the same velocity in free fall. Therefore, we use the mean velocity $v_{\text{mean}} = \frac{1}{n}\sum_{i=1}^{n} v_i$ as the common velocity, ensuring physically consistent and visually coherent results.

3.2 | Energy-Based Physical Constraints

After obtaining the initial particle positions and dynamic properties, we require a physically constrained solver that can guarantee fluid incompressibility. Therefore, we employ continuous convolution as a scalar constraint in the energy model to enforce minimal energy at the predicted fluid positions.

Drawing on density conservation principles and Position-Based Dynamics (PBD), we replace conventional iterative steps with physically informed choices that ensure strict adherence to incompressibility constraints, leading to more accurate fluid flow predictions. Inference in this framework involves finding values of unobserved variables that minimize the energy function. This optimization problem can be formulated as follows:

$$Y^* = \arg\min_{Y \in \mathcal{Y}} E(Y, X) \tag{3}$$

where $X = \{x^t, v^t\}$ denotes model inputs, E is the energy model, $Y = \{x^{t+1}, v^{t+1}\}$ represents a candidate future state, and $\mathcal{Y}$ is the space of all possible states. Here, Y^* denotes the optimal predicted output state that minimizes the energy $E(Y, X)$, with lower energy values corresponding to more accurate predictions.

Energy models E typically employ an iterative inference process to find the solution. Starting with an initial guess Y^0, the solution is refined over $\mathcal{L}$ steps. A single prediction step is usually expressed as a gradient descent update on the energy function:

$$Y^l = Y^{l-1} - \lambda \nabla_Y E_\theta\left(X, Y^{l-1}\right) \tag{4}$$

where θ denotes the learnable parameters of the continuous convolution neural network, λ denotes the step size for each gradient descent step, and l is the iteration number. The process continues until Y^{l-1} reaches a local minimum of the energy landscape, indicating $E_\theta\left(X, Y^l\right) \approx E_\theta\left(X, Y^{l-1}\right)$, at which point the final refined state Y^l serves as the optimal prediction Y^*.

For simple tasks such as image generation, λ can remain constant. However, fluid prediction in 3D simulations with complex dynamics constraints requires a more sophisticated approach that satisfies density conservation and incorporates additional physical properties. In PBD, density conservation during simulation requires solving a nonlinear system of constraints governing each particle's behavior. All constraints are modeled as functions of particle positions and their neighbors. The density constraint for particle i is:

$$C_i\left(p_1, \ldots, p_n\right) = \frac{\rho_i}{\rho_0} - 1 \tag{5}$$

where ρ_0 is the rest density and ρ_i is calculated using the standard SPH density estimator:

$$\rho_i = \sum_j m_j W\left(p_i - p_j, \ \ h\right) \tag{6}$$

We assume equal mass for all particles m_j, omitting the mass term in subsequent equations. The kernel function W typically employs the Poly6 kernel for density estimation and the Spiky kernel for gradient calculations, where h denotes the smoothing length (support radius). The goal is to find a position correction Δp that satisfies $C(p + \Delta p) = 0$. By performing a first-order Taylor expansion and taking steps along the constraint gradient, the required position correction for particle k due to constraint i is derived as:

$$\Delta p_k \approx \lambda_i \nabla_{p_k} C_i \tag{7}$$

$$C_i(p + \Delta p) \ \approx C_i(p) + \sum_k \nabla_{p_k} C_i \cdot \Delta p_k = 0 \tag{8}$$

From Equation (8), the Lagrange multiplier λ_i can be calculated as:

$$\lambda_i = -\frac{C_i\left(p_1, \ldots, p_n\right)}{\sum_k \| \nabla_{p_k} C_i\|^2 + \varepsilon} \tag{9}$$

where k iterates over particle i and its neighboring particles, and ε is a relaxation parameter that mitigates potential vanishing gradients at the smoothing kernel boundary when particles separate.

3.3 | Continuous Convolution Solver With Physical Constraints

Building on the continuous convolution approach [14], we implement a physically constrained solver f_c based on an energy model. The core of this approach is to generalize discrete convolutions to operate on continuous, off-grid data such as particle positions. Given a set of particles, each with a position p_i and a feature vector h_i, the continuous convolution computes an updated feature h_i' for each particle by aggregating information from its neighbors, including both fluid particles and static boundary particles. Our solver f_c is implemented as a neural network composed of multiple continuous convolution layers. It takes the current fluid state (positions and velocities) together with the static boundary as input and outputs a single non-negative scalar representing the system's energy. Notably, we pretrain the parameters of f_c to improve generalization, and then further refine the model using the input video data to achieve more accurate predictions.

Particle trajectories are represented as a sequence of states $\left(\mathbf{X}_1, \ldots, \mathbf{X}_T\right)$, each comprising position, instantaneous velocity, and mass. For dynamic scenes, the simulator uses past flow conditions $\mathbf{X}_{\le t}$ (historical trajectory up to time t) to predict the next fluid position $\hat{\mathbf{X}}_{t+1} = s\left(\mathbf{X}_{\le t}\right)$, where $s(\cdot)$ denotes the learned state transition function of our simulator. This predicted state is appended to the sequence, and the process repeats to generate the full trajectory.

The solver consists of a predictor and an updater. The predictor, modeled as a Markov chain, learns position biases and maps input positions and velocities to an update value $\hat{\mathbf{Y}}$ (e.g., position corrections or velocity residuals), similar to iterative SPH methods. The updater then advances the current state using $\hat{\mathbf{Y}}$:

$$\hat{\mathbf{X}}_{t+1} = \text{UPDATE}\left(\mathbf{X}_t, \hat{\mathbf{Y}}\right) \tag{10}$$

Unlike standard explicit Euler integration, we incorporate an energy model and incompressibility constraints, making the forward calculation an implicit iteration $f_c\left(\mathbf{X}_t, \hat{\mathbf{Y}}\right) = c$, where c is a scalar representing the system's energy. The optimal update value $\hat{\mathbf{Y}}^*$ is formulated as:

$$\hat{\mathbf{Y}}^* = \arg\min_{\hat{\mathbf{Y}}} f_c\left(\mathbf{X}_t, \hat{\mathbf{Y}}\right) \tag{11}$$

with $\mathbf{X}_t$ including both position and velocity. Subsequently, the next state is obtained via Equation (10).

