

---

# Hybrid Neural-MPM for Interactive Fluid Simulations in Real-Time

---

Anonymous Author(s)

Affiliation

Address

email

## Abstract

1 We propose a neural physics system for real-time, interactive fluid simulations.  
2 Traditional physics-based methods, while accurate, are computationally intensive  
3 and suffer from latency issues. Recent machine-learning methods reduce com-  
4 putational costs while preserving fidelity; yet most still fail to satisfy the latency  
5 constraints for real-time use and lack support for interactive applications. To bridge  
6 this gap, we introduce a novel hybrid method that integrates numerical simulation,  
7 neural physics, and generative control. Our neural physics jointly pursues low-  
8 latency simulation and high physical fidelity by employing a fallback safeguard to  
9 classical numerical solvers. Furthermore, we develop a diffusion-based controller  
10 that is trained using a reverse modeling strategy to generate external dynamic  
11 force fields for fluid manipulation. Our system demonstrates robust performance  
12 across diverse 2D/3D scenarios, material types, and obstacle interactions, achieving  
13 real-time simulations at high frame rates ( $-11 \sim 29\%$  latency) while enabling  
14 fluid control guided by user-friendly freehand sketches. We present a significant  
15 step towards practical, controllable, and physically plausible fluid simulations for  
16 real-time interactive applications. We promise to release both models and data  
17 upon acceptance.

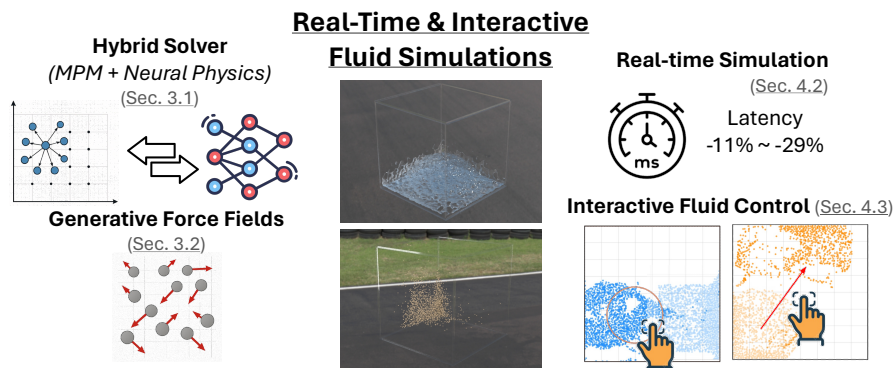


Figure 1: We target real-time, interactive fluid simulations. Our hybrid solver integrates a numerical simulator and neural physics (Section 3.1), enabling real-time simulation (Section 4.2). In addition, we generate external force fields (Section 3.2) to support users to control fluids interactively via freehand sketches (Section 4.3).

## 1 Introduction

19 Modeling fluid behavior is essential for advancing diverse engineering fields, including entertain-  
20 ment [30], urban planning [2], fashion design [33], and virtual reality (VR) [29]. Moreover, con-

21 trollability, aiming to instruct movements and shapes of fluids, is also a very important attribute for  
 22 volumetric effects, character animations, and fluid-solid coupling [25]. Realizing compelling and  
 23 interactive physics simulations in real-time has been the long-standing objective for years in order to  
 24 deliver transformative user experiences.

25 Traditional simulation methods, though powerful, often demand significant implementation efforts and  
 26 computational costs [5]. Recent neural physics and machine learning approaches present a promising  
 27 path forward by learning from data, delivering transformative changes for use cases such as fluid  
 28 interactions and animations [27]. However, fidelity and latency in these neural-based methods are not  
 29 well-balanced. Moreover, most methods only focus on the accuracy of non-interactive applications,  
 30 and their computational complexity still remains generally high for real-time scenarios [4].

31 Motivated by the above challenges, we ask two scientific questions:

- 32 ***Q1:** Can neural physics accelerate real-time fluid simulations and interactions?*

***Q2:** Can neural physics and generative methods be optimized for interactive fluid control?*

33 We aim to explore a novel paradigm: neural physics for interactive simulations in real-time (Figure 1).  
 34 We provide affirmative answers. The core idea is to **proactively marry the strengths of numerical**  
 35 **simulation (high fidelity), neural physics (low latency), and generative control (interactivity)**  
 36 to deliver authentic and diverse fluid simulations. Specifically, neural physics is responsible for  
 37 significantly low-latency fluid simulation with tolerant errors, and numerical simulation will serve  
 38 as a fallback solution when fluid dynamics is increasingly complex. Furthermore, to make fluid  
 39 animation compatible with user-friendly control, we introduce another diffusion-based controller to  
 40 generate external force fields to assist manipulations. We summarize our contributions below:

- 41 1. We improve the **error-latency trade-off** of fluid simulation. First, to accelerate neural physics, we  
 42 seek to build our graph neural network at low spatiotemporal resolution without substantial degrada-  
 43 tion in simulation accuracy (Section 3.1.1). Second, to preserve simulation fidelity and avoid  
 44 error accumulation during unrolling, we make our neural physics hybrid with a safeguard condition  
 45 and fallback mechanism to the classic MPM (Material Point Method) algorithm (Section 3.1.2).
- 46 2. We further aim to support **users’ flexible freehand sketches** that specify desired trajectories or  
 47 shapes of fluid particles to be controlled. To this end, our novel reverse simulation strategy enables  
 48 the automated generation of realistic fluid control data (Section 3.2.2), which is used to train our  
 49 diffusion-based generative controller (Section 3.2.3).
- 50 3. Across **diverse scenarios** (2D/3D, particle materials, presence of rigid obstacles, see Table 2),  
 51 our hybrid simulator can **significantly accelerate simulations** ( $-11 \sim 29\%$  latency) while  
 52 maintaining low errors (Section 4.2), and can **control fluid particles to align with user sketches**  
 53 (Section 4.3), paving the way for promising advances towards engaging interactive simulations in  
 54 real-time.

## 55 2 Background

56 We first introduce the necessary components on which our method is built, and how they can be made  
 57 real-time and controllable in Section 3.

### 58 2.1 Fluid Simulations with Material Point Method (MPM)

59 The Material Point Method (MPM) [17, 14, 13, 15] is a hybrid Eulerian-Lagrangian numerical  
 60 technique for simulating complex interactions between solid and fluid materials, especially under  
 61 large deformations and topological changes (snow, landslides, cloth, etc.). It extends the FLuids-  
 62 Implicit-Particle (FLIP) [3] from Computational Fluid Dynamics (CFD) to solid mechanics by  
 63 representing materials as a set of Lagrangian particles that carry mass, velocity ( $\dot{\mathbf{p}}_{i,t}$ ), position ( $\mathbf{p}_{i,t}$ ),  
 64 and possible internal states. These particle quantities are first transferred to a background Eulerian  
 65 grid using a particle-to-grid mapping (p2g). The equations of motion are then solved on this grid,  
 66 after which updated values are mapped back to particles through grid-to-particle transfer (g2p). The particle  
 67 positions ( $\mathbf{p}$ ) are then advanced using the updated velocities ( $\dot{\mathbf{p}}$ ), e.g.,  $\mathbf{p}_{i,t+1} = \mathbf{p}_{i,t} + \Delta t \cdot \dot{\mathbf{p}}_{i,t+1}$ .

## 2.2 GNN-based Neural Physics for Particle Simulations

We denote the state of particle  $i$  at time step  $t$  as  $\mathbf{x}_{i,t}$  (position  $\mathbf{p}$ , velocity  $\dot{\mathbf{p}}$ , acceleration  $\ddot{\mathbf{p}}$ , etc.), and the state of  $N$  particles as  $\mathbf{X}_t = [\mathbf{x}_{1,t}, \dots, \mathbf{x}_{N,t}]$ . A *simulator*  $s$  maps  $T_{\text{in}}$  input states to causally consequent future states, and can iteratively compute  $\mathbf{X}_{t_{\text{in}}+1} = s(\mathbf{X}_{t_1}, \mathbf{X}_{t_2}, \dots, \mathbf{X}_{t_{\text{in}}})$  to simulate a rollout trajectory. Following [27], our learnable simulator  $s_\theta$  adopts a particle-based representation of the physical system, which can be viewed as message-passing via a graph neural network (GNN).

**Input.** Our neural physics simulator  $s_\theta$  takes the input of particle  $i$  as: the position, a sequence of  $T_{\text{in}} = 6$  previous velocities, and features that capture static material properties (e.g., water, sand, rigid, boundary particle), i.e.,  $\mathbf{x}_{i,t_k-T_{\text{in}}:t_k} = [\dot{\mathbf{p}}_{i,t_k-T_{\text{in}}+1}, \dots, \dot{\mathbf{p}}_{i,t_k}, \mathbf{f}_i]$  at time step  $t_k$  (Figure 2).

