GPWM: GENERATIVE PARTICLE WORLD MODELS FOR 3D PREDICTION AND PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce GPWM, a generative world model that learns 3D physical dynamics directly from particle trajectories. Its shared representation and architecture covers rigid, deformable, cloth, and granular materials, in both passive physics and robotic interaction. GPWM is an object-aware multi-resolution transformer that combines flow matching with diffusion forcing to enable streaming generation far beyond the training horizon. We train and evaluate GPWM on PARTICLEWORLDS, a 25.5-million-frame corpus spanning diverse geometries, physical properties, interactions, and robot embodiments. When trained within individual domains, GPWM reduces prediction error by 29–53% relative to the strongest applicable prior learned-dynamics models. Going beyond the domain-specific setting of prior methods, a single *Unified* GPWM jointly models heterogeneous physical regimes within one set of weights. GPWM further generalizes to unseen geometries, physical properties, object counts, and rollout horizons. Beyond prediction, GPWM enables contact-rich planning. An object-space goal reward is backpropagated through the learned interaction dynamics to the latent noise, steering actuator motion and the contacts it establishes. Executed open-loop in simulation, the resulting plans outperform unguided generation and planning with prior learned dynamics models on pushing, picking, and granular manipulation. Anonymous project page: https://gpwm.github.io/

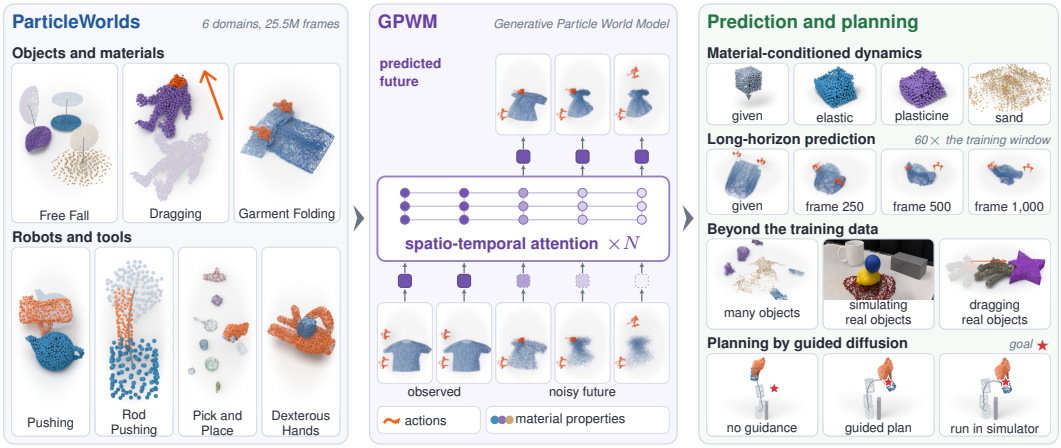

Figure 1: **GPWM learns physical dynamics across materials and embodiments.** From the observed history and, when available, actuator motion and material properties, GPWM denoises future particle trajectories. It enables material-conditioned prediction, generalization to unseen geometries, rollouts far beyond the training horizon, and planning via guided diffusion.

1 INTRODUCTION

A general-purpose physical world model should predict how a 3D scene will evolve and how to act to make it evolve toward a desired state. This requires modeling diverse objects and materials, maintaining physical consistency over long horizons, and reasoning through contact. Recent generative video world models capture rich and multimodal scene evolution (NVIDIA, 2026; Wan et al., 2025;

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Bruce et al., 2024; Guo et al., 2026), but pixel-space prediction lacks an explicit representation of 3D geometry, object state, and physical interaction. Particle- and point-based models instead represent physical state directly in 3D (Li et al., 2019; Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021; Huang et al., 2025a), yet existing learned 3D dynamics models typically remain limited along one or more axes: deterministic prediction, fixed short generation horizons, or specialization to particular materials or interaction regimes (Sanchez-Gonzalez et al., 2020; Shi et al., 2024; Zhang et al., 2025; Wang et al., 2025a; Chen et al., 2026). We ask: *can generative particle world models provide a common framework for diverse physical dynamics, long-horizon prediction, and contact-rich planning?*

We introduce GPWM, a generative world model of 3D physical trajectories (Figure 1). GPWM represents deformable dynamics through particle displacements and rigid dynamics through object-level transformations, providing a common architecture for rigid, deformable, and granular systems. It conditions on tracked 3D scene history and optionally physical properties and future actuator motion. Actuators share the same geometric representation as other scene components, so their trajectories can either condition physical prediction or be generated jointly with the scene for planning.

Scaling generative particle modeling presents two challenges: scenes may contain thousands of particles across multiple interacting objects, and small prediction errors can compound over long rollouts. GPWM addresses the first with object-aware multi-resolution tokenization and spatio-temporal attention in relative 3D coordinates. To address the second, we combine flow matching with *diffusion forcing*, independently varying the noise level across future frames. This enables *streaming generation*: near-future states are progressively resolved and committed while farther-future states remain uncertain and revisable.

We train GPWM on PARTICLEWORLDS, a 25.5-million-frame dataset of object and object–actuator interactions collected from physics solvers across diverse material behaviors, scene geometries, object interactions, and robot embodiments spanning rigid, elastic, elastoplastic, granular, and cloth dynamics. Prior learned dynamics models in our comparison are specialized to individual physical domains (Sanchez-Gonzalez et al., 2020; Shi et al., 2024; Zhang et al., 2025). When trained within the corresponding domains, GPWM reduces prediction error by 29–53% relative to the strongest evaluated baseline. Going beyond this domain-specific setting, we additionally train a single *Unified* GPWM across all six collections with one set of weights. GPWM also generalizes to unseen geometries, physical properties, and object counts.

Beyond predicting what *will* happen, a generative physical world model can be guided toward what *should* happen. When actuator motion is withheld, GPWM jointly generates actuator and passive-object trajectories. By backpropagating reward gradients to the initial trajectory noise (Eyring et al., 2024; Ben-Hamu et al., 2024; Karunratanakul et al., 2024), the same world model becomes a *contact-rich planner* without specifying contact modes or training a separate policy for each goal. We show that such latent noise optimization in GPWM produces the desired pushing, picking, and granular manipulation behaviors while sampling-based baselines fail to do so.

Our contributions are as follows. **(1)** We introduce **GPWM**, a generative 3D particle world-modeling framework spanning rigid, deformable, cloth, and granular dynamics, with stable streaming generation far beyond its training horizon. **(2)** We introduce **PARTICLEWORLDS**, a 25.5M-frame corpus spanning heterogeneous physical regimes, interactions, and robot embodiments. **(3)** GPWM achieves **state-of-the-art physical prediction** across diverse domains, generalizes to unseen geometries, physical properties, object counts, and rollout horizons, and enables a single model to be trained jointly across dramatically heterogeneous physical regimes. **(4)** We demonstrate **contact-rich planning through latent noise optimization**, outperforming MPC-based planning with prior learned dynamics models.

2 RELATED WORK

Physics simulators and learned 3D dynamics. Physics engines and differentiable simulators use explicit physical models for prediction and control (Todorov et al., 2012; Hu et al., 2020; Du et al., 2021), while learned simulators infer dynamics from trajectory data using particle and mesh representations (Li et al., 2019; Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021). RoboCraft (Shi et al., 2024) and RoboCook (Shi et al., 2023) model elastoplastic manipulation; ParticleFormer (Huang et al., 2025a) models multi-object, multi-material interactions; RigidFormer (Dou et al., 2026) targets multi-object rigid dynamics; and Particle-Grid Neural Dynamics (Zhang et al., 2025) uses a hybrid particle-grid representation for deformable dynamics. PointWorld (Huang et al., 2026) repre-

sents scene motion and robot actions as 3D point flows for forecasting and model-predictive control. These learned 3D dynamics models are predominantly domain-specific and deterministic. Deterministic prediction becomes increasingly restrictive as physical diversity grows: contact and interaction uncertainty can induce multiple plausible futures that must otherwise collapse to a single trajectory. GPWM instead learns a generative distribution over dynamics within a common architecture spanning rigid, deformable, cloth, and granular systems.

Generative physical dynamics and long-horizon prediction. The closest works to GPWM are PhysCtrl (Wang et al., 2025a) and PhysiFormer (Chen et al., 2026). PhysCtrl generates 3D point trajectories conditioned on physical parameters and forces across rigid, elastic, sand, and plasticine materials, while PhysiFormer uses a diffusion transformer with attention factorized across space, time, and objects to generate multi-object rigid and elastic trajectories. Both are trained over fixed, relatively short generation horizons and retain the full set of 3D points or vertices throughout the network, limiting scalability to densely represented scenes; PhysiFormer, for example, trains on scenes with at most 356 vertices. Both are also trained on substantially smaller datasets than PARTICLEWORLDS. In contrast, GPWM progressively compresses particles through a multi-resolution hierarchy, scaling to scenes with up to 12,000 particles, and combines this spatial scalability with diffusion forcing (Chen et al., 2024) for streaming generation far beyond the training horizon.

Generative models for planning. Diffuser (Janner et al., 2022) and Decision Diffuser (Ajay et al., 2022) formulate decision making as trajectory generation conditioned or guided by task objectives. Subsequent work explores adaptive planning (Liang et al., 2023), hierarchical trajectory generation (Li et al., 2023), and joint state–action generation for control (Huang et al., 2025b). Classifier- and reward-guided sampling typically steer intermediate noisy states during generation. In contrast, *latent noise optimization* (LNO), including D-Flow (Ben-Hamu et al., 2024) and Diffusion Noise Optimization (Karunratanakul et al., 2024), differentiates through the generative process to optimize the initial noise toward a downstream objective. GPWM applies LNO to a generative 3D physical world model: object-space rewards backpropagate through the coupled evolution of actuators and passive objects to guide actuator motion and the contacts it establishes.

3 METHOD

We formulate world modeling as conditional generation of 3D scene trajectories. A scene contains N tracked particles with positions $\mathbf{P}_t \in \mathbb{R}^{N\times 3}$ at frame t, grouped into objects, including K rigid bodies. We assume particle correspondences and object assignments are known. Given C observed frames $\mathbf{P}_{0:C-1}$, GPWM models a distribution over the next F frames, optionally conditioned on material properties and prescribed actuator motion.

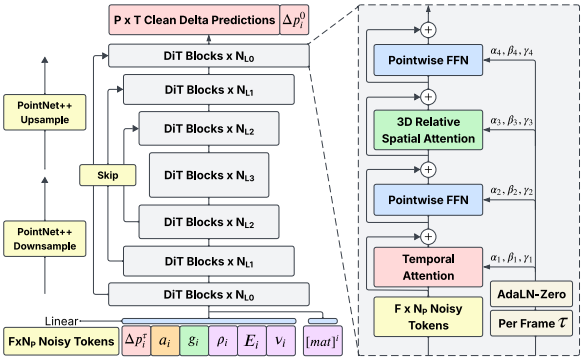

Figure 2: **GPWM Architecture Diagram.**

GPWM represents motion through particle displacements and rigid-body transformations (Section 3.1), processed by an object-aware, multi-resolution spatio-temporal transformer (Section 3.2). We train the model with flow matching and diffusion forcing (Section 3.3) and use a streaming denoising schedule to generate trajectories beyond the training window (Section 3.4). For planning, task-specific models jointly generate actuator and object trajectories, allowing rewards on object outcomes to guide actuator motion (Section 3.5).

3.1 PARTICLE AND RIGID-BODY REPRESENTATION

GPWM represents scene motion through particle displacements and rigid-body transformations. The generative variables reflect each component's physical structure, while particle geometry provides a shared representation for modeling interactions.

Particle representation. Let N_p denote the number of particles represented by individual displacement variables, with positions $\mathbf{P}^p_t \in \mathbb{R}^{N_p\times 3}$. Their frame-to-frame displacements are $\Delta\mathbf{P}^p_t = \mathbf{P}^p_{t+1} - \mathbf{P}^p_t$, and the clean future trajectory is $\mathbf{x}_0 = [\Delta\mathbf{P}^p_{C-1}, \ldots, \Delta\mathbf{P}^p_{C+F-2}] \in \mathbb{R}^{F\times N_p\times 3}$. Future positions are recovered by accumulating these displacements from the last observed frame. This makes the motion targets invariant to global translation. Where available, we condition on material

type and physical parameters: Young's modulus E, Poisson's ratio ν, and density ρ. Section 3.2 describes how motion, geometry, and conditioning information are embedded into particle tokens.

Rigid-body representation. Each rigid body k is represented by both a pose token and particle tokens. The pose token encodes its frame-to-frame transformation, $\Delta\mathbf{q}_k^t = [\Delta\mathbf{t}_k^t, \Delta\mathbf{r}_k^t] \in \mathbb{R}^9$, where $\Delta\mathbf{t}_k^t \in \mathbb{R}^3$ is a translation increment and $\Delta\mathbf{r}_k^t \in \mathbb{R}^6$ represents the relative rotation (Zhou et al., 2019). The clean future rigid-body trajectory is $\mathbf{y}_0 = [\Delta\mathbf{q}^{C-1}, \ldots, \Delta\mathbf{q}^{C+F-2}] \in \mathbb{R}^{F\times K\times 9}$. We recover future poses from the last observed pose by accumulating translations and composing the rotation matrices decoded from these increments. Applying each pose to fixed body-frame particles preserves rigid shape by construction. These particles do not receive their own noise; they move with the noisy pose of their rigid body. Pose and particle trajectories are denoised jointly by the multi-resolution spatio-temporal transformer.

Actuator conditioning. We represent actuator commands using the same geometric variables as scene motion: prescribed particle displacements $\mathbf{a}_i^t$ or rigid-body transformations. Rods, grippers, and articulated hands are therefore described through the motion of their physical components. When future actuator motion is supplied, GPWM conditions on it to generate the resulting object trajectories. For planning, task-specific models jointly generate actuator and object trajectories, allowing rewards on future object configurations to guide actuator motion (Section 3.5).

3.2 Multi-Resolution Spatio-Temporal Transformer

Direct attention over all particles and frames is prohibitively expensive for scenes containing thousands of particles. We therefore use a multi-resolution architecture that progressively compresses dense particles into a small set of anchor tokens, performs most global interaction modeling at coarse resolution, and propagates the resulting features back to the original particles (Figure 2).

Particle embeddings. Each particle i at frame t is embedded as

$$\mathbf{h}_i^t = \mathrm{MLP}\big([\Delta\mathbf{p}_i^t,\, \mathbf{g}_i^t,\, \boldsymbol{\phi}_i,\, \mathbf{a}_i^t]\big) + \mathbf{e}_{m_i},$$

where $\Delta\mathbf{p}_i^t$ is its frame-to-frame displacement, $\mathbf{g}_i^t$ encodes its displacement from its initial position together with its absolute height, $\boldsymbol{\phi}_i = [\log_{10} E_i, \nu_i, \rho_i]$ contains available physical parameters, and $\mathbf{e}_{m_i}$ is a learned embedding of material type m_i. For context frames, $\Delta\mathbf{p}_i^t$ is observed; for future frames, it is the corresponding noisy generative variable.

