

WorldParticle: Unified World Simulation of Lagrangian Particle Dynamics via Transformer

CAOLIWEN WANG*, University of British Columbia, Canada

MINGHAO GUO*†, MIT CSAIL, USA

SIYUAN CHEN*, University of British Columbia, Canada

HENG ZHANG, University of British Columbia, Canada

MENGDI WANG, Georgia Institute of Technology, USA

XINGYU NI, Inria, France

HANSON SUN, University of British Columbia, Canada

KUNYI WANG, University of British Columbia, Canada

ZHERONG PAN, Meta, USA

KUI WU, Independent Researcher, USA

LINGJIE LIU, University of Pennsylvania, USA

YIN YANG, University of Utah, USA

CHENFANFU JIANG, University of California Los Angeles, USA

TAKU KOMURA, University of Hong Kong, Hong Kong

WOJCIECH MATUSIK, MIT CSAIL, USA

PETER YICHEN CHEN†, University of British Columbia, Canada

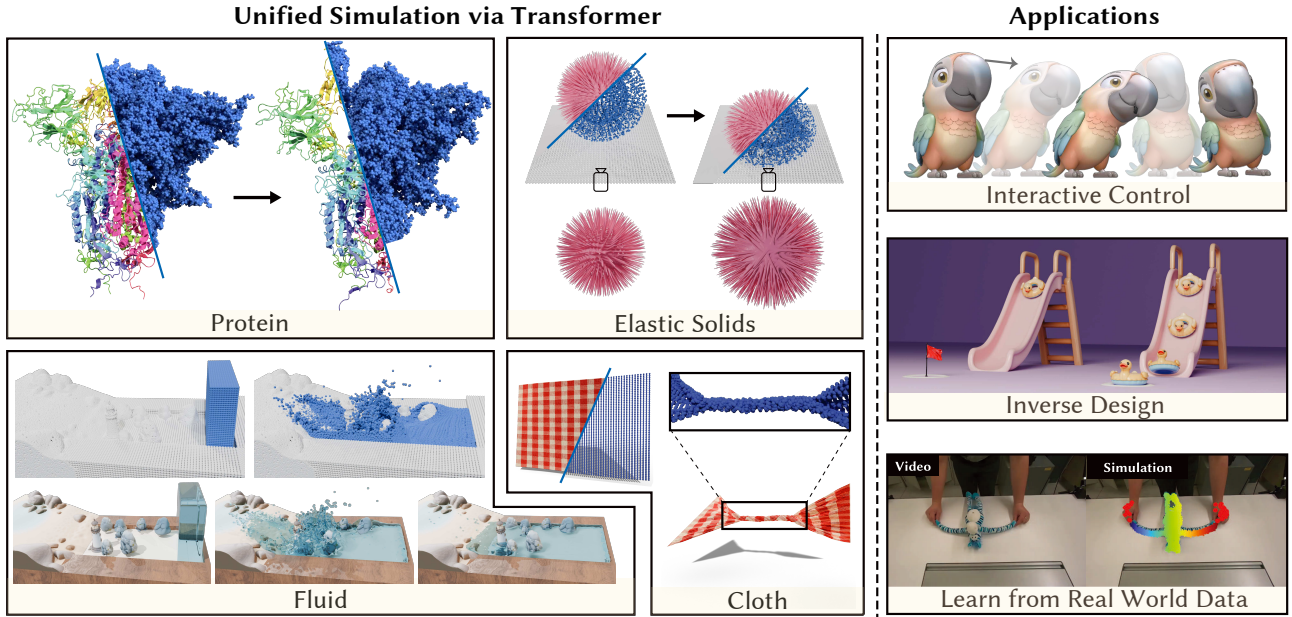


Fig. 1. **We propose a unified transformer-based neural simulator for Lagrangian particle dynamics.** *Left:* Our model handles diverse particle systems, including proteins, elastic solids, fluids, and cloth, within a single architecture. *Right:* The learned simulator supports downstream tasks including interactive control, inverse design, and learning from real-world observations.

*Equal contribution

†Corresponding author

Authors' Contact Information: [Caoliwen Wang](mailto:Caoliwen.Wang@ubc.ca), wclw1021@gmail.com, University of British Columbia, Canada; [Minghao Guo](mailto:Minghao.Guo@mit.edu), guomh2014@gmail.com, MIT CSAIL, USA; [Siyuan Chen](mailto:Siyuan.Chen@ubc.ca), chensiyuan030105@gmail.com, University of British Columbia, Canada; [Heng Zhang](mailto:Heng.Zhang@ubc.ca), mediosrity@gmail.com, University of British Columbia, Canada; [Mengdi](mailto:Mengdi.Wang@gtt.org)

[Wang, m.wang.13@outlook.com](mailto:Wang.m.wang.13@outlook.com), Georgia Institute of Technology, USA; [Xingyu Ni](mailto:Xingyu.Ni@inria.fr), xingyu.ni@inria.fr, Inria, France; [Hanson Sun](mailto:Hanson.Sun@ubc.ca), jsun33@student.ubc.ca, University of British Columbia, Canada; [Kunyi Wang](mailto:Kunyi.Wang@ubc.ca), kunyi.wang252750@gmail.com, University of British Columbia, Canada; [Zherong Pan](mailto:Zherong.Pan@meta.com), zherong.pan.usa@gmail.com, Meta, USA; [Kui Wu](mailto:Kui.Wu@walker.ai), walker.kui.wu@gmail.com, Independent Researcher, USA; [Lingjie Liu](mailto:Lingjie.Liu@seas.upenn.edu), lingjie.liu@seas.upenn.edu, University of Pennsylvania, USA; [Yin Yang](mailto:Yin.Yang@utah.edu), yangzzzy@gmail.com, University of Utah, USA; [Chenfanfu Jiang](mailto:Chenfanfu.Jiang@ucla.edu), chenfanfu.jiang@gmail.com, University

A unified simulator that can model diverse physical phenomena without solver-specific redesign is a long-standing goal across simulation science. We present a learning-based particle simulator built on a single transformer architecture to model cloth, elastic solids, Newtonian and non-Newtonian fluids, granular materials, and molecular dynamics. Our model follows a prediction-correction design on a shared Lagrangian particle representation. An explicit predictor first advances particles under the known external forces, producing an intermediate state that captures externally driven motion but not inter-particle interactions. A learned corrector then predicts the residual position and velocity updates through three stages: a particle tokenizer that encodes local particle-particle, particle-boundary, and topology-guided interactions; a super-token encoder that hierarchically merges particle tokens into a compact set of super tokens via alternating self-attention and token merging; and a super-token decoder that lifts these super tokens back to particle resolution through cross-attention to predict per-particle position and velocity corrections. Progressive token merging reduces the attention cost at successive encoder layers by halving the token count at each level, and the decoder communicates through the compact super-token set rather than full particle-to-particle attention. Across the six dynamics categories, the same architecture generalizes to unseen materials, boundary configurations, initial conditions, and external forces. We further demonstrate downstream interactive control, inverse design, and learning from real-world manipulation data, reducing the need for per-phenomenon solver engineering.

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

Additional Key Words and Phrases: Neural Simulation; Transformer

1 Introduction

Building a *unified simulation*, a single framework that captures the full diversity of physical phenomena, has long been pursued across simulation science, from computer graphics and engineering simulation to molecular modeling, but remains elusive in practice. Different phenomena are traditionally modeled with different discretizations, from Eulerian grids for fluids, to Lagrangian meshes and the finite-element method for elastic solids, to particle samples for free-surface flow, and by different governing equations, from Navier–Stokes to nonlinear constitutive laws, granular rheology, and atomistic potentials, many exhibiting strong nonlinearity and multiscale coupling. Modern engines such as NVIDIA Newton [Contributors 2025] push toward broader coverage by combining several solver backends in one system, yet each phenomenon still relies on its own dedicated solver internally. This *per-phenomenon*, *per-solver* paradigm makes cross-phenomenon coupling difficult and constrains how broadly a single classical pipeline can be applied across the diverse physics of real-world scenes.

A promising direction is to unify the underlying discrete representation. Position-based dynamics [Müller et al. 2007] uses particles with energy constraints to simulate cloth, soft bodies, and fluids within a single framework, at the cost of physical fidelity. The material point method [Stomakhin et al. 2013; Sulsky et al. 1994] adopts a hybrid particle-grid representation and captures fluid-solid coupling with higher accuracy, but still relies on per-scene parameter tuning and is computationally expensive. Despite these tradeoffs, both demonstrate that a particle-based Lagrangian representation is a natural common language across physical phenomena: material

points, mesh vertices, SPH samples, and atoms can all be abstracted as particles carrying position, velocity, and per-particle physical properties. Particles, however, only unify the representation layer; the complex nonlinearity and problem-specific parameterization of the underlying governing equations remain, and no analytical model covers all phenomena at once.

Machine learning offers a complementary answer on the modeling side: rather than hand-designing constitutive and interaction laws, a network can learn dynamics directly from trajectory data, bypassing explicit PDEs and per-scene tuning. Existing neural simulators [Li et al. 2019; Pfaff et al. 2021; Sanchez-Gonzalez et al. 2020; Ummenhofer et al. 2020], however, are typically designed and trained for a specific phenomenon or equation class, and reduced-order methods [Barbič and James 2005; Treuille et al. 2006] further tie their learned bases to a fixed geometry or shape family. Changing the physics often requires changing the model design.

We propose a particle-based prediction-correction transformer whose architecture remains *entirely fixed* across phenomena, only the training data changes. The same architecture simulates cloth, elastic solids, Newtonian and non-Newtonian fluids, granular materials, and molecular dynamics. Following the prediction-correction scheme common in particle simulation [Müller et al. 2007], an explicit predictor integrates known external forces, and a learned corrector then predicts interaction corrections for position and velocity. The corrector has three stages: a *particle tokenizer* encodes local interactions through learnable kernels over spatial, boundary, and topological neighborhoods; inspired by multigrid hierarchical coarsening [Vaněk et al. 1996], a *super-token encoder* compresses these tokens into a compact set of super tokens by alternating self-attention and token merging, progressively halving the token count; and a *super-token decoder* lifts super-token information back to full particle resolution through cross-attention, producing residual position and velocity corrections. Because both the architecture and the prediction-correction interface are equation-agnostic, the model treats each phenomenon as a different training distribution over the same input-output format, rather than requiring a new model design for each governing equation. The pipeline is trained with an autoregressive rollout loss.

The same architecture, trained on the six Lagrangian dynamics categories above, generalizes to unseen materials, boundary configurations, initial conditions, and external forces. Our model delivers higher fidelity and more stable long-horizon rollouts than existing neural simulators while covering a broader range of physical phenomena in a single architecture. Beyond forward simulation, we demonstrate three downstream applications: interactive control of deformable objects under user-specified forces; inverse design of friction parameters via differentiable rollout; and real-world manipulation prediction on particle trajectories extracted with Phys-Twin [Jiang et al. 2025] using 3D Gaussian Splatting [Kerbl et al. 2023]. Together, our model handles phenomena whose classical counterparts each require dedicated FEM, SPH, or MPM solvers, using a single differentiable architecture and workflow, at the cost of some per-phenomenon accuracy compared to those specialized methods. Our contributions are:

of California Los Angeles, USA; Taku Komura, taku@cs.hku.hk, University of Hong Kong, Hong Kong; Wojciech Matusik, wojciech@csail.mit.edu, MIT CSAIL, USA; Peter Yichen Chen, pyc@csail.mit.edu, University of British Columbia, Canada.

- A particle-based prediction-correction transformer that simulates cloth, elastic solids, Newtonian and non-Newtonian fluids, granular materials, and molecular dynamics, in a single architecture.
- A particle tokenizer for local neighborhood interactions combined with a super-token encoder-decoder for global communication through progressive token merging.
- Experiments across six dynamics categories showing lower roll-out errors and more stable long-horizon predictions than existing neural simulators, together with three applications: interactive control, inverse design, and real-world manipulation prediction.

2 Related work

We review three lines of work: learning-based physics simulators that operate on particles and meshes, reduced-order and operator-learning methods that accelerate simulation through compact representations, and transformer architectures applied to graphics.