**GNN Design.** We first build the initial graph  $G^{(0)}$  by assigning a node to each particle and connecting particles as edges within a fixed “connectivity radius”  $R$ . The edge embeddings are learned from relative positional displacement and the magnitude  $\mathbf{r}_{i,j} = [(\mathbf{p}_i - \mathbf{p}_j), \|\mathbf{p}_i - \mathbf{p}_j\|]$ . Our neural physics consists of a stack of  $L = 10$  GNN layers. The decoder predicts the per-particle acceleration,  $\ddot{\mathbf{p}}_i$ . The training loss is the particle-level

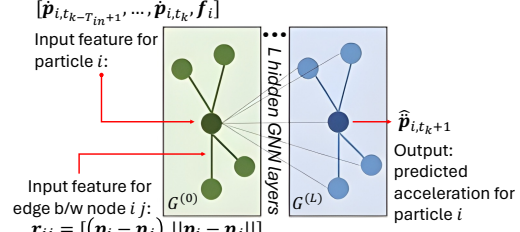


Figure 2: GNN as our neural physics simulator.

RMSE $\ddot{\mathbf{p}} \equiv \frac{1}{N} \sum_{i=1}^N \frac{\|\hat{\ddot{\mathbf{p}}}_i - \ddot{\mathbf{p}}_i\|_2}{\|\ddot{\mathbf{p}}_i\|_2}$ , where  $\hat{\ddot{\mathbf{p}}}_i$  is the predicted acceleration from  $s_\theta$ . The future position and velocity are updated using an Euler integrator. See Appendix A for further details.

## 3 Methods

We aim at real-time fluid simulations (Section 3.1) with interactive control (Section 3.2). Our method is overviewed in Figure 3.

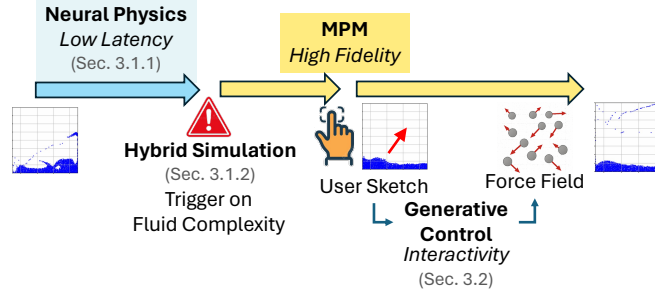


Figure 3: Method Overview. To achieve real-time simulations, we cut latency by learning neural physics at a coarse spatiotemporal resolution, while safeguarding fidelity by automatically falling back to an MPM solver when complex fluid phenomena arise (Section 3.1). For interactive control, we train a diffusion-based generative model that infers external force fields directly from user sketches (Section 3.2).

### 3.1 Hybrid Real-Time Fluid Simulation

#### 3.1.1 Learning Real-Time Neural Physics at Low Spatiotemporal Resolution

To accelerate the simulation, we train our neural physics at low spatiotemporal resolution. As shown in Figure 4, we consider learning the neural physics on simulations with both a downsampled number of particles (ratio  $r_p \in (0, 1)$ ) and also with a larger time step (i.e. coarser temporal discretization rate  $r_t \in \mathbb{N}, r_t > 1$ ).

However, a key pitfall is that once the number of particles is downsampled ( $N_h$  particles are merged via clustering into  $N_l$ , see Appendix B), we will lose the particle-wise correspondence, i.e.,  $\hat{\ddot{\mathbf{p}}}_i$  ( $i \in [1, N_l]$ ) and  $\ddot{\mathbf{p}}_j$  ( $j \in [1, N_h]$ ) cannot align in the particle-level RMSE $\ddot{\mathbf{p}}$  (Section 2.2). As a result, RMSE $\ddot{\mathbf{p}}$  can no longer

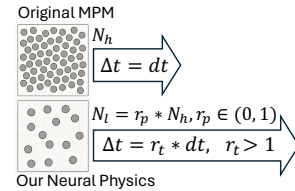


Figure 4: Our neural physics accelerates simulations by learning and inferring at low spatial ( $N_l$  num. particles) and temporal ( $\Delta t$  time steps) resolutions, with downsampling ratios as  $r_p, r_t$ .

quantify the simulation’s fidelity to the ground truth of the original spatial resolution [16]. To mitigate this issue, we use normalized grid-level RMSE  $\tilde{m} \equiv \frac{1}{N} \sum_{i=1}^N \frac{\|\hat{\tilde{m}}_i - \tilde{m}_i\|_2}{\|\tilde{m}_i\|_2}$  as the evaluation metric, which essentially quantifies the mass distribution.  $\tilde{m}$  is the normalized grid mass ( $\tilde{m}_i = \frac{m_i}{\sum_{i=1}^N m_i}$ ) converted from particles to the grid via p2g, and  $\hat{\tilde{m}}$  is the prediction by  $s_\theta$ .  $\tilde{m}_i$  and  $\hat{\tilde{m}}$  share the same grid size but can represent mass distributions from different resolutions (number of particles). During training, we continue to optimize the surrogate loss RMSE $_{\tilde{p}}$  at the low spatial resolution, thereby avoiding additional p2g operations.

In Figure 6 (a-c), we can see that by tuning spatiotemporal downsampling ratios  $r_p, r_t$ , we can improve the trade-off between simulation errors and latency. Based on this ablation study, we will choose  $r_p = 1/1.75$  and  $r_t = 2$ . With this configuration, on Water 2D, we can reduce the latency of the original neural physics ( $r_p = r_t = 1$ ) by over 78.8% (from 1.954ms to 0.4048ms).

### 3.1.2 Hybrid Simulator with Safeguard

Traditional numerical methods like MPM offer high fidelity but are computationally expensive. Though inferring neural physics at a low spatiotemporal resolution enables significantly faster simulations, it often comes at the cost of increased simulation errors. For example, most dots in Figure 6 (a) and (b) are above the original neural physics ( $r_p = r_t = 1$ ) and MPM.

To fuse the strengths of both approaches, we make our simulator hybrid. We primarily leverage neural physics for fast updates, but incorporate a safeguard mechanism to fall back to MPM in challenging scenarios and to empirically ensure simulation quality:

$$X_{t+1} = \begin{cases} \text{Neural Physics Update} & \text{if update is "good"} \\ \text{Fallback to MPM Update} & \text{otherwise.} \end{cases} \quad (1)$$

**Fluid Complexity Measures.** Intuitively, when the current fluid dynamics is simple, the neural physics should generalize well. In contrast, if the particles behave chaotically, their dynamics become out-of-distribution (OOD) samples that neural physics may be able to generalize. We thus trigger the fallback condition based on the complexity of the current fluid dynamics being simulated by neural physics. Moreover, the safeguard should be computationally cheap, since we need to densely monitor them during the simulation of neural physics.

Specifically, we consider the cosine similarity of per-particle acceleration over a window of history (window size as  $\delta t = 10$  steps by default):  $\frac{1}{N} \sum_i \cos(\ddot{\mathbf{p}}_{i,t-2\delta t:t-\delta t}, \ddot{\mathbf{p}}_{i,t-\delta t:t})$ . In contrast, we also tried to monitor the divergence of particles’ velocity [9], which is also used to quantify the quality of incompressible fluid simulations in previous works. However, its computation is significantly more expensive due to the use of finite difference methods, resulting in increased latency. We show the negative correlation between this cosine similarity and the neural physics simulation error in Figure 5, which indicates that whenever particles’ accelerations start diverging, we should fall back to MPM.

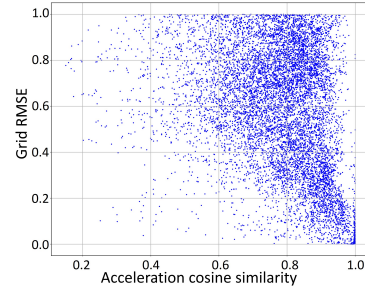


Figure 5: Negative correlation between “cosine similarity of particle accelerations over frames” vs. “simulation errors of neural physics”. Scenario: Water 2D. Spearman correlation: -0.3902.

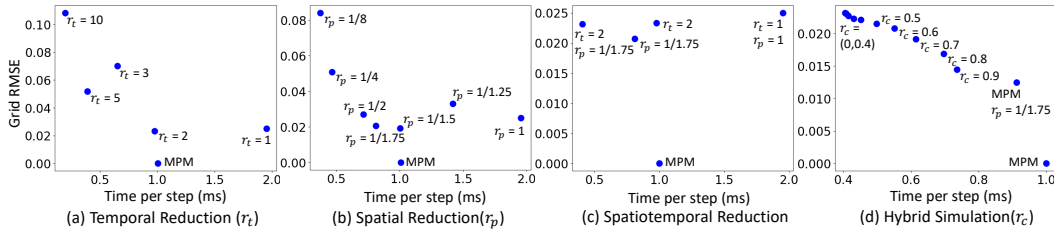


Figure 6: Ablation studies of the trade-off between grid-level RMSE $_{\tilde{m}}$  vs. simulation latency. Left to right: temporal reduction  $r_t$  (train neural physics with reduced particles  $N_l$ ), spatial reduction  $r_p$  (train neural physics with larger time step  $\Delta t$ ), spatiotemporal reduction (combine  $r_t = 2$  and  $r_p = 1/1.75$ ), and hybrid with MPM (at  $r_p = 1/1.75$ ) with different thresholds  $r_c$ . Scenario: Water 2D.