Our neural fluid simulator integrates an energy model with PBD-based incompressibility conditions via a continuous convolution neural network f_c. The prediction process comprises

two stages: first, a gradient-based method computes the optimal update $\hat{\mathbf{Y}}^*$ that iteratively satisfies physical constraints; second, particle positions are updated from $\mathbf{X}_t$ to $\mathbf{X}_{t+1}$ using this update.

ALGORITHM 1 | Overall Simulation Pipeline.

Require: Monocular video sequence $V = \{I_1, \ldots, I_T\}$.
Require: Pretrained dynamics solver f_c.
Require: Number of prediction frames N_{pred}.
Part 1: Reconstruction and Initialization
Segment video frames I_t into foreground F_t and background BG_t using a segmentation model[31].
Reconstruct a sequence of fluid point clouds $\{P_1, \ldots, P_T\}$ from $\{F_1, \ldots, F_T\}$.
Aggregate background points to form a static boundary point cloud $\mathbf{B}$.
for $t = 1$ to T **do**
  Refine point cloud P_t using MLS.
end for
for $t = 1$ to $T - 1$ **do**
  Find optimal matching between P_t and P_{t+1} using the Hungarian algorithm.
  Compute velocity field v_t from displacements.
end for
Set initial state $\mathbf{X}_T = \{P_T, v_{T-1}\}$ and boundary $\mathbf{B}$.
Refine solver f_c using the sequence $\{\mathbf{X}_t\}_{t=1}^{T}$.
Part 2: Predictive Simulation
for $t = T$ to $T + N_{pred} - 1$ **do**
  $\hat{\mathbf{Y}}^* \leftarrow \text{SolveDynamics}(\mathbf{X}_t, \mathbf{B}, f_c)$ ▷ See Algorithm 2
  $\mathbf{X}_{t+1} \leftarrow \text{UPDATE}(\mathbf{X}_t, \hat{\mathbf{Y}}^*)$
end for
Ensure: Predicted fluid states $\{\mathbf{X}_{T+1}, \ldots, \mathbf{X}_{T+N_{pred}}\}$.

The complete end-to-end process of our method is detailed in Algorithm 1. This pipeline is divided into two main parts. Part 1, "Reconstruction and Initialization," outlines the procedure for processing an input video to generate the initial state for the simulation. This includes segmenting the fluid and boundaries, reconstructing them into point clouds, and estimating the initial velocity field. Part 2, "Predictive Simulation," describes how the simulator takes this initial state and iteratively predicts future fluid motion by repeatedly calling the dynamics solver.

The core of the predictive simulation is the dynamics solver step, detailed in Algorithm 2. This algorithm describes an iterative optimization process where, at each time step, the solver refines an initial state update by minimizing the system's energy, which is computed by the neural network f_c. The iteration terminates early if the system energy c falls below a predefined tolerance $c_{threshold}$, indicating that physical constraints, particularly incompressibility, are sufficiently satisfied. The final particle state is then advanced using the optimized update, as shown in Equation (10).

ALGORITHM 2 | Dynamics Solver Step (SolveDynamics).

Require: Current state $\mathbf{X}_t$, boundary $\mathbf{B}$, solver f_c.
Initialize update $\hat{\mathbf{Y}}^{(0)}$ with current velocity from $\mathbf{X}_t$.
for $i \leftarrow 0$ to $N_{max\_iter} - 1$ **do**
  $c, \nabla_{\hat{\mathbf{Y}}} f_c = f_c(\mathbf{X}_t, \mathbf{B}, \hat{\mathbf{Y}}^{(i)})$
  if $c < c_{threshold}$ **then**
    break
  end if
  $\lambda = -\dfrac{c}{\sum \|\nabla_{\hat{\mathbf{Y}}} f_c\|^2 + \varepsilon}$
  $\Delta\mathbf{Y} = -\lambda \nabla_{\hat{\mathbf{Y}}} f_c$
  $\hat{\mathbf{Y}}^{(i+1)} = \hat{\mathbf{Y}}^{(i)} + \Delta\mathbf{Y}$
end for
Ensure: Optimized update $\hat{\mathbf{Y}}^*$.

4 | Experiment

4.1 | Experimental Setup

We evaluated our approach using both simulated and real-world videos. For simulated data, we used the Liquid3D dataset [14] and generated 200 additional scenes, rendering all results with Blender. For real-world data, we captured videos using a smartphone camera at 1280 × 720 resolution and 30 frames per second. All experiments were conducted in PyTorch on an Nvidia RTX 3090 GPU, with about 24 h training.

Our evaluation focused on: (1) the accuracy of point cloud reconstruction from videos, and (2) the accuracy of predicted fluid motion. To assess the effectiveness of our monocular point cloud generation, we evaluated the reconstructed 3D fluid particles using visual inspection, Chamfer Distance (CD), and Earth Mover's Distance (EMD). To prevent gradient vanishing or explosion, we applied orthogonal regularization to the feature maps. The orthogonal loss L_{Orth} is:

$$L_{\text{Orth}} = \sum_{i \in \{1,2,3\}} \| U_i^T U_i - I \|_2^2 \tag{12}$$

The CD loss is:

$$L_{\text{CD}}(P_1, P_2) = \frac{1}{|P_1|} \sum_{x \in P_1} \min_{y \in P_2} \| x - y \|_2^2 + \frac{1}{|P_2|} \sum_{y \in P_2} \min_{x \in P_1} \| y - x \|_2^2 \tag{13}$$

The total training loss is:

$$L = L_{\text{CD}} + \alpha L_{\text{Orth}} \tag{14}$$

where α is set to 100.