Multi-resolution particle hierarchy. Particle tokens remain attached to physical particles at every resolution. Coarse anchors are selected by farthest-point sampling (FPS) within each object from the initial geometry. PointNet++-style layers (Qi et al., 2017) transfer information between resolutions. During downsampling, each anchor a aggregates features from a neighborhood within the same object. Suppressing feature projections, normalization, and noise-dependent gates, this aggregation is written schematically as

$$\tilde{\mathbf{h}}_a^t = \max_{j\in\mathcal{N}(a,t)} \mathrm{MLP}\big([\mathbf{h}_j^t,\, \mathbf{h}_j^t - \mathbf{h}_a^t,\, \mathbf{p}_j^t - \mathbf{p}_a^t]\big), \qquad \mathbf{h}_a^t \leftarrow \mathbf{h}_a^t + \tilde{\mathbf{h}}_a^t.$$

During upsampling, coarse features are projected to the finer feature width and interpolated back to finer particles using inverse-distance weighting,

$$\mathbf{h}_i^t = \sum_{a\in\mathcal{N}(i,t)} w_{ia}\mathbf{h}_a^t, \qquad w_{ia} = \frac{(\|\mathbf{p}_i^t - \mathbf{p}_a^t\| + \epsilon)^{-1}}{\sum_{b\in\mathcal{N}(i,t)}(\|\mathbf{p}_i^t - \mathbf{p}_b^t\| + \epsilon)^{-1}}.$$

Here $\epsilon > 0$ handles coincident points. The interpolated features are fused with the corresponding encoder features through a gated MLP. Neighborhoods are recomputed at each frame from particle positions obtained by integrating the current noisy displacement trajectory. Anchor identities remain fixed throughout the temporal window while their positions evolve. Feature width increases as particle token count decreases, and most transformer blocks operate at coarse resolutions. Rigid-body pose tokens are retained at every resolution and participate in attention alongside particle tokens. Linear projections change their feature width between resolutions without pooling them.

Factorized spatio-temporal attention. At each resolution, transformer blocks factorize attention into temporal and spatial stages (Arnab et al., 2021). Temporal attention operates along each token trajectory within the denoising window, while spatial attention connects tokens within each frame. At full resolution, spatial attention is restricted within each object; at coarser resolutions, it spans the entire scene, allowing the model to capture interactions between objects and actuators. Each attention stage is followed by a feed-forward network, and blocks are modulated with adaLN-Zero (Peebles & Xie, 2023) using the frame-specific noise level. Encoder and decoder stages are connected by Hunyuan-DiT-style long skips (Li et al., 2024), preserving fine-scale information across the multi-resolution hierarchy.

Relative 3D geometry. Particle inputs encode displacement from the initial position, and PointNet++ aggregation uses offsets relative to each anchor. Spatial attention applies rotary position embeddings (Heo et al., 2024) to the 3D coordinates of particle and pose tokens. Absolute height is retained to represent vertical position relative to the ground plane.

3.3 Flow Matching with Diffusion Forcing

We train GPWM with flow matching (Lipman et al., 2022) and diffusion forcing (Chen et al., 2024). Rather than assigning a single noise level to the entire future trajectory, diffusion forcing samples an independent noise level $\tau_t \in [0, 1]$ for each future frame, collected in $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_F)$. We normalize particle displacements and rigid-body increments before adding noise. In this section, $\mathbf{x}_0$ and $\mathbf{y}_0$ denote these normalized trajectories. For the particle and rigid-body trajectories, respectively,

$$\mathbf{x}_{\boldsymbol{\tau}}^t = (1 - \tau_t)\mathbf{x}_0^t + \tau_t \boldsymbol{\epsilon}_x^t, \qquad \mathbf{y}_{\boldsymbol{\tau}}^t = (1 - \tau_t)\mathbf{y}_0^t + \tau_t \boldsymbol{\epsilon}_y^t,$$

where $\boldsymbol{\epsilon}_x^t, \boldsymbol{\epsilon}_y^t \sim \mathcal{N}(0, I)$. Thus different future frames can be presented to the model at different stages of the generative process.

Similar to Li & He (2026), we use an endpoint parameterization: rather than predicting the flow velocity directly, the network predicts the clean particle and rigid-body trajectories,

$$(\hat{\mathbf{x}}_0, \hat{\mathbf{y}}_0) = \mathcal{F}_\theta\left(\mathbf{x}_{\boldsymbol{\tau}}, \mathbf{y}_{\boldsymbol{\tau}}, \boldsymbol{\tau}, \mathbf{c}\right),$$

where $\mathbf{c}$ contains the observed context and any available physical properties and action conditioning. The flow field used for sampling is recovered from the noisy state and predicted clean endpoint. We do not reweight this objective in velocity space and instead directly optimize

$$\mathcal{L}(\theta) = \mathbb{E}\left[\|\hat{\mathbf{x}}_0 - \mathbf{x}_0\|_2^2 + \gamma\|\hat{\mathbf{y}}_0 - \mathbf{y}_0\|_2^2\right],$$

where γ balances the particle and rigid-body terms. No auxiliary physics-based losses are used.

3.4 Streaming Long-Horizon Generation

Independent per-frame noise levels allow different future frames to occupy different stages of denoising at inference time. Following diffusion forcing (Chen et al., 2024), we use a staggered denoising schedule: near-future frames are progressively resolved, while farther-future frames remain at higher noise levels and continue to be revised. Once the earliest frames are fully denoised, they are committed, the prediction window advances, and fresh Gaussian noise is appended at the far end of the horizon. Repeating this process allows GPWM to generate trajectories beyond the F-frame training horizon while keeping the model's temporal window fixed.

3.5 Reward-Guided Planning

For planning, task-specific models jointly generate actuator and object trajectories conditioned on the scene. A differentiable reward specifies the desired final configuration of the passive objects. With the world model fixed, the object-space reward guides actuator motion through the learned joint dynamics, without requiring a reference actuator trajectory or training a goal-specific policy.

We implement reward guidance through latent noise optimization (Eyring et al., 2024; Ben-Hamu et al., 2024; Karunratanakul et al., 2024). Let $\mathbf{z}$ collect the Gaussian noise used to initialize an H-frame rollout, and let $\hat{\mathbf{s}}_{1:H}(\mathbf{z})$ denote the generated scene trajectory in physical coordinates, reconstructed by denormalizing and integrating the predicted motion. With Π_{pass} selecting the passive-object configuration from a scene state, a gradient ascent step is

$$\mathbf{z} \leftarrow \mathbf{z} + \eta \nabla_{\mathbf{z}} r(\Pi_{\text{pass}}(\hat{\mathbf{s}}_H(\mathbf{z}))),$$

where η is the guidance step size.

For horizons $H > F$, we generate the rollout using the streaming procedure in Section 3.4. Gradients propagate through the rollout across window boundaries, allowing the terminal reward to guide earlier actuator motion. In our experiments, the resulting actuator trajectory is executed open-loop in simulation (Section 5.3). Additional details are given in Appendix E.3.

4 ParticleWorlds Dataset

We introduce ParticleWorlds, a dataset of 25.5 million simulated training frames spanning rigid objects, elastic and elastoplastic solids, granular materials, and cloth. Its six domains cover passive dynamics and robotic manipulation with rods, parallel-jaw grippers, and articulated hands (Appendix A). Scenes range from isolated objects to cluttered environments, with representations

containing 256–12,000 particles and geometries including irregular objects and garments from ten categories. This breadth supports both domain-specific models and a unified world model trained across materials and robot embodiments.

We generate new trajectories for falling objects, deformable dragging, garment manipulation, and dexterous manipulation, and incorporate existing recordings of rod and gripper pushing and of pick-and-place. To combine these sources, we convert simulator outputs into the common particle representation of Section 3.1, preserving particle identities and object membership over time. Deformable bodies use tracked material or mesh points; rigid bodies and actuator links use fixed surface samples transformed by their recorded poses. Dataset sources, generation procedures, and split definitions are detailed in Appendix A.

5 Experiments

We evaluate GPWM along three dimensions: **(1) prediction** across rigid, deformable, granular, and cloth dynamics; **(2) generalization** across geometry, physical properties, scene complexity, and rollout horizon, including joint training across heterogeneous regimes; and **(3) contact-rich planning** from sparse object-space rewards.

Baselines. We compare against four learned 3D dynamics models: GNS (Sanchez-Gonzalez et al., 2020), a graph-based particle simulator; PGND (Zhang et al., 2025), a hybrid particle–grid model; PointWorld (Huang et al., 2026), a point-cloud world model; and PhysCtrl (Wang et al., 2025a), a generative model of physical dynamics. Each is evaluated where its conditioning applies: PhysCtrl on all domains, GNS on non-action-conditioned domains, PointWorld on robot-actuated domains, and PGND on robot-actuated domains and Dragging. We report domain-specific GPWM models (**Specialist**) for direct comparison and a single **Unified** model trained across all domains.

Experimental setup. Unless stated otherwise, GPWM observes four context frames and predicts 16 future frames per window. Predictions are rolled forward to the end of each trajectory. Recorded actuator motion is provided for action-conditioned tasks; passive tasks require no action input. All methods use the same evaluation trajectories and available actuator conditioning.

Metrics. For physical prediction, we report mean Euclidean position error (mm, ↓) over passive-object particles, averaging over particles and frames and then equally across trajectories. We evaluate one sample from GPWM. Because physical interactions can be multimodal, error against a single recorded future does not fully capture physical plausibility; we therefore provide extensive qualitative rollouts on our project webpage `https://gpwm.github.io/`. For cloth, we report symmetric Chamfer distance (↓). Planning is instead evaluated by task outcome, using final object-to-goal error, directional displacement, or task success as appropriate.

5.1 Prediction across Physical Regimes

Table 1 compares prediction across six physical settings spanning rigid, deformable, granular, and cloth dynamics.

State-of-the-art prediction across diverse physics. GPWM specialists outperform the strongest applicable baseline on five of six settings, reducing error by **29–53%**, and matching performance on Dexterous Hands (24.7 vs. 24.8 mm). The gains span qualitatively different dynamics: error decreases from 66.4 to 44.4 mm on Free Fall, 12.8 to 8.8 mm on Dragging, and 24.5 to 17.5 mm on Pushing. Together, these results show that the same generative particle formulation achieves strong prediction accuracy across substantially different physical regimes.

One model across heterogeneous physics. A single Unified GPWM spans heterogeneous materials, interactions, and actuator embodiments within one set of weights. It outperforms all evaluated prior methods on Dragging, Garment Folding, and Pick and Place, and achieves the best overall performance on Pick and Place, improving even over the GPWM specialist. Domain-specific specialists remain more accurate on the other settings, revealing a specialization gap while demonstrating that substantially different physical regimes can be captured within a single model.

GPWM maintains stability far beyond its training horizon. We next evaluate long-horizon stability in Figure 3. Although trained with only 16-frame prediction windows, GPWM maintains coherent cloth dynamics over **1,000-frame rollouts** ($\approx 60\times$ the training horizon). At frame 1,000, GPWM achieves 34.8 mm symmetric Chamfer distance, compared with 73.4 mm for PointWorld, 123.3 mm for PhysCtrl, and 159.4 mm for PGND. Removing diffusion forcing increases error to 75.5 mm despite comparable short-horizon accuracy. The widening gap with rollout length shows that diffusion forcing is critical for long-horizon stability.

Table 1: **Prediction across diverse physical settings.** Prediction error in millimeters (↓). Baselines and Ours (Specialist) are trained separately for each setting; Ours (Unified) uses a single model across heterogeneous settings. Graph/grid-based denotes GNS or PGND, depending on the setting. †Garment Folding reports average symmetric Chamfer distance at the 1,000th predicted frame.

Method	Free Fall	Dragging	Garment Folding†	Pushing	Pick and Place	Dexterous Hands
Graph/grid-based	97.2	48.6	159.4	158.7	–	–
PointWorld	–	–	73.4	32.7	171.2	27.4
PhysCtrl	66.4	12.8	123.3	24.5	34.1	24.8
Ours (Specialist)	**44.4**	**8.8**	**34.8**	**17.5**	20.3	**24.7**
Ours (Unified)	73.2	11.4	64.0	27.2	**17.7**	32.3

5.2 Generalization across Geometry and Physics

We conduct controlled generalization experiments in the Free Fall setting, where geometry, object count, and material properties can be varied independently. Table 2 evaluates GPWM as each factor moves beyond the training distribution. GPWM achieves the lowest prediction error across all evaluated conditions.

GPWM generalizes to unseen shapes and larger scenes. Free Fall training scenes contain one to four objects. At test time, we introduce unseen geometries and increase scene complexity to five to seven objects, reaching 7,168 particles. Unseen shapes increase specialist error by less than 7%, while combining unseen shapes with larger scenes raises error only from 41.6 to 51.8 mm. In contrast, PhysCtrl increases from 58.7 to 138.6 mm. The Unified model is also stable across these shifts and ranks second on both larger-scene settings.

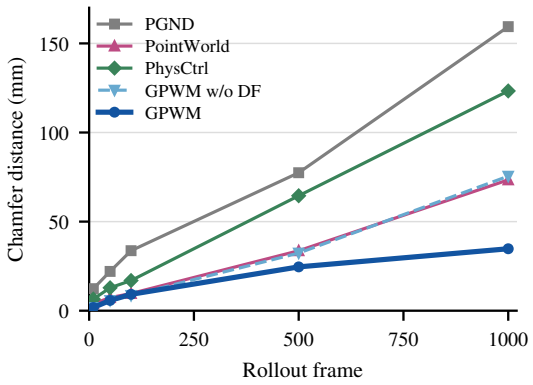

Figure 3: **Long-horizon cloth prediction.** Chamfer distance over the rollout horizon, given recorded gripper motion.

Table 2: **Generalization on Free Fall.** Mean particle position error in millimeters (↓). Training uses one to four objects per scene. Stiffness values are seen (10, 100, 1,000 kPa), unseen within the training range (30, 300 kPa), or outside it (3 kPa, 3 MPa). Material types are weighted equally; plasticine is excluded at 3 kPa because simulation fails on impact.

	1–4 objects		5–7 objects		Stiffness		
Method	Seen shapes	Unseen shapes	Seen shapes	Unseen shapes	Seen	Unseen	Out of range
GNS	96.8	97.2	100.9	97.6	116.7	113.2	110.6
PhysCtrl	58.7	66.4	136.0	138.6	69.3	67.7	64.1
Ours (Specialist)	**41.6**	**44.4**	**50.5**	**51.8**	**51.5**	**52.3**	**50.7**
Ours (Unified)	69.0	73.2	73.0	73.3	90.8	87.5	84.2

GPWM generalizes to unseen material stiffness. We next vary Young's modulus from 3 kPa to 3 MPa, covering values seen during training, unseen values within the training range, and extrapolation beyond it. Models receive the material parameters and initial observations for each simulation. GPWM achieves the lowest error in all three regimes—51.5, 52.3, and 50.7 mm, respectively—with less than 2 mm variation across them. This demonstrates robust prediction under both interpolation and extrapolation of material stiffness.