### Triggering MPM by Fluid Complexity.

With our fluid complexity metric, we need to trigger the MPM fallback mechanism in principle. In Table 1, we see that when increasing our threshold  $r_c$  (i.e. MPM will be more frequently triggered), the simulation fidelity will be corrected by MPM (RMSE $_{\tilde{m}}$  is improved), and the latency will increase due to heavy computations of MPM. Thus, we need to choose a threshold  $r_c$  such that we can improve our trade-off between RMSE $_{\tilde{m}}$  and latency. In Figure 6 (d), we tune this threshold, and choose  $r_c = 0.8$  to balance the improvements over RMSE $_{\tilde{m}}$  and latency.

We finalize our hybrid solver using this threshold. In Figure 7, we demonstrate trajectories simulated by the original neural physics and our hybrid solver (from the same initial condition, of the same number of steps  $T$ ). Although the original neural physics ( $r_p = r_t = 1$ ) shows lower rollout error in the early stage (black curve, due to simulation at high resolution), it quickly accumulates long-term errors. In contrast, after triggering the fallback to MPM (yellow areas), our error is suppressed and we finish the simulation much faster. Thus, our hybrid solver improves both rollout RMSE $_{\tilde{m}}$  and latency.

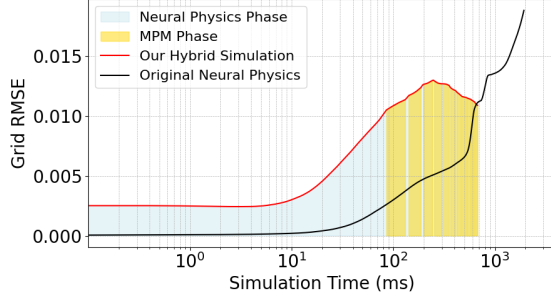


Figure 7: Error trajectories during simulation (Water 2D). Simulating the same number of steps ( $T = 1000$ ), our hybrid solver takes significantly less time (676.4ms) than the original neural physics (1931.1ms), and the final error is also reduced (grid RMSE $_{\tilde{m}}$ ) (0.0109 vs. 0.0188).

**Table 1:** Grid RMSE $_{\tilde{m}}$  vs. time per step with hybrid simulations triggered by different thresholds (Water 2D).

Threshold $r_c$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Grid RMSE $_{\tilde{m}}$	0.0232	0.0230	0.0227	0.0223	0.0221	0.0215	0.0208	0.0192	0.0169	0.0144
Time per step (ms)	0.4048	0.4081	0.4147	0.4301	0.4516	0.4977	0.5509	0.6137	0.6966	0.7356

## 3.2 Interactive Fluid Control

### 3.2.1 Use Cases

Fluid control is essential in computer graphics, where liquid animations convey expressive, story-driven scenes and key visual ideas like splash shapes or motion [38]. Manual fluid control produces unnatural effects and forces artists to rely on slow, trial-and-error methods [22]. This underscores the need for intuitive tools that let users shape visuals directly, without complex physics. Yet, achieving the desired appearance of fluid control remains difficult. Fluid dynamics are intrinsically chaotic and unpredictable. Setup and tuning of fluid control is tedious and repetitive. Moreover, recording real fluid motion is also expensive and hard to customize.

In our paper, we mainly consider the following use case: during a fluid simulation, a user would like to draw a simple sketch and provide it as a control signal, following which the fluid particles should move, as shown in Figure 8 bottom panel. However, how to *artistically* manipulate fluid particles to follow the user’s sketch should be automatically designed by our system.

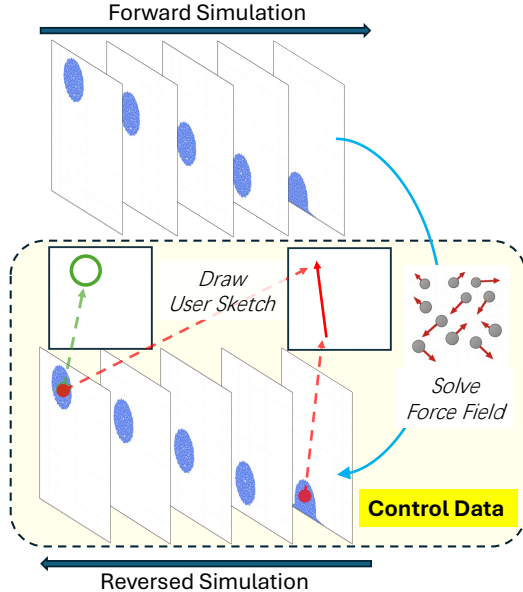


Figure 8: We prepare our training data for generative control via solving external force fields that can reverse a forward simulation. We also prepare user sketches (arrow, ellipse) that depict movements or target shapes of particles (see Appendix B for implementation details)

### 3.2.2 Data Generation via Reversed Simulation

The key to making our fluid control possible is to automatically collect training data in principle. Specifically, we have two highly nontrivial sub-tasks: 1) Design a large number of diverse scenarios of fluid particles with artistic control effects (i.e. fluid particles move along a desired direction or fill a pre-defined shape, in an organized manner, rather than in a chaotic manner); 2) Solve a spatiotemporal external force field that will be applied to the particles, such that the artistic control effect can be fulfilled driven by the composition of gravity, particle interactions, and the proposed force field.

We address these challenges with a reverse simulation strategy. The core idea is to solve the required force fields that can reverse the fluid dynamics of artistic effects. We have the following steps:

**1) Forward Simulation.** We randomly simulate a trajectory of fluid dynamics  $\mathbf{X} = (X_1, X_2, \dots, X_{T_{\text{ctr}}})$ , with different initial conditions (positions or velocities of particles).

**2) Reversed Simulation.** We iteratively solve the required acceleration<sup>1</sup> that can restore *positions* of each fluid particle reversely, from  $X_{T_{\text{ctr}}}$  to  $X_1$ :

$$\ddot{\mathbf{p}}_t = \frac{(\mathbf{p}_{t-1} - \mathbf{p}_t) - \dot{\mathbf{p}}_t \cdot \Delta t}{(\Delta t)^2} - \mathbf{g} \quad (2)$$

**3) Generation of Control Sketches.** Finally, ased on  $\mathbf{X}$ , we generate the user’s sketch that depicts the general movements of particles. We support both directional arrows for movement guidance and one-stroke freehand oval shapes to indicate target regions, as shown in Figure 8. See our Appendix B for details of implementing freehand arrows and oval shapes. Note that in 3D scenarios, we use the arrow width to indicate depth [22].

For simplicity, we will by default control the fluid particles for 100 MPM steps ( $T_{\text{ctr}} = 100$ ). That means all our control trajectory will have 100 steps. While it is possible to employ dynamic neural architectures [41] to adaptively adjust the number of MPM steps for this control based on the control complexity, we leave it as a future work.

### 3.2.3 Diffusion-based Fluid ControlNet

Inspired by the recent success of conditioned video generation [40, 35, 12, 43, 34, 39, 37, 11, 36, 42], we choose to train a conditioned diffusion model to control fluid particles. We by default control the simulations of MPM instead of our neural physics, since the controlled particles may lead to challenging simulations, which are essentially “OOD” settings to neural physics. That means, whenever a user provides a control sketch, we will fall back to MPM, and continue the MPM simulation under the control.

We show our architecture design in Figure 9. Our diffusive controller shares the same backbone and input particle features as our neural physics (Section 2.2). The output of our controller is an external force field that will be applied to particles on top of gravity and particle interactions. Along the MPM simulation, our controller will unroll the subsequent temporal force fields. The training target will be the ground truth force fields we simulate in Section 3.2.2. Parallel to the backbone, we extract the embeddings of the user’s sketch input using a convolutional neural network (CNN) and concatenate them with the diffusion timestep embeddings to guide the generation process. We also embed the current control time step into a latent space and integrate it into the initial noise. See Appendix B for details of our controller’s architecture.

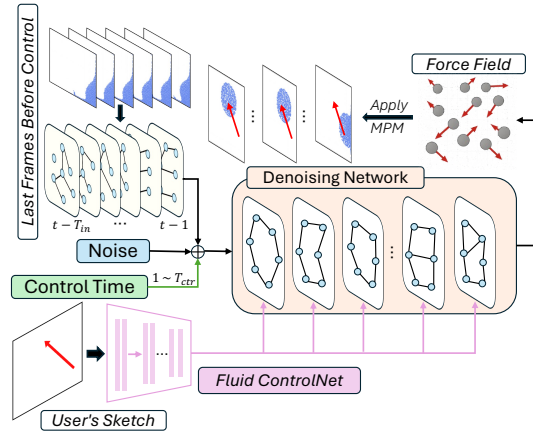


Figure 9: Architecture design of our fluid controller.