We compared the inferred 3D fluid velocities with ground truth (GT) using Mean Squared Error (MSE). The model predicts particle positions for the next two time steps, $t + 1$ and $t + 2$, and we computed the corresponding validation errors, denoted as $Error_{t+1}$ and $Error_{t+2}$. To evaluate inference capability, we used the first frame of the test set to iteratively predict fluid particle positions for the next T frames, and measured the discrepancy

$\hat{\mathbf{X}}^T$, between the predicted positions at frame T, and the GT positions, $\mathbf{X}^T$, as the inference loss $Error_{\mathrm{T}}$:

$$Error_{\mathrm{T}} = \| \mathbf{X}^T - \hat{\mathbf{X}}^T \|_2^2 \tag{15}$$

4.2 | 3D Particle Reconstruction and Refinement

We evaluated our method both visually and quantitatively using GT data. Experiments were conducted on a simulated dataset featuring four representative fluid shapes in a falling scenario from a frontal perspective, comparing our approach with PSGN [33], GX [32], and 3DA [30]. Figure 2a,b show two sample scenes. Visually, our method produces fluid shapes with smoother edges and higher fidelity than competing methods. For quantitative evaluation, Table 1 reports that our method consistently achieves the lowest CD and EMD values across all tested scenarios, albeit with a slight increase in computational time due to the refinement process.

We also compared the 3D fluid velocities estimated by our physical velocity inference module with GT data, using MSE as the metric. Both shortest distance matching and a Hungarian algorithm-based assignment were evaluated. During the initial falling phase, gravity enforces uniform velocities, so we use the average vector sum of all particles as the mean velocity. As shown in Table 2, the Hungarian approach consistently outperforms shortest distance matching. Notably, for the more complex Hemispherical, Donut, and Spherical Wedge shapes, the greedy nature of the shortest distance matching method fails to find correct particle correspondences, leading to anomalously high errors as it does not converge to a physically plausible solution. In contrast, the Hungarian algorithm provides a globally optimal assignment, drastically reducing the error. Mean-based estimation further improves accuracy, particularly for spherical wedge scenarios. Therefore, we adopt the Hungarian algorithm with the mean method for velocity inference in the optimized monocular reconstruction point cloud.

4.3 | Dynamics Solver

We evaluated the inference loss of our method and CConv [14] on a test set with $T = 800$. As shown in Table 3, our approach consistently achieves lower errors across all three metrics. This

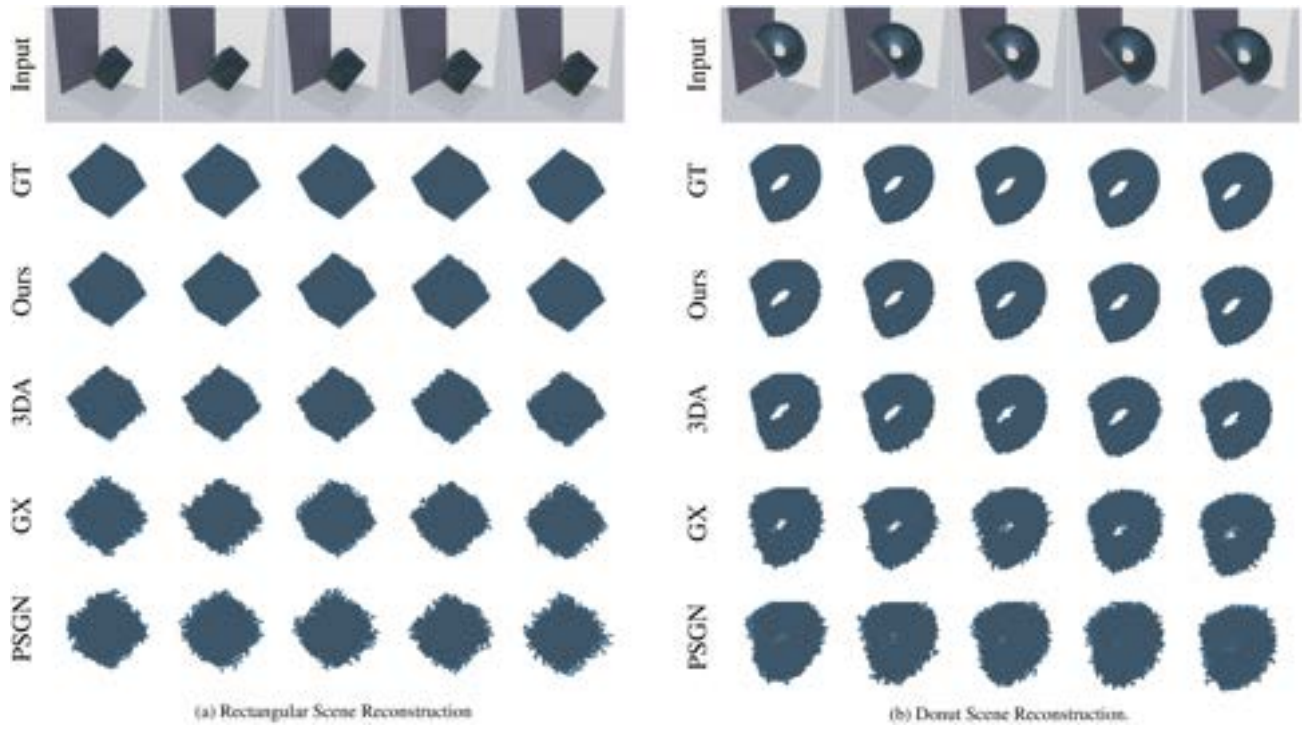

FIGURE 2 | Visual comparison of 3D particle reconstruction for different scenes. (a) Rectangular scene. (b) Donut scene.

TABLE 1 | Evaluation of monocular view 3D reconstruction (CD $\times 10^{-4}$, EMD $\times 10^{-2}$). Time is measured in milliseconds (ms) per frame.

