GPWM: Modeling the World as 3D Particle Motion

Robert Ren[1,*], Matthew Bronars[1,*], He Zhu[1], Yikai Tang[1], Jianing Zhang[1], Sergey Zakharov[2], Pavel Tokmakov[2], Katerina Fragkiadaki[1]

[1]Carnegie Mellon University, [2]Toyota Research Institute

*Equal contribution, ordered by coin flip

We introduce GPWM, a **G**enerative **P**article **W**orld **M**odel that predicts how 3D scenes evolve by generating the motion of particles and rigid bodies over time. Unlike video world models that generate scene appearance, GPWM operates directly in 3D physical space. A single representation and architecture cover 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 generate coherent physical trajectories far beyond its training horizon. We train and evaluate GPWM on ParticleWorlds, a 25.5-million-frame corpus spanning diverse geometries, materials, interactions, and robot embodiments. GPWM outperforms the best prior learned-dynamics baseline in each of six physical domains, reducing prediction error by 31% on average. Breaking from the domain-specific setting of prior methods, a single *Unified* GPWM models these heterogeneous regimes with one set of weights, largely outperforming per-domain baselines and even surpassing our own specialist models in two domains. GPWM further generalizes to unseen geometries, physical properties, object counts, and rollout horizons. Beyond prediction, GPWM enables contact-rich planning by learning a generative prior over joint actuator–object motion. Instead of searching over unconstrained actions, planning searches the model's latent noise for interactions that achieve a desired object-space goal. With the same world model, noise-space planning substantially outperforms action-space search on pushing and picking, where complex contact sequences are hard to discover. Executed open-loop in simulation, the resulting plans achieve goal-directed manipulation without reference action trajectories or goal-specific policies.

Contact: {robertre, mbronars, katef}@andrew.cmu.edu
Project page: https://robertren1122.github.io/gpwm/

Carnegie Mellon University

TOYOTA RESEARCH INSTITUTE

1 Introduction

A general-purpose physical world model should predict how a 3D scene evolves and how to act to bring it toward a desired state. This requires modeling diverse objects and materials, maintaining coherent dynamics over long horizons, and reasoning through physical contact. Recent generative video world models capture rich and multimodal scene evolution (NVIDIA, 2026; Wan et al., 2025; Bruce et al., 2024; Guo et al., 2026), but generate visual appearance rather than explicitly representing the motion and interactions of physical objects in 3D. 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; Duisterhof et al., 2026). However, existing learned 3D dynamics models are typically limited by deterministic prediction, short generation horizons, or specialization to particular materials and interactions (Sanchez-Gonzalez et al., 2020; Shi et al., 2024; Zhang et al., 2025; Wang et al., 2025a; Chen et al., 2026). We ask: *can a generative model of 3D particle motion capture diverse physical dynamics, predict over long horizons, and support planning through contact?*

We introduce GPWM, a generative world model that represents 3D scenes as particles and rigid bodies and predicts their motion over time (Figure 1). Deformable objects are modeled through particle displacements and rigid bodies through body-level transformations, providing a common representation for rigid, deformable, cloth, and granular dynamics. GPWM conditions on observed 3D scene history and, when available, physical properties and future actuator motion. Actuators share the same geometric representation as other scene components, allowing their trajectories to either condition physical prediction or be generated jointly with object motion for planning.

Learning generative particle dynamics at scale presents two challenges: scenes may contain thousands of particles across interacting objects, and prediction errors accumulate over long rollouts. GPWM addresses the first with object-