<sup>1</sup>Equivalently, the force field if all particles have the same constant mass

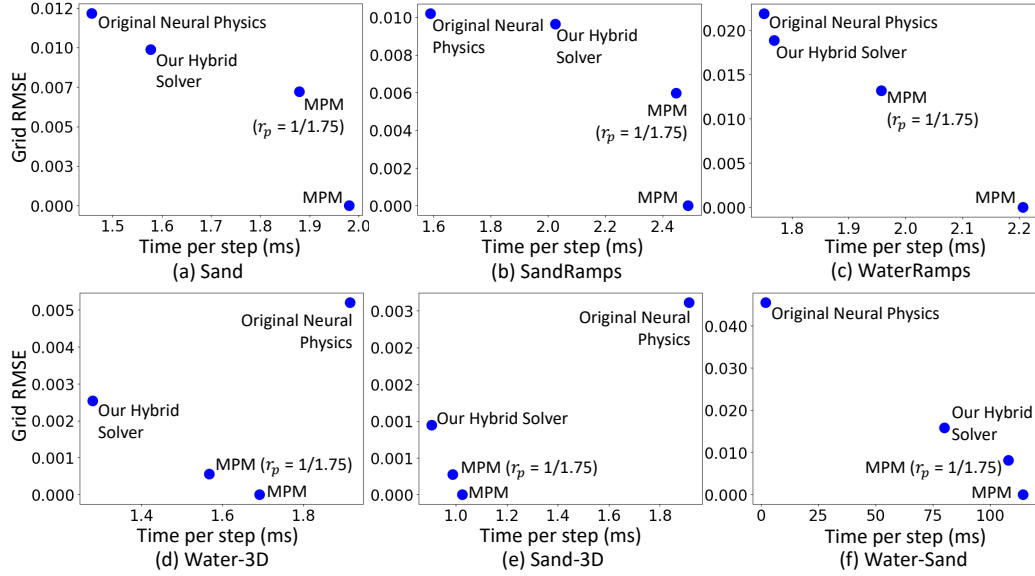


Figure 10: Trade-off between simulation error (grid  $\text{RMSE}_{\tilde{m}}$ ) and latency, comparing different methods. (a) Sand (2D); (b) SandRamps (2D); (c) WaterRamps (2D); (d) Water (3D); (e) Sand (3D); (f) Water-Sand (2D).

## 4 Experiments

### 4.1 Settings

**Physical Domains and Simulations.** To build our hybrid simulator, we prepare our own ground truth simulations with the Taichi package [14, 13, 15] on GPUs, with settings closely aligned with [27]. We summarize our scenarios in Table 2. We include diverse initial conditions (position, velocity) and numbers of particles. We fix our grid size as  $128 \times 128$  for 2D and  $64 \times 64 \times 64$  for 3D. We use time step  $dt = 2.5\text{ms}$  in our simulations.

Table 2: Datasets.  $N_h$ : Max number of particles at the original spatial resolution.  $T$ : total time steps.  $M$ : number of simulation trajectories.

Domain	$N_h$	$T$	$M$
Water (2D)	4k	1k	1k
WaterRamps (2D)	3.3k	600	1k
Sand (2D)	4k	320	1k
SandRamps (2D)	3.3k	400	1k
Water (3D)	4k	800	1k
Sand (3D)	4k	350	1k
Water-Sand (2D)	4k	500	1k

**Evaluation.** To report quantitative results, we evaluated our models by computing **rollout** metrics on held-out test trajectories, drawn from the same distribution of initial conditions used for training. As discussed in Section 3.1.1, we use grid-level  $\text{RMSE}_{\tilde{m}}$  to compare predictions at lower spatial resolution with the original ground truth.

### 4.2 Fluid Simulation Acceleration

Our hybrid simulator can consistently achieve real-time fluid simulations with preserved simulation fidelity across both 2D and 3D cases. We show the trade-off between simulation error and latency in Figure 10, where we compare our hybrid solver with the original neural physics ( $r_p = r_t = 1$ ) [27], MPM [14, 13, 15], and another MPM that also simulates at low spatial resolution ( $r_p = 1/1.75$ ). On 2D scenarios, our hybrid solver consistently balances the neural physics and MPM, achieving both reduced simulation latency and preserved simulation errors. For example, on multiple materials (Water-Sand 2D), our hybrid solver can accelerate MPM from 0.114s per frame to 0.08s, with a 29.8% reduction. On 3D, the neural physics at  $r_p = r_t = 1$  is extremely slow, whereas our hybrid solver improves both latency and errors. For example, on Sand 3D, we reduce the latency of MPM by 11.8%, from 1.02ms to 0.90ms.

### 4.3 Generative Fluid Control

We show visualizations of our generative fluid control in Figure 11. We compare with a baseline, where particles are controlled with a spatiotemporal constant force field, with the force magnitude and orientation solved by moving particles from  $X_{T_{\text{ctrl}}}$  to  $X_1$ .

Table 3: Grid  $\text{RMSE}_{\bar{m}}$  between ground truth and predictions at the last time during fluid control.

Method	Water (2D)	Sand (2D)	Water (3D)	Sand (3D)
Baseline	0.0908	0.1151	0.0019	0.0022
Ours	0.0802	0.0924	0.0013	0.0019

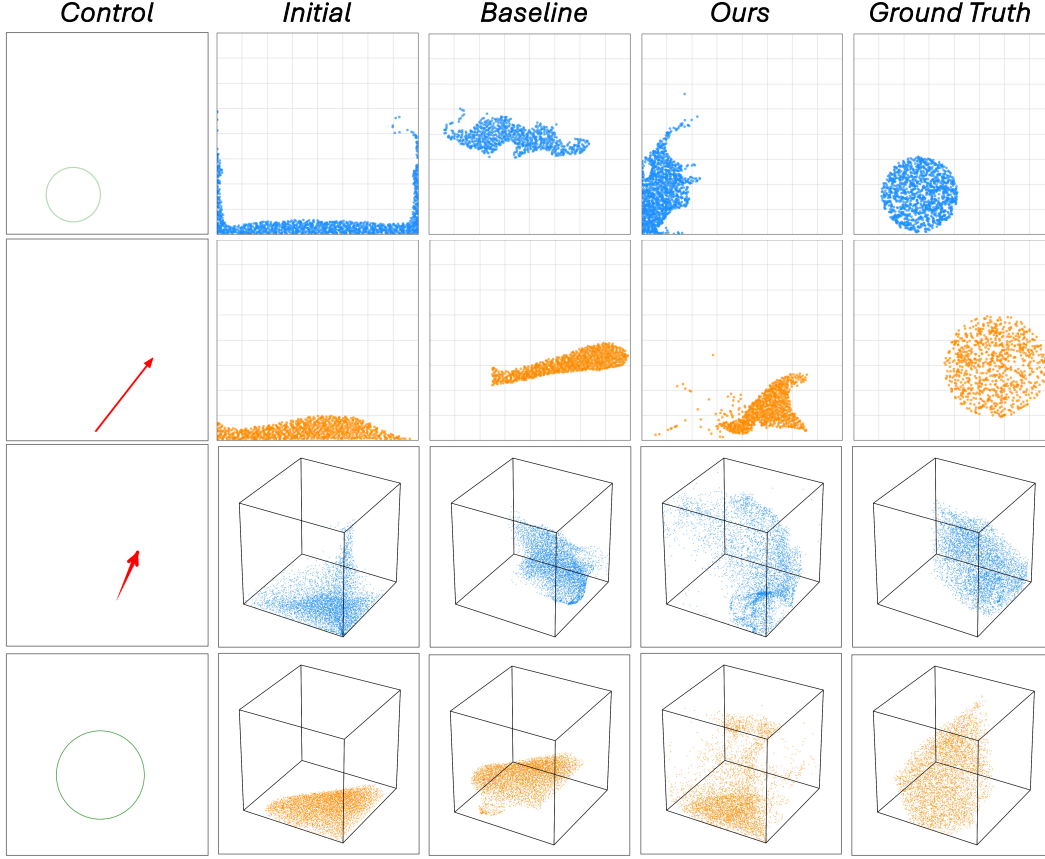


Figure 11: Visualization of generative fluid control. Rows from top to bottom: Water (2D), Sand (2D), Water (3D), Sand (3D).

We also quantitatively evaluate the control in Table 3, where we calculate the grid-level  $\text{RMSE}_{\bar{m}}$  between the ground truth and the prediction at the last time step, since our main concern is the recovery of the shape of the ground truth at the end of the simulation. In sum, we can see that our diffusion controller can move particles to better align with the user sketches.

#### 4.4 Complete Results: Hybrid Simulation + Fluid Control

Finally, we present the result from our complete pipeline in Figure 12. Particles are first simulated by our hybrid solver, where we start with the neural physics (at low spatiotemporal resolution) and is triggered to MPM once the fluid complex is high. Then, a user draws a sketch to control, and our diffusion-based controller takes both this sketch and recent particle states as inputs, and generates external force fields to control particles.

## 5 Related Works

**Fluid Modeling and Animation** Learning-based fluid simulators have progressed from graph-based models to hybrid, physics-informed approaches. DPI-Net [21] introduced dynamic interaction graphs with hierarchical message passing to model interactions across particles. This was unified

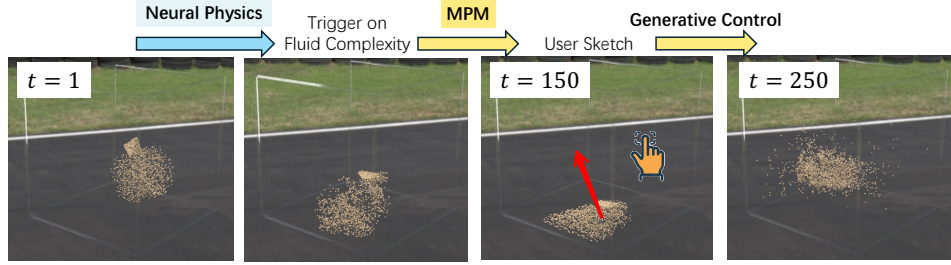


Figure 12: Complete results: hybrid simulation + fluid control. We start the simulation with our neural physics, which is then triggered to MPM. At  $t = 150$ , a user presents the control sketch.

in GNS [27, 24, 19, 18], enabling generalized simulation of fluids, solids, and deformables. Hybrid solvers like MPMNet [20] and NeuralMPM [26] adopt the Material Point Method for scalability. Neural SPH [32] integrates SPH priors to stabilize rollouts, while NeuroFluid [10] combines learned dynamics and rendering from videos. These advances balance physical accuracy with real-time performance. In our work, we propose a hybrid approach that combines neural and numerical methods to enable accelerated and high-fidelity fluid simulation.