Scene	Hemispherical			Rectangular			Donut			Spherical Wedge		
Method	CD (↓)	EMD (↓)	Time (↓)	CD (↓)	EMD (↓)	Time (↓)	CD (↓)	EMD (↓)	Time (↓)	CD (↓)	EMD (↓)	Time (↓)
GX [32]	10.60	5.17	65.2	10.30	5.26	64.8	10.80	5.64	65.5	10.60	5.35	65.0
PSGN [33]	9.52	4.19	61.7	9.58	4.22	61.9	9.86	4.38	62.1	9.72	4.18	61.5
3DA [30]	7.61	3.12	**52.6**	7.57	3.11	**52.3**	7.69	3.16	**52.8**	7.62	3.06	**52.4**
Ours	**2.97**	**1.59**	71.2	**2.99**	**1.60**	71.0	**3.00**	**1.60**	71.5	**2.98**	**1.58**	71.4

TABLE 2 | MSE of the fluid velocity field under different estimation approaches ($\times 10^{-3}$).

Method	Hemispherical (↓)	Rectangular (↓)	Donut (↓)	Spherical Wedge (↓)
Shortest distance matching	2510	3.40	3880	4030
Hungarian + Non-mean method	3.30	**3.00**	3.70	11.4
Hungarian + Mean method	**3.10**	**3.00**	**3.50**	**4.40**

TABLE 3 | Inference loss evaluation ($\times 10^{-3}$) and time per step (ms).

Method	$Error_{t+1}$ (↓)	$Error_{t+2}$ (↓)	$Error_{\mathrm{T}}$ (↓)	Time (↓)
CConv [14]	0.859	2.42	35.5	**15.2**
Ours	**0.760**	**2.18**	**30.8**	18.5

Note: Lower values are better (↓), and the best results are highlighted in bold.

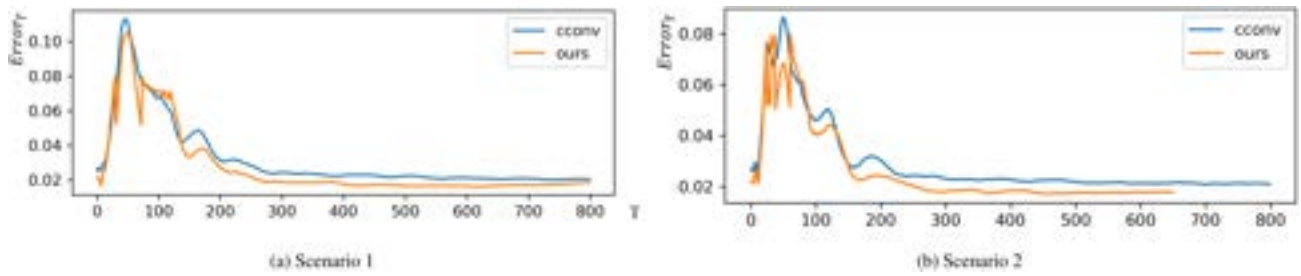

FIGURE 3 | Smoothed inference error curves.

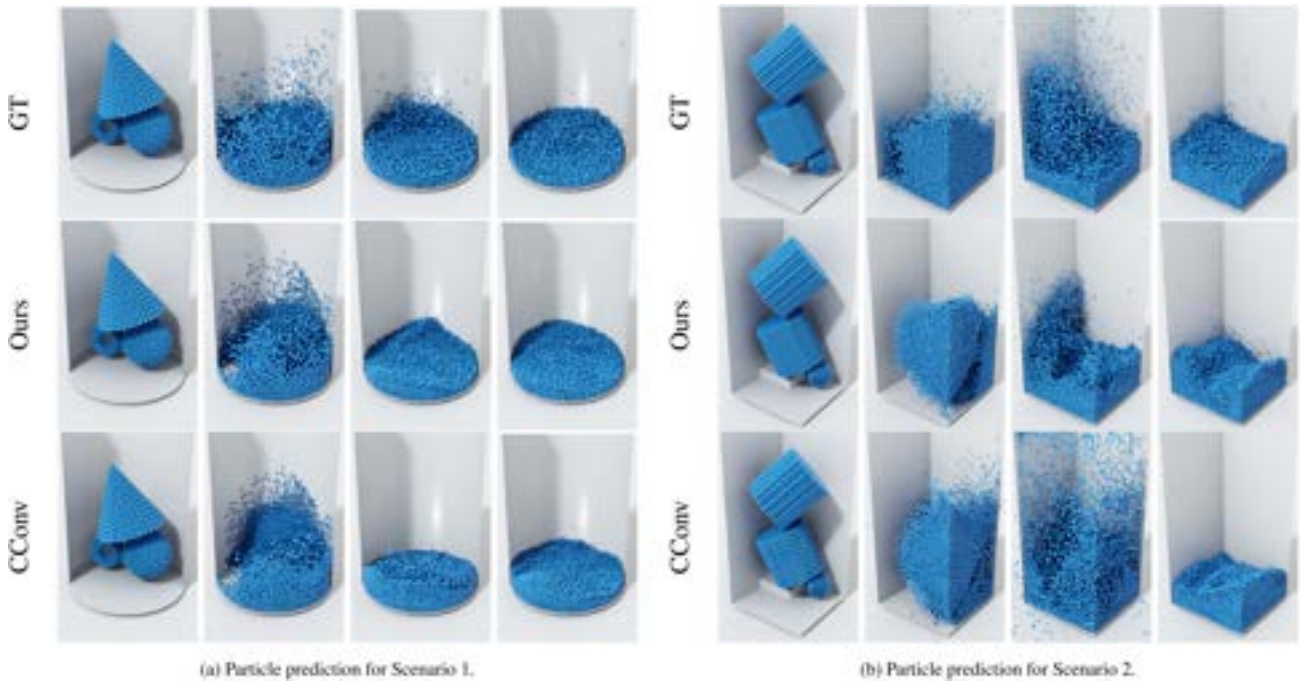

FIGURE 4 | Particle predictions for two scenarios. (a) Scenario 1. (b) Scenario 2.

enhanced performance comes at the cost of a minor increase in computation time per step, as our method incorporates stronger physical constraints.