Learning-based physics simulation. Learning-based simulators have progressed from object-level prediction to models operating directly on particles and meshes. Interaction networks [Battaglia et al. 2016] and neural physics engines [Chang et al. 2017] predict dynamics of predefined object sets but require manually specified interaction graphs. SPNets [Schenck and Fox 2018] and DPI-Net [Li et al. 2019] learn particle-level updates across multiple material types. Continuous-convolution networks [Ummenhofer et al. 2020] replace hand-crafted SPH kernels with learned ones. GNS [Sanchez-Gonzalez et al. 2020] and MeshGraphNets [Pfaff et al. 2021] learn local message passing on particle radius graphs and simulation meshes, respectively. Domain-specific neural simulators further exploit physical structure for Eulerian-fluid acceleration [Tompson et al. 2017], smoke synthesis [Chu and Thuerey 2017], fluid super-resolution [Xie et al. 2018], cloth deformation [Bertiche et al. 2021, 2022], garment collision handling [Liao et al. 2024], and constitutive-law learning within differentiable solvers [Ma et al. 2023]. A common limitation is that global information must propagate through many local hops. Our work retains the Lagrangian particle representation and local neighborhood structure, but introduces a super-token encoder-decoder that provides global communication in every correction step without deep message-passing chains.

Reduced-order modeling and neural operators. A complementary direction accelerates simulation by compressing the state or the solution operator. Latent-space models encode high-dimensional fields into compact coordinates and learn their temporal evolution [Kim et al. 2019; Morton et al. 2018; Wiewel et al. 2019]. Neural-field and subspace methods learn reduced representations spanning specific geometry families or solution classes [Chang et al. 2025, 2026, 2023; Chen et al. 2023, 2025, 2026; Huang et al. 2026; Liu et al. 2025; Modi et al. 2024; Zong et al. 2023], but new geometries or physics typically require rebuilding the basis. Neural operators learn maps between function spaces [Li et al. 2021; Lu et al. 2021; Raissi et al. 2019], with extensions to multiscale problems [Kossaifi et al. 2024; Li et al. 2020; Raonić et al. 2023; Tran et al. 2023], complex geometries [Li et al. 2023; Wang et al. 2024], and physical consistency [Li et al. 2024]; they predict fields over PDE families rather than evolving individual particle states. Our model performs particle-level evolution: super

tokens serve as intermediate communication variables regenerated from the full dynamical state at every timestep, rather than a fixed reduced basis or a learned solution operator.

Transformers in graphics and simulation. Transformers provide data-dependent, nonlocal communication on geometric domains. In point-cloud processing, self-attention supports geometric reasoning, shape completion, and pretraining [Pang et al. 2022; Wang 2023; Wu et al. 2024a, 2022; Yang et al. 2023; Yu et al. 2021, 2022; Zhao et al. 2021]. In rendering, attention aggregates multiview and ray information for view synthesis and global illumination [Jin et al. 2025; Kulhánek et al. 2022; Wang et al. 2021; Zeng et al. 2025]. Transformers have also been applied to physics surrogates for capturing long-range dependencies on general domains [Holzschuh et al. 2026; Wu et al. 2024b]. Video and generative world models produce visually plausible motion but operate on pixels or latent frames rather than explicit particle states [Wang et al. 2025]. Our method applies transformer attention to particle-level simulation, covering a wide range of physical phenomena within a single architecture.

3 Method

We model one step of Lagrangian particle dynamics as a *prediction-correction* procedure [Ladický et al. 2015; Müller et al. 2007]. An explicit prediction step integrates known external forces, producing an intermediate state that accounts for external acceleration but not for inter-particle interactions. A learned correction step then predicts the interaction correction. The corrector consists of three components: a *particle tokenizer* that encodes particle-particle, particle-boundary, and topological interactions, a *super-token encoder* that compresses particle tokens into dynamically generated super tokens, and a *super-token decoder* that lifts super-token information back to particle resolution to produce position and velocity corrections. Together, they separate communication into two scales: the tokenizer captures *local* interactions, while the encoder and decoder handle *global* coupling through the super tokens.

Let $X_t, V_t, F_t \in \mathbb{R}^{N \times 3}$ denote positions, velocities, and external forces of the N simulated particles, and let $C \in \mathbb{R}^{N \times C_P}$ denote their per-particle attributes such as mass, material parameters, or atom types. Separately, N_b static boundary particles (X^b, C^b) represent walls, obstacles, or contact surfaces, with attributes such as normals and friction coefficients. $\mathcal{T}$ denotes rest-shape topology when available (e.g., for cloth and elastic solids). The prediction step integrates the external forces over one timestep,

$$\tilde{V}_t = V_t + \Delta t M^{-1} F_t, \quad \tilde{X}_t = X_t + \frac{\Delta t}{2} (V_t + \tilde{V}_t), \quad (1)$$

where M is the diagonal mass matrix assembled from C . The correction step takes the predicted state and outputs residual updates:

$$(\Delta X_t, \Delta V_t) = \mathcal{F}_\theta(\tilde{X}_t, \tilde{V}_t, C, \mathcal{T}, X^b, C^b), \quad (2)$$

yielding $X_{t+\Delta t} = \tilde{X}_t + \Delta X_t$ and $V_{t+\Delta t} = \tilde{V}_t + \Delta V_t$. Predicting both residuals lets the model represent effects from implicit solves, contact projections, pressure coupling, and damping that cannot be expressed as a single explicit acceleration. Inputs unavailable in a domain, such as $\mathcal{T}$ for fluids or boundary particles for proteins, are zero-filled so that the architecture remains fixed. Throughout, bold uppercase letters denote ordered sets of particles (e.g., X) and

bold lowercase with subscript i denotes the i -th element (e.g., $\mathbf{x}_i$). A superscript (ℓ) indicates the encoder or decoder layer; when omitted it defaults to layer 0 (the original particle resolution), so in particular $N^{(0)} = N$. The following subsections describe one timestep and drop the time subscript; lowercase symbols such as $(\tilde{\mathbf{x}}_i, \tilde{\mathbf{v}}_i, \mathbf{c}_i)$ denote the i -th row of $(\tilde{\mathbf{X}}, \tilde{\mathbf{V}}, \mathbf{C})$. The components of $\mathcal{F}_\theta$ are shown in Fig. 2.

3.1 Particle Tokenizer

The particle tokenizer encodes local interactions before global attention. It follows the locality principle underlying particle methods: interactions are evaluated over compact neighborhoods, while the filters inside those neighborhoods are learned from data rather than prescribed as SPH kernels [Koschier et al. 2022]. For particle i , we use up to three supports: a spatial neighborhood $\mathcal{N}_i^s$ from radius search around $\tilde{\mathbf{x}}_i$, a boundary neighborhood $\mathcal{N}_i^b$ from nearby boundary samples, and a topological neighborhood $\mathcal{N}_i^t$ defined by the adjacency of the rest-shape tessellation $\mathcal{T}$. Each branch aggregates neighbor contributions through a learnable kernel $\mathbf{W}_k$:

$$\mathbf{a}_i^k = \sum_{j \in \mathcal{N}_i^k} \mathbf{W}_k(\mathbf{r}_{ij}^k) \mathbf{u}_{ij}^k, \quad k \in \{\text{s, b, t}\}, \quad (3)$$

where $\mathbf{W}_k$ maps a relative displacement to a mixing matrix, parameterized as a learnable 3D grid with trilinear interpolation and compact support of radius [Ummenhofer et al. 2020] (details in the appendix). Each branch defines displacement $\mathbf{r}_{ij}^k$ and feature $\mathbf{u}_{ij}^k$ as:

$$\begin{aligned} \text{Particle-Particle: } \mathbf{r}_{ij}^s &= \tilde{\mathbf{x}}_j - \tilde{\mathbf{x}}_i, & \mathbf{u}_{ij}^s &= [\tilde{\mathbf{v}}_j, \mathbf{c}_j], \\ \text{Particle-Boundary: } \mathbf{r}_{ij}^b &= \mathbf{x}_j^b - \tilde{\mathbf{x}}_i, & \mathbf{u}_{ij}^b &= \mathbf{c}_j^b, \\ \text{Topology-Guided: } \mathbf{r}_{ij}^t &= \mathbf{x}_j^0 - \mathbf{x}_i^0, & \mathbf{u}_{ij}^t &= [\tilde{\mathbf{v}}_j, \mathbf{c}_j, \mathbf{x}_j^0 - \mathbf{x}_i^0], \end{aligned} \quad (4)$$

where $\mathbf{x}_i^0$ denotes the particle's rest-pose position. The particle token $\mathbf{h}_i$ concatenates all available branches with the particle's own state:

$$\mathbf{h}_i = \text{MLP}([\mathbf{a}_i^s, \mathbf{a}_i^b, \mathbf{a}_i^t, \tilde{\mathbf{v}}_i, \mathbf{c}_i]). \quad (5)$$

3.2 Super-Token Encoder

The particle tokens $\{\mathbf{h}_i\}$ capture local interactions, but many physical phenomena, such as pressure waves, elastic stresses, and molecular motion, require global, long-range communication. Self-attention provides such coupling [Vaswani et al. 2017], but applying it repeatedly over all N particles is computationally expensive. Inspired by the hierarchical coarsening principle of multi-grid methods [Vaněk et al. 1996], the super-token encoder alternates global self-attention at the current resolution with *token merging* [Bolya et al. 2022], so later attention layers operate on progressively fewer tokens.

Let $\mathbf{H}_e$ and $\tilde{\mathbf{X}}$ be the particle tokens and positions from the tokenizer. Each encoder layer $\ell = 1, \dots, L_e$ first applies multi-head self-attention with 3D RoPE [Su et al. 2024] to contextualize all tokens at the current resolution:

$$\tilde{\mathbf{H}}_e^{(\ell)} = \text{Self-Attn}(\mathbf{H}_e^{(\ell-1)}, \tilde{\mathbf{X}}^{(\ell-1)}). \quad (6)$$

with $\tilde{\mathbf{h}}_i^{(\ell)}$ denotes the i -th token in $\tilde{\mathbf{H}}_e^{(\ell)}$. We then perform token merging, which halves the sequence from $N^{(\ell-1)}$ to $N^{(\ell)} = \lceil N^{(\ell-1)} / 2 \rceil$. Specifically, the current tokens $\{\tilde{\mathbf{h}}_i^{(\ell)}\}$ are split into two disjoint halves by alternating index. Each token in the first half is greedily matched to its most cosine-similar token in the second half, and matched pairs are merged while unmatched tokens are kept as-is.

Let $\mathcal{M}_i^{(\ell)}$ denote the set of tokens collapsed into the i -th output token. The merged token and its positional anchor are computed by weighted averaging:

$$(\mathbf{h}_i^{(\ell)}, \tilde{\mathbf{x}}_i^{(\ell)}) = \sum_{j \in \mathcal{M}_i^{(\ell)}} \frac{m_j^{(\ell-1)}}{m_i^{(\ell)}} (\tilde{\mathbf{h}}_j^{(\ell-1)}, \tilde{\mathbf{x}}_j^{(\ell-1)}), \quad \mathbf{h}_i^{(\ell)} \leftarrow \text{MLP}(\mathbf{h}_i^{(\ell)}). \quad (7)$$

where $m_i^{(\ell)} = \sum_{j \in \mathcal{M}_i^{(\ell)}} m_j^{(\ell-1)}$ is the multiplicity of token i at layer ℓ , initialized as $m_i^{(0)} = 1$. The merged tokens and positions form $\mathbf{H}_e^{(\ell)}$ and $\tilde{\mathbf{X}}^{(\ell)}$ with $N^{(\ell)}$ tokens, which become the input to layer $\ell + 1$. Applying self-attention *before* merging is important: similarity is computed after each token has seen current-scale context, rather than from raw local features alone. Thus, particles with similar local neighborhoods but different global roles (e.g., one near a free surface, one deep in the interior) are less likely to be merged.