**Fluid Control** Recent work in fluid control aims to make simulation more intuitive and accessible. Traditional methods using space-time optimization were costly and hard to tune. Yan et al.[38] addressed this with a sketch-based system using conditional GANs to generate liquid splashes. Pan et al.[22] enabled interactive control through sketching and mesh dragging. Chu et al.[7] used GANs to infer fluid motion from static fields with semantically controllable features. Schoentgen et al.[28] introduced reusable templates for particle-based animations. These approaches shift toward flexible, artist-friendly tools. We tackle the case where only a freehand sketch is given, and the generative controller is tasked with producing the intended artistic fluid behavior.

**Controllable Video Generation** Controllable video generation has advanced rapidly with diffusion models, especially in disentangling motion control. DragNUWA [40] enabled trajectory-based editing, while MotionCtrl [35] and Direct-a-Video [39] decoupled camera and object motion. CameraCtrl [12] and CamCo [37] refined camera control using geometric cues. MotionDirector [43] and Boximator [34] allowed user-customized motion, and SparseCtrl [11] enabled sparse, entity-level conditioning. Tora [42] unified text, image, and trajectory inputs for physics-aware generation. Inspired by these approaches, we leverage forward simulations and compute control forces via reversed simulation.

## 6 Conclusion

In this work, we introduced a novel hybrid neural physics framework that effectively bridges the gap between high-fidelity physical simulation and real-time interactive control. By combining learned graph-based neural simulators with a fallback to classical MPM solvers, we achieved robust, low-latency fluid dynamics capable of handling complex scenarios without sacrificing accuracy. Additionally, we developed a diffusion-based generative controller trained via reverse modeling, enabling intuitive user interaction through freehand sketches for dynamic fluid control. Extensive experiments across 2D and 3D domains demonstrate that our approach not only accelerates fluid simulations but also provides controllable and physically plausible outcomes. This hybrid paradigm represents a step forward in making real-time, artist-friendly fluid simulation practical for applications in graphics, design, and virtual environments.

## 7 Limitations

Our current limitations are: 1) The control step  $T_{\text{ctrl}}$  is fixed at 100 and is not adaptive to the difficulty of the control scenario; 2) Errors are introduced by the inference of neural physics at low resolution. The potential solutions are: 1) Training the diffusion-based controller to unroll different numbers of steps to adapt to challenging control scenarios; 2) Training a super-resolution model to correct errors introduced by simulating neural physics at low spatial resolution. However, addressing these limitations is beyond the scope of this paper, and we plan to study them in our immediate future work.

## References

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [2] Bert Blocken and Ted Stathopoulos. Cfd simulation of pedestrian-level wind conditions around buildings: Past achievements and prospects. *Journal of Wind Engineering and Industrial Aerodynamics*, 121:138–145, 2013.
- [3] Jeremiah U Brackbill and Hans M Ruppel. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational physics*, 65(2):314–343, 1986.
- [4] Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.
- [5] Robert Bridson. *Fluid simulation for computer graphics*. AK Peters/CRC Press, 2015.
- [6] Jie Chen, Haw-ren Fang, and Yousef Saad. Fast approximate kNN graph construction for high dimensional data via recursive lanczos bisection. *Journal of Machine Learning Research*, 10(Sep):1989–2012, 2009.
- [7] Mengyu Chu, Nils Thuerey, Hans-Peter Seidel, Christian Theobalt, and Rhaleb Zayer. Learning meaningful controls for fluids. *ACM Transactions on Graphics (TOG)*, 40(4):1–13, 2021.
- [8] Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web*, pages 577–586, 2011.
- [9] Yue Gao, Hong-Xing Yu, Bo Zhu, and Jiajun Wu. Fluidnexus: 3d fluid reconstruction and prediction from a single video. *arXiv preprint arXiv:2503.04720*, 2025.
- [10] Shanyan Guan, Huayu Deng, Yunbo Wang, and Xiaokang Yang. Neurofluid: Fluid dynamics grounding with particle-driven neural radiance fields. In *International conference on machine learning*, pages 7919–7929. PMLR, 2022.
- [11] Yuwei Guo, Ceyuan Yang, Anyi Rao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Sparsectrl: Adding sparse controls to text-to-video diffusion models. In *European Conference on Computer Vision*, pages 330–348. Springer, 2024.
- [12] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for text-to-video generation. *arXiv preprint arXiv:2404.02101*, 2024.
- [13] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. DiffTaichi: Differentiable programming for physical simulation. *ICLR*, 2020.
- [14] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):201, 2019.
- [15] Yuanming Hu, Jiafeng Liu, Xuanda Yang, Mingkuan Xu, Ye Kuang, Weiwei Xu, Qiang Dai, William T. Freeman, and Frédo Durand. Quantaichi: A compiler for quantized simulations. *ACM Transactions on Graphics (TOG)*, 40(4), 2021.
- [16] Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *arXiv preprint arXiv:2104.03311*, 2021.
- [17] Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015.
- [18] Krishna Kumar and Yonjin Choi. Accelerating particle and fluid simulations with differentiable graph networks for solving forward and inverse problems. In *Proceedings of the SC’23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, pages 60–65, 2023.



- [19] Krishna Kumar and Joseph Vantassel. Gns: A generalizable graph neural network-based simulator for particulate and fluid modeling. *arXiv preprint arXiv:2211.10228*, 2022.
- [20] Jin Li, Yang Gao, Ju Dai, Shuai Li, Aimin Hao, and Hong Qin. Mpmnet: A data-driven mpm framework for dynamic fluid-solid interaction. *IEEE Transactions on Visualization and Computer Graphics*, 2023.
- [21] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018.
- [22] Zherong Pan, Jin Huang, Yiyang Tong, Changxi Zheng, and Hujun Bao. Interactive localized liquid motion editing. *ACM Transactions on Graphics (TOG)*, 32(6):1–10, 2013.
- [23] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.
- [24] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International conference on learning representations*, 2020.
- [25] Karthik Raveendran, Nils Thuerey, Christopher J Wojtan, and Greg Turk. Controlling liquids using meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2012.
- [26] Omer Rochman-Sharabi, Sacha Lewin, and Gilles Louppe. A neural material point method for particle-based simulations. 2024.
- [27] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.
- [28] Arnaud Schoentgen, Pierre Poulin, Emmanuelle Darles, and Philippe Meseure. Particle-based liquid control using animation templates. In *Computer Graphics Forum*, volume 39, pages 79–88. Wiley Online Library, 2020.
- [29] Serkan Solmaz and Tom Van Gerven. Interactive cfd simulations with virtual reality to support learning in mixing. *Computers & Chemical Engineering*, 156:107570, 2022.
- [30] Jos Stam. Stable fluids. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 779–786. 2023.
- [31] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web*, pages 287–297, 2016.
- [32] Artur P Toshev, Jonas A Erbesdobler, Nikolaus A Adams, and Johannes Brandstetter. Neural sph: Improved neural modeling of lagrangian fluid dynamics. *arXiv preprint arXiv:2402.06275*, 2024.
- [33] Pascal Volino, Frederic Cordier, and Nadia Magnenat-Thalmann. From early virtual garment simulation to interactive fashion design. *Computer-aided design*, 37(6):593–608, 2005.
- [34] Jiawei Wang, Yuchen Zhang, Jiaxin Zou, Yan Zeng, Guoqiang Wei, Liping Yuan, and Hang Li. Boximator: Generating rich and controllable motions for video synthesis. *arXiv preprint arXiv:2402.01566*, 2024.
- [35] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video generation. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.
- [36] Weijia Wu, Zhuang Li, Yuchao Gu, Rui Zhao, Yefei He, David Junhao Zhang, Mike Zheng Shou, Yan Li, Tingting Gao, and Di Zhang. Draganything: Motion control for anything using entity representation. In *European Conference on Computer Vision*, pages 331–348. Springer, 2024.