To further support these results, we selected two scenarios from the test set to compare the inference error curves $Error_{\mathrm{T}}$ and visualize predictions from both methods. Using only the first frame's data, we iteratively predicted the subsequent 800 frames for each scenario. As illustrated in Figure 3, our method maintains notably lower inference errors than CConv, particularly during the critical frames 20–150 when particle collisions with solid boundaries trigger considerable splashing effects.

We also visualized the particles predicted by our method and by CConv, as shown in Figure 4a,b. In Figure 4a, the fluid flow predicted by our method more closely matches the GT, especially in

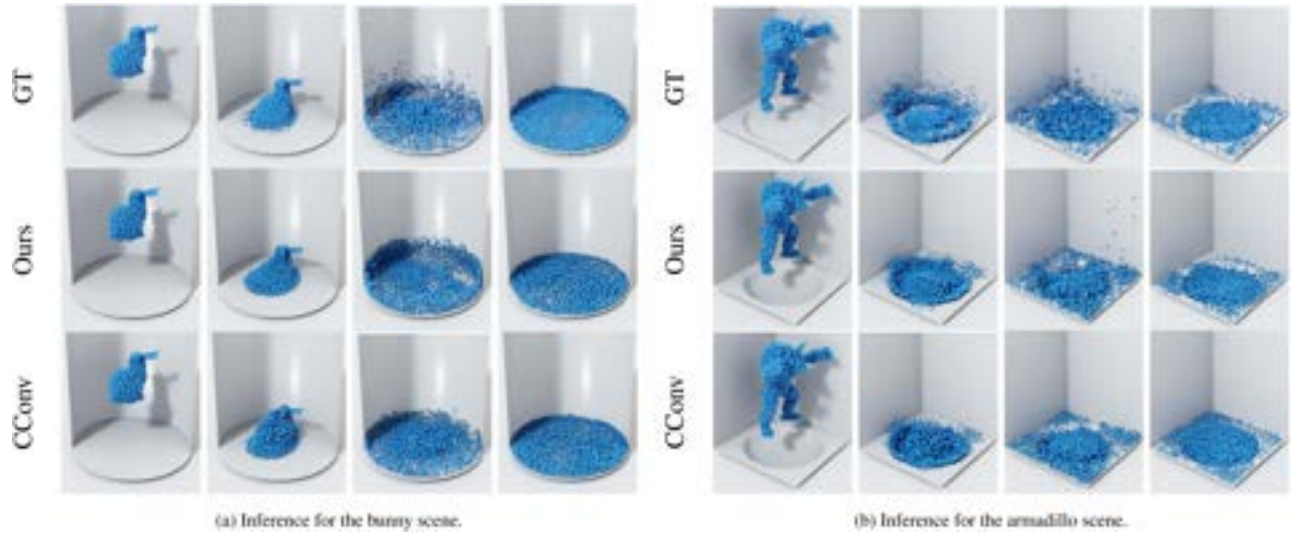

FIGURE 5 | Inference results for two scenes. (a) Bunny scene. (b) Armadillo scene.

the splash region, demonstrating better alignment with the true particle configuration.

Additionally, we tested our model on previously unseen scenes featuring bunny and armadillo fluid shapes with varied solid boundaries (Figure 5a,b). While both methods accurately replicate the initial and final steady states from the GT, CConv exhibits greater divergence from the incompressibility condition during intermediate frames, likely due to weaker physical constraints. For instance, in the bunny scene (Figure 5a), our approach better preserves the fluid shape's features after impact and more accurately captures intense fluid–solid collisions. Similarly, in the armadillo scene (Figure 5b), our method maintains incompressibility more effectively and yields fluid–solid interactions closer to the GT, underscoring its superior predictive accuracy in complex, unseen scenarios.

4.4 | Video-Oriented 3D Fluid Simulation

In this subsection, we validate our method for 3D fluid flow simulation using a simulated cubic fluid block within a complex boundary containing obstacles. First, we present the reconstruction results for the cubic block, optimized by the dynamics-guided inference module. Figure 6 (top row: input images, bottom row: refined reconstructions) shows that our method effectively preserves fluid shapes and maintains well-arranged particles. The dynamics-guided inference module estimates the particle velocity field from the reconstructed data, which the physically constrained continuous convolution solver uses to predict subsequent fluid flow. As shown in Figure 7, the simulation captures varied fluid interactions. Notably, even under vigorous fluid–solid collisions, no particles penetrate the boundaries, demonstrating the effectiveness of our constraints. The simulation also shows particles gradually losing energy through collisions and settling realistically over time.

We further evaluated our method using real video recordings to demonstrate its practical applicability. Figure 8 shows the reconstruction process for a water droplet falling into a circular

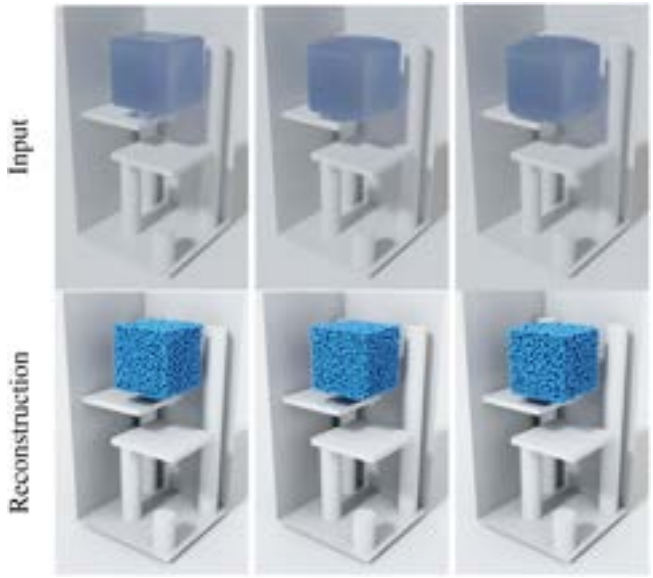

FIGURE 6 | Particle reconstruction from simulation video.

pool (top: input frames, bottom: 3D reconstructions). These reconstructed particles serve as input for our solver to predict subsequent fluid motion. As illustrated in Figure 9, our predictions (top: particle states, bottom: rendered visualizations) consistently adhere to physical constraints, validating the accuracy and robustness of our approach in real-world scenarios.