After L_e layers, the remaining tokens are the *super tokens* $\bar{\mathbf{H}} \triangleq \mathbf{H}_e^{(L_e)}$ with corresponding positions $\bar{\mathbf{X}} \triangleq \tilde{\mathbf{X}}^{(L_e)}$, as shown in Fig. 3. Compared to classical multigrid, which relies on local smoothing operators and achieves linear per-iteration cost, our encoder uses global self-attention at each level and retains a quadratic cost at every resolution. The trade-off is expressiveness: global attention captures long-range coupling more directly than purely local operators, and the rapid halving ensures that the total cost is dominated only by the first full-resolution layer.

Super tokens as a queryable latent field. The pair $(\bar{\mathbf{H}}, \bar{\mathbf{X}})$ forms a sparse, queryable latent representation of the current particle system. Each super token aggregates local features, velocities, and material information from its represented particle group into a latent feature anchored at the corresponding position $\bar{\mathbf{x}}_i$. Unlike an implicit neural field stored entirely in network weights [Chen et al. 2023], or a neural eigenbasis precomputed for a specific shape family [Chang et al. 2025], these super tokens are generated online from the current particle state at every timestep. This distinction is the key to generalization: within a trained dynamics category, the encoder can process held-out configurations, since super tokens are produced from whatever configuration is presented. The decoder (Sec. 3.3) queries this representation by cross-attending from particle tokens to super tokens.

3.3 Super-Token Decoder

The decoder lifts coarse context back to particle resolution through L_d layers. The particle tokens $\mathbf{H}_d^{(0)} = \{\mathbf{h}_i\}$ and positions $\tilde{\mathbf{X}}$ from the tokenizer are passed directly to the decoder, bypassing the encoder. At decoder layer ℓ , particle tokens query the super tokens through cross-attention [Jaegle et al. 2021]:

$$\tilde{\mathbf{H}}_d^{(\ell)} = \text{Cross-Attn}(\mathbf{H}_d^{(\ell-1)}, \tilde{\mathbf{X}}; \bar{\mathbf{H}}, \bar{\mathbf{X}}). \quad (8)$$

Self-attention then refines the tokens:

$$\mathbf{H}_d^{(\ell)} = \text{Self-Attn}(\tilde{\mathbf{H}}_d^{(\ell)}, \tilde{\mathbf{X}}), \quad \mathbf{H}_d^{(\ell)} \leftarrow \text{MLP}(\mathbf{H}_d^{(\ell)}). \quad (9)$$

All attention blocks use 3D RoPE [Su et al. 2024], residual connections, and feed-forward networks. Thus the decoder performs a learned coarse-to-fine lifting: the attention weights act as data-adaptive interpolation weights from super-token anchors to particles. This resembles coarse-to-fine reconstruction in multilevel

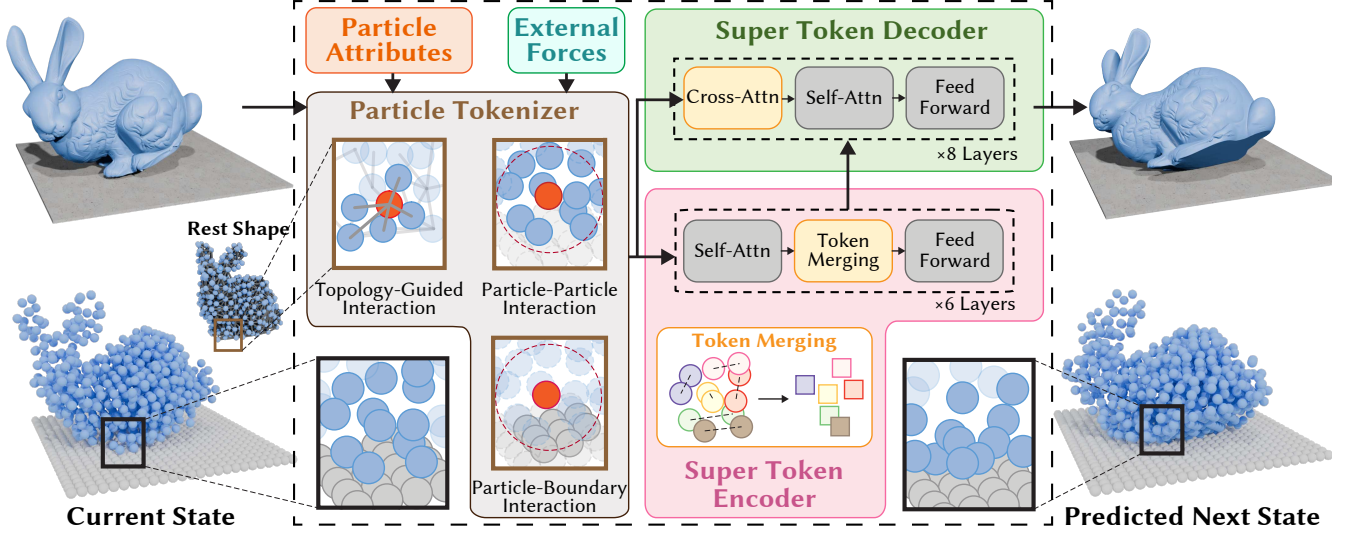


Fig. 2. **Prediction-correction particle transformer.** Given the intermediate state from the prediction step (Eq. 1), the correction step proceeds in three stages within one timestep: the *particle tokenizer* encodes local interactions into per-particle tokens, the *super-token encoder* compresses them into dynamically generated super tokens via self-attention and token merging, and the *super-token decoder* refines particle tokens through alternating self-attention and cross-attention to produce position and velocity corrections (Eq. 2).

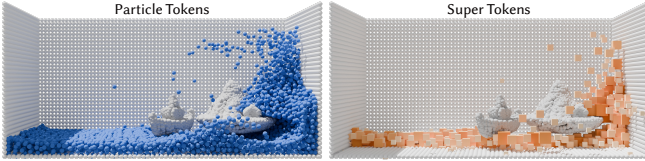


Fig. 3. **Token merging visualization.** Particle tokens (blue spheres, left) are merged into super tokens (yellow cubes, right). Cube size and color intensity reflect how many original particles each super token represents.

methods and reduced-order models [Chang et al. 2025; Chen et al. 2023; Fulton et al. 2019], but the weights here are feature-dependent and recomputed for every timestep rather than fixed by geometry or a precomputed basis. A final particle-wise MLP maps $H_d^{(L_d)}$ to $(\Delta x_i, \Delta v_i)$. Full specifications are provided in the appendix.

3.4 Training

We use the same architecture across all dynamics categories and train one network per category. For a rollout window of W states, including the initial ground-truth state ($W \leq 5$), we start from (X_t, V_t) and unroll the prediction-correction update autoregressively for $W-1$ steps, feeding each predicted state back as input. The predicted states $\{(\tilde{X}_{t+n\Delta t}, \tilde{V}_{t+n\Delta t})\}_{n=1}^{W-1}$ are supervised against the ground-truth states $\{(X_{t+n\Delta t}, V_{t+n\Delta t})\}_{n=1}^{W-1}$. The training loss is:

$$\mathcal{L} = \mathcal{L}_{\text{roll}} + \lambda_{\text{phys}} \mathcal{L}_{\text{phys}}, \quad (10)$$

where $\mathcal{L}_{\text{roll}}$ is the mean per-particle, per-step position and velocity error averaged over the rollout, and $\mathcal{L}_{\text{phys}}$ optionally supervises domain-specific physical constraints, e.g., $\mathcal{L}_{\text{phys}} = \mathcal{L}_{\text{div}}$, a divergence-free regularizer for incompressible fluids. Training is conducted on 8 NVIDIA A100 GPUs (80 GB VRAM per GPU). The

ALGORITHM 1: Predictor-corrector transformer rollout

Require: Rollout window W , timestep Δt , mass matrix M

Require: Initial state (X_t, V_t) , **particle attributes** C , **external forces** $\{F_{t+n\Delta t}\}_{n=0}^{W-2}$, rest-shape topology $\mathcal{T}$, boundary particles (X^b, C^b)

Ensure: Predicted trajectory $\{(X_{t+n\Delta t}, V_{t+n\Delta t})\}_{n=1}^{W-1}$

```

1: for  $n = 0$  to  $W - 2$  do
2:   // Prediction step (Eq. 1)
3:    $\tilde{V} \leftarrow V_{t+n\Delta t} + \Delta t M^{-1} F_{t+n\Delta t}$ ,  $\tilde{X} \leftarrow X_{t+n\Delta t} + \frac{\Delta t}{2} (V_{t+n\Delta t} + \tilde{V})$ 
4:   // Correction step (Eq. 2)
5:   Build neighborhoods  $N^s, N^b, N^r$  from  $(\tilde{X}, X^b, \mathcal{T})$ 
6:    $H \leftarrow \text{ParticleTokenizer}(\tilde{X}, \tilde{V}, C, N^s, N^b, N^r, C^b)$ 
7:    $(\bar{H}, \bar{X}) \leftarrow \text{SuperTokenEncoder}(H, \tilde{X})$ 
8:    $(\Delta X, \Delta V) \leftarrow \text{SuperTokenDecoder}(H, \tilde{X}, \bar{H}, \bar{X})$ 
9:   // State update
10:   $X_{t+(n+1)\Delta t} \leftarrow \tilde{X} + \Delta X$ ,  $V_{t+(n+1)\Delta t} \leftarrow \tilde{V} + \Delta V$ 
11: end for
12: return  $\{(X_{t+n\Delta t}, V_{t+n\Delta t})\}_{n=1}^{W-1}$ 

```

model has 156.73M parameters. Optimization details are provided in the appendix.

4 Experiments

We evaluate our model on six categories of particle dynamics, comparing it against state-of-the-art neural simulators and ablating key architectural design choices; model and training details are provided in the appendix. For each dynamics category, we train a separate model using data generated by established solvers: Newtonian fluids from DFSPH [Bender and Koschier 2015], cloth from GIPC [Huang et al. 2024, 2025], granular sand and non-Newtonian fluids from MPM [Stomakhin et al. 2013], elastic shapes from VBD [Chen et al. 2024], and protein molecular dynamics from OpenMM [Eastman

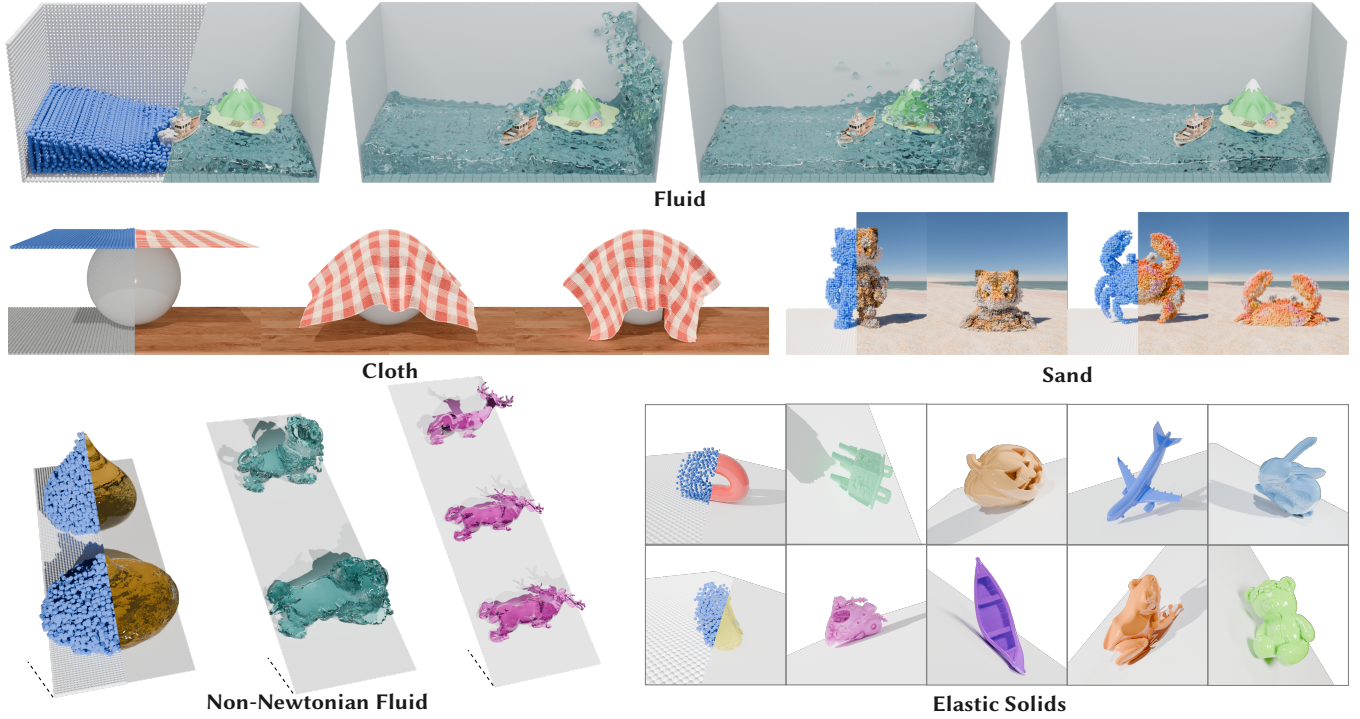


Fig. 4. **Qualitative results across five of six dynamics categories: Newtonian fluids, cloth, granular sand, non-Newtonian fluids, and elastic solids.** Example sequences show Newtonian fluids flowing with varying viscosity and obstacle placements in a container, cloth colliding with a sphere, granular sand collapsing with varied initial shapes and friction, non-Newtonian fluids with varying rheological properties on different slopes, and elastic solids with varying initial rotation on a fixed slope. All test sequences use unseen configurations. In the first figure of each sequence, blue spheres indicate predicted particles, and white spheres indicate boundary samples. Results for protein are shown in Fig. 6.