- 415 [37] Dejia Xu, Weili Nie, Chao Liu, Sifei Liu, Jan Kautz, Zhangyang Wang, and Arash Vah-  
416 dat. Camco: Camera-controllable 3d-consistent image-to-video generation. *arXiv preprint*  
417 *arXiv:2406.02509*, 2024.
- 418 [38] Guowei Yan, Zhili Chen, Jimei Yang, and Huamin Wang. Interactive liquid splash modeling by  
419 user sketches. *ACM Transactions on Graphics (TOG)*, 39(6):1–13, 2020.
- 420 [39] Shiyuan Yang, Liang Hou, Haibin Huang, Chongyang Ma, Pengfei Wan, Di Zhang, Xiaodong  
421 Chen, and Jing Liao. Direct-a-video: Customized video generation with user-directed camera  
422 movement and object motion. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–12, 2024.
- 423 [40] Shengming Yin, Chenfei Wu, Jian Liang, Jie Shi, Houqiang Li, Gong Ming, and Nan Duan.  
424 Dragnuwa: Fine-grained control in video generation by integrating text, image, and trajectory.  
425 *arXiv preprint arXiv:2308.08089*, 2023.
- 426 [41] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural  
427 networks. *arXiv preprint arXiv:1812.08928*, 2018.
- 428 [42] Zhenghao Zhang, Junchao Liao, Menghao Li, Zuozhuo Dai, Bingxue Qiu, Siyu Zhu, Long Qin,  
429 and Weizhi Wang. Tora: Trajectory-oriented diffusion transformer for video generation. *arXiv*  
430 *preprint arXiv:2407.21705*, 2024.
- 431 [43] Rui Zhao, Yuchao Gu, Jay Zhangjie Wu, David Junhao Zhang, Jia-Wei Liu, Weijia Wu, Jussi  
432 Keppo, and Mike Zheng Shou. Motiondirector: Motion customization of text-to-video diffusion  
433 models. In *European Conference on Computer Vision*, pages 273–290. Springer, 2024.

## A Details of Neural Physics Simulator

### A.1 Particle Simulations as Message-Passing on a Graph

We denote the state of a particle  $i$  at time step  $t$  as  $\mathbf{x}_{i,t} \in \mathbb{R}^D$ , and the collective state of  $N$  particles as  $\mathbf{X}_t = [\mathbf{x}_{1,t}, \dots, \mathbf{x}_{N,t}] \in \mathbb{R}^{N \times D}$ . Applying physical dynamics over multiple timesteps yields a trajectory of particle states,  $\mathbf{X}_{t_1:t_{T_{\text{in}}}} = [\mathbf{X}_{t_1}, \mathbf{X}_{t_2}, \dots, \mathbf{X}_{t_{T_{\text{in}}}}] \in \mathbb{R}^{T_{\text{in}} \times N \times D}$ . In essence, the simulator  $s : \mathbb{R}^{T_{\text{in}} \times N \times D} \rightarrow \mathbb{R}^{N \times d}$  ( $d = 2$  or  $3$  for 2D/3D) leverages the current  $T_{\text{in}}$  particle states as input to predict their future motion, capturing the underlying dynamics using methods ranging from simple Euler integration to advanced numerical or data-driven techniques. If a simulator is learnable, it can be represented as  $s_\theta$ , a parameterized function approximator. The simulator then iteratively computes future states, such as  $\tilde{\mathbf{X}}_{t_{T_{\text{in}}+1}} = s(\tilde{\mathbf{X}}_{t_1}, \tilde{\mathbf{X}}_{t_2}, \dots, \tilde{\mathbf{X}}_{t_{T_{\text{in}}}})$ , where each newly predicted state is appended to simulate a rollout trajectory over time.

Our learnable simulator  $s_\theta$  represents the physical system as interacting particles, where dynamics emerge from exchanges of energy and momentum with neighbors. To ensure robust simulation quality,  $s_\theta$  must generalize across diverse interaction patterns and physical scenarios. This particle-based approach naturally maps to message passing on a graph, with particles as nodes and pairwise interactions as edges, making graph neural networks (GNNs) a suitable modeling choice.

### A.2 Details of Graph-based Neural Physics

Following [27], we implement our neural physics with GNN, and use standard nearest neighbor algorithms [8, 6, 31] to construct the graph.

**Input.** In our learnable simulator  $s_\theta$ , the input state vector for each particle  $i$  at time step  $t_k$  includes a sequence of 5 previous velocities (via finite differences from  $T_{\text{in}} = 6$  previous locations), and static features representing material properties (e.g., water, sand, rigid, boundary particle). In practice, only the position vectors  $\mathbf{p}_i$  are stored in our datasets; the velocities  $\dot{\mathbf{p}}_i$  and accelerations  $\ddot{\mathbf{p}}_i$  are computed on the fly using finite differences when needed. Formally, the node feature is defined as

$$\mathbf{x}_{i,t_k-T_{\text{in}}:t_k} = [\dot{\mathbf{p}}_{i,t_k-T_{\text{in}}+2}, \dots, \dot{\mathbf{p}}_{i,t_k}, \mathbf{f}_i] \in \mathbb{R}^D,$$

where  $\mathbf{f}_i$  denotes the concatenated material-specific features and scene boundary indicators. Specifically, the dimension of the encoded node feature vector is  $D = 30$  for 2D simulations (5 2-dim velocities by finite differences, i.e.,  $5 \times 2 = 10$ ; 4 distances from the boundary; 16-dim embedding for the particle type), or  $D = 37$  for 3D simulations (5 3-dim velocities by finite differences, i.e.,  $5 \times 3 = 15$ ; 6 distances from the boundary; 16-dim embedding for the particle type). See Figure 2 for an illustration. It is important to note that in our Fluid Controlnet (Section 3.2.3), the input feature dimension  $D$  will increase by 16, where we embed the current control timestep into the latent space with another 2-layer MLP with SiLU activation.

To obtain more informative edge features  $\mathbf{r}_{i,j}$ , we use the relative positional displacement between a pair of adjacent particles  $i$  and  $j$ , along with its magnitude:

$$\mathbf{r}_{i,j} = [(\mathbf{p}_i - \mathbf{p}_j), \|\mathbf{p}_i - \mathbf{p}_j\|].$$

Edges are added between particles that lie within a predefined *connectivity radius*  $R = 0.015$ , which captures local particle interactions.  $R$  is kept constant for all 2D scenarios. In different 3D scenarios, a larger radius can be used to accommodate higher-resolution environments. Although  $R$  is fixed in simulations, edges in the graph are still dynamically updated by comparing the current particle-wise distances to  $R$ . For full details of these input and target features, we refer readers to [27].

The  $\text{ENCODER} : \mathbb{R}^{N \times D} \rightarrow \mathcal{G}$  embeds particle-based states, it can be formulated as:  $G^{(0)} = (\mathbf{V}^{(0)}, \mathbf{E}^{(0)}) = \text{ENCODER}(\mathbf{X}, \mathbf{r}_{i,j})$ . The node embeddings  $\mathbf{V}^{(0)} = \text{ENCODER}_V(\mathbf{X})$  are learned functions of the particles' states. The edge embeddings,  $\mathbf{E}_{i,j}^{(0)} = \text{ENCODER}_E(\mathbf{r}_{i,j})$ , are learned functions of the pairwise properties of the corresponding particles. We implement  $\text{ENCODER}_V$  and  $\text{ENCODER}_E$  as multilayer perceptrons (MLP), which encode node features and edge features into the latent vectors,  $\mathbf{V}_i$  and  $\mathbf{E}_{i,j}$ , of size 128.

The  $\text{PROCESSOR} : \mathcal{G} \rightarrow \mathcal{G}$  computes interactions among nodes through  $L$  steps of learned message passing and outputs the final graph,  $G^{(L)} = \text{PROCESSOR}(G^{(0)})$ . Message passing enables information propagation among particles. Our  $\text{PROCESSOR}$  consists of a stack of  $L = 10$  GNN layers,

each using separate (non-shared) MLPs for updating node and edge features, along with residual connections between the input and output latent attributes of both nodes and edges. For the fluid controller setting, an additional MLP layer is used to encode the diffusion timestep and control image features; see Appendix B.2 for details.

The DECODER :  $\mathcal{G} \rightarrow \mathbb{R}^{N \times d}$  extracts dynamics information (of the future state) from the nodes of the final latent graph,  $\hat{X} = \text{DECODER}(\mathbf{V}^{(L)})$ . Our DECODER is an MLP that outputs accelerations  $\ddot{\mathbf{p}}_i$ . The future position and velocity are updated using an Euler integrator.

All MLPs in PROCESSOR have two hidden layers with ReLU, followed by an output layer without activation, with a width of 128. All MLPs are followed by a LayerNorm [1].

## B Implementations

### B.1 Latency Measurements

**Latency of Neural Physics.** We utilize the TensorRT library to convert the PyTorch model into an ONNX model to accelerate model inference and align it with the acceleration of MPM on the Taichi kernel. However, since TensorRT does not support the aggregation operation in GNNs (i.e., aggregating information from edges to adjacent nodes), when measuring the latency, we approximate the time cost of this aggregation operation with a matrix multiplication between an adjacency matrix  $A \in \mathbb{R}^{N \times N}$  (where  $N$  denotes the number of nodes, i.e. particles), and node features ( $\mathbf{o}$ ), such that the aggregation becomes  $A \cdot \mathbf{o}$ . All reported latency measurements are based on the median number of nodes across different scenarios in our test datasets.

**Latency of Taichi.** To enable a fair comparison under MPM simulation setting, we applied a matching latency reduction strategy to the Taichi implementation by skipping non-essential overhead. Specifically, we excluded the time spent on initializing the MPM state (initial positions and velocities of particles) and the cost of initializing the Taichi kernel at the beginning of the simulation. As a result, our comparison focuses solely on the runtime per simulation step after the CUDA or Taichi kernel has been initialized.