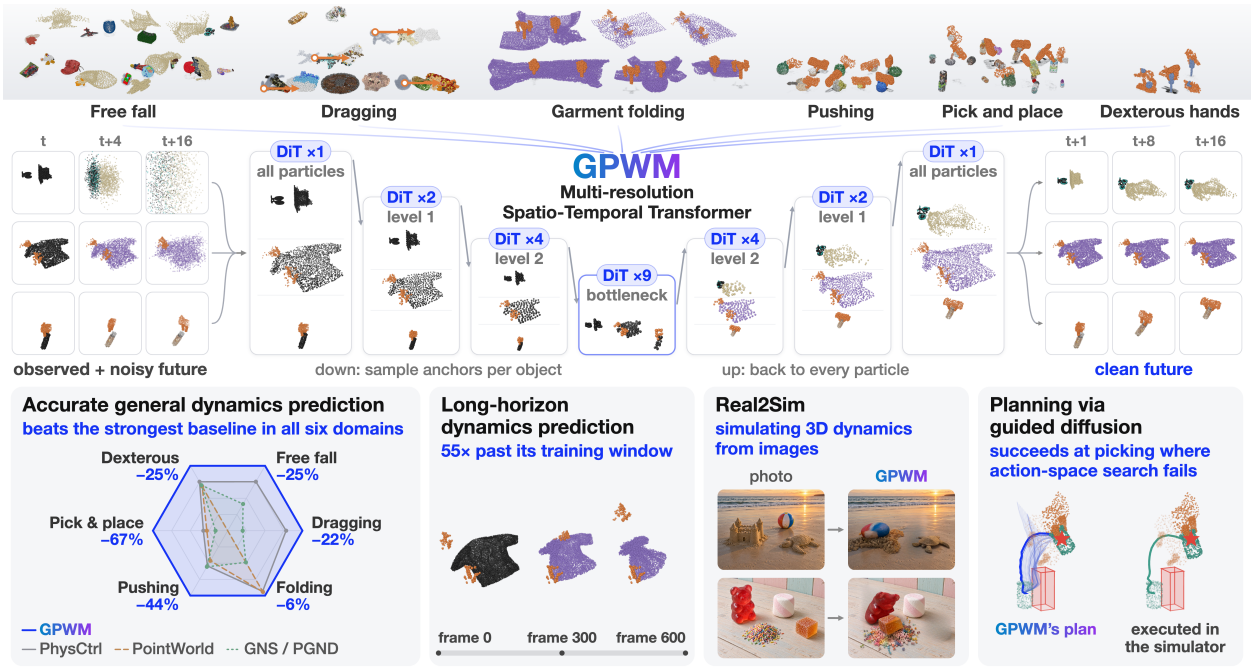


Figure 1 GPWM learns 3D particle dynamics across six physical domains. Top: samples from the six domains of ParticleWorlds. Middle: given the observed frames and a noisy future, a U-shaped multi-resolution spatio-temporal transformer denoises the future motion of every particle. Bottom: GPWM (the better of its specialist and unified models in each domain) lowers prediction error relative to the strongest baseline in every domain, by 31% on average, rolls out 55× past its 16-frame training window, simulates 3D dynamics from images, and plans via guided diffusion, succeeding at picking where action-space search fails.

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 noise levels across future frames. This enables *streaming generation*, where near-future states are progressively resolved and committed while farther-future states remain uncertain and revisable, allowing generation far beyond the training horizon.

We train and evaluate GPWM on ParticleWorlds, a corpus of 25.5 million simulated frames spanning rigid, elastic, elastoplastic, granular, and cloth dynamics, with diverse geometries, physical properties, interactions, and robot embodiments. Unlike prior learned dynamics models trained for individual physical domains (Sanchez-Gonzalez et al., 2020; Zhang et al., 2025; Huang et al., 2026; Wang et al., 2025a), we also train a single *Unified* GPWM across heterogeneous physical regimes. Across six domains, the best GPWM variant reduces prediction error by 31% on average relative to the strongest applicable prior baseline. Remarkably, the Unified model outperforms domain-specific specialists on Dragging and Pick and Place, demonstrating that heterogeneous physical dynamics can be learned within one set of weights. GPWM further generalizes to unseen geometries, physical properties, object counts, and long rollout horizons.

Beyond prediction, GPWM learns a *generative prior over physical interactions* that enables contact-rich planning. Conventional action-space planning searches actuator commands and predicts their consequences, but can struggle to discover complex contact sequences or exploit world-model errors through out-of-distribution actions. By jointly generating actuator and object trajectories, GPWM instead allows planning in its latent noise space, biasing search toward learned interaction patterns. We optimize this noise using either gradient-based latent noise optimization (Eyring et al., 2024; Ben-Hamu et al., 2024; Karunratanakul et al., 2024) or gradient-free sampling, guided by sparse rewards on desired object configurations. Using the same world model, noise-space planning substantially outperforms action-space search on pushing and picking. The resulting actuator trajectories execute successfully in simulation without reference action trajectories or goal-specific policies.