et al. 2023]. Within each category, we randomize material parameters, initial conditions, and boundary configurations across training sequences, and evaluate on held-out test scenes that vary these factors. Specifically, fluid scenes vary viscosity and obstacle placement; cloth scenes vary stiffness and density under sphere collisions; sand scenes vary initial pile shape and friction; non-Newtonian scenes vary rheological properties and slope angle; elastic solid scenes vary initial orientation on a fixed slope; and protein scenes vary initial atomic conformations. Representative rollouts across all categories are shown in Fig. 1, 4, and 6, illustrating predicted results under varying materials, initial conditions, and boundary configurations. Quantitative results are reported in Table 1, containing rollout mean-squared errors (MSE) for particle positions and velocities, averaged over all particles, frames, and test sequences, together with the degrees of freedom (DOF) and per-step compute cost.

4.1 Comparison with Baselines

We compare against GNS [Sanchez-Gonzalez et al. 2020], DeepLagrangian [Ummenhofer et al. 2020], and Neural Operator [Li et al. 2021; Viswanath et al. 2024] on fluid, cloth, and sand tasks. As shown in Fig. 15, all baselines exhibit artifacts such as floating sand particles, explosive cloth behavior, and interpenetration for long rollouts, whereas our model maintains stable simulations that closely match the ground truth.

Table 1. **Accuracy and inference cost on test sets.** Auto-regressive rollout MSE for particle positions and velocities, averaged over particles, frames, and test sequences. DOF: degrees of freedom. Throughput is reported in frames/s; per-step compute cost is in TFLOPs. All measurements are conducted on a single NVIDIA A100 GPU (80 GB VRAM).

Metric	Fluid	Cloth	Sand	Non-Newt.	Elastic	Protein
# of Frames	600	90	100	250	100	100
DOFs	42,120	29,400	60,000	60,000	52,542	61,884
Pos. Error ($\times 10^{-3}$)	516.585	0.077	0.054	0.706	0.158	0.001
Vel. Error ($\times 10^{-3}$)	263.149	0.454	1.442	1.760	3.530	0.092
Speed (frames/s)	16.176	24.050	11.500	11.168	11.708	14.695
Cost (TFLOP/step)	1.323	0.923	1.886	1.886	1.651	0.944

We attribute the improvement to three design choices. First, the prediction-correction decomposition factors out known external forces, so the learned corrector handles only inter-particle interactions. This separation also enables generalization to unseen force configurations at inference time. Second, the particle tokenizer captures fine-grained local interactions, while the super-token encoder and decoder provide global communication through self-attention and cross-attention; baselines typically offer one scale, requiring

either many message-passing steps for long-range effects or sacrificing local detail. Third, progressive token merging supplies multigrid-like multi-resolution processing, resolving both fine contact details and long-range pressure or stress coupling simultaneously.

4.2 Generalization

We evaluate generalization along four axes: unseen initial and boundary conditions, long rollouts beyond the training horizon, unseen geometries, and unseen particle sampling densities. The architecture is designed to support: super tokens are generated from the current particle state at every timestep; the tokenizer uses relative displacements, making the model translation-equivariant; material parameters enter as per-particle attributes; and the prediction-correction separation factors out known external forces so the corrector learns transferable interaction patterns.

Initial and boundary conditions. Fig. 8 shows two sequences from the dataset of [Ummenhofer et al. 2020]. The first row is a training sequence whose initial fluid configuration is the closest match to the test case in the second row; despite this proximity, the fluid volume, container geometry, and obstacle placement all differ substantially (highlighted in orange). Our 800-frame rollout on the test case remains stable and closely tracks the ground truth, capturing splash patterns, fluid-obstacle interaction, and the long-term settling behavior, demonstrating generalization capabilities to unseen boundary and initial conditions.

Long rollouts. Fig. 7 shows a cloth-twisting experiment where boundary vertices are driven by external torques of varying magnitudes. The model was trained on 200-frame sequences but tested here at 250 frames, requiring extrapolation beyond its training horizon. The cloth continues to accumulate twist without diverging, collapsing, or developing interpenetration, producing coherent winding motion. Long-horizon stability is a known difficulty for autoregressive neural simulators as errors compound; the prediction-correction decomposition helps by providing a consistent intermediate state from the predictor before the corrector is applied.

Geometries. As shown in Fig. 4, the tiger-shaped sand pile never appeared in the training set, which contains only simple shapes such as cylinders and boxes, yet our model stably predicts its collapse with plausible granular flow and pile spreading. This suggests the model learns local interaction rules rather than memorizing shape-specific trajectories, so it transfers to novel geometries whose local particle neighborhoods remain within the training distribution. The same observation holds for the cloth category, where test meshes with unseen vertex counts and aspect ratios produce stable rollouts.

Particle sampling density. Fig. 12 tests the fluid model, trained on approximately 7,000 particles, with altered initial arrangements and increased particle counts at higher resolutions. Rollouts remain stable across all tested densities. This robustness follows from the use of relative displacements in the tokenizer, which depend on inter-particle distance, allowing the model to process denser or sparser configurations than seen during training.

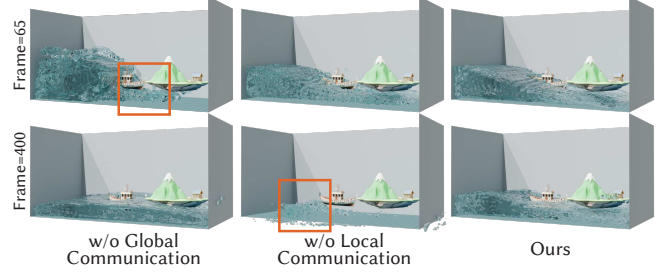


Fig. 5. **Ablation on local and global communication.** Three variants on the fluid task: w/o the super-token encoder-decoder (local only), w/o the tokenizer’s neighborhood branches (global only), and the full model. Removing either component degrades rollout quality.

4.3 Ablation Studies

The ablations below isolate two core design choices: the interplay between local tokenization and global super-token communication, and the physics-informed regularization loss.

Local and global communication. We examine the effect of local and global communication on the fluid task. The *local-only* model removes the super-token encoder and decoder and predicts the correction quantity from particle tokens alone, reducing to the Deep Lagrangian approach [Ummenhofer et al. 2020] without multi-level information exchange. The *global-only* model removes the neighborhood aggregation branches in the tokenizer, so each token encodes only the single particle under consideration rather than summarizing information from a local neighborhood, and embeds positions and velocities directly into token space. As shown in Fig. 5, the local-only model fails to propagate long-range pressure information, while the global-only model loses fine-grained particle interactions and accumulates error over time. The full model, combining both pathways, produces the most accurate rollouts.

Physical regularization. We test the effect of adding a divergence-free regularizer $\mathcal{L}_{\text{div}}$ to $\mathcal{L}_{\text{phys}}$ on the fluid task (Fig. 4). The divergence is computed using the standard SPH estimator with the same kernel used by the DFSPH reference solver [Bender and Koschier 2015], applied to the predicted particle positions and velocities at each rollout step. The inset table reports rollout position and velocity MSE alongside the divergence residual. Adding $\mathcal{L}_{\text{div}}$ reduces the divergence error while maintaining comparable trajectory accuracy. We also ablate network hyperparameters and architectural choices on the cloth task (Fig. 4); full results are reported in the appendix.

Metric	w/o $\mathcal{L}_{\text{div}}$	w/ $\mathcal{L}_{\text{div}}$
Pos. Loss	0.517	0.535
Vel. Loss	0.263	0.292
$\mathcal{L}_{\text{div}}$	0.015	0.008

4.4 Applications

Our architecture unlocks various downstream applications: the predictor’s explicit external-force integration (Eq. 1) enables generalization to unseen force configurations for interactive control; the differentiable neural corrector enables gradient-based optimization of physical parameters for inverse design; and the generic particle

representation allows training on real-world point-cloud observations. Fast single-GPU inference makes these applications practical.

Interactive Control. Because the prediction step explicitly integrates external forces, the model generalizes to force configurations not seen during training. Combined with fast inference, this enables interactive applications. As shown in Fig. 10, we build a real-time interface using Polyscope [Sharp et al. 2019] where users apply forces of arbitrary direction, location, and magnitude.

Inverse Design. Since gradients can be backpropagated through the neural rollout, we can optimize physical parameters by backpropagating through auto-regressive rollouts. We consider a setting where a duck-shaped inflatable mesh slides down a ramp and glides along the ground; the goal is to recover the friction coefficient μ that brings the duck to rest at a prescribed target location. We freeze the trained model and optimize μ alone via gradient descent through the rollout. Fig. 13 compares trajectories under the initial and optimized friction values alongside the optimization curve: the recovered μ places the duck at the target, and the loss converges smoothly.

Learning from Real-World Data. Since our model operates on particle representations, we can train it directly on real-world point cloud observations without a classical simulator. Fig. 11 shows three manipulation sequences from PhysTwin [Jiang et al. 2025], comparing real-world observations with our model’s predicted trajectories. The predicted rollouts qualitatively match the observed motion.

5 Conclusion and Discussion

We presented a prediction-correction particle transformer for Lagrangian dynamics, trained on cloth, elastic solids, Newtonian and non-Newtonian fluids, granular materials, and molecular dynamics. The architecture combines local particle tokenization with a super-token encoder-decoder, capturing both local interactions and global coupling within a single fixed architecture. The key design choice is that super tokens are constructed from the full dynamical state at every timestep, making the global communication input-dependent rather than tied to a fixed basis or geometry. The model generalizes to unseen materials, boundary configurations, initial conditions, and external forces, and supports downstream tasks including interactive control, inverse design, and learning from real-world observations.

Several directions remain for future work. The main limitation is memory: the first encoder layer applies self-attention at full particle resolution, currently supporting up to approximately 50,000 particles on a single 80GB A100 GPU. Memory-efficient attention variants such as sparse attention [Child et al. 2019] could extend this budget to larger scenes. We currently train one parameter set per dynamics category; training a single model with shared weights across all categories would move from a unified architecture to a unified model, though scaling to the combined data volume and maintaining per-category accuracy will pose additional challenges. Extending the per-particle attribute to include quantities such as temperature or strain history would broaden the range of addressable phenomena, and tighter integration with differentiable reconstruction pipelines [Hafner et al. 2023] could further close the gap between learned simulation and real-world deployment.