### B.2 Design of Fluid ControlNet

In our fluid controller, the control signal  $\mathcal{C} \in \mathbb{R}^{H \times W \times 3}$  is encoded using our Fluid ControlNet. The encoded embedding is then injected into the graph-based diffusion model to guide the generation of the external field of accelerations. The Fluid ControlNet consists of 8 convolutional layers and 3 downsampling operations. It extracts multi-scale features from the control signal  $\mathcal{C}$ , projects each scale to a different dimensional space, and then concatenates the projected features into control embedding representation of dimension size 44. The resulting embedding is then integrated into the PROCESSOR module of the graph-based diffusion model. Notably, to better condition the diffusion process on the control signal, we draw inspiration from DiT [23] and concatenate the embedding of the control signal to the diffusion time step embedding. This design choice ensures that the control condition is effectively incorporated at each diffusion step, thereby generating high-fidelity acceleration fields that can align fluid particles to the target motion or shape.

### B.3 Training

Following [27], we normalize the input velocity to the GNNs, and apply random noises to input positions ( $\mathbf{p}_{t_1:t_{T_m}}$ ) during training. For both neural physics and fluid controller, we train with the Adam optimizer and a learning rate at  $1 \times 10^{-4}$  with exponential decay. Our training batch size is 1, and we train for 2 million gradient descent steps.

**Table 4:** Training Costs (GPU hours) across different scenarios.

GPU Hours	Water (2D)	Sand (2D)	Water (3D)	Sand (3D)
Neural Physics (Section 3.1)	17.27h	17.94h	19.71h	19.67h
Fluid ControlNet (Section 3.2)	69.87h	76.36h	184.12h	151.03h

We include our training costs in Table 4. Neural physics requires approximately one day on a single NVIDIA 4090 GPU. For the Fluid ControlNet, training takes around three days for 2D scenarios on a single NVIDIA 4090 and six days for 3D scenarios on a single NVIDIA A40. We train both neural physics and Fluid ControlNet with the particle-level RMSE loss on predicted accelerations  $\text{RMSE}_{\ddot{\mathbf{p}}} \equiv \frac{1}{N} \sum_{i=1}^N \frac{\|\hat{\ddot{\mathbf{p}}}_i - \ddot{\mathbf{p}}_i\|_2}{\|\ddot{\mathbf{p}}_i\|_2}$ , which was defined in Section 2.2.

#### B.4 Generating Users' Freehand Sketches (Arrows and Oval Shapes)

Arrows are computed by connecting the centroid ( $\bar{\mathbf{p}} = \frac{1}{N} \sum_{i=1}^N \mathbf{p}_i$ ) of fluid particles at  $t = 1$  ( $\bar{\mathbf{p}}_1$ ) and  $t = T_{\text{ctr}}$  ( $\bar{\mathbf{p}}_{T_{\text{ctr}}}$ ). Based on the mean displacement vector  $\Delta\bar{\mathbf{p}} = \bar{\mathbf{p}}_1 - \bar{\mathbf{p}}_{T_{\text{ctr}}}$ , we derive the arrow length  $\|\Delta\bar{\mathbf{p}}\|$  and orientation  $\theta = \tan^{-1}(\Delta\bar{\mathbf{p}}_y/\Delta\bar{\mathbf{p}}_x)$ . In 3D, we use the arrow width to indicate depth [22]. A multi-segment arrow with varying line width is implemented as  $n$ -segment polyline with width modulation, where each segment's width  $w_i$  ( $i \in [1, n]$ ) is  $w_i = w_{\min} + (w_{\max} - w_{\min}) \cdot \frac{\Delta\bar{\mathbf{p}}_{z,i} - \Delta\bar{\mathbf{p}}_{z,\min}}{\Delta\bar{\mathbf{p}}_{z,\max} - \Delta\bar{\mathbf{p}}_{z,\min}}$ . The arrowhead adopts perspective-correct scaling.

For 2D oval sketches, shapes of particles at  $t = T_{\text{ctr}}$  are represented as elliptical outlines centered at  $\bar{\mathbf{p}}_{T_{\text{ctr}}}$ , with radii corresponding to  $\pm 2\sigma$ , where  $\sigma$  is the standard deviation of particle positions along each principal axis. This statistically-grounded ellipse captures approximately 95% particles' positions while being visually simple. Meanwhile, 2D oval-shaped control sketches can indeed be ambiguous in 3D, since it is infeasible to depict 3D volumes with a simple one-stroke 2D sketch.

#### B.5 Enforcing Smoothness on Target Accelerations

We observe that ground truth accelerations solved by Equation 2 are typically complicated (see the temporal-wise cosine similarity in Figure 13), which will be challenging to learn. We thus further enforce a certain level of smoothness of the acceleration across temporal steps:

$$\ddot{\mathbf{p}}_{t,\text{smooth}} = \ddot{\mathbf{p}}_t - \lambda \cdot \exp\left(-\beta \cdot \frac{\ddot{\mathbf{p}}_t \cdot \ddot{\mathbf{p}}_{t+1}}{\|\ddot{\mathbf{p}}_t\| \cdot \|\ddot{\mathbf{p}}_{t+1}\|}\right) \cdot (\ddot{\mathbf{p}}_t - \ddot{\mathbf{p}}_{t+1}) \quad (3)$$

Essentially, Equation 3 enforces decoupled smoothness over the magnitude and the orientation of accelerations over temporal steps. We choose  $\lambda = 0.1$  and  $\beta = 2$  in our work.

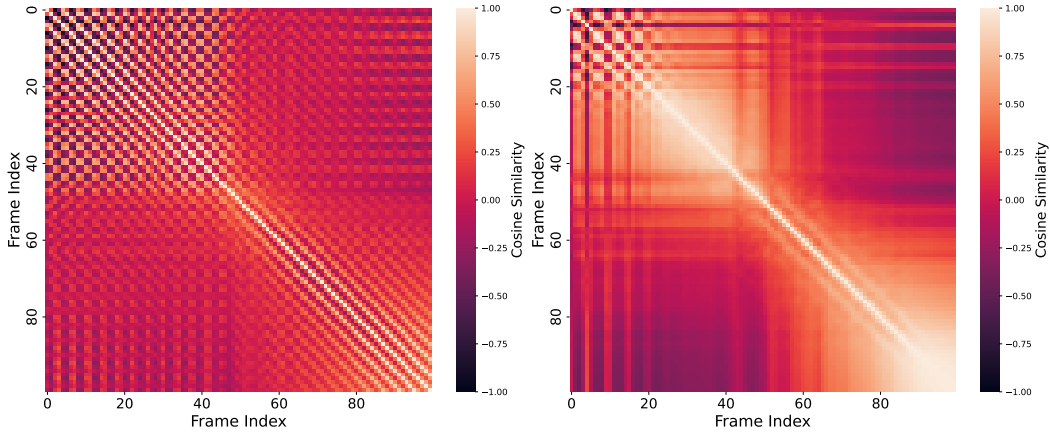


Figure 13: Step-wise correlations of ground-truth accelerations for fluid control. Left: before enforcing smoothness; Right: after enforcing smoothness.

### C More Results

#### C.1 Grid $\text{RMSE}_{\bar{m}}$ of Fluid Simulations over Random Seeds

To ensure a fairer comparison, we conducted experiments using three different random seeds. The results, as shown in Table 5, demonstrate that our hybrid solver consistently outperforms the original neural physics across all datasets.

**Table 5:** Grid RMSE $_{\tilde{m}}$  of fluid simulations on different scenarios, over three random runs.

Grid RMSE $_{\tilde{m}}$	Water (2D)	Sand (2D)	SandRamps (2D)	WaterRamps (2D)	Water (3D)	Sand (3D)	Water-Sand (2D)
Neural Physics	0.0263 (1.15e-6)	0.0125 (2.59e-7)	0.0101 (3.23e-8)	0.0229 (2.09e-6)	0.0048 (6.58e-7)	0.0025 (2.11e-8)	0.0441 (3.51e-6)
Our Hybrid Solver	0.0186 (8.17e-6)	0.0116 (6.88e-8)	0.0096 (1.00e-9)	0.0171 (3.16e-6)	0.0022 (1.77e-8)	0.0013 (1.08e-7)	0.0149 (2.38e-6)

## C.2 More Visualizations

**Fluid Simulations.** Figure 14 presents the visualizations of all models discussed throughout the paper. Here, we show a comparison of intermediate frames from a single trajectory. It is evident that, due to the hybrid design of our hybrid solver, our method produces visual results that are more similar to MPM ( $r_p = 1/1.75$ ) simulations. Since MPM ( $r_p = 1/1.75$ ) is highly consistent with MPM (ground truth), the outputs of our Hybrid solver also align better with MPM compared to the original neural physics. This demonstrates that our approach effectively balances computational efficiency and accuracy.