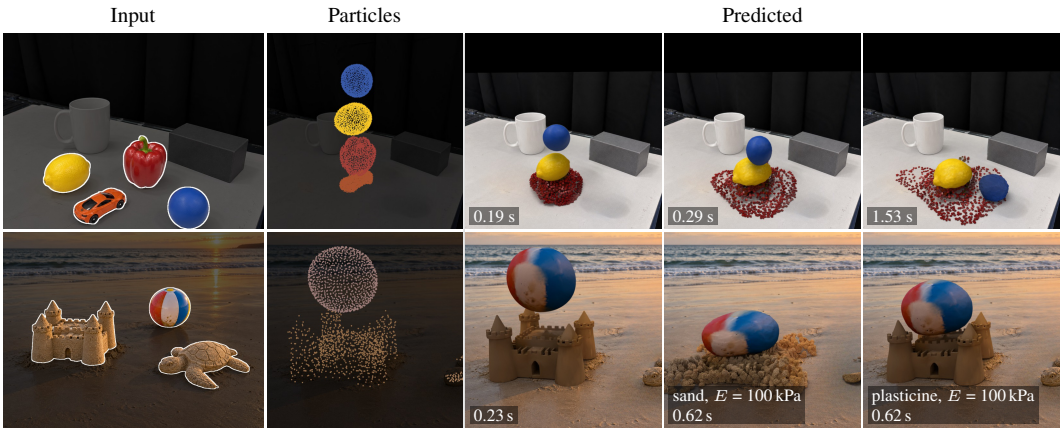

Figure 4: **Predictions from a single image.** Each row shows the input with the reconstructed objects outlined, their particles, and the predicted motion. Top, a tower of a toy car (elastic, 1 MPa), a pepper (sand, 100 kPa), a lemon (plasticine, 100 kPa), and a ball (elastic, 100 kPa). Bottom, an elastic ball (10 kPa) dropped onto a sand or a plasticine castle (100 kPa).

GPWM generalizes to objects reconstructed from images. We also test GPWM on previously unseen object geometries reconstructed from single RGB images (Figure 4). SAM 3 (Carion et al., 2025) segments the objects, SAM 3D (SAM 3D Team et al., 2025) reconstructs their meshes, and MoGe-2 (Wang et al., 2025b) estimates their metric scene placement. We sample surface particles from the reconstructed meshes, assign physical properties, and use the Free Fall specialist to predict their dynamics. The resulting trajectories are rendered back into the scene, demonstrating that GPWM can operate on reconstructed real-world geometry rather than only simulator-native assets.

5.3 Reward-Guided Planning

We next ask whether GPWM can use its learned physical dynamics not only to predict interactions, but to generate actuator motions that achieve desired object-space outcomes. We use frozen task-specific world models: rigid-object models trained on pushing and pick-and-place interactions, and a granular model trained on random paddle–pile interactions. Each task specifies only a terminal reward on the passive-object configuration. Following Section 3.5, latent noise optimization (LNO) backpropagates this reward through the jointly generated object–actuator trajectory to guide actuator motion and the contacts it establishes. The resulting actuator trajectory is then executed **open-loop in simulation**; thus, success requires plans optimized through the learned world model to transfer to the underlying simulator.

Tasks and comparisons. **Pushing** requires moving a rigid object toward a target over 60 steps; **Pile Manipulation** requires displacing the centroid of a granular pile in a specified direction over 150 steps; and **Picking** requires grasping and lifting an object to within 6 cm of its target. We report final object-to-target distance, displacement in the target direction, and success rate, respectively.

We compare LNO against sampling-based planning with PointWorld, GNS, and unguided GPWM. These baselines follow an MPPI-style procedure (Williams et al., 2016): candidate actuator trajectories are evaluated by rolling them out in the corresponding learned world model over a 5-step lookahead, and planning is repeated over the task horizon. PointWorld uses randomly sampled action candidates, while for GNS and unguided GPWM the candidate actuator trajectories are generated jointly with the predicted object motion. We cap planning time at 200 s and report runtime on a single L40S GPU. Further details are provided in Appendix E.

LNO in GPWM turns object-space goals into actuator trajectories. Given only a desired object-space outcome, LNO recovers actuator trajectories that realize the goal through learned physical interactions. On Picking, guided GPWM succeeds in **21/25 trials (84%)**, compared with 4/25 (16%) without guidance and 0/25 for PointWorld and GNS (Table 3). Guidance also nearly halves Pushing error from 8.26 to 4.26 cm and increases Pile Manipulation displacement from 1.48 to 3.07 cm (**107%**). Importantly, the inferred actuator trajectories are executed open-loop in the simulator: their success therefore demonstrates that LNO can translate object-space goals into actuator

Table 3: **Reward-guided planning.** Task performance and planning time (max 200 s) after open-loop execution of generated actuator trajectories in simulation. Results are aggregated over 50 seeds for Pushing and Pile Manipulation and 25 seeds for picking (5 lift targets for 5 objects).

	Pushing		Pile Manip.		Picking	
Method	Error (cm) ↓	Time (s)	Disp. (cm) ↑	Time (s)	Success ↑	Time (s)
PointWorld	8.30 ± 1.83	65	1.28 ± 1.07	189	$0/25$	200
GNS	8.19 ± 2.00	5	0.36 ± 0.19	200	$0/25$	200
Ours	8.26 ± 4.33	61	1.48 ± 0.76	152	$4/25$	190
Ours w/ guidance	$\mathbf{4.26 \pm 2.29}$	43	$\mathbf{3.07 \pm 0.99}$	114	$\mathbf{21/25}$	127

motions—including the contacts needed to achieve them—without a reference action trajectory or goal-specific policy.

Additional qualitative results—including comparisons with baselines, generalization to real-world images, and reward-guided planning examples—are available on our anonymous project page: https://gpwm.github.io/.

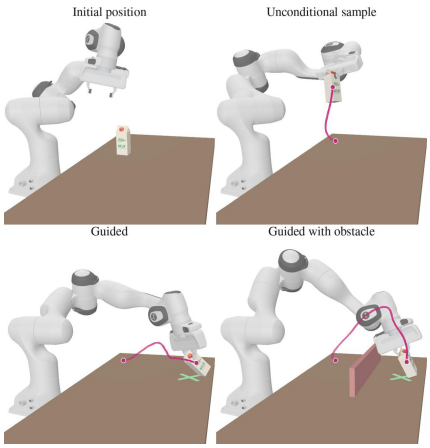

Figure 5: **GPWM Guidance.** Latent noise optimization steers a frozen GPWM to move an object to a target (green ×). When a wall blocks the direct path (bottom right), guidance finds a trajectory over it.

6 Limitations and Future Directions

Several limitations remain. First, GPWM assumes access to complete, temporally tracked 3D scene state. Our experiments therefore isolate physical prediction from perception and do not address recovering persistent particles and object identities from raw visual observations. An important next step is to integrate 3D reconstruction and tracking, ultimately learning physical world models directly from RGB or RGB-D video. Second, although a single Unified GPWM can model heterogeneous physical regimes within one set of weights, it currently underperforms domain-specific models on several settings. This specialization gap suggests interference between heterogeneous dynamics and motivates better data balancing, better-matched point density across domains, domain-aware routing or mixture-of-experts architectures, and increased model capacity as the diversity of physical experience grows. Finally, latent noise optimization requires differentiating through the full generative process and, for long-horizon tasks, through multiple streaming windows. Its computational cost therefore grows with planning horizon and currently limits rapid online control. Faster generative parameterizations, including flow maps (Boffi et al., 2025a;b), could substantially reduce the cost of long-horizon reward-guided planning, an exciting direction for future work.

7 Conclusion

We introduced GPWM, a generative 3D particle world-modeling framework for physical prediction and contact-rich planning. Across rigid, deformable, granular, and cloth dynamics, domain-specific GPWM models outperform prior learned dynamics approaches, while a single Unified model demonstrates that heterogeneous physical regimes can be modeled within one set of weights. Flow matching with diffusion forcing enables streaming generation far beyond the training horizon, including coherent deformable-object prediction for more than $60\times$ the trained prediction window. Beyond prediction, reward guidance turns the same generative formulation into a contact-rich planner: sparse objectives on desired object configurations backpropagate through learned object–actuator interactions to guide actuator motion and the contacts it establishes. Together, these results establish generative particle world models as a common framework for long-horizon physical prediction and goal-directed interaction in 3D.

AI USE STATEMENT

In this work, we used generative AI tools for cleaning and reformatting datasets (filtering, indexing and downsampling the simulated collections), designing and giving feedback on experiments and evaluation protocols, and debugging model training. A generative 3D reconstruction model, SAM 3D (SAM 3D Team et al., 2025), produced the object models reconstructed from single photographs in our real world experiments. The training and evaluation data of PARTICLEWORLDS are produced by physics simulators rather than by generative models. We have not used generative AI tools to develop theoretical models or conceptual frameworks, and formulating or proving mathematical claims, translation, and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools for tasks with recommended disclosure, namely creating and editing scientific figures and software code, suggesting the structure of sections, and searching for information. We have reviewed all AI-assisted work. Every reported number is computed by scripted evaluation from saved predictions on fixed lists of test trajectories. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Training and quantitative evaluation use simulated physical trajectories generated with physics simulators and 3D assets. Our qualitative real-image experiments use object photographs for 3D reconstruction and contain no human subjects or personally identifiable information.

REPRODUCIBILITY STATEMENT

We will publicly release our code and the PARTICLEWORLDS dataset. Section 3 describes the model architecture, the flow-matching and diffusion-forcing training objective (Section 3.3), and the streaming inference (Section 3.4) and reward-guided planning (Section 3.5) procedures. Section 4 and Appendix A describe the dataset, Section 5 describes the baselines and evaluation metrics for every experiment, and Appendices B and C give model, training, and evaluation details. Appendix E.2 details the planning baselines.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6836–6846, 2021.

Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-Flow: Differentiating through flows for controlled generation. *arXiv preprint arXiv:2402.14017*, 2024.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation, 2025a. URL https://arxiv.org/abs/2505.18825.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models, 2025b. URL https://arxiv.org/abs/2406.07507.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

Nicolas Carion et al. SAM 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

Yiming Chen, Yushi Lan, and Andrea Vedaldi. PhysiFormer: Learning to simulate mechanics in world space. *arXiv preprint arXiv:2606.27364*, 2026. URL https://arxiv.org/abs/2606.27364.

Zhiyang Dou, Minghao Guo, Haixu Wu, Doug Roble, Tuur Stuyck, and Wojciech Matusik. RigidFormer: Learning rigid dynamics using transformers. *arXiv preprint arXiv:2605.09196*, 2026.

Tao Du, Kui Wu, Pingchuan Ma, et al. DiffPD: Differentiable projective dynamics. *ACM Transactions on Graphics*, 2021.

Luca Eyring, Shyamgopal Karthik, Karsten Roth, Alexey Dosovitskiy, and Zeynep Akata. ReNO: Enhancing one-step text-to-image models through reward-based noise optimization. *Advances in Neural Information Processing Systems*, 37:125487–125519, 2024.

Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis.

Yanjiang Guo, Lucy Shi, Jianyu Chen, and Chelsea Finn. Ctrl-World: A controllable generative world model for robot manipulation. In *ICLR*, 2026.

Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *European Conference on Computer Vision*, pp. 289–305. Springer, 2024.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, et al. DiffTaichi: Differentiable programming for physical simulation. *ICLR*, 2020.

Suning Huang, Qianzhong Chen, Xiaohan Zhang, Jiankai Sun, and Mac Schwager. ParticleFormer: A 3D point cloud world model for multi-object, multi-material robotic manipulation, 2025a. URL https://arxiv.org/abs/2506.23126.

Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. PointWorld: Scaling 3D world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782*, 2026.

Xiaoyu Huang, Takara Truong, Yunbo Zhang, Fangzhou Yu, Jean Pierre Sleiman, Jessica Hodgins, Koushil Sreenath, and Farbod Farshidian. Diffuse-CLoC: Guided diffusion for physics-based character look-ahead control, 2025b. URL https://arxiv.org/abs/2503.11801.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Korrawe Karunratanakul, Konpat Preechakul, Emre Aksan, Thabo Beeler, Supasorn Suwajanakorn, and Siyu Tang. Optimizing diffusion noise can serve as universal motion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1334–1345, 2024.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*, pp. 20035–20064. PMLR, 2023.

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *ICLR*, 2019.

Zhimin Li, Jianwei Zhang, Qin Lin, Jiangfeng Xiong, Yanxin Long, Xinchi Deng, Yingfang Zhang, Xingchao Liu, Minbin Huang, Zedong Xiao, Dayou Chen, Jiajun He, Jiahao Li, Wenyue Li, Chen Zhang, Rongwei Quan, Jianxiang Lu, Jiabin Huang, Xiaoyan Yuan, Xiaoxiao Zheng, Yixuan Li, Jihong Zhang, Chao Zhang, Meng Chen, Jie Liu, Zheng Fang, Weiyan Wang, Jinbao Xue, Yangyu Tao, Jianchen Zhu, Kai Liu, Sihuan Lin, Yifu Sun, Yun Li, Dongdong Wang, Mingtao Chen,

Zhichao Hu, Xiao Xiao, Yan Chen, Yuhong Liu, Wei Liu, Di Wang, Yong Yang, Jie Jiang, and Qinglin Lu. Hunyuan-DiT: A powerful multi-resolution diffusion transformer with fine-grained Chinese understanding, 2024. URL https://arxiv.org/abs/2405.08748.

Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. AdaptDiffuser: Diffusion models as adaptive self-evolving planners. *arXiv preprint arXiv:2302.01877*, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

NVIDIA. Cosmos 3: Omnimodal world models for physical AI, 2026. URL https://arxiv.org/abs/2606.02800.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03409.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

SAM 3D Team, Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jiawei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. SAM 3D: 3Dfy anything in images, 2025. URL https://arxiv.org/abs/2511.16624.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, et al. Learning to simulate complex physics with graph networks. In *ICML*, 2020.

Haochen Shi, Huazhe Xu, Samuel Clarke, et al. RoboCook: Long-horizon elasto-plastic object manipulation with diverse tools. *arXiv preprint arXiv:2306.14447*, 2023.

Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. RoboCraft: Learning to see, simulate, and shape elasto-plastic objects in 3D with graph networks. *The International Journal of Robotics Research*, 43(4):533–549, 2024.

Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with Fourier convolutions. In *WACV*, 2022.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033, 2012.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL https://arxiv.org/abs/2503.20314.