References

- Jernej Barbič and Doug L. James. 2005. Real-Time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph.* 24, 3 (July 2005), 982–990. doi:10.1145/1073204.1073300
- Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. 2016. Interaction Networks for Learning about Objects, Relations and Physics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*.
- Jan Bender et al. [n.d.]. *SPLiSHSPLasH Library*. <https://github.com/InteractiveComputerGraphics/SPLiSHSPLasH>
- Jan Bender and Dan Koschier. 2015. Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, California) (SCA ’15). Association for Computing Machinery, New York, NY, USA, 147–155. doi:10.1145/2786784.2786796
- Hugo Bertiche, Meysam Madadi, and Sergio Escalera. 2021. PBNS: physically based neural simulation for unsupervised garment pose space deformation. *ACM Trans. Graph.* 40, 6, Article 198 (Dec. 2021), 14 pages. doi:10.1145/3478513.3480479
- Hugo Bertiche, Meysam Madadi, and Sergio Escalera. 2022. Neural Cloth Simulation. *ACM Trans. Graph.* 41, 6, Article 220 (Nov. 2022), 14 pages. doi:10.1145/3550454.3555491
- Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461* (2022).
- Michael B. Chang, Tomer Ullman, Antonio Torralba, and Joshua B. Tenenbaum. 2017. A Compositional Object-Based Approach to Learning Physical Dynamics. In *Proceedings of the 5th International Conference on Learning Representations*.
- Yue Chang, Otman Benckroun, Maurizio M. Chiaramonte, Peter Yichen Chen, and Eitan Grinspun. 2025. Shape Space Spectra. *ACM Trans. Graph.* 44, 4, Article 121 (July 2025), 16 pages. doi:10.1145/3731148
- Yue Chang, Peter Yichen Chen, Eitan Grinspun, and Maurizio M. Chiaramonte. 2026. Low-Rank Koopman Deformables with Log-Linear Time Integration. *arXiv:2602.07687* [cs.GR]
- Yue Chang, Peter Yichen Chen, Zhecheng Wang, Maurizio M. Chiaramonte, Kevin Carlberg, and Eitan Grinspun. 2023. LiCROM: Linear-Subspace Continuous Reduced Order Modeling with Neural Fields. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney, NSW, Australia) (SA ’23). Association for Computing Machinery, New York, NY, USA, Article 111, 12 pages. doi:10.1145/3610548.3618158
- Anka He Chen, Ziheng Liu, Yin Yang, and Cem Yuksel. 2024. Vertex Block Descent. *ACM Trans. Graph.* 43, 4, Article 116 (July 2024), 16 pages. doi:10.1145/3658179
- Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, G. A. Pershing, Henrique Teles Maia, Maurizio M. Chiaramonte, Kevin Carlberg, and Eitan Grinspun. 2023. CROM: Continuous Reduced-Order Modeling of PDEs Using Implicit Neural Representations. In *Proceedings of the 11th International Conference on Learning Representations*.
- Siyuan Chen, Yixin Chen, Jonathan Panuelos, Otman Benckroun, Yue Chang, Eitan Grinspun, and Zhecheng Wang. 2025. Fast Subspace Fluid Simulation with a Temporally-Aware Basis. *ACM Trans. Graph.* 44, 4, Article 43 (July 2025), 18 pages. doi:10.1145/3730826
- Siyuan Chen, Zhecheng Wang, Yixin Chen, Yue Chang, Peter Yichen Chen, Eitan Grinspun, and Jonathan Panuelos. 2026. Factorized Neural Implicit DMD for Parametric Dynamics. *arXiv:2603.10995* [cs.LG]
- Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating Long Sequences with Sparse Transformers. In *arXiv preprint arXiv:1904.10509*.
- Mengyu Chu and Nils Thuerey. 2017. Data-driven synthesis of smoke flows with CNN-based feature descriptors. *ACM Trans. Graph.* 36, 4, Article 69 (July 2017), 14 pages. doi:10.1145/3072959.3073643
- The Newton Contributors. 2025. *Newton: GPU-accelerated physics simulation for robotics and simulation research*. <https://github.com/newton-physics/newton>
- Tri Dao. 2024. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *Proceedings of the 12th International Conference on Learning Representations*.
- Peter Eastman, Raimondas Galvelis, Raúl P. Peláez, Charles R. A. Abreu, Stephen E. Farr, Emilio Gallicchio, Anton Gorenko, Michael M. Henry, Frank Hu, Jing Huang, Andreas Krämer, Julien Michel, Joshua A. Mitchell, Vijay S. Pande, João PGLM Rodrigues, Jaime Rodriguez-Guerra, Andrew C. Simmonett, Sukrit Singh, Jason Swails, Philip Turner, Yuanqing Wang, Ivy Zhang, John D. Chodera, Gianni De Fabritiis, and Thomas E. Markland. 2023. OpenMM 8: Molecular Dynamics Simulation with Machine Learning Potentials. *arXiv:2310.03121* [physics.chem-ph]
- Lawson Fulton, Vismay Modi, David Duvenaud, David IW Levin, and Alec Jacobson. 2019. Latent-space dynamics for reduced deformable simulation. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 379–391.
- Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. 2023. Mastering Diverse Domains through World Models. *arXiv preprint arXiv:2301.04104* (2023).
- Benjamin Holzschuh, Georg Kohl, Florian Redinger, and Nils Thuerey. 2026. P3D: Scalable Neural Surrogates for High-Resolution 3D Physics Simulations with Global

- Context. In *Proceedings of the 14th International Conference on Learning Representations*.
- Kemeng Huang, Floyd M. Chitalu, Huancheng Lin, and Taku Komura. 2024. GIPC: Fast and Stable Gauss-Newton Optimization of IPC Barrier Energy. *ACM Trans. Graph.* 43, 2, Article 23 (March 2024), 18 pages. doi:10.1145/3643028
- Kemeng Huang, Xinyu Lu, Huancheng Lin, Taku Komura, and Minchen Li. 2025. StiffGIPC: Advancing GPU IPC for Stiff Affine-Deformable Simulation. *ACM Trans. Graph.* 44, 3, Article 31 (May 2025), 20 pages. doi:10.1145/3735126
- Shixun Huang, Eitan Grinspun, and Yue Chang. 2026. Odd-DC: Generalizable Neural Model Reduction via Odd Difference-of-Convex Structure. arXiv:2511.18241 [cs.GR]
- Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. In *International conference on machine learning*. PMLR, 4651–4664.
- Hanxiao Jiang, Hao-Yu Hsu, Kaifeng Zhang, Hsin-Ni Yu, Shenlong Wang, and Yunzhu Li. 2025. Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7219–7230.
- Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. 2025. LVSM: A Large View Synthesis Model with Minimal 3D Inductive Bias. In *Proceedings of the 13th International Conference on Learning Representations*.
- Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.
- Byungsoo Kim, Vinicius C. Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. 2019. Deep Fluids: A Generative Network for Parametrized Fluid Simulations. *Computer Graphics Forum* 38, 2 (2019), 59–70. doi:10.1111/cgfg.13619
- Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. 2022. A survey on SPH methods in computer graphics. In *Computer graphics forum*, Vol. 41. Wiley Online Library, 737–760.
- Jean Kossaifi, Nikola B. Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. 2024. Multi-Grid Tensorized Fourier Neural Operator for High-Resolution PDEs. *Transactions on Machine Learning Research* (2024).
- Jonáš Kulháněk, Erik Derner, Torsten Sattler, and Robert Babuška. 2022. ViewFormer: NeRF-free Neural Rendering from Few Images Using Transformers. In *Proceedings of the 17th European Conference on Computer Vision*.
- L'ubor Ladický, SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, and Markus Gross. 2015. Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–9.
- Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. 2019. Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids. In *Proceedings of the 7th International Conference on Learning Representations*.
- Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhat-tacharya, Andrew Stuart, and Anima Anandkumar. 2020. Multipole Graph Neural Operator for Parametric Partial Differential Equations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*.
- Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhat-tacharya, Andrew Stuart, and Anima Anandkumar. 2021. Fourier Neural Operator for Parametric Partial Differential Equations. In *Proceedings of the 9th International Conference on Learning Representations*.
- Zongyi Li, Nikola B. Kovachki, Christopher B. Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Animashree Anandkumar. 2023. Geometry-Informed Neural Operator for Large-Scale 3D PDEs. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*.
- Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. 2024. Physics-Informed Neural Operator for Learning Partial Differential Equations. *ACM/IMS Journal of Data Science* 1, 3 (2024), 1–27. doi:10.1145/3648506
- Zhouyingcheng Liao, Sinan Wang, and Taku Komura. 2024. SENC: Handling Self-collision in Neural Cloth Simulation. In *Proceedings of the 18th European Conference on Computer Vision*.
- Mengfei Liu, Yue Chang, Zhecheng Wang, Peter Yichen Chen, and Eitan Grinspun. 2025. Precise Gradient Discontinuities in Neural Fields for Subspace Physics. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers* (Vancouver, Canada) (SA Conference Papers '25). Association for Computing Machinery, New York, NY, USA, Article 26, 11 pages. doi:10.1145/3757377.3763810
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *Proceedings of the 7th International Conference on Learning Representations*.
- Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. 2021. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence* 3 (2021), 218–229. doi:10.1038/s42256-021-00302-5
- Pingchuan Ma, Peter Yichen Chen, Bolei Deng, Joshua B. Tenenbaum, Tao Du, Chuang Gan, and Wojciech Matusik. 2023. Learning Neural Constitutive Laws from Motion Observations for Generalizable PDE Dynamics. In *Proceedings of the 40th International Conference on Machine Learning*.
- Vismay Modi, Nicholas Sharp, Or Perel, Shinjiro Sueda, and David I. W. Levin. 2024. Simplicits: Mesh-Free, Geometry-Agnostic Elastic Simulation. *ACM Trans. Graph.* 43, 4, Article 117 (July 2024), 11 pages. doi:10.1145/3658184
- Jeremy Morton, Freddie D. Witherden, Antony Jameson, and Mykel J. Kochenderfer. 2018. Deep dynamical modeling and control of unsteady fluid flows. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montreal, Canada) (NIPS'18). Curran Associates Inc., Red Hook, NY, USA, 9278–9288.
- Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratchiff. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation* 18, 2 (2007), 109–118.
- Yatian Pang, Wenxiao Wang, Francis E. H. Tay, Wei Liu, Yonghong Tian, and Li Yuan. 2022. Masked Autoencoders for Point Cloud Self-supervised Learning. In *Proceedings of the 17th European Conference on Computer Vision*.
- Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. 2021. Learning Mesh-Simulation with Graph Networks. In *Proceedings of the 9th International Conference on Learning Representations*.
- Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* 378 (2019), 686–707. doi:10.1016/j.jcp.2018.10.045
- Bogdan Raonić, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézénac. 2023. Convolutional Neural Operators for Robust and Accurate Learning of PDEs. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*.
- Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. 2020. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 784, 10 pages.
- Connor Schenck and Dieter Fox. 2018. SPNets: Differentiable Fluid Dynamics for Deep Neural Networks. In *Proceedings of the 2nd Conference on Robot Learning*.
- Nicholas Sharp et al. 2019. Polyscope. www.polyscope.run.
- Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomput.* 568, C (Feb. 2024), 12 pages. doi:10.1016/j.neucom.2023.127063
- D. Sulsky, Z. Chen, and H.L. Schreyer. 1994. A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering* 118, 1 (1994), 179–196. doi:10.1016/0045-7825(94)90112-0
- Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. 2017. Accelerating Eulerian Fluid Simulation With Convolutional Networks. In *International Conference on Learning Representations Workshop*.
- Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. 2023. Factorized Fourier Neural Operators. In *Proceedings of the 11th International Conference on Learning Representations*.
- Adrien Treuille, Andrew Lewis, and Zoran Popović. 2006. Model reduction for real-time fluids. *ACM Trans. Graph.* 25, 3 (July 2006), 826–834. doi:10.1145/1141911.1141962
- Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. 2020. Lagrangian Fluid Simulation with Continuous Convolutions. In *Proceedings of the 8th International Conference on Learning Representations*.
- Yann Vander Meersche, Gabriel Cretin, Aria Gheeraert, Jean-Christophe Gelly, and Tatiana Galochkina. 2024. ATLAS: protein flexibility description from atomistic molecular dynamics simulations. *Nucleic Acids Research* 52, D1 (01 2024), D384–D392. doi:10.1093/nar/gkad1084
- Petr Vaněk, Jan Mandel, and Marian Brezina. 1996. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing* 56, 3 (1996), 179–196.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*.
- Prishikesh Viswanath, Yue Chang, Aleksey Panas, Julius Berner, Peter Yichen Chen, and Aniket Bera. 2024. Reduced-order neural operators: Learning lagrangian dynamics on highly sparse graphs. arXiv preprint arXiv:2407.03925 (2024).
- Chen Wang, Chuhaio Chen, Yiming Huang, Zhiyang Dou, Yuan Liu, Jiatao Gu, and Lingjie Liu. 2025. PhysCtrl: Generative Physics for Controllable and Physics-Grounded Video Generation. In *Proceedings of the 39th International Conference on Neural Information Processing Systems*.
- Haixin Wang, Jiaxin Li, Anubhav Dwivedi, Kentaro Hara, and Tailin Wu. 2024. BENO: Boundary-Embedded Neural Operators for Elliptic PDEs. In *Proceedings of the 12th International Conference on Learning Representations*.
- Peng-Shuai Wang. 2023. OctFormer: Octree-based Transformers for 3D Point Clouds. *ACM Trans. Graph.* 42, 4, Article 155 (July 2023), 11 pages. doi:10.1145/3592131

- Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snaveley, and Thomas Funkhouser. 2021. IBRNet: Learning Multi-View Image-Based Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Steffen Wiewel, Moritz Becher, and Nils Thuerey. 2019. Latent Space Physics: Towards Learning the Temporal Evolution of Fluid Flow. *Computer Graphics Forum* 38, 2 (2019), 71–82. doi:10.1111/cgf.13620
- Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. 2024b. Transolver: A Fast Transformer Solver for PDEs on General Geometries. In *Proceedings of the 41st International Conference on Machine Learning*.
- Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. 2024a. Point Transformer V3: Simpler, Faster, Stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. 2022. Point Transformer V2: Grouped Vector Attention and Partition-based Pooling. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.
- You Xie, Aleksandra Franz, Mengyu Chu, and Nils Thuerey. 2018. tempoGAN: a temporally coherent, volumetric GAN for super-resolution fluid flow. *ACM Trans. Graph.* 37, 4, Article 95 (July 2018), 15 pages. doi:10.1145/3197517.3201304
- Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. 2023. Swin3D: A Pretrained Transformer Backbone for 3D Indoor Scene Understanding. arXiv:2304.06906 [cs.CV]
- Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. 2021. PoinTr: Diverse Point Cloud Completion with Geometry-Aware Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.
- Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. 2022. Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Chong Zeng, Yue Dong, Pieter Peers, Hongzhi Wu, and Xin Tong. 2025. RenderFormer: Transformer-based Neural Rendering of Triangle Meshes with Global Illumination. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers* (Vancouver, Canada) (*SIGGRAPH Conference Papers '25*). Association for Computing Machinery, New York, NY, USA, Article 48, 11 pages. doi:10.1145/3721238.3730595
- Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. 2021. Point Transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.
- Zeshun Zong, Xuan Li, Minchen Li, Maurizio M. Chiamonte, Wojciech Matusik, Eitan Grinspun, Kevin Carlberg, Chenfanfu Jiang, and Peter Yichen Chen. 2023. Neural Stress Fields for Reduced-order Elastoplasticity and Fracture. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney, NSW, Australia) (*SA '23*). Association for Computing Machinery, New York, NY, USA, Article 78, 11 pages. doi:10.1145/3610548.3618207

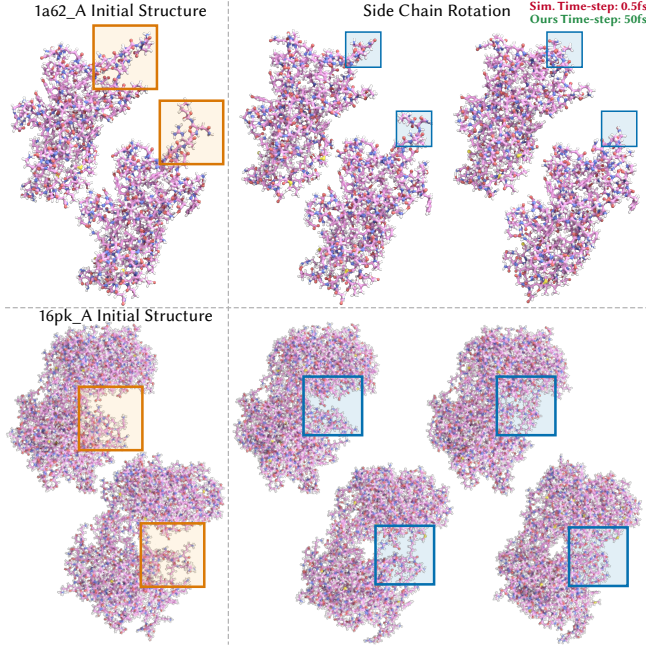


Fig. 6. **Side-chain rotation prediction on proteins 1a62_A and 16pk_A.** Starting from different initial conformations (orange), the model predicts side-chain rotational dynamics (blue) at 50 fs timesteps, 100× larger than the 0.5 fs steps used in traditional simulations.

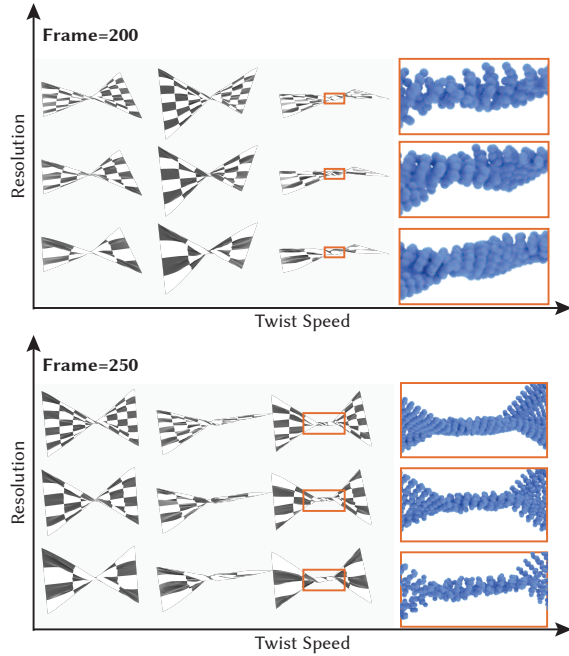


Fig. 7. **Long rollouts and unseen actuation.** Cloth-twisting sequences under actuation strengths not seen during training. The model remains stable up to 250 frames, beyond the 200-frame training horizon.

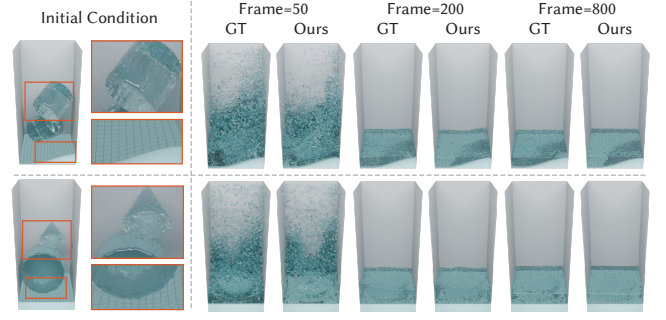


Fig. 8. **Generalization to unseen initial and boundary conditions.** First row: the training sequence whose initial configuration is closest to the test case in the second row. Orange boxes highlight differences in fluid volume and container geometry. Despite substantial differences, the model produces a stable 800-frame rollout that closely tracks the ground truth.

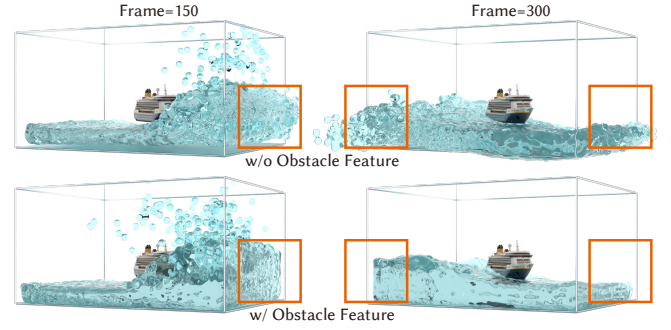


Fig. 9. **Ablation on particle-boundary interaction.** Fluid rollouts without (top) and with (bottom) the boundary branch of the particle tokenizer. Without boundary features, the model produces penetration.

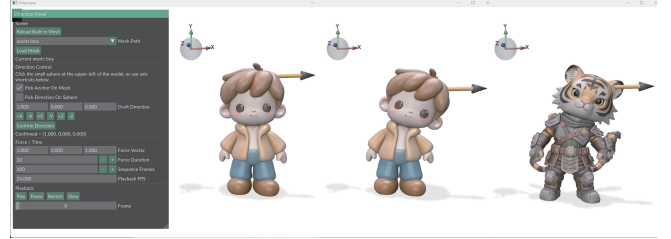


Fig. 10. **Interactive force control.** Left: user-specified force parameters (application region, direction, and duration). Right: three examples of the resulting model rollouts under forces not seen during training.

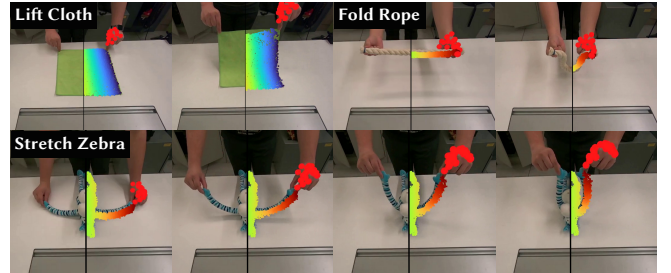


Fig. 11. **Real-world manipulation.** Predictions on three tasks from Phys-Twin [Jiang et al. 2025]: lifting cloth, stretching a zebra toy, and folding rope. Each subfigure: real-world observation (left) and predicted particle trajectories (right).

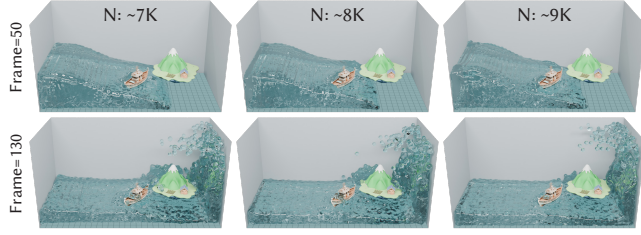


Fig. 12. **Generalization to particle sampling density.** The fluid model, trained on approximately 7k particles, produces stable rollouts under altered initial particle arrangements (7k) and increased particle counts (8k, 9k).

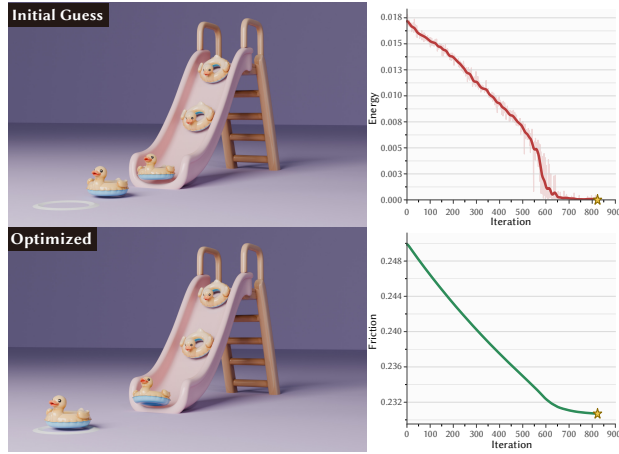


Fig. 13. **Inverse design.** Left: forward rollout under the initial friction guess (top) and the optimized friction (bottom). Right: design objective and recovered friction coefficient over optimization iterations.