Our contributions are as follows. **(1)** We introduce **GPWM**, a generative world model that predicts the motion of particles and rigid bodies across heterogeneous physical regimes, with streaming generation far beyond its training

horizon. **(2)** We introduce **ParticleWorlds**, a 25.5M-frame corpus spanning diverse physical dynamics, 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 demonstrates that a single Unified model can outperform domain-specific specialists. **(4)** We demonstrate **contact-rich planning through generative interaction priors**, showing that searching the learned distribution of actuator–object trajectories is substantially more effective than direct action-space optimization.

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) represents 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. 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 stable auto-regressive 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, while latent noise optimization (LNO), including D-Flow (Ben-Hamu et al., 2024) and Diffusion Noise Optimization (Karunratanakul et al., 2024), optimizes the initial noise by differentiating through the generative process.

GPWM extends these ideas to contact-rich planning in 3D physical environments. By jointly generating actuator and object trajectories, it learns a prior over physical interactions that can be searched using either gradient-based LNO or sampling-based optimization. Unlike direct action-space search, which can propose out-of-distribution actions and exploit world-model errors, searching this generative prior biases planning toward plausible interaction trajectories. Our experiments show that noise-space planning substantially outperforms action-space optimization even when both use the same world model.

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.

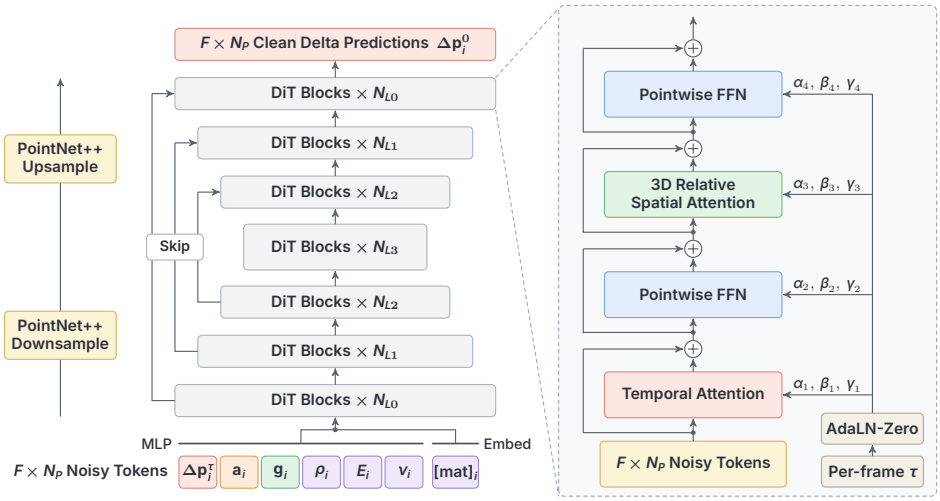


Figure 2 GPWM architecture. Noisy per-particle tokens, which embed displacement, actuator motion, geometry, physical parameters, and material, pass through a multi-resolution DiT. PointNet++ layers pool particles into per-object anchors on the way down and interpolate back on the way up, and long skips connect matching levels. Each block (right) applies temporal attention, 3D relative spatial attention, and pointwise feed-forward layers, modulated by AdaLN-Zero from the per-frame noise level τ, and the network outputs clean future displacements.

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 $\mathbf{P}_t^p \in \mathbb{R}^{N_p \times 3}$. Their frame-to-frame displacements are $\Delta\mathbf{P}_t^p = \mathbf{P}_{t+1}^p - \mathbf{P}_t^p$, and the clean future trajectory is $\mathbf{x}_0 = [\Delta\mathbf{P}_{C-1}^p, \ldots, \Delta\mathbf{P}_{C+F-2}^p] \in \mathbb{R}^{F \times N_p \times 3}$. 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 Planning with Generative Interaction Priors

Planning uses task-specific GPWM models (Appendix E.2). GPWM supports planning in either action space or the latent space of its generative model. In action-space planning, candidate actuator trajectories are evaluated using the action-conditioned world model. Alternatively, by jointly generating actuator and passive-object motion, GPWM learns a prior over physical interactions that enables planning directly in latent noise space. This biases search toward actuator–object trajectories represented by the learned distribution, rather than unconstrained action sequences.