Overall, these experiments indicate our approach robustly models and predicts fluid behavior from both simulated and real video data, confirming its potential applicability in diverse real-world scenarios.

4.5 | Complex Large-Scaling Scene 3D Fluid Simulation

To further demonstrate the scalability and generalizability of our method, we applied it to a complex large-scale real-world fluid

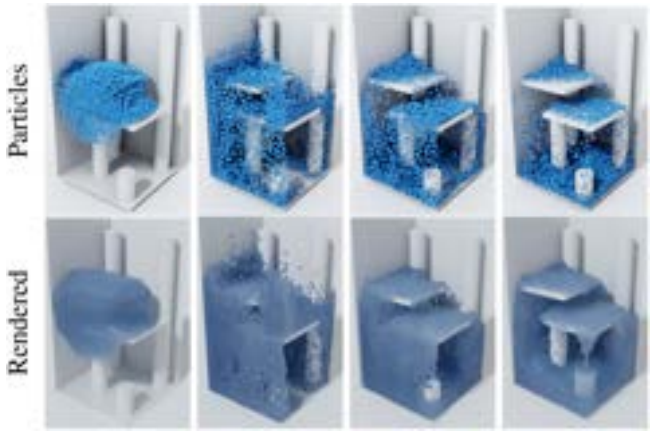

FIGURE 7 | Fluid prediction after simulation video in Figure 6.

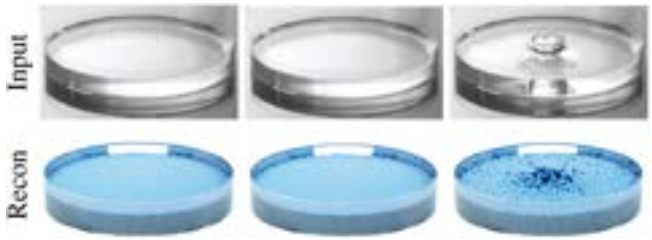

FIGURE 8 | Particle reconstruction from real video.

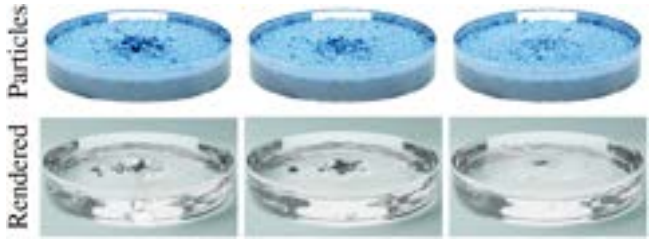

FIGURE 9 | Fluid prediction after real video in Figure 8.

scene, lava flow. The scene is sampled from a video titled "Volcano & Lava," uploaded to YouTube by Scenic Relaxation. It was recorded via a single moving aerial camera capturing volcanic eruptions. The video has a resolution of 3180×2160 and a frame rate of 30 fps. The selected segment lasts about 15 s, as shown in the first row of Figure 10.

We first reconstructed the moving lava particles from the video using our Kinetic Estimation and Optimization module. In this scene, because the camera moves continuously over time, the module may occasionally fail to correctly pair particles across frames. Therefore, we apply a Statistical Outlier Removal filter to eliminate outliers in the reconstructed particles. The particles corresponding to the video frames are shown in the second row of Figure 10, where they remain well aligned and preserve the overall shape of the lava flow. Since we obtain a sequence of reconstructed particles from the video, we can further refine the pretrained model described in Section 4.4 for this specific scene to better capture its unique dynamics. In this scene, the viscosity of lava is significantly higher than that of water, leading to slower flow and more pronounced interactions with the terrain. By fine-tuning our model on the reconstructed lava particles, we adapt it to these specific physical properties and improve prediction accuracy.

Then, we predict the subsequent fluid flow after the video segment, using our refined continuous convolution solver, to predict the scene when the lava eruption stops. As shown in the third row of Figure 10, when the eruption ceases, the last ejected lava impacts the crater rim and creates a splashing effect on the lava surface. The lava level slowly drops and gradually flows towards the base of the mountain. The lava exhibits significant viscosity, causing it to flow slowly down the inner walls of the volcano with a noticeable adhesion effect.

Despite the complexity and scale of the scene, our approach effectively captures the intricate flow patterns and interactions of the lava, demonstrating generalizability and its capability to handle large-scale fluid simulations in real-world scenarios.

5 | Discussion

In this section, we conduct a series of experiments to demonstrate the efficacy of each component of our simulator. We then present end-to-end experiments on video inputs to demonstrate the ability of our method to reconstruct and simulate fluid flows from videos. Moreover, we apply our method to a complex, large-scale real-world fluid scene (lava flow) to further demonstrate its scalability and generalizability.

Although our method can effectively capture fluids from videos and predict their physical properties, several limitations remain. First, our boundary handling assumes static and rigid environments; consequently, scenarios with moving containers or deformable obstacles are currently out of scope. Second, the overall performance is bounded by the quality of upstream video-to-3D reconstruction and segmentation and remains sensitive to viewing conditions. Missing regions due to occlusions, imperfect masks, or depth ambiguities in monocular capture can propagate to velocity estimation and subsequently affect long-horizon rollouts. Third, motion estimation relies on frame-to-frame correspondences and optical-flow cues; under high-speed motion or low frame rates, correspondences can become ambiguous and may yield temporal jitter. Moreover, MLS-based refinement introduces a trade-off between robustness and detail preservation.

Future work will focus on mitigating these limitations by improving temporally consistent reconstruction and correspondence under fast motion, incorporating additional geometric cues (e.g., multi-view or depth when available), and extending the framework to support dynamic boundaries via per-frame boundary reconstruction and tighter fluid–boundary coupling. We will also explore adaptive refinement strategies that better preserve fine-scale structures while maintaining robustness to reconstruction noise.