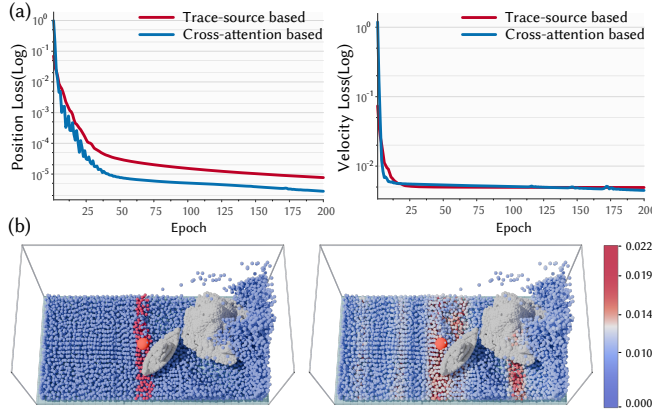


Fig. 14. **Cross-attention decoder vs. trace-source decoder.** (a) Training loss of two diagnostic probes attached to the same frozen super-token encoder. The cross-attention probe achieves lower position and velocity errors than the trace-source probe. (b) Influence maps for the two probes: warmer colors indicate stronger influence. The trace-source decoder shows primarily local support, while the cross-attention decoder captures non-local, motion-aware coupling.

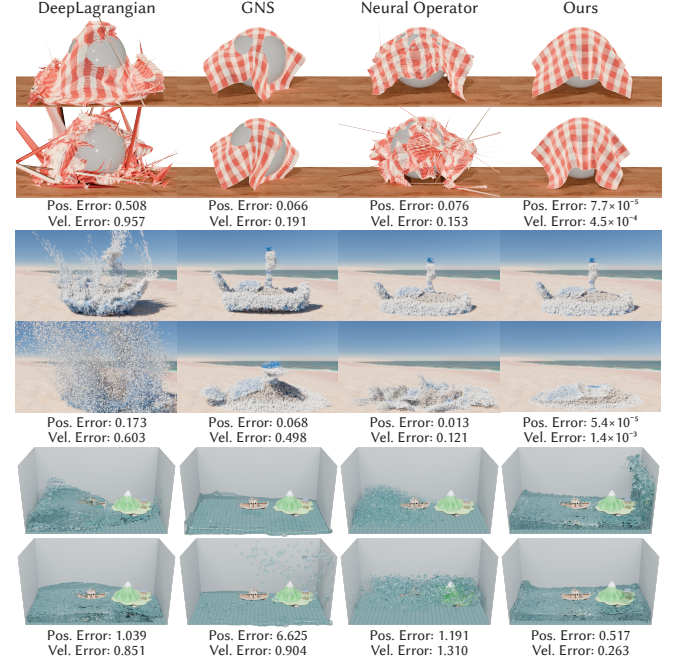


Fig. 15. **Baseline comparison on fluid, cloth, and sand.** Rollout sequences for DeepLagrangian [Ummenhofer et al. 2020], GNS [Sanchez-Gonzalez et al. 2020], Neural Operator [Li et al. 2021; Viswanath et al. 2024], and our model, compared against ground truth (GT). Reported errors are auto-regressive rollout MSE for position and velocity, averaged over particles, frames, and test sequences.

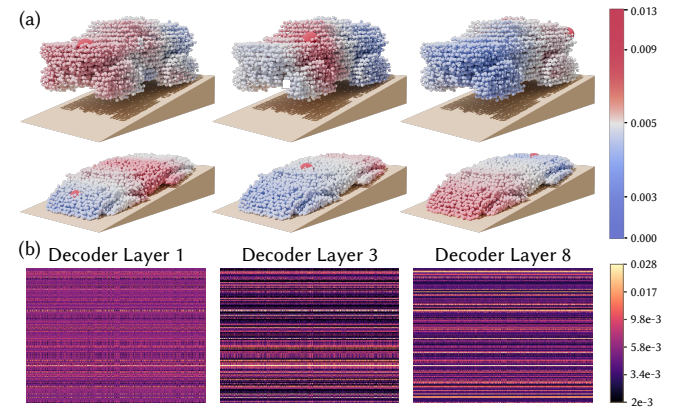


Fig. 16. **Decoder attention patterns.** (a) Influence maps of selected super tokens on particles at two frames across 8 decoder layers; warmer colors indicate stronger influence. Each super token affects spatially distributed particles, reflecting global coupling. (b) Cross-attention heat maps from decoder layers 1, 3, and 8 (rows: super tokens, columns: particles). Horizontal high-response bands indicate that super tokens induce globally coupled updates across many particles. Experiment uses the non-Newtonian model from Fig. 4.

A Method Details

A.1 Learnable Kernel for Particle Tokenizer

For branch $k \in \{s, b, \tau\}$, the kernel $\mathbf{W}_k$ is parameterized by a learnable 3D lattice

$$\Theta_k \in \mathbb{R}^{G \times G \times G \times C_{in}^k \times C_{out}^k},$$

where each vertex stores a $C_{in}^k \times C_{out}^k$ mixing matrix and G is the lattice resolution per axis. Given relative displacement $\mathbf{r}_{ij}^k$ with $\|\mathbf{r}_{ij}^k\| \leq R_k$ (zero otherwise), we normalize to grid coordinates:

$$\hat{\mathbf{r}}_{ij}^k = \mathbf{r}_{ij}^k / R_k, \quad \xi_d = \frac{\hat{r}_d + 1}{2}(G - 1), \quad d \in \{x, y, z\},$$

and evaluate $\mathbf{W}_k$ by trilinear interpolation over the eight enclosing lattice vertices:

$$\mathbf{W}_k(\mathbf{r}_{ij}^k) = \sum_{\delta \in \{0,1\}^3} \omega_{\delta}(t) \Theta_k[\mathbf{n} + \delta], \quad (11)$$

$$\omega_{\delta}(t) = \prod_{d \in \{x,y,z\}} (\delta_d t_d + (1 - \delta_d)(1 - t_d)), \quad (12)$$

where $\mathbf{n} = \lfloor \xi \rfloor$, $t = \xi - \mathbf{n}$, and $\Theta_k[\mathbf{n} + \delta]$ is the mixing matrix at lattice corner $\mathbf{n} + \delta$. The resulting kernel is continuous and piecewise linear inside the support radius R_k . Each neighbor contributes $\mathbf{W}_k(\mathbf{r}_{ij}^k) \mathbf{u}_{ij}^k$, and the branch output is $\mathbf{a}_i^k = \sum_{j \in \mathcal{N}_i^k} \mathbf{W}_k(\mathbf{r}_{ij}^k) \mathbf{u}_{ij}^k$.

A.2 Model Specification

The particle tokenizer uses three neighborhood branches with kernels on a $4 \times 4 \times 4$ learnable lattice at base width 384. The spatial and topology-guided branches are concatenated as two 192-channel streams, the boundary branch contributes 384 channels, and the particle self-feature branch contributes 384 channels, yielding $\mathbf{h}_i \in \mathbb{R}^{1152}$.

The super-token encoder applies $L_e = 6$ layers of self-attention and token merging at width 1152, with 16 attention heads (head dimension 72) and bipartite merging ($k = 2$). 3D RoPE with rotary dimension 72 is applied at every layer using current token coordinates.

The super-token decoder uses $L_d = 8$ layers, each consisting of cross-attention (particle queries, super-token keys/values), particle self-attention, and a feed-forward network, at width 1152 with 12 heads (head dimension 96), FFN hidden width 512, and dropout 0.1. 3D RoPE with rotary dimension 48 is applied in both cross-attention and self-attention, using particle positions for queries and super-token positions for keys.

The final prediction head maps $\mathbf{H}_d^{(L_d)}$ to $(\Delta \mathbf{x}_i, \Delta \mathbf{v}_i) \in \mathbb{R}^6$ through a five-layer MLP ($1152 \rightarrow 512 \rightarrow 512 \rightarrow 512 \rightarrow 6$), where the first four layers use Linear+ReLU+LayerNorm and the last is linear without activation.

The parameter count by module: tokenizer 249,600; super-token encoder 55,842,060; super-token decoder 99,255,296; prediction head 1,385,478; total 156,732,450.

B Experiment Details

B.1 Training Details

We train with AdamW [Loshchilov and Hutter 2019] ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 5×10^{-4} , base learning rate 1×10^{-4}). Training uses 8 NVIDIA A100 GPUs (80 GB each) with FlashAttention-2 [Dao

2024]. The learning rate follows a linear warmup over the first 8,000 steps (start factor 0.01), then cosine annealing over 400,000 steps to 5×10^{-6} . Batch size is 1; the number of epochs varies by category. One model is trained per dynamics category, shared across all systems within that category.

B.2 Main Results

Newtonian Fluids. We simulate dam-break dynamics in a container of size $4 \times 2 \times 2$ ($x \in [-2, 2]$, $y, z \in [-1, 1]$), where fluid interacts with two fixed rigid bodies: a boat and an island. Each sequence uses randomized fluid viscosity $\nu \in [4.0 \times 10^{-3}, 4.0 \times 10^{-2}]$ and is simulated for 600 frames at $\Delta t = 0.01$ s using DFSPH [Bender and Koschier 2015] in SPlisHSPlasH [Bender et al. [n.d.]]. The boat and island rotations around the y -axis are randomized; the boat centroid is sampled in $x, z \in [-0.3, 0.3]$, $y \in [0.38, 0.45]$; the island centroid is sampled in $x \in [0.8, 1.2]$, $z \in [-0.3, 0.3]$ with fixed $y = 0.45$.

Per-particle attributes $\mathbf{C}$ consist of the normalized viscosity $\hat{\nu} = (\nu - \mu_{\nu}) / (\sigma_{\nu} + 10^{-8})$, where μ_{ν} and σ_{ν} are the mean and standard deviation across sequences. Boundary particles ($\mathbf{X}^b, \mathbf{C}^b$) are sampled from the surfaces of the container walls (54,000 samples), boat (3,000), and island (3,000), with outward surface normals as boundary attributes $\mathbf{C}^b$. The dataset contains 200 sequences, split into 160 training, 20 validation, and 20 test.

Cloth. A triangular cloth mesh with 4,900 vertices falls onto a fixed sphere (radius 0.3, center height 0.3) above a ground plane at height -0.01 , released from height 0.62. For each sequence, Young’s modulus $E \in [1.0 \times 10^4, 2.0 \times 10^5]$ and density $\rho \in [100, 400]$ are randomized. Stiff-GIPC [Huang et al. 2024, 2025] generates 90-frame trajectories at $\Delta t = 0.02$ s.

Each cloth vertex is treated as a simulated particle. Per-particle attributes $\mathbf{C}$ consist of the normalized (E, ρ) repeated across all vertices. The topology $\mathcal{T}$ is the initial triangular mesh connectivity. Boundary particles ($\mathbf{X}^b, \mathbf{C}^b$) are sampled from the sphere surface (5,000 samples) and ground plane (5,000), with outward normals as $\mathbf{C}^b$. The dataset contains 200 sequences, split into 160 training, 20 validation, and 20 test.

Granular Sand. Sand piles are dropped onto a rigid ground plane, simulated with MPM [Stomakhin et al. 2013] in NVIDIA Newton [Contributors 2025]. Each sequence is initialized from one of 20 predefined mesh geometries, with the interior voxel-sampled into 10,000 particles. The internal friction coefficient is randomized within $\mu \in [0.20, 1.00]$; other parameters are fixed at density $\rho = 1000$, Young’s modulus $E = 1.0 \times 10^{15}$, and Poisson’s ratio $\nu = 0.3$. The ground plane has friction $\mu_g = 0.5$. Each sequence runs for 400 steps at $\Delta t = 0.005$ with 2 substeps per step.

Per-particle attributes $\mathbf{C}$ consist of the normalized particle radius $\hat{r}$ and normalized internal friction $\hat{\mu}$, each standardized by mean and standard deviation across sequences. Boundary particles ($\mathbf{X}^b, \mathbf{C}^b$) are 5,000 samples on the ground plane with upward-pointing normals as $\mathbf{C}^b$. The dataset contains 400 sequences, split into 360 training, 20 validation, and 20 test.