Chen Wang, Chuhao Chen, Yiming Huang, Zhiyang Dou, Yuan Liu, Jiatao Gu, and Lingjie Liu. PhysCtrl: Generative physics for controllable and physics-grounded video generation. *NeurIPS*, 2025a.

Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. MoGe-2: Accurate monocular geometry with metric scale and sharp details. *arXiv preprint arXiv:2507.02546*, 2025b.

Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016.

Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Particle-grid neural dynamics for learning deformable object models from RGB-D videos. *arXiv preprint arXiv:2506.15680*, 2025.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5745–5753, 2019.

Yunsong Zhou, Hangxu Liu, Xuekun Jiang, Xing Shen, Yuanzhen Zhou, Hui Wang, Baole Fang, Yang Tian, Mulin Yu, Qiaojun Yu, Li Ma, Hengjie Li, Hanqing Wang, Jia Zeng, and Jiangmiao Pang. SIM1: Physics-aligned simulator as zero-shot data scaler in deformable worlds, 2026. URL https://arxiv.org/abs/2604.08544.

A PARTICLEWORLDS DATASET DETAILS

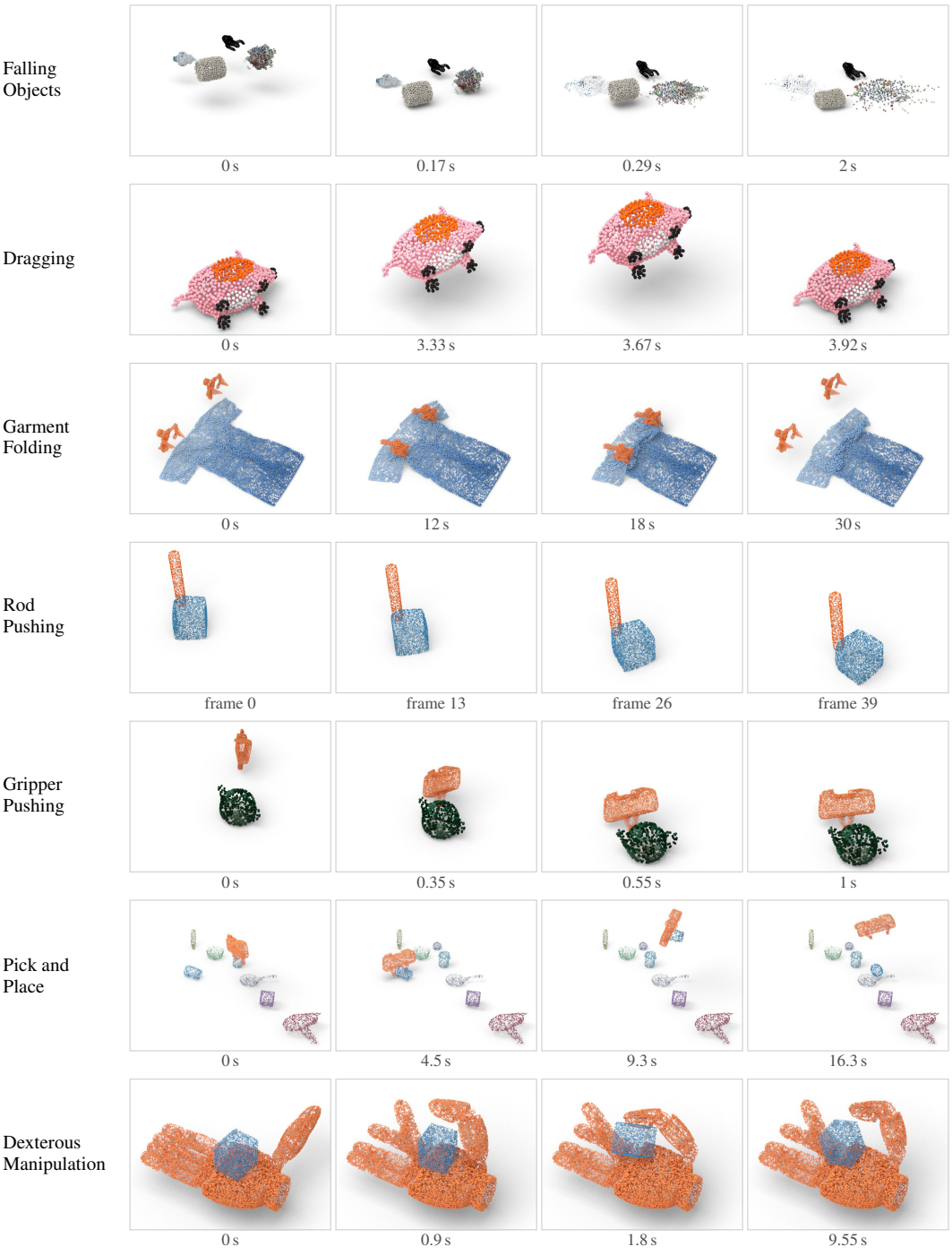

Figure 6: **Examples from the PARTICLEWORLDS domains.** Each row shows one recorded episode at four moments, labeled with the time since its first frame; Pushing has one row for each of its two sources. From the top, four objects of three materials fall onto the table, a piggy bank is lifted by a patch on its back and released, two grippers fold a shirt, a rod pushes a block, a gripper pushes an object, a gripper picks up an object from a cluttered table and places it, and a hand turns a cube. Actuated parts are orange, and objects keep the colors of their meshes where available. Rigid bodies are drawn with extra points that follow their recorded poses, denser than the points the model receives. Rod Pushing records no time step, so its panels give frame indices.

Table 4: **Overview of PARTICLEWORLDS.** Pushing combines rod and gripper recordings, and Pick and Place combines single-object and cluttered recordings. Frame counts refer to usable training episodes at each domain's training sampling rate, including recorded settling frames.

Domain	Materials	Actuation	Training frames
Free Fall	Elastic, elastoplastic, granular	—	2,317,112
Dragging	Elastic, elastoplastic	Material-point velocities	183,949
Garment Folding	Cloth	Two grippers	4,068,020
Pushing	Rigid	Rod or gripper	5,958,397
Pick and Place	Rigid	Gripper	7,257,150
Dexterous Hands	Rigid	Articulated hand	5,679,049
Total			**25,463,677**

Domains and counting convention. The six domains in Table 4 group eight sources: Pushing combines rod and gripper pushing, and Pick and Place combines single-object and cluttered recordings. The corpus contains 267,864 usable training episodes and 25,463,677 scene frames, counted once at each loader's temporal rate, including recorded settling prefixes. Counts exclude test episodes and do not multiply frames by overlapping windows, repeated training draws, or particle counts. Rod and gripper pushing contribute 3,706,040 and 2,252,357 frames, and single-object and cluttered Pick and Place contribute 1,690,120 and 5,567,030 frames, respectively.

Generation and source trajectories.

Free Fall and Dragging. We generate both domains with the Genesis material-point solver (Genesis Authors, 2024). Falling scenes contain one to four objects with 1,024 material points each and span elastic recovery, permanent deformation, and granular rearrangement. Dragging uses elastic and elastoplastic objects: prescribed velocities drive a persistent patch of material points through pull, hold, and release events. Falling episodes contain 49 recorded states; dragging episodes contain 49, 81, or 113 states, depending on the number of events, with a recorded interval of 0.0417 seconds.

Garment Folding. We generate scripted garment manipulation with two grippers in SIM1 (Zhou et al., 2026). After filtering, the domain covers 988 garments from ten categories. Each clip contains approximately 1000 states at 30 Hz. The representation uses a persistent farthest-point subset of the garment mesh with 8 mm spacing, capped at 10,000 cloth points, together with 1,000 points per gripper. The same garment subset is used across frames and clips. These are numerically screened manipulation trajectories, not verified successful folds.

Rigid manipulation. Pushing has two sources. Rod Pushing uses existing recordings of scripted rod motion against a block, with 40 states and 256 points per episode. Gripper Pushing uses existing reinforcement-learning policy recordings of successful object pushes. Single-object Pick and Place combines recordings of four policies for picking, picking convex objects, bringing objects upright, and placing held objects. Its cluttered counterpart includes the target and surrounding objects, which may move or remain still. For Dexterous Hands, we generate trajectories from five teacher policies in Genesis using XHand1 and Sharpa hands for object rotation, reorientation, and screwdriver turning. The rigid gripper and hand recordings are consumed at 20 Hz.

Inputs, targets, and physical properties. Action-conditioned prediction receives scene history and the future motion of rigid actuator links, and predicts passive-object dynamics. Cluttered Pick and Place includes the surrounding objects in the prediction target. Dragging instead supplies commanded material-point velocities, together with indicators distinguishing a stationary hold from a release. These commands need not equal the achieved motion: the target includes every material point, including the driven patch. Passive falling scenes have no action input. Recorded physical properties are supplied where available. The MPM domain varies stiffness over three values while fixing Poisson's ratio and density; garment parameters and the randomized mass and friction of Gripper Pushing and Dexterous Hands are not recorded for conditioning.

Splits and coverage. The release holds out geometries for Free Fall, Dragging, single-object Pick and Place, and Dexterous Hands, and splits Garment Folding by garment identity. Cluttered Pick

and Place uses held-out scenes. Rod Pushing tests new trajectories on the same geometry. Gripper Pushing holds out objects.

Quality control. Source-specific screening removes numerical failures and conspicuous simulation artifacts. MPM checks cover unsettled starts, wall interactions, collapse, and persistent fragmentation during dragging; cloth clips undergo numerical screening. Rigid-object audits detect table penetration, abrupt motion, excessive hand–object overlap, and other failures, with exclusion manifests recording rejected episodes and reasons. We also check pose and surface-template reconstruction against recorded particles. Screening reduces artifacts but does not establish solver convergence.

B Model, Training, and Baseline Details

B.1 Architecture

Every reported model uses the multi-resolution network of Section 3.2, with the sizes in Table 5. The hierarchy has four levels. The first holds every particle. The second keeps, per object, 256 anchors for elastic, plasticine, and sand objects, 128 for rigid bodies and robot links, and 512 for cloth. The third and fourth levels divide these counts by 4 and then by 2, so a Free Fall object of 1,024 particles is represented by 256, 64, and 32 anchors. Downsampling aggregates the 8 nearest tokens of each anchor within its object, and upsampling interpolates from the 3 nearest anchors. From the third level on, each block also carries one object's worth of register tokens. Temporal attention is bidirectional within the window in every reported model.

Table 5: **Network sizes.** The six specialists share one configuration. Levels run from the particles (first) to the coarsest anchors (fourth).

	Specialist	Unified
Parameters	103.1 M	502.2 M
Anchors per object at the second level	256 deformable, 128 rigid body or link, 512 cloth	
Reduction to the third and fourth level	4 and 2	
Feature width per level	64, 128, 256, 512	128, 320, 640, 1280
Encoder blocks per level	1, 2, 4, 9	1, 2, 4, 8
Decoder blocks per level	1, 2, 4	1, 2, 4
Attention head width	64	64
Dropout	0.1	0.1

B.2 Training

Each specialist is trained on one domain. The unified model is trained on all of them, and every batch comes from one domain drawn with equal probability. A training window has 4 context frames and 16 future frames. Particle displacements and rigid-body increments are normalized per domain by the 99th percentile of their per-frame magnitude, separately in the horizontal plane and along the vertical axis.

Future frames are noised with diffusion forcing. In 60% of the windows every future frame draws its own noise level uniformly, and in the other 40% the noise level rises linearly along the window, so nearer frames are less noisy than farther ones, as during streaming inference. Context frames stay clean, except that the history of passive objects receives Gaussian jitter with a standard deviation of 0.05 in normalized units, which imitates the error of predicted context during rollouts. The network predicts the clean trajectories, and the particle and rigid-body terms of the loss are weighted equally ($\gamma = 1$).

All models are optimized with AdamW (learning rate 10^{-4}, weight decay 10^{-3}, $\beta = (0.9, 0.999)$), a 5,000-step linear warmup followed by cosine decay to 1% of the peak rate, gradient clipping at 1.0, an exponential moving average of the weights with decay 0.9999, and bf16 mixed precision. We train each specialist on two NVIDIA RTX PRO 6000 GPUs for 150k iterations, and the unified model on four for 300k iterations.

B.3 Baselines

Each baseline is trained per domain on the same training episodes and evaluated on the same test clips, with the same given frames and actuator motion as our model unless noted.

PhysCtrl (Wang et al., 2025a). We build on the authors' implementation and keep its spatiotemporal transformer, and its diffusion objective. We scale it to about 100 M parameters (8 blocks with hidden size 640) and train it from scratch on our data. The model receives the same history of 4 observed frames as ours, instead of the single frame it was designed for, predicts the same 16 frames per window, and processes every particle of the scene without subsampling.
PhysCtrl takes an action as regions of particles and the motion applied to each region. We cluster the

actuator's particles by position into a few regions, with farthest point sampling choosing the region centers. The action is then given region by region. For every region and every predicted frame, the model receives two conditioning tokens, the position of the region's center and the region's average displacement in that frame. These tokens join the spatial attention of every frame, so every particle can attend to them. We do not give PhysCtrl rigid-body poses, since its interface does not support them. As for our model, the actuator's future motion is an input and is never predicted.
All objects share a single point cloud, and the spatial attention is all-to-all over the particles of each frame. Through PhysCtrl's per-particle conditioning path, each particle is also tagged with the object it belongs to and that object's material parameters $(\log E, \nu, \log \rho)$.

PointWorld (Huang et al., 2026). We train the PointWorld model from scratch with its default PTv3-base backbone (409.8M parameters) on the same episodes, splits, and start frames our own loader uses. PointWorld voxel-downsamples every point cloud before its backbone, by default on a 15 mm grid. On a number of our domains this would discard much of the objects' geometry, so we train at 1 mm for this initial voxelization. For evaluation and rollout purposes, we duplicate each surviving point's prediction onto the points merged into its voxel. We also drop the input features our data does not provide, such as color and surface normals. Each scene point therefore carries only its positions over time and its distance to the robot, and each actuator point its positions, velocity and acceleration. PointWorld receives the actuator's points at every frame and predicts only the scene. It observes a single context frame, where our models receive 4, and the released model predicts 10 frames but we extend the horizon to 16 to match our models prediction. Longer evaluation windows are rolled out autoregressively in chunks anchored on the model's own previous prediction.

GNS (Sanchez-Gonzalez et al., 2020). We train GNS on the same episodes, splits, and start frames our own loader uses, keeping the published architecture: 10 message-passing steps, 128-dimensional latents and a 0.015 m connectivity radius. It is a one step model so we roll out autoregressively at every step to get a full length prediction. GNS carries the learned embedding of a discrete particle type plus a material channel that is hard-coded to one scalar. We widen that channel to a per-particle vector and feed in each particle's normalized material properties. For the planning experiments we give the gripper a unique particle type so it is predicted like any other body. Finally, since the predicted actuator loses rigidity over a long horizon, we recover an executable robot action by Kabsch fitting to the initial actuator geometry and doing a rigid re-projection at every step of the rollout.