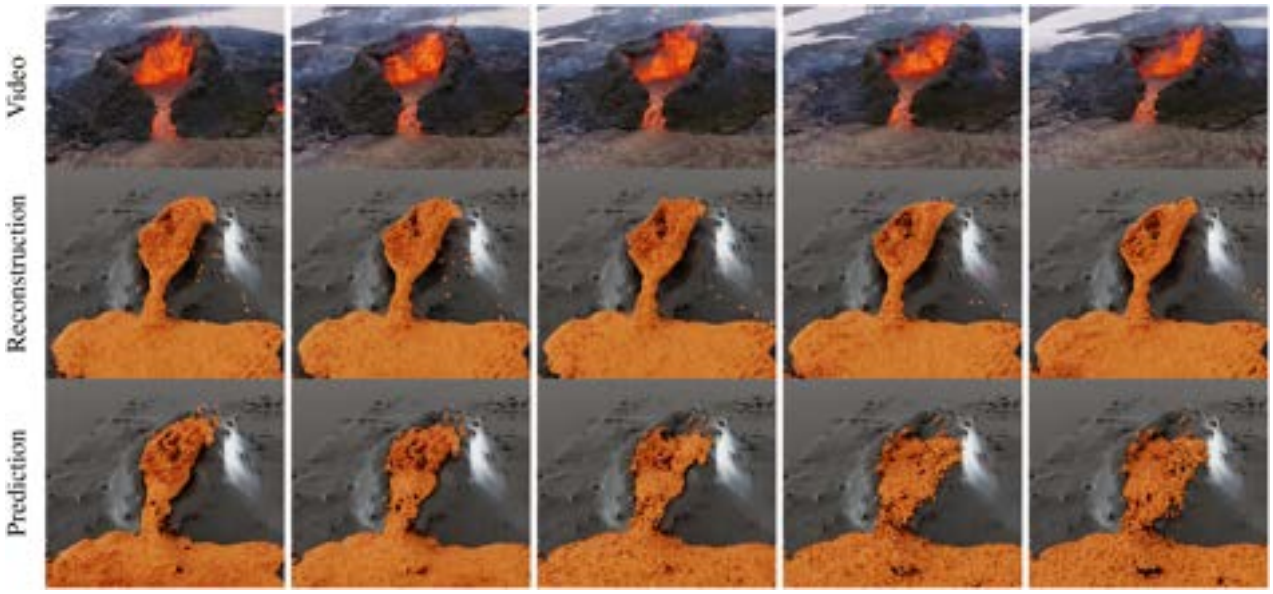

FIGURE 10 | Particle reconstruction and fluid prediction on the real-world Lava test set. The first row shows the original video, the second row shows corresponding fluid particles reconstructed from the video, and the third row shows fluid predictions after the video.

6 | Conclusion

In this paper, we present a neural fluid simulator that integrates visual priors from 2D image sequences with a physically constrained continuous convolution solver. Extensive experiments on both simulated and real-world datasets demonstrate superior accuracy in fluid motion prediction. By effectively combining visual data with physical constraints, our approach addresses the limitations of both traditional physics-based and purely data-driven methods, providing a robust and efficient solution for real-world fluid simulations.

Author Contributions

Feilong Du was responsible for the methodology, formal analysis, and writing of the original draft. Xiaojuan Ban contributed to funding acquisition, as well as the review and editing of the manuscript. Yuhang Xu handled the code implementation, experimental validation, and visualization. Angelos Chatzimparmpas provided supervision and also contributed to the manuscript's review and editing. Yalan Zhang guided the investigation, methodology, and funding acquisition. All authors have read and approved the final version of the manuscript.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62306032), the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2022A1515110350), and the Interdisciplinary Research Project for Young Teachers of USTB (Grant No. FRF-IDRY-GD24-003). Computational resources were partially provided by the MAGICOM Platform of the Beijing Advanced Innovation Center for Materials Genome Engineering. We sincerely thank all funding agencies and institutions for their support.

Funding

This work was supported by the National Natural Science Foundation of China, 62306032; Basic and Applied Basic Research Foundation of Guangdong Province, 2022A1515110350; University of Science and Technology Beijing, FRF-IDRY-GD24-003.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. J. Stam, "Stable Fluids," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (ACM Press/Addison-Wesley Publishing Co, 1999), 121–128.

2. T. Ciano, M. Ferrara, M. Babanezhad, A. Khan, and A. Marjani, "Prediction of Velocity Profile of Water Based Copper Nanofluid in a Heated Porous Tube Using CFD and Genetic Algorithm," *Scientific Reports* 11, no. 1 (2021): 10623.

3. Y. Li, T. Lin, K. Yi, et al., "Visual Grounding of Learned Physical Models," *PMLR* (2020): 5927–5936.

4. L. Prantl, B. Ummenhofer, V. Koltun, and N. Thuerey, "Guaranteed Conservation of Momentum for Learning Particle-Based Fluid Dynamics," *Advances in Neural Information Processing Systems* 35 (2022): 6901–6913.

5. A. J. Chorin, "Numerical Solution of the Navier–Stokes Equations," *Mathematics of Computation* 22, no. 104 (1968): 745–762, https://doi.org/10.1090/S0025-5718-1968-0242392-2.

6. J. H. Ferziger, M. Perić, and R. L. Street, *Computational Methods for Fluid Dynamics*, 4th ed. (Springer, 2020).

7. X. Xie, Y. Gao, F. Hou, T. Cheng, A. Hao, and H. Qin, "Fluid Inverse Volumetric Modeling and Applications From Surface Motion," *IEEE Transactions on Visualization and Computer Graphics* 31, no. 3 (2025): 1785–1801.

8. M. Okabe, Y. Dobashi, K. Anjyo, and R. Onai, "Fluid Volume Modeling From Sparse Multi-View Images by Appearance Transfer," *ACM Transactions on Graphics* 34, no. 4 (2015): 1–10.