Non-Newtonian Fluids. A non-Newtonian fluid flows over a sloped rigid surface, simulated with MPM [Stomakhin et al. 2013] in NVIDIA

Newton [Contributors 2025]. Fluid particles are uniformly sampled within one of 10 predefined mesh geometries and released from height $z = 0.2$. The slope angle is randomized as $\theta \in [0^\circ, 6^\circ]$, with 0° and 6° guaranteed for each geometry and remaining angles sampled from the same range. Material parameters are randomized per sequence: Young’s modulus $E \in [9, 15]$ and damping $d \in [60, 90]$. Slope friction is fixed at 0.5. Each sequence runs for 250 frames at $\Delta t = 0.01$ s.

Per-particle attributes C consist of normalized Young’s modulus $\hat{E} = 0.2(E - 7) - 1$ and damping $\hat{d} = 0.04(d - 50) - 1$, replicated across all particles. Boundary particles (X^B, C^B) are 5,000 samples on the sloped ground with surface normals as C^B . The dataset contains 400 sequences, split into 360 training, 20 validation, and 20 test.

Elastic Solids. Elastic objects interact with a rigid sloped ground (10°), simulated with VBD [Chen et al. 2024] in NVIDIA Newton [Contributors 2025]. Each sequence uses one of 10 tetrahedralized assets with randomized initial rotation and zero initial velocity. Material parameters are fixed at Young’s modulus $E = 6.5 \times 10^5$ and Poisson’s ratio $\nu = 0.3$. Trajectories are 160 frames at $\Delta t = 0.005$ s with 80 substeps per frame and 150 VBD iterations per substep.

Each tetrahedral vertex is a simulated particle. Per-particle attributes C consist of normalized $(\hat{E}, \hat{\nu})$ replicated across all vertices. The topology $\mathcal{T}$ is the tetrahedral mesh connectivity. Boundary particles (X^B, C^B) are 5,000 samples on the sloped ground with surface normals as C^B . The dataset contains 400 sequences, split into 360 training, 20 validation, and 20 test.

Proteins. Molecular dynamics of proteins requires extremely small timesteps (e.g., 0.5 fs) due to stiff interatomic potentials, while biologically meaningful events such as side-chain rotations and folding occur on microsecond to millisecond timescales. Our model operates as a one-step predictor at a fixed coarser Δt , enabling rapid exploration of conformational changes without prohibitively long conventional simulations.

We generate trajectories for proteins 1a62_A (~2k atoms) and 16pk_A (~6k atoms) from the ATLAS dataset [Vander Meersche et al. 2024] using OpenMM [Eastman et al. 2023]. For each run, we first equilibrate in implicit solvent (AMBER14 + OBC1) using a Langevin integrator at 300 K, friction $\gamma = 1.0 \text{ ps}^{-1}$, and timestep 0.5 fs for 50,000 steps. From the equilibrated candidates, we select the initial state whose total energy is closest to the median among states satisfying $|E_k - \bar{E}|/|\bar{E}| \leq 0.01$, where $\bar{E}$ is the mean energy over all candidates sampled every 10 steps.

Starting from the selected state, we run Hamiltonian dynamics in vacuum with the amber14-all force field using Velocity Verlet at 0.5 fs. Each rollout contains 10,000 integration steps, with states recorded every 100 steps, yielding trajectories of length 100. The governing equations are Hamilton’s equations:

$$H(\mathbf{x}, \mathbf{p}) = \sum_i \frac{\|\mathbf{p}_i\|^2}{2m_i} + U(\mathbf{x}), \quad \dot{\mathbf{x}}_i = \frac{\partial H}{\partial \mathbf{p}_i}, \quad \dot{\mathbf{p}}_i = -\frac{\partial H}{\partial \mathbf{x}_i}. \quad (13)$$

Per-atom attributes are $\mathbf{c}_i = [m_i, \tau_i]$, where m_i is the atomic mass and $\tau_i \in \{0, \dots, 5\}$ is the element-type index (C, H, O, N, S, P $\mapsto$ 0:5). Topology $\mathcal{T}$ and boundary particles (X^B, C^B) are not used for this category. We repeat the procedure for 1,000 independent runs per

protein, each with a different initial state, and split into 800 training, 100 validation, and 100 test trajectories.

B.3 Generalization

Initial and boundary conditions. We use the dataset released by [Ummenhofer et al. 2020]. Each sequence selects one of 10 box-shaped containers and initializes 1, 2, or 3 fluid bodies with randomized scales, orientations, and initial velocities, following the original data-generation protocol. Sequences are 800 frames at $\Delta t = 0.02$ s.

Each SPH particle is a simulated particle. Per-particle attributes C consist of mass $m = 0.125$ and viscosity $\nu = 0.01$. Boundary particles (X^B, C^B) are sampled from the container surfaces with outward normals as C^B .

Long rollouts. We simulate boundary-driven cloth twisting. Each sequence uses a triangulated square cloth at one of three resolutions (40×40 , 50×50 , or 60×60 vertices). The boundary rotation-speed scale is randomized from $[\frac{1}{6}, \frac{5}{6}]$: forces are applied to two opposite boundary strips to induce counter-rotational motion, with larger angular velocity corresponding to stronger applied force. Initial velocity is zero. Trajectories are simulated using VBD [Chen et al. 2024] in NVIDIA Newton [Contributors 2025] with 4 iterations per substep, producing 250-frame sequences at $\Delta t = \frac{1}{30}$ s with 10 substeps per frame.

Each cloth vertex is a simulated particle. Per-particle attributes C consist of per-vertex mass. The topology $\mathcal{T}$ is the triangular mesh connectivity. Boundary particles (X^B, C^B) are the two actuated boundary strips, with direction features encoding the applied twisting action. The model is trained on the first 200 frames and evaluated with auto-regressive rollout up to 250 frames.

B.4 Ablation Studies

All models in this section are trained for 6 epochs, and the checkpoint with the lowest validation loss (auto-regressive rollout loss on the validation set) is selected. The default configuration uses training rollout horizon 5, spatial neighborhood radius $R_s = 0.007$, token embedding dimension 384, and encoder/decoder layers $L_e/L_d = 6/8$; its results are reported in Tab. 3 for reference.

Training rollout horizon. We vary the rollout horizon in $\{2, 3, 4, 6\}$ while keeping all other settings fixed. As reported in Tab. 2, longer rollouts provide stronger long-horizon supervision and reduce cumulative errors, at the cost of increased training time and memory.

Spatial neighborhood radius. The radius R_s defines the support of the spatial branch in the particle tokenizer, controlling each particle’s local interaction range. We vary $R_s \in \{0.002, 0.015, 0.030\}$ with other settings fixed. As shown in Tab. 2, a very small radius limits neighborhood coverage and degrades local interaction modeling. Increasing R_s improves rollout stability but increases the number of neighbors per particle and thus computational cost.

Token embedding dimension. We vary the embedding dimension while keeping $L_e/L_d = 6/8$ fixed. As shown in Tab. 3, increasing dimensionality initially reduces rollout error by providing greater

Table 2. **Hyperparameter ablation.** Effect of training rollout horizon and spatial neighborhood radius R_s on the cloth task. Position and velocity errors are rollout MSE ($\times 10^{-4}$), averaged over particles, frames, and test sequences. Peak Mem.: per-GPU peak training memory. Time/Epoch: single-GPU wall-clock time.

Variant	Peak Mem. (GB)	Time/Ep. (h)	Pos. MSE	Vel. MSE
<i>Rollout horizon</i>				
2	6.89	0.92	57.43	94.05
3	11.37	1.74	5.058	13.18
4	15.86	2.57	1.816	4.673
6	20.34	3.36	0.652	2.424
<i>Spatial radius R_s</i>				
0.002	20.34	3.29	0.944	3.289
0.015	20.35	3.52	0.701	2.018
0.030	20.41	4.10	0.404	1.062

Table 3. **Architectural ablation.** Effect of decoder composition, token embedding dimension, and encoder/decoder layer allocation on the cloth task. Position and velocity errors are rollout MSE ($\times 10^{-4}$), averaged over particles, frames, and test sequences. Peak Mem.: per-GPU peak training memory. Time/Ep.: single-GPU wall-clock time.

Variant	Peak Mem. (GB)	Time/Ep. (h)	Pos. MSE	Vel. MSE
<i>Decoder composition</i>				
Full (default)	20.33	3.35	0.831	3.058
w/o self-attn	12.17	2.39	10.50	23.48
w/o FFN	17.03	3.17	1.309	3.874
w/o self-attn & FFN	8.77	2.24	14.10	31.44
<i>Embedding dimension</i>				
192	10.19	1.88	1.006	3.054
256	14.04	2.17	0.817	2.664
384 (default)	20.35	3.36	0.831	3.058
512	27.73	4.75	0.907	2.472
<i>Layer allocation (L_e/L_d)</i>				
4 / 10	24.21	3.70	10.26	26.14
6 / 8 (default)	20.33	3.35	0.831	3.058
8 / 6	16.41	3.02	11.23	27.92
10 / 4	18.36	3.19	20.01	41.33

representational capacity. Beyond a certain width, however, performance degrades, likely due to overfitting at fixed data and training budget. Wider tokens also increase computational cost.

Layer configuration. We ablate the decoder block composition and the allocation of layers between encoder and decoder. Removing self-attention or the feed-forward network from the decoder block reduces cost but increases rollout error, confirming that both components contribute to the decoder’s capacity. We then vary the encoder-decoder depth split while keeping the total layer count fixed. Allocating too few layers to the encoder yields higher error due to insufficient token coarsening, while a shallow decoder limits the model’s ability to transfer super-token information back to particle resolution.

Interactive Control. Young’s modulus $E \in [1.0 \times 10^4, 3.5 \times 10^4]$ and Poisson’s ratio $\nu \in [0.24, 0.40]$ are randomized per sequence. External forces with magnitude $\|F\| \in [0.7, 1.6]$ are applied in the positive x half-plane, with direction, temporal window $([10, 30]$

frames), and application region randomized. The force region is an ellipsoidal surface patch centered at an anchor sampled in normalized bounding-box coordinates $(\alpha_x, \alpha_y, \alpha_z) \in [0.18, 0.82] \times [0.80, 0.98] \times [0.18, 0.82]$, with patch radius in $[0.10, 0.18]$. Surface vertices below the anchor y -coordinate are pinned. Each sequence is 100 frames at $\Delta t = 1/60$ s with 10 substeps per frame, simulated with VBD [Chen et al. 2024] in NVIDIA Newton [Contributors 2025]. We use 20 meshes with 20 sequences each, totaling 400 training trajectories.

Each tetrahedral vertex except pinned surface points is a simulated particle. Per-particle attributes C consist of normalized $(\hat{E}, \hat{\nu})$ replicated across all vertices. The topology $\mathcal{T}$ is the tetrahedral mesh connectivity. Boundary particles (X^B, C^B) are the pinned surface points with outward normals as C^B .

Inverse Design. We define the body particles at the final frame as $\mathcal{B} = \{i \mid z_{T,i}(\mu) \leq 0.2\}$ and compute their mean terminal x -position:

$$s(\mu) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} x_{T,i}(\mu).$$

The design objective minimizes the squared distance to a target stop position $x^* = 0.92$:

$$\mathcal{E}(\mu) = (s(\mu) - x^*)^2. \quad (14)$$

Gradients with respect to the friction coefficient μ are obtained by backpropagating through the full rollout. The coefficient is updated via gradient descent and mapped through a sigmoid to remain in a physically plausible range.

Training data consists of 10 sequences simulated with VBD [Chen et al. 2024] in NVIDIA Newton [Contributors 2025], varying only the friction coefficient $\mu \in [0.20, 0.40]$ over 10 uniformly spaced values. Other parameters are fixed: density $\rho = 1200$, Young’s modulus $E = 8.0 \times 10^5$, Poisson’s ratio $\nu = 0.3$, and damping $d = 0.004$. Each sequence is 99 rollout steps at $\Delta t = 0.02$ s with 50 substeps per frame; we subsample every fifth exported frame.

Per-particle attributes C consist of the friction coefficient μ , shared across all particles in a sequence. The topology $\mathcal{T}$ is the tetrahedral mesh connectivity. Boundary particles (X^B, C^B) are sampled from the ramp (10,000 points) and ground plane (10,000 points), with outward normals as C^B .