PGND (Zhang et al., 2025). We train PGND with 4 given and 16 predicted frames. Its actuator representation needed the most change. PGND either drives material with a sphere collider or shows the network a rigid 500-point gripper template posed by the end-effector, so its fingers never articulate. We instead feed the recorded actuator points every frame so finger motion is represented. The Dragging domain has no tool at all, since a patch of 100 material points is driven by a commanded velocity, so there we use PGND's sphere centered on that patch. This sphere moves with the object material. PGND does not condition on physics properties, so we do not provide them to the model either. We exclude 61 training episodes whose objects exceed PGND's fixed 0.92 m simulation box. Other than these exclusions, we train PGND with the same episodes, splits, and start frames as our own loader.

C EVALUATION PROTOCOL

Prediction protocol and metrics. All baseline models are domain-specific specialists, as are the GPWM models in the Specialist row of Table 1. The Unified row uses a single model trained jointly across all six PARTICLEWORLDS domains. Methods are evaluated on the same test trajectories with the same available actuator conditioning. On action-conditioned tasks, recorded future actuator motion is supplied throughout the rollout; passive-dynamics tasks require no action conditioning. Predictions are fed back as context for subsequent windows. For particle-tracked dynamics, we average Euclidean position error over passive-object particles and predicted rollout frames, then average equally across trajectories. Prescribed actuator motion is excluded from the error. Cloth is evaluated using symmetric Chamfer distance, since particle correspondence is not shared across all methods and surface geometry is the quantity of interest. Appendix D shows predictions on every domain.

Test sets. Table 6 lists the test clips behind each column. Every rollout starts from 4 recorded frames and runs to the last recorded frame of its clip, so horizons vary within a domain. A clip's error is its mean over the predicted frames and the passive particles, and a column averages clips with equal weight. Actuator particles, which are given, are not scored.

Table 6: **Test clips.** Held out column indicates how the test set differs from training. Predicted frames give the range of rollout lengths.

Column	Clips	Held out	Predicted frames	Frame interval
Free Fall, 1 to 4 objects	100	shapes	45	1/24 s
Free Fall, 5 to 7 objects (Table 2)	100	shapes and object count	45	1/24 s
Dragging	100	shapes	45 to 109	1/24 s
Pushing	200	objects	16 to 93	0.05 s
Pick and Place	200	scenes	233 to 921	0.05 s
Dexterous Hands	50	objects	35 to 396	0.05 s
Garment Folding	8	garments	1,000	1/30 s

D Additional Prediction Results

This section shows test clips for every domain of Table 1. Every model rolls out from the same four recorded frames with the same actuator motion. The error under each panel is the mean position error of the predicted particles at that frame.

D.1 Free Fall

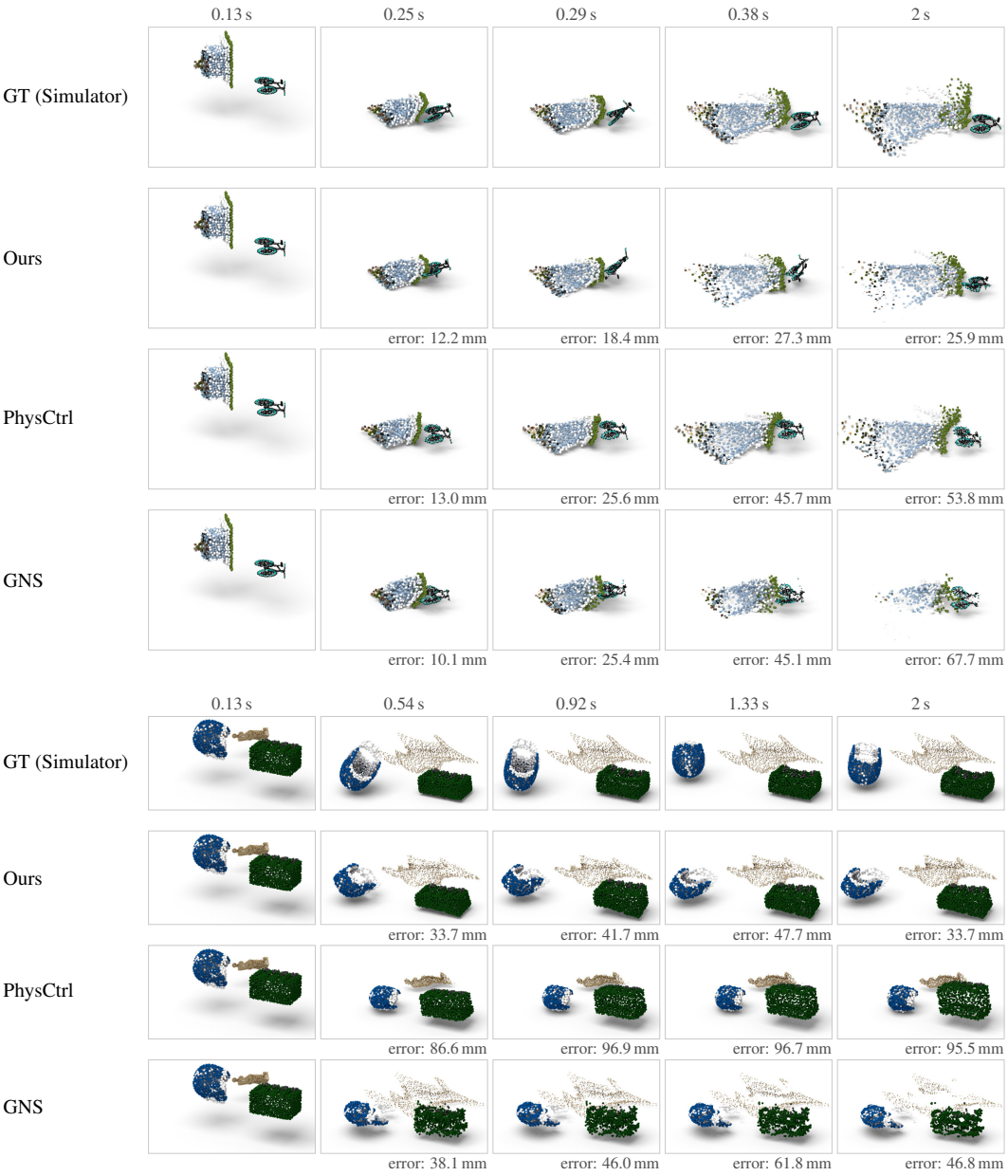

Figure 7: **Free Fall.** Two test clips with unseen shapes. Rows show the simulator's recording and three models, and columns show the last given frame, three moments of the fall and the last predicted frame, labeled with the time since release. Ours has the lowest clip error on 172 of the 200 Free Fall test clips, 84 of 100 with one to four objects and 88 of 100 with five to seven.

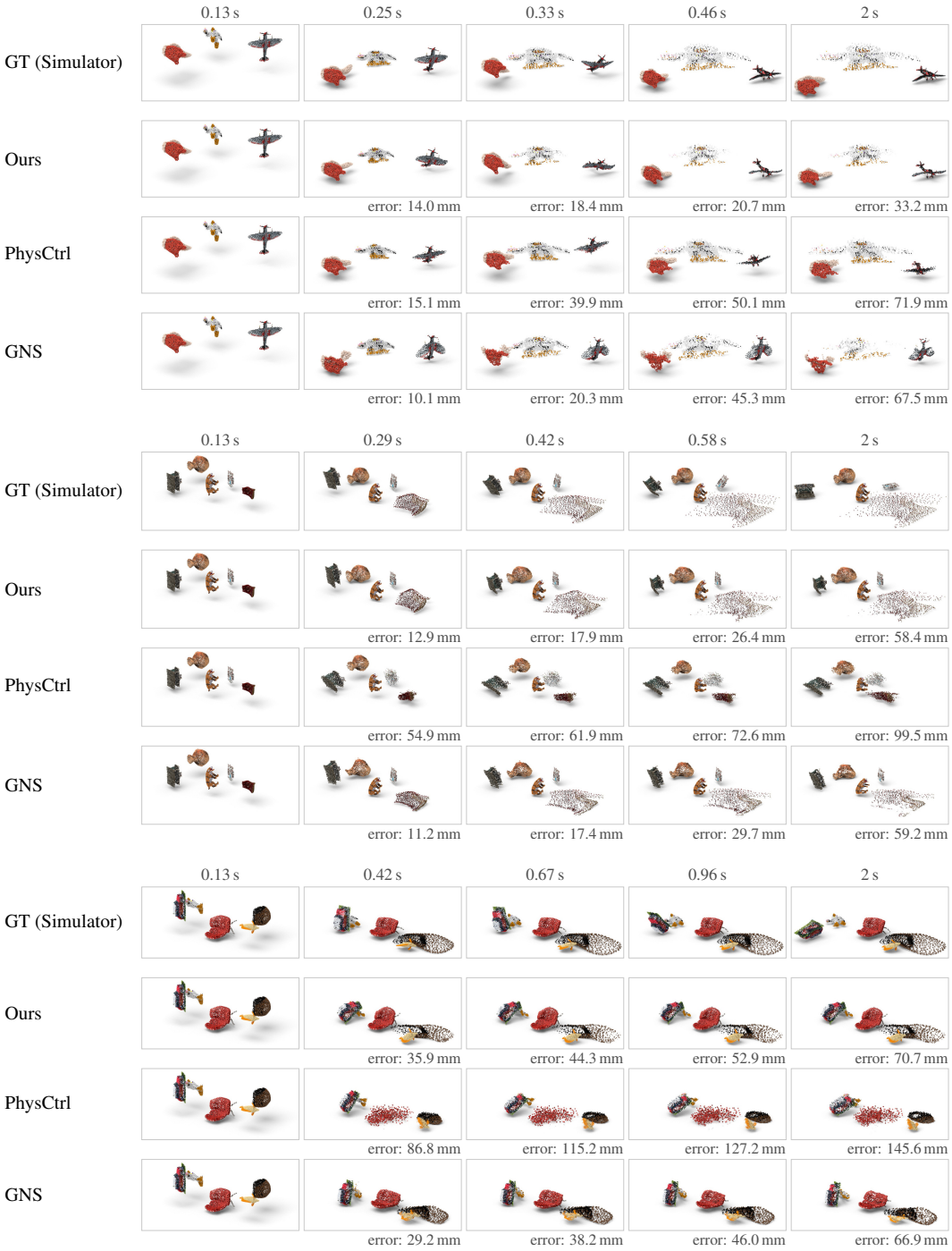

Figure 8: **Free Fall, continued.**

D.2 DRAGGING

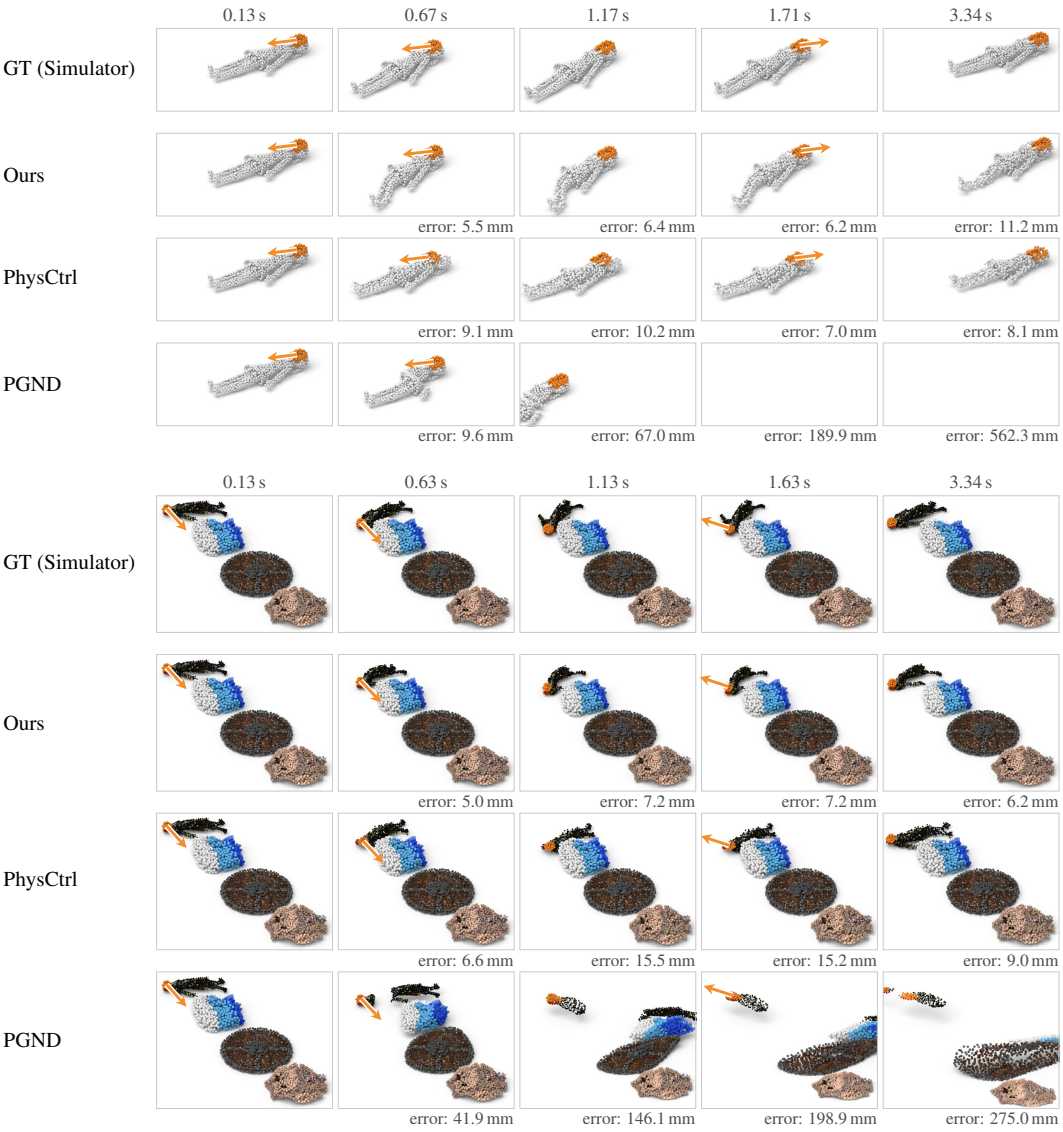

Figure 9: **Dragging.** Two test clips with unseen shapes, laid out as in Figure 7. The orange arrow shows the direction in which the dragged patch (orange) is pulled at that moment, from the recorded command that every model receives. Ours has the lowest clip error on 81 of the 100 Dragging test clips.