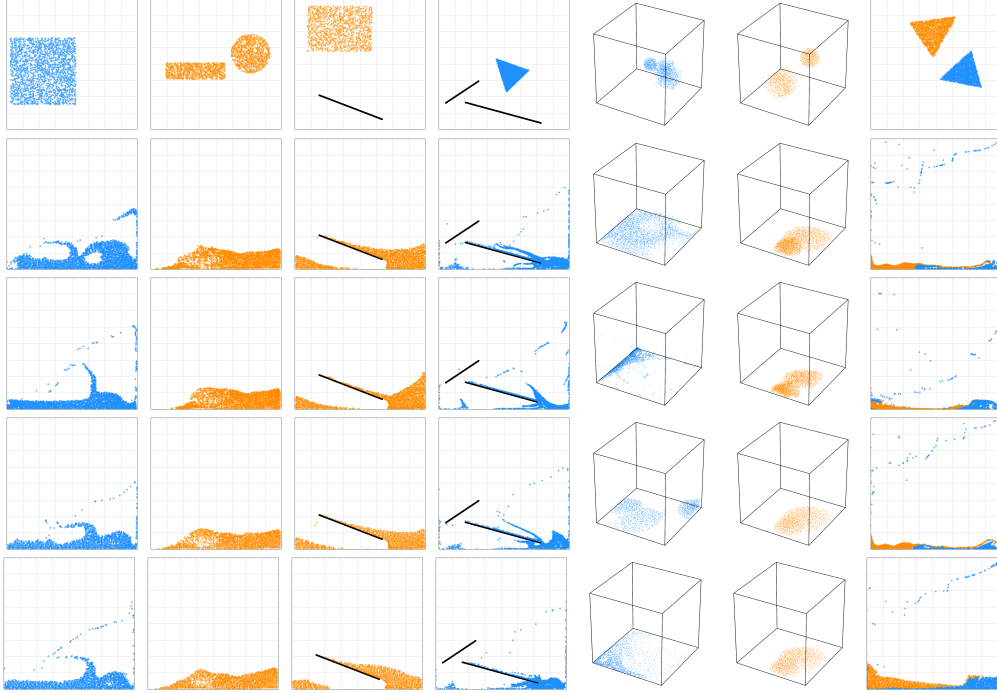


Figure 14: Visualizations of fluid simulations by different methods, over different scenarios. From left to right: Water (2D), Sand (2D), SandRamps (2D), WaterRamps (2D), Water (3D), Sand (3D), Water-Sand (2D). From top to bottom: Initial, MPM (ground truth), Original Neural Physics, MPM ( $r_p = 1/1.75$ ), Our Hybrid Solver.

**More Visualizations of Fluid Control.** Figure 15 presents additional visualizations of generative fluid control across a variety of tasks, both 2D and 3D control signals. We can see that our approach consistently generates physically plausible and visually accurate outcomes that align closely with the target controls across all fluid types and dimensions, demonstrating strong control capability. These results further confirm the effectiveness of our method in achieving both visually appearing and physically plausible fluid control.



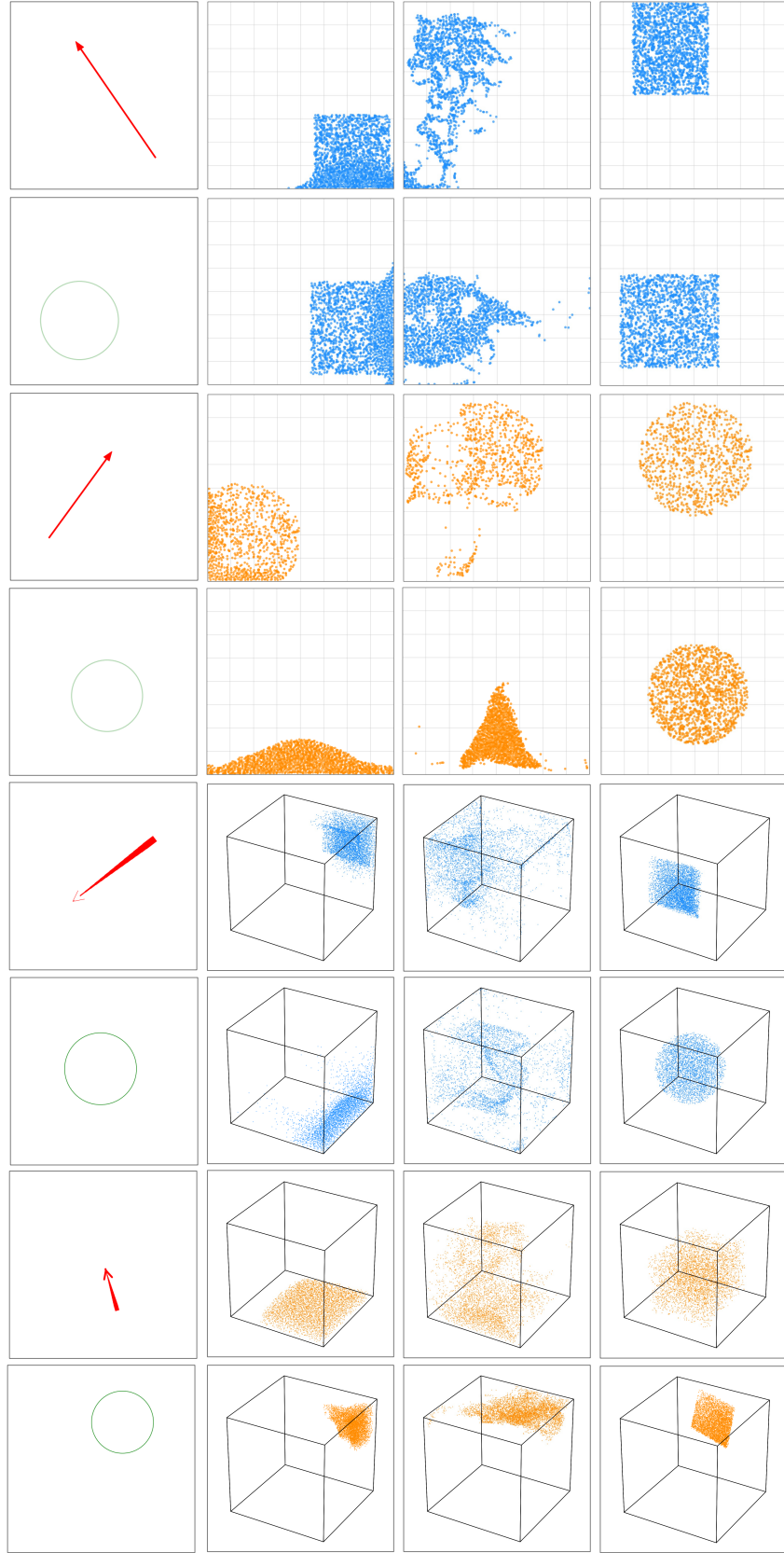


Figure 15: More visualization of generative fluid control. From top to bottom: Water2D, Sand2D, Water3D, and Sand3D, each with two types of control signals (arrows for motion direction, and oval shapes for target spatial positions). From left to right: control signal, initial, ours, ground truth.

## NeurIPS Paper Checklist

### 1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and Introduction explicitly state the three main contributions (hybrid neural + MPM solver, diffusion-based controller, and real-time performance) and the results sections back each of these claims with empirical evidence.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

### 2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We include a “Limitation” section in the main paper.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren’t acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

### 3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper is entirely empirical; it introduces no theorems or formal proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

#### 4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4 details datasets, grid sizes, time-steps, and initial-condition sampling (Table 2), while the Appendix lists optimizer, learning-rate schedule, batch size, and total training steps. We also attach our code in the supplement. Together these give enough information to rerun the experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

## 5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We attach our code in the supplement. We will also release both our data, code, model upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

## 6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Dataset splits, particle counts, grid resolutions, dt, network architecture, and all training hyper-parameters are specified in Section 3, Section 4, and Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

## 7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We present standard deviations of results in our Figure 10 over three random runs in Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

## 8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In the Appendix we give our training times (1 GPU-day for neural physics, 3–6 GPU-days for the controller) on a single GPU.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

## 9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The work uses only synthetic simulation data and involves no personal or sensitive information, so it aligns with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

## 10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: There is no societal impact of the work performed.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

## 11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

## 12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No]

Justification: External libraries (e.g., Taichi) are cited, and the CC BY 4.0 license has been chosen on OpenReview.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.



- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, [paperswithcode.com/datasets](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

### 13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: We promise to release all our data upon acceptance.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

### 14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[NA\]](#)

Justification: No human-subject or crowdsourced data are used.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

### 15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [\[NA\]](#)

Justification: Not applicable—no human-subject studies were conducted.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- 878           • We recognize that the procedures for this may vary significantly between institutions  
879           and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the  
880           guidelines for their institution.  
881           • For initial submissions, do not include any information that would break anonymity (if  
882           applicable), such as the institution conducting the review.

883 **16. Declaration of LLM usage**

884       Question: Does the paper describe the usage of LLMs if it is an important, original, or  
885       non-standard component of the core methods in this research? Note that if the LLM is used  
886       only for writing, editing, or formatting purposes and does not impact the core methodology,  
887       scientific rigorousness, or originality of the research, declaration is not required.

888       Answer: [NA]

889       Justification: The core method development in this research does not involve LLMs as any  
890       important, original, or non-standard components.

891       Guidelines:

- 892           • The answer NA means that the core method development in this research does not  
893           involve LLMs as any important, original, or non-standard components.  
894           • Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>)  
895           for what should or should not be described.