We specify planning objectives through a differentiable reward on the desired final configuration of passive objects. With the world model frozen, we optimize the initial Gaussian noise through *latent noise optimization* (LNO) (Eyring et al., 2024; Ben-Hamu et al., 2024; Karunratanakul et al., 2024). Let $\mathbf{z}$ denote the initial noise for an H-frame rollout and $\hat{\mathbf{s}}_{1:H}(\mathbf{z})$ the generated scene trajectory, reconstructed in physical coordinates by denormalizing and integrating predicted motion. With Π_{pass} selecting the passive-object configuration, LNO performs

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

where η is the guidance step size. As a gradient-free alternative, we also apply the cross-entropy method (CEM) (Rubinstein, 1999) in the same latent noise space, iteratively sampling and selecting noise vectors according to their predicted terminal rewards.

For horizons $H > F$, we use the streaming generation procedure of Section 3.4. In LNO, gradients propagate through successive generation windows, allowing terminal object-space rewards to guide earlier actuator motion. The actuator component of the optimized joint trajectory defines the resulting plan, which is executed open-loop in simulation (Section 5.3). Further details are provided 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 trajectories for falling objects, deformable dragging, garment manipulation, dexterous manipulation, rod and gripper pushing, and pick-and-place (Zhu et al., 2026). 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, up to 898 frames for Garment Folding (over $55\times$ the prediction window). 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, $\downarrow$) 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 page. For cloth, we report symmetric Chamfer distance ($\downarrow$). Planning is instead evaluated by task progress toward the goal (Section 5.3).

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 outperforms the strongest applicable baseline in all six settings, reducing error by **6–67%** (31% on average) with the better of its specialist and unified models (Table 1). The gains span qualitatively different dynamics: error decreases from 53.0 to 40.0 mm on Free Fall, 7.8 to 6.1 mm on Dragging, and 23.9 to 13.3 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 five of the six settings, and achieves the best overall performance on Dragging and Pick and Place, improving even over the GPWM

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. –: not applicable. †Garment Folding reports symmetric Chamfer distance averaged over 898 predicted frames on 20 test trajectories.

Method	Free Fall	Dragging	Garment Folding†	Pushing	Pick and Place	Dexterous Hands
Graph/grid-based	97.2	30.4	20.5	23.9	68.2	26.7
PointWorld	–	–	10.6	28.7	39.8	27.4
PhysCtrl	53.0	7.8	10.7	24.5	34.1	24.8
Ours (Specialist)	**40.0**	6.5	**10.0**	**13.3**	12.6	**18.7**
Ours (Unified)	41.5	**6.1**	10.8	18.0	**11.1**	22.6

specialists (Table 1). 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.

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 generalization 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 37.5 to 45.2 mm. In contrast, PhysCtrl increases from 48.5 to 120.9 mm. The Unified model is also stable across these shifts and ranks second in all four shape and object-count conditions.

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) or unseen within the training range (30, 300 kPa). Material types are weighted equally.

	1–4 objects		5–7 objects		Stiffness	
Method	Seen shapes	Unseen shapes	Seen shapes	Unseen shapes	Seen	Unseen
GNS	96.8	97.2	100.9	97.6	116.7	113.2
PhysCtrl	48.5	53.0	122.7	120.9	51.2	51.7
Ours (Specialist)	**37.5**	**40.0**	**45.1**	**45.2**	**47.3**	**48.9**
Ours (Unified)	39.0	41.5	47.8	47.5	54.5	53.1

GPWM generalizes to unseen material stiffness. We next vary Young's modulus from 10 kPa to 1 MPa, covering values seen during training and unseen values within the training range. Models receive the material parameters and initial observations for each simulation. GPWM's error rises by less than 2 mm on unseen values (47.3 to 48.9 mm) while remaining below the strongest baseline. This demonstrates robust prediction under interpolation of material stiffness.

GPWM generalizes to objects reconstructed from images. We also test GPWM on previously unseen object geometries reconstructed from single RGB images (Figure 3). 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.