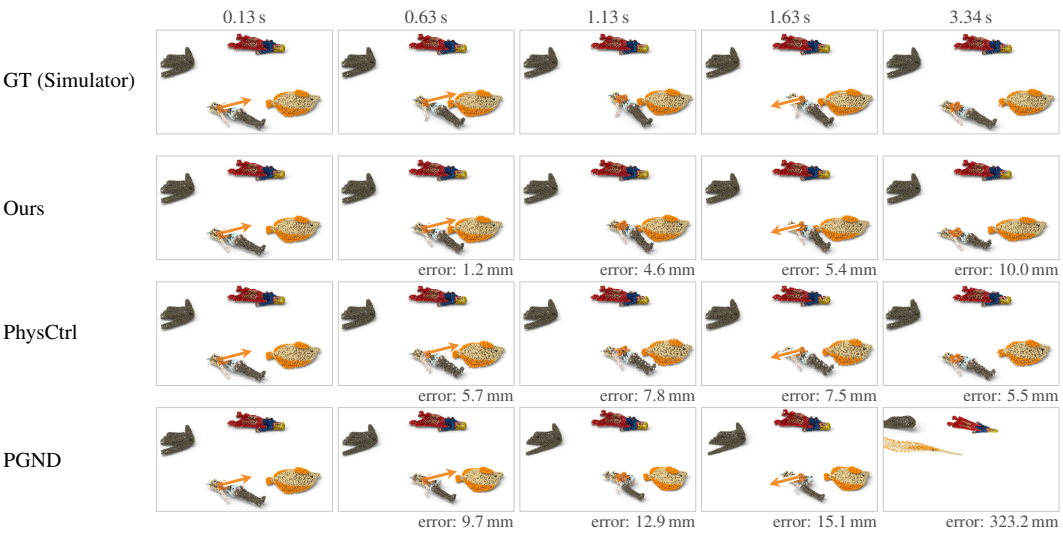

Figure 10: **Dragging, continued.** On this clip PhysCtrl has the lowest error.

D.3 GARMENT FOLDING

Figures 11 and 12 render predictions back onto the simulator's own garment mesh. They use a model trained on the Garment Folding domain of PARTICLEWORLDS and four of its held-out test garments.

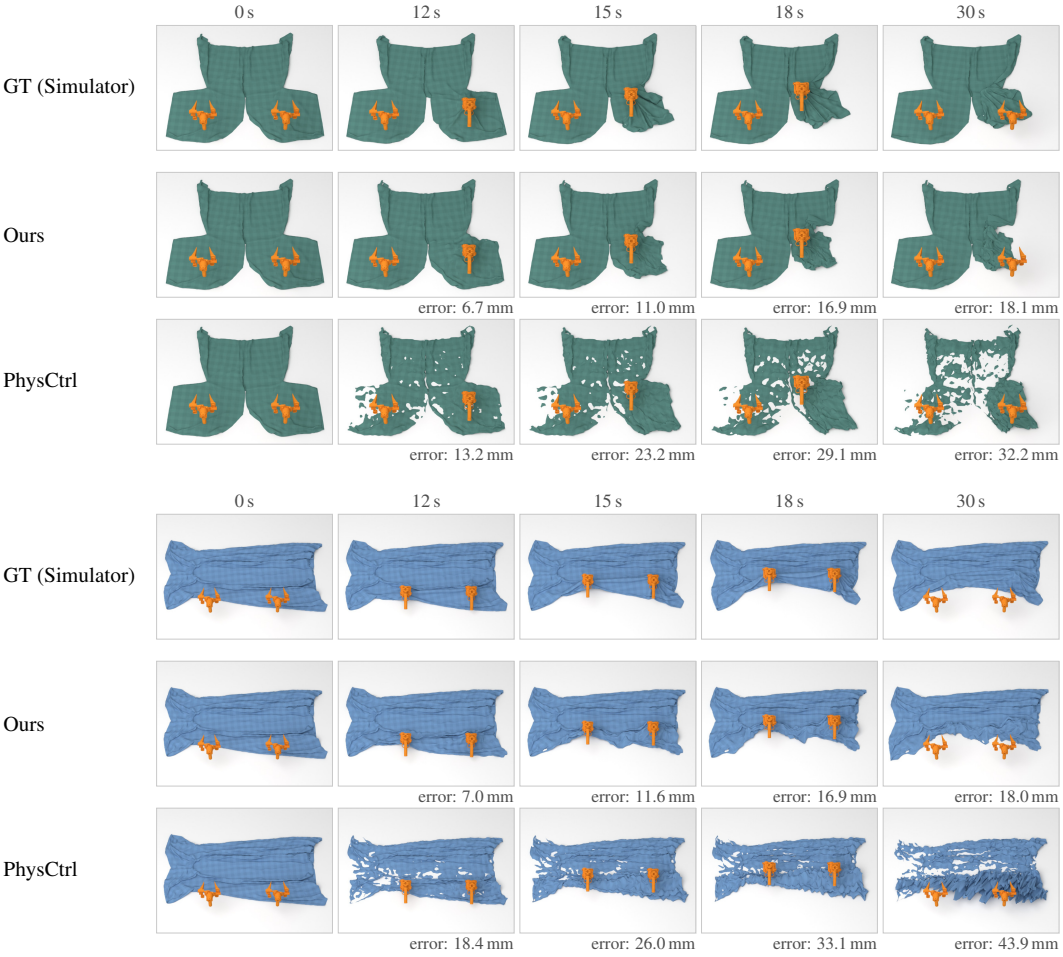

Figure 11: **Garment Folding, rendered back to the simulation mesh.** A jacket and a T-shirt not seen in training, 0 to 30 s after the given history frames. Rows show the simulator and each model's particles driving the same mesh, with the grippers (orange) at their recorded poses. The error under each panel is the mean position error of the cloth particles at that moment. All rows are smoothed for better mesh visualization, which moves vertices by 1.5 to 6 mm on average. Gaps in the PhysCtrl rows are where its cloth sinks below the floor.



Figure 12: **Garment Folding, continued.** Pants and a hooded jacket.

D.4 PUSHING

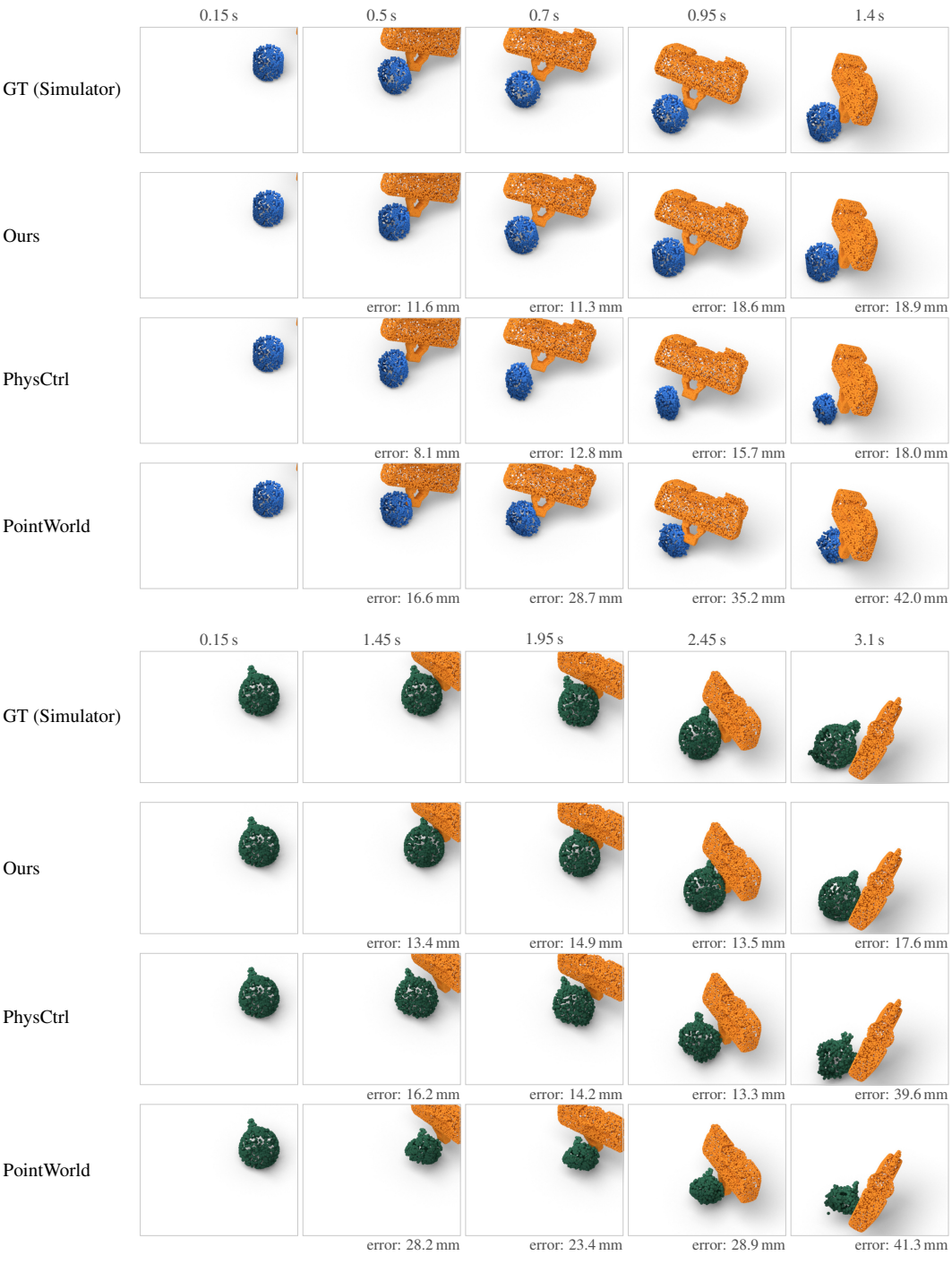

Figure 13: **Pushing.** Test clips with objects held out from training, laid out as in Figure 7. The gripper (orange) is given to every model and drawn at its recorded pose. Ours has the lowest clip error on 165 of the 200 Pushing test clips.

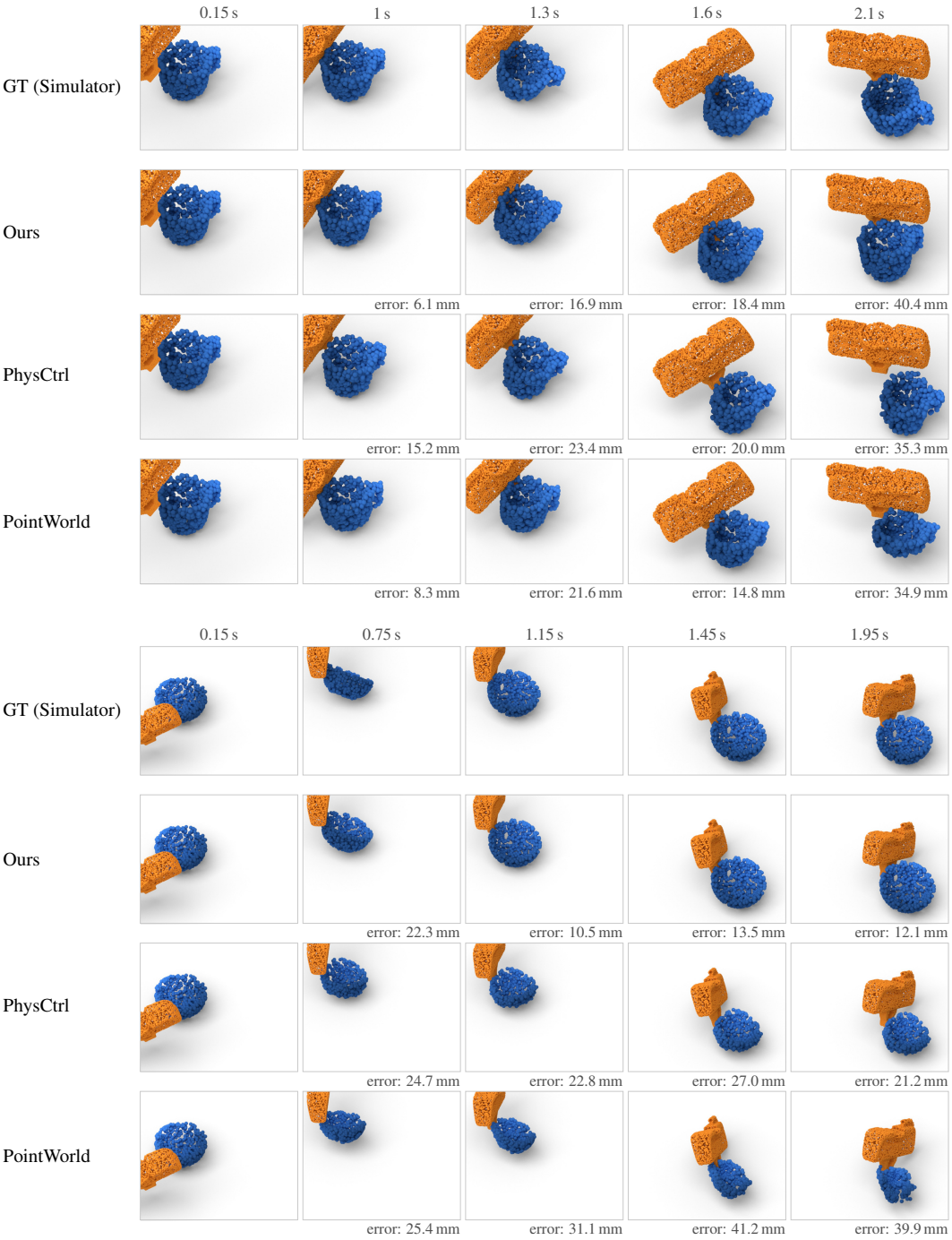

Figure 14: **Pushing, continued.**

D.5 PICK AND PLACE



Figure 15: **Pick and Place.** Successful picks in held-out cluttered scenes, laid out as in Figure 13. Ours has the lowest clip error on 185 of the 200 Pick and Place test clips.