9. K. S. Bhat, S. M. Seitz, J. Popović, and P. K. Khosla, *Computing the Physical Parameters of Rigid-Body Motion From Video* (Springer, 2002), 551–565.

10. S. Bayraktar, U. Güdükbay, and B. Özgüç, "Particle-Based Simulation and Visualization of Fluid Flows Through Porous Media," *Journal of Visualization* 13, no. 4 (2010): 327–336, https://doi.org/10.1007/s12650-010-0041-2.

11. S. Bayraktar, U. Güdükbay, and B. Özgüç, "GPU-Based Neighbor-Search Algorithm for Particle Simulations," *Journal of Graphics, GPU, & Game Tools* 14, no. 1 (2009): 31–42.

12. U. Güdükbay, S. Bayraktar, Ç. Koca, and B. Özgüç, "Particle-Based Simulation of the Interaction Between Fluid and Knitwear," *Signal, Image and Video Processing* 8, no. 3 (2014): 415–422, https://doi.org/10.1007/s11760-012-0308-2.

13. A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to Simulate Complex Physics With Graph Networks," in *119 of Proceedings of Machine Learning Research* (PMLR, 2020), 8459–8468.

14. B. Ummenhofer, L. Prantl, N. Thuerey, and V. Koltun, *Lagrangian Fluid Simulation With Continuous Convolutions* (ICLR, 2020).

15. T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, "Learning Mesh-Based Simulation With Graph Networks," in *International Conference on Learning Representations* (Cornell University, 2021).

16. Z. Li, N. B. Kovachki, K. Azizzadenesheli, et al., "Fourier Neural Operator for Parametric Partial Differential Equations," in *International Conference on Learning Representations* (Cornell University, 2021).

17. M. Babanezhad, I. Behroyan, A. T. Nakhjiri, A. Marjani, and S. Shirazian, "Performance and Application Analysis of ANFIS Artificial Intelligence for Pressure Prediction of Nanofluid Convective Flow in a Heated Pipe," *Scientific Reports* 11, no. 1 (2021): 902.

18. A. Franz, B. Solenthaler, and N. Thuerey, "Global Transport for Fluid Reconstruction With Learned Self-Supervision," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Cornell University, 2021), 1632–1642.

19. P. Ma, P. Y. Chen, B. Deng, et al., "Learning Neural Constitutive Laws from Motion Observations for Generalizable PDE Dynamics," in *Proceedings of the 40th International Conference on Machine Learning* (PMLR, 2023), 23279–23300.

20. F. Regazzoni, S. Pagani, M. Salvador, Dede' L, and A. Quarteroni, "Learning the Intrinsic Dynamics of Spatio-Temporal Processes Through Latent Dynamics Networks," *Nature Communications* 15, no. 1 (2024): 1834.

21. X. Li, Y. Cao, M. Li, Y. Yang, C. Schroeder, and C. Jiang, "Plasticitynet: Learning to Simulate Metal, Sand, and Snow for Optimization Time Integration," *Advances in Neural Information Processing Systems* 35 (2022): 27783–27796.

22. M. Chu, L. Liu, Q. Zheng, et al., "Physics Informed Neural Fields for Smoke Reconstruction With Sparse Data," *ACM Transactions on Graphics* 41, no. 4 (2022): 1–14.

23. W. Si, J. Qin, Z. Chen, X. Liao, Q. Wang, and P. A. Heng, "Thin-Feature-Aware Transport-Velocity Formulation for SPH-Based Liquid Animation," *IEEE Transactions on Multimedia* 20, no. 11 (2018): 3033–3044.

24. M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," *Journal of Computational Physics* 378 (2019): 686–707.

25. S. Xiao, C. Tong, Q. Zhang, Y. Cen, F. W. Li, and X. Liang, "Laplacian Projection Based Global Physical Prior Smoke Reconstruction," *IEEE Transactions on Visualization and Computer Graphics* (2024).

26. J. Li, Y. Gao, J. Dai, S. Li, A. Hao, and H. Qin, "MPMNet: A Data-Driven MPM Framework for Dynamic Fluid-Solid Interaction," *IEEE Transactions on Visualization and Computer Graphics* 30, no. 8 (2024): 4694–4708.

27. H. X. Yu, Y. Zheng, Y. Gao, Y. Deng, B. Zhu, and J. Wu, "Inferring Hybrid Neural Fluid Fields From Videos," in *Proceedings of the 37th International Conference on Neural Information Processing Systems* (Curran Associates Inc, 2023), 2777.

28. Z. Li and A. B. Farimani, "Graph Neural Network-Accelerated Lagrangian Fluid Simulation," *Computers and Graphics* 103 (2022): 201–211.

29. S. Xiong, X. He, Y. Tong, Y. Deng, and B. Zhu, "Neural Vortex Method: From Finite Lagrangian Particles to Infinite Dimensional Eulerian Dynamics," *Computers and Fluids* 258 (2023): 105811.

30. X. Wen, J. Zhou, Y. S. Liu, H. Su, Z. Dong, and Z. Han, "3D Shape Reconstruction From 2D Images With Disentangled Attribute Flow," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Cornell University, 2022), 3793–3803.

31. N. Ravi, V. Gabeur, Y. T. Hu, et al., "SAM 2: Segment Anything in Images and Videos," in *The Thirteenth International Conference on Learning Representations* (Cornell University, 2025).

32. D. Nguyen, S. Choi, W. Kim, and S. Lee, "GraphX-Convolution for Point Cloud Deformation in 2D-to-3D Conversion," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (IEEE, 2019), 8627–8636.

33. H. Fan, H. Su, and L. Guibas, "A Point Set Generation Network for 3D Object Reconstruction From a Single Image," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2017), 2463–2471.

Supporting Information

Additional supporting information can be found online in the Supporting Information section. **Video S1:** Demonstration of the proposed Neural Fluid Simulator in a complex large-scale lava scene. We extract spatiotemporally continuous fluid particles from the input video and predict the subsequent simulation.