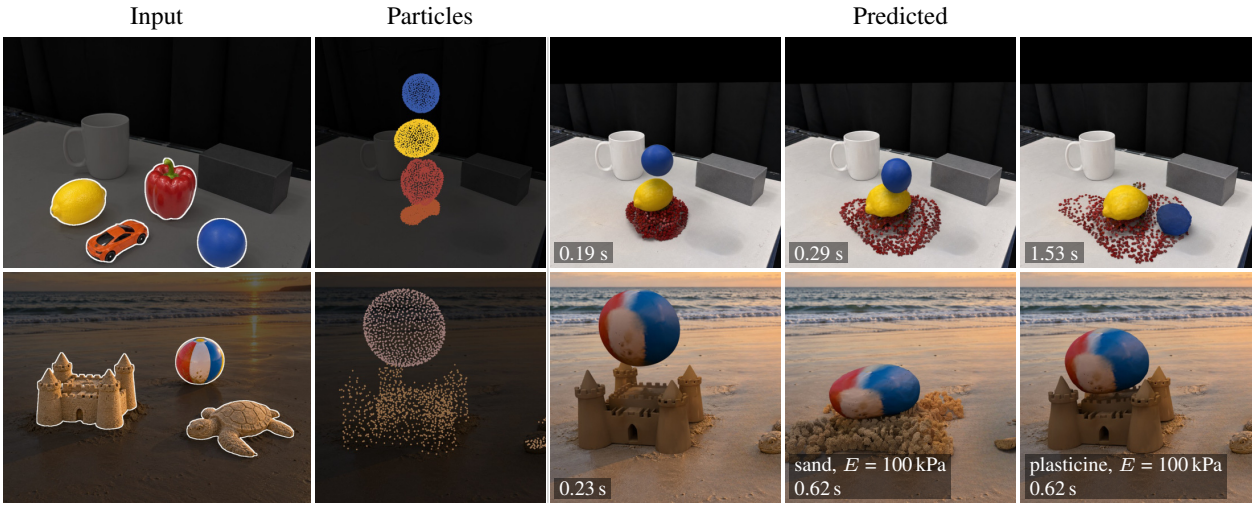


Figure 3 Predictions from a single image. Each row shows the input with the reconstructed objects outlined, their particles, and the predicted motion. Top, a tower of a toy car (elastic, 1 MPa), a pepper (sand, 100 kPa), a lemon (plasticine, 100 kPa), and a ball (elastic, 100 kPa). Bottom, an elastic ball (10 kPa) dropped onto a sand or a plasticine castle (100 kPa).

5.3 Planning with Generative Interaction Priors

GPWM supports two approaches to model-based planning. In *action-space planning*, candidate actuator trajectories are evaluated using an action-conditioned world model. However, searching unconstrained action sequences presents two challenges: complex contact-rich behaviors are difficult to discover without a suitable action proposal policy, and optimization may exploit prediction errors by selecting actions outside the model's training distribution.

GPWM offers an alternative: by jointly generating actuator and passive-object trajectories, it learns a *generative prior over physical interactions*. Planning can then search the model's latent noise rather than raw actuator commands, biasing optimization toward plausible actuator–object trajectories. We optimize this noise using either gradient-based latent noise optimization (LNO) (Eyring et al., 2024; Ben-Hamu et al., 2024; Karunratanakul et al., 2024) or gradient-free sampling. In LNO, a differentiable terminal reward on the passive-object configuration is backpropagated through the generated trajectory to the initial noise (Sec. 3.5).

We compare action- and noise-space planning on three contact-rich tasks, using task-specific world models and rewards defined solely on desired object outcomes (Fig. 5). All generated actuator trajectories are executed **open-loop in simulation**, testing whether plans optimized through the learned dynamics succeed under the underlying simulator.

Tasks and metrics. We evaluate three tasks requiring different contact strategies. **Pushing** requires a rod to reposition around a rigid block before pushing it toward a target over 60 steps. **Pile Manipulation** requires a paddle to move a granular pile toward a target over 150 steps, typically through multiple pushes. **Picking** requires establishing a stable grasp and lifting an object