Figure 16: **Pick and Place, continued.**

D.6 DEXTEROUS HANDS

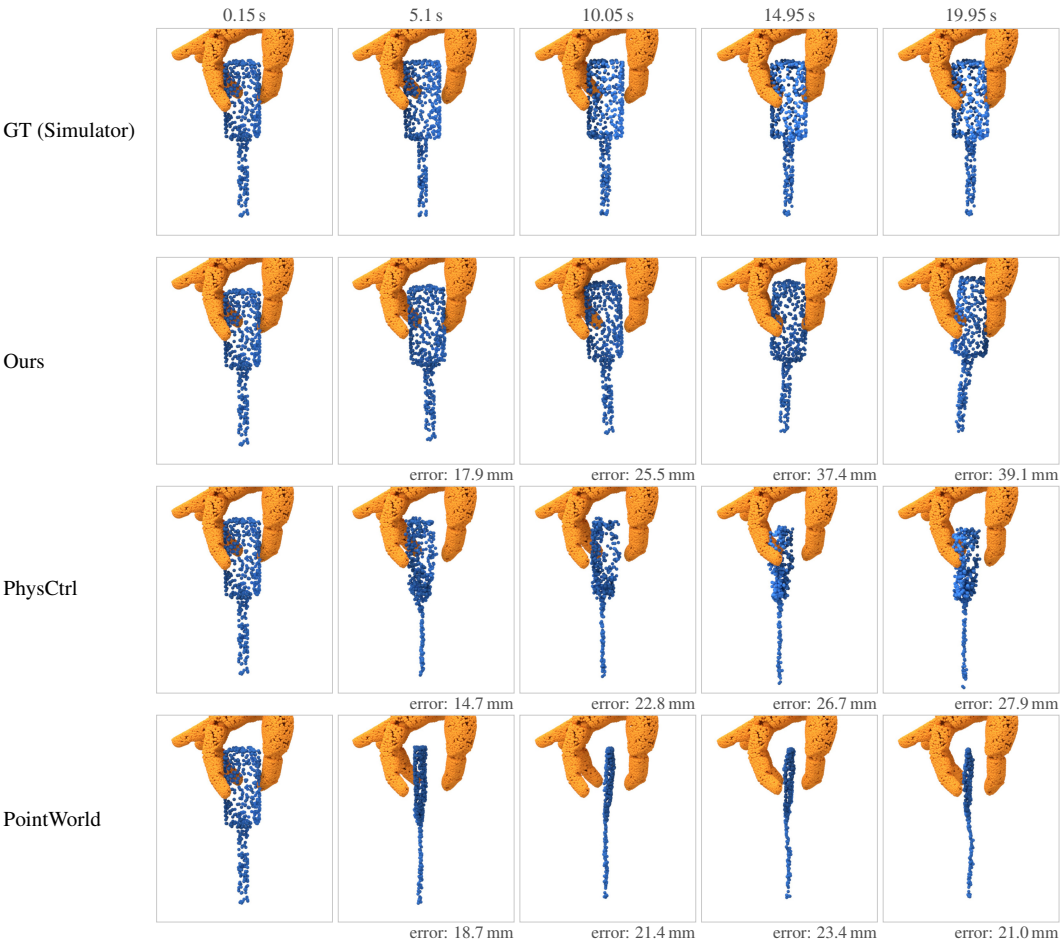

Figure 17: **Dexterous Hands.** A held-out screwdriver turned by the Sharpa hand. On this clip PhysCtrl and PointWorld have lower mean error than ours, but both move every point freely and thin the screwdriver.

E PLANNING

E.1 DETAILS ON PLANNING TASKS

In all our diffusion based planning tasks, an actuator must bring a target object into a target configuration within a specified time horizon. A *sparse* reward is defined only on the final trajectory frame: the distance from the object's centroid to its target. World models must find an open-loop plan, which is then carried out in the base simulator. For models that predict future actuator states (GNS and GPWM), we execute these states with a simple state tracking controller. We evaluate on rigid, deformable and grasping tasks:

- **Pushing** — Rigid manipulation task where a rod pushes a block around a workspace. This task is challenging with a sparse reward because the rod must reposition around the block to push the object to the target. So random actions will typically produce no signal for task progress. Planning horizon is 60 steps ($3\times$ our model window). Plans are executed in ManiSkill.
- **Pile Manipulation** — Deformable manipulation task in which a paddle shifts the centroid of a soft granular mass. World models are trained on $6{,}000$ trajectories of 300 frames each from a scripted policy that randomly pushes and presses the pile with the paddle. Planning horizon is 150 steps ($7.5\times$ our model window). Plans are executed in Genesis.
- **Picking** — A parallel-jaw gripper must grasp a tabletop object and carry it to a target. These targets are spawned 15 cm above and up to 15 cm to the side of its start (5 objects $\times$ 5 targets). Planning horizon is 60 steps ($3\times$ our model window). Plans are executed with a floating Franka hand in Isaac Lab.

Metrics. For block pushing we report the final distance to the target. For pile manipulation we report progress: how much closer the pile's centroid ends to its target than it started. For picking we report the number of objects carried to within 6 cm of their target. We also report the time taken to generate a plan on a single L40S GPU.

E.2 DETAILS ON BASELINES

All baselines use MPC-style receding-horizon planning *fully within* their learned dynamics model. Once a plan is committed, the action sequence is executed open-loop in the base simulator from the scenario's exact start state. No baseline ever plans beyond the remaining task budget, and we commit 5 frames per replan. Rewards are evaluated only on the final frame of the horizon.

PointWorld is action-conditioned, so candidates are action sequences drawn as Gaussian noise around a nominal sequence. We use the argmax candidate as the nominal sequence as we find this more effective than the softmax weighted average. On pushing, every candidate is rolled out to the final frame of the horizon and the lowest-cost candidate is executed; the pusher's per-step displacement is clamped to 6 mm, the largest step in the training data. We use N = 128 on pushing, N = 256 on pile manipulation, and N = 88 on picking.

GNS and GPWM predict the actuator's motion together with the scene, so their candidates are sampled trajectories. A trajectory-space weighted update is not meaningful for them (averaging the predicted clouds yields physically inconsistent intermediate states), so again we commit the argmin candidate. This deterministic baseline is meant to show that just modeling the average motion in a multi-modal scene with actuated objects is not sufficient for planning. We use N = 256 on pushing, N = 64 on pile manipulation, and N= 256 on picking.

GPWM w/o guidance draws N candidates of the remaining horizon at every replan from the same streaming sampler used by guidance, with the committed frames given to the streaming buffer as clean context. Near the end of the horizon the buffer slides back over committed frames instead of shrinking, so every model call sees the full trained window. We use $N = 32$ on pushing, $N = 6$ on pile manipulation and $N = 20$ on picking.

E.3 Planning with Latent Noise Optimization

We plan by optimizing the noisy trajectory buffer x_t with the weights of GPWM frozen (Fig. 18). To score a candidate x_t, the streaming lookahead of Section 3.4 strides the model window across the full horizon of T frames and returns a clean trajectory estimate $\hat{x}_0(x_t)$. The reward is sparse, $r = -\|c(\hat{x}_{0,T}) - g\|_2$ where $c(\hat{x}_{0,T})$ is the forcasted location of the passive object's centroid at the final frame of the trajectory and g is the goal. This reward and its gradient is back-propagated through every model call of the lookahead. Because GPWM generates the actuator and the passive objects jointly, the plan is simply the actuator part of the optimized trajectory.

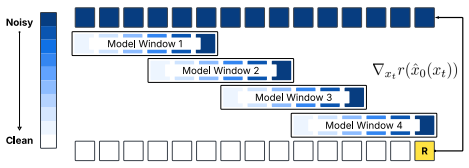

Figure 18: **Guidance step with streaming lookahead.** For a gradient step on noisy sample x_t, we get an estimate of the clean trajectory using streaming lookahead. Back-propagating through the lookahead allows guidance to optimize for long horizon tasks with sparse rewards only on the final frame.

We optimize noise in the rollout buffer at every $t \geq t_{\text{freeze}}$. Standard latent noise optimization just optimizes with $t_{\text{freeze}} = 1$, but we find it helpful to set $t_{\text{freeze}} = 0.8$. When optimizing a draw z, we take a gradient step $z \leftarrow z + \eta \nabla_z r$ and then rescaled the noise to its original norm. A population of P independent samples is optimized for I steps. Pile manipulation is the longest horizon task and we run LNO with $P = 3$, $I = 6$, and $\eta = 0.1$. For pushing and picking we are able to increase the values of P and I while still staying within the planning budget. Each member keeps its highest-reward trajectory and we execute the member with the highest predicted reward; the simulator never enters planning.

E.4 Additional Analysis of Planning Results

Block Pushing: For the block pushing task baselines primarily fail to solve the task at all. The sparse rewards make it difficult for them to settle on a plan that reaches the target, and the actuator just takes random motions. Our method on the other hand learns to reposition to the other side of the block in order to execute a successful push. Visualizations in Fig. 19.

Pile Manipulation:For the pile manipulation task, trajectory optimization in PointWorld amplifies errors in the world model's predictions, eventually leading to a degenerate solution with highly inaccurate dynamics. The deterministic GNS only learns the average actions in the dataset, simply moving around the middle of the pile. Guiding over our learned multi-modal distribution allows us to find a successful plan that stays within distribution of our world model and the dynamics of the system. Visualizations in Fig. 20.

Picking: Because grasping is a difficult and multi-modal action, PointWorld completely fails to pick the object and it suffers from the same error amplification as it did in the Pile Manipulation task. Being deterministic, GNS suffers from similar model prediction collapse. Our baseline without guidance effectively captures the picking modes, so it is able to successfully lift the object. But without guidance, the model cannot be effectively guided to the correct pick location. Visualizations in Fig 21 and Fig 22.

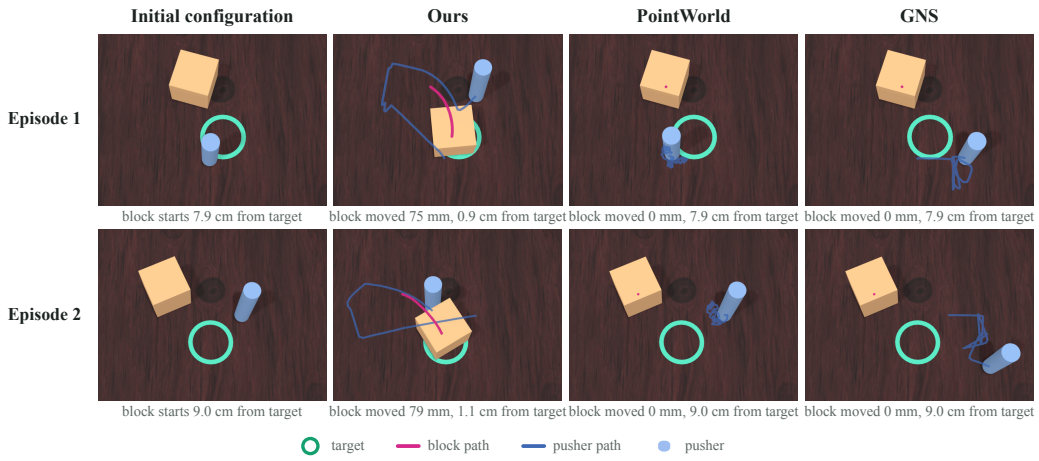

Figure 19: **Representative Baseline Failure Cases for BlockPush** GNS and Pointworld both fail to reposition around the other side of the block to complete the push, but our model succeeds.

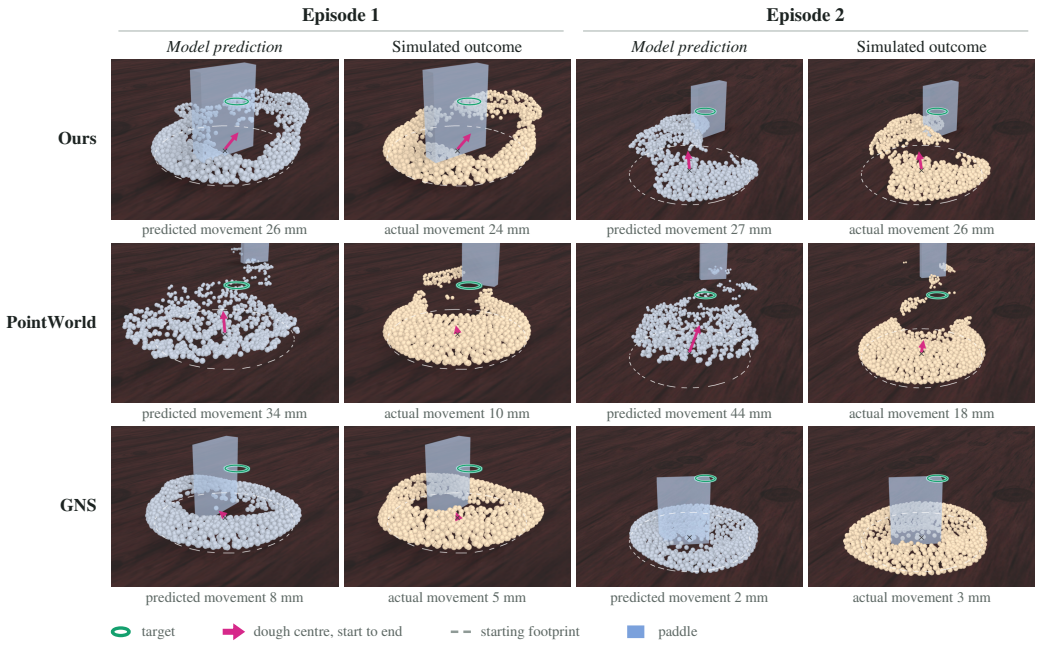

Figure 20: **Representative Baseline Failure Cases for Pile Manipulation** Optimization within PointWorld leads to a greedy solution that throws the world model out of distribution. GNS just predictions the average motion in the domain and fails to make task progress. Our method finds an in-distribution solution that makes significant task progress.

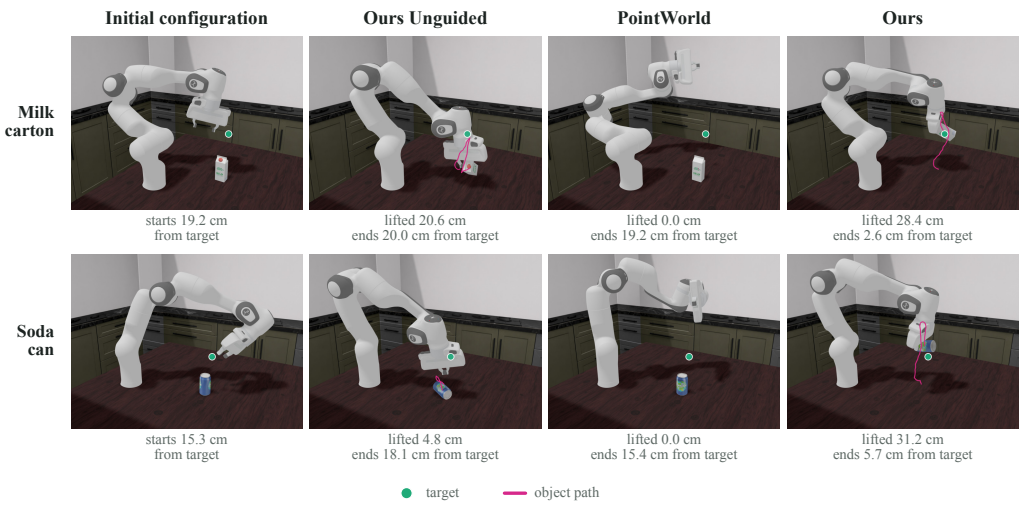

Figure 21: **Representative Baseline Failure Cases for Picking** Our baseline without guidance will sometimes lift the object, but fails to bring it to the target location. PointWorld fails to complete a pick at all. Our method successfully lifts and brings the object to the target location. Full Franka arm is rendered just for visualization purposes (picks carried out with floating gripper).

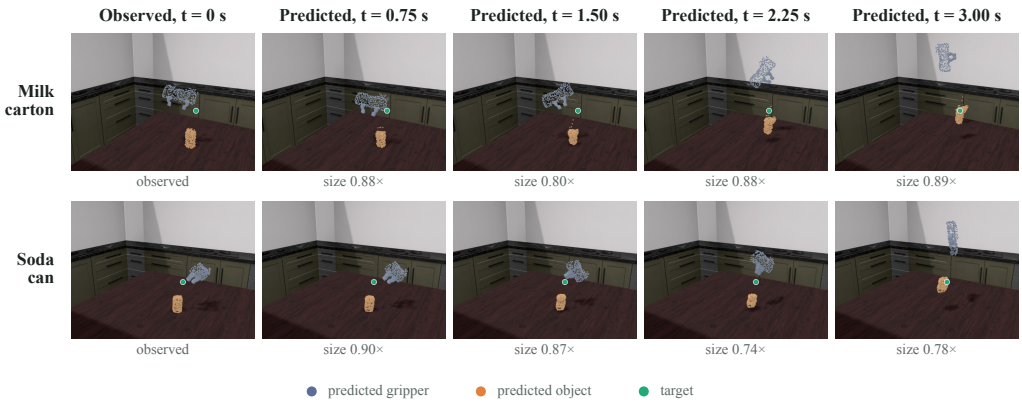

Figure 22: **Representative Model Prediction Failure Cases for PointWorld** Optimization within PointWorld pushes the world model out of distribution and causes errors. This leads to the rigid object breaking apart and a photom lift of the object. Size represents the percentage volume preserved from the initial frame's shape.

F PREDICTIONS FROM A SINGLE IMAGE

This section gives the details of the predictions from single images in Section 5.2 and more examples.

F.1 RECONSTRUCTION AND SETUP

Reconstruction. SAM 3 (Carion et al., 2025) segments each object named in a text prompt, SAM 3D (SAM 3D Team et al., 2025) reconstructs it as a mesh, and MoGe-2 (Wang et al., 2025b) estimates the metric depth and camera that place the meshes in the scene. Each object is represented by 1,024 particles sampled on its surface, as in the training data.

Scale, materials, and release. Each scene is rescaled so that its objects match the particle spacing of the training objects, by 1.7 times for the desk photograph, and times are reported at the original scale. Each object is assigned a material type and a Young's modulus. The four given frames describe free fall from the chosen release pose.

Rendering. Image pixels of moving objects are filled with LaMa (Suvorov et al., 2022). Each object mesh follows its predicted particles, every vertex moving with its four nearest particles, and is path traced with the camera of the image. Sand is drawn as grains, and sand particles that end below the table are drawn on its surface.

F.2 MORE EXAMPLES

Figure 23 shows more drops from the desk photograph and from five generated images, two of them stylized images beyond the training data.

F.3 CHANGING THE OBSERVED HISTORY

The model receives four frames of history, so editing them changes the predicted future. In Figure 24 a ball dropped onto a pepper comes to rest on top. We then rewrite only the ball's four input frames so that it moves sideways at 9 cm/s from where it rests, and let the model continue. No actuator is modeled.

F.4 DRAGGING A RECONSTRUCTED OBJECT

In Figure 25, a sloth reconstructed from a photograph is dragged into a foam star. Every model receives the same initial scene and commanded drag, and the arrow shows the drag direction.

F.5 FOLDING GARMENTS FROM GENERATED IMAGES

Figure 26 applies the Garment Folding specialist to garments from generated images. Each garment is segmented from a top-down image, represented as two cloth layers at the particle spacing of the training data, and placed on the most similar training garment, which sets its size and supplies the recorded gripper motion. The specialist predicts all frames from that motion, which moves the grasped edge 16 to 20 cm, so each garment ends in a partial fold.

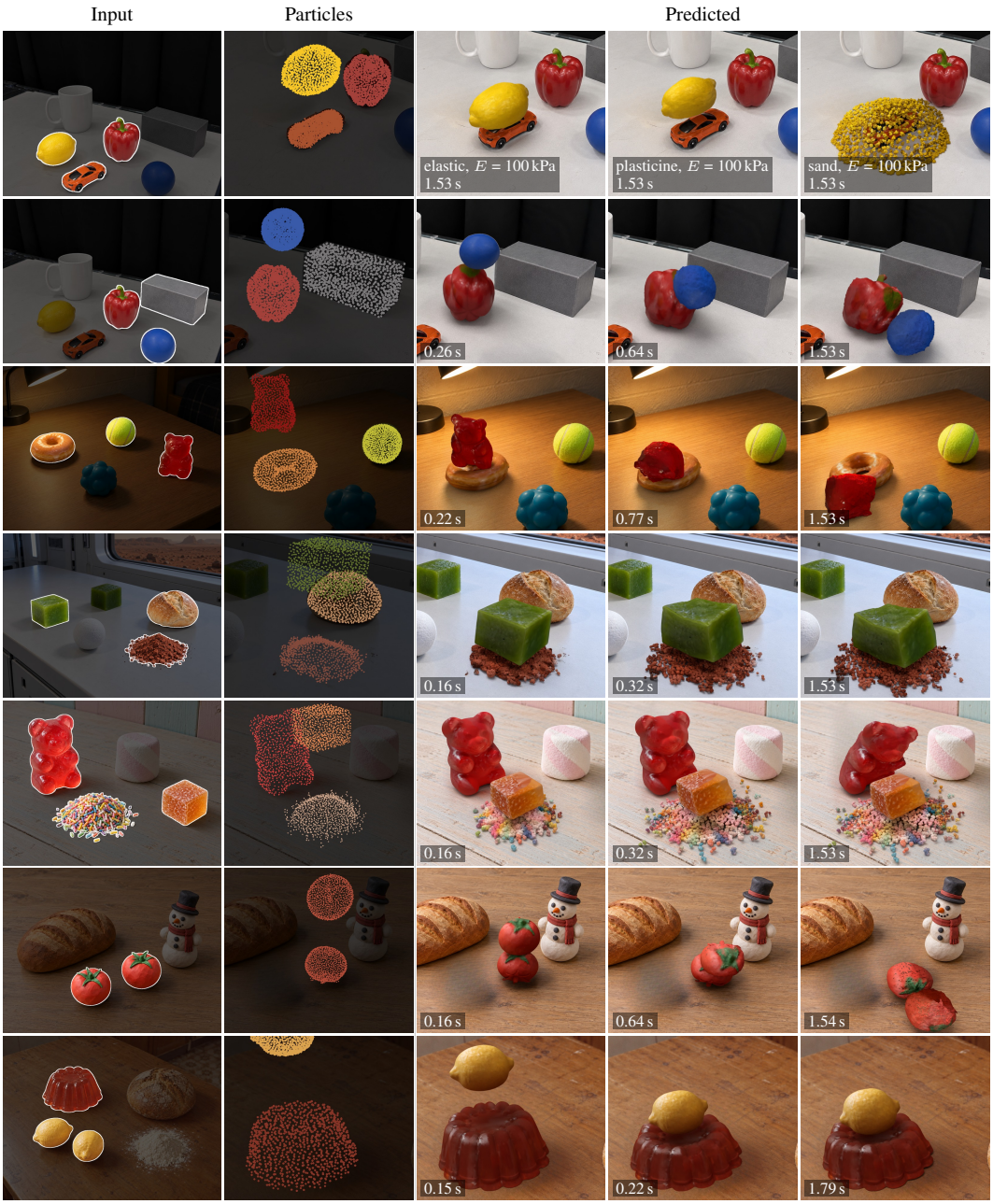

Figure 23: **More drops from images.** Each row shows the input image with the simulated objects outlined, their particles at the last input frame, and three predicted frames. Row 1, the lemon from the desk photograph falls onto the toy car as an elastic, a plasticine, and a sand object, shown 1.53 s after release. Row 2, a plasticine ball lands on the pepper, rolls off and pushes it aside. Rows 3 to 7 use generated images. Row 3, an elastic gummy bear falls onto a plasticine donut on a dorm desk, flattens and slides off. It stays flattened, a limitation of the model for soft elastic objects. Row 4, an elastic gel cube falls onto a pile of Mars regolith simulated as sand and scatters it. Row 5, an elastic jelly cube falls onto sprinkles simulated as sand, and the gummy bear beside it slumps by 1.53 s, the same limitation. Row 6, in a claymation scene, a plasticine tomato lands on an elastic one, and the two topple and come to rest apart. Row 7, in a 1970s photograph, an elastic lemon lands on an elastic jelly mold and comes to rest on top. Rows 6 and 7 are stylized images beyond the training data. Young's moduli are 10 kPa for the gummy bears, donut, gel cube, bread roll, jelly cube, and jelly mold, 1 MPa for the toy car and the block, and 100 kPa for all other objects. The car, pepper, block, and tennis ball are elastic. Times are at each scene's original scale.

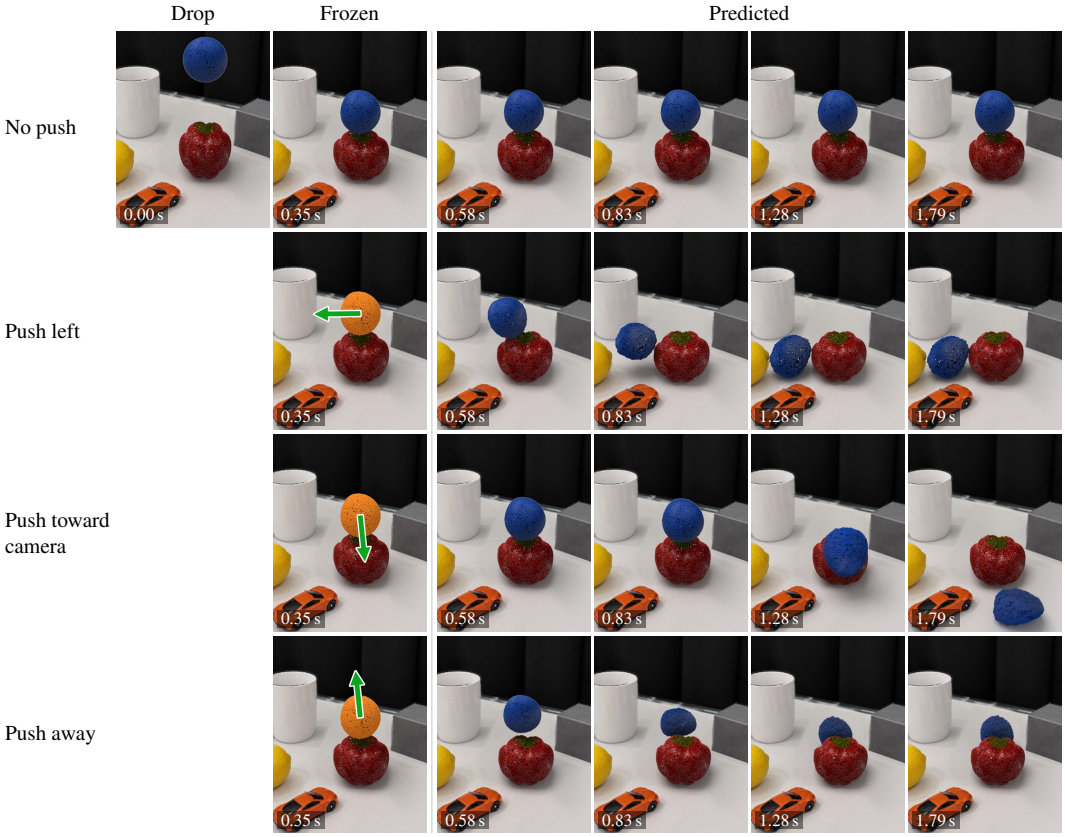

Figure 24: **Changing the observed history.** Every row continues the same drop of the blue ball onto the pepper. In the top row the ball is released, lands, and rests on top at 0.35 s, when we freeze the scene. Without an edit it stays there. In each row below, only the ball's four input frames at the freeze are rewritten so that it moves in the direction of the green arrow (the edited ball is drawn orange), and the model predicts the rest. Times are since the release at the original scale. The last column is 1.44 s after the freeze.

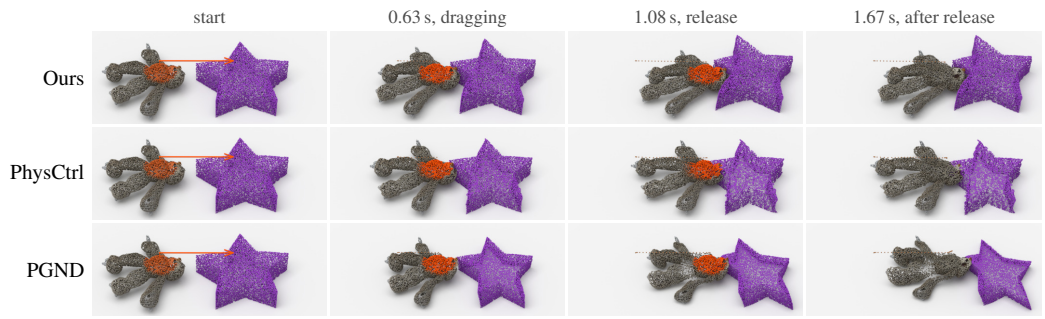

Figure 25: **Dragging a sloth into a foam star.** A sloth reconstructed from a photograph is dragged into a foam star, and every model gets the same start and drag. The arrow shows the commanded drag. Objects are drawn with dense points that follow each model's predicted particles.

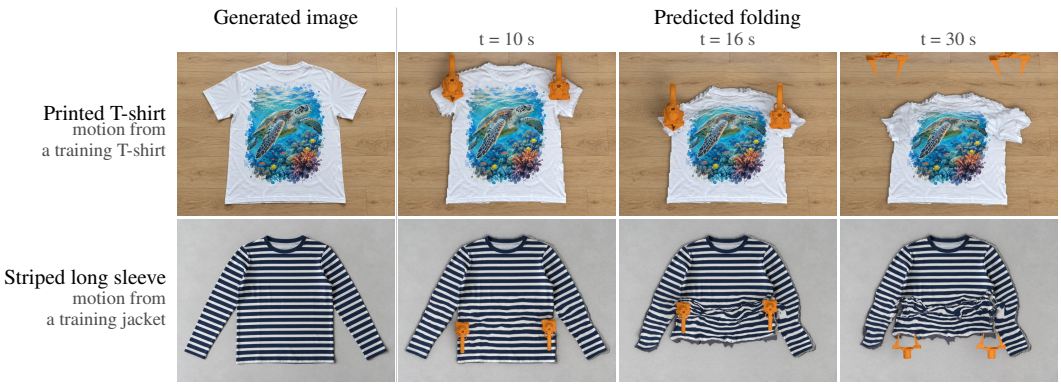

Figure 26: **Folding garments from generated images.** The grippers (orange) follow the motion recorded for a training garment, and their points are given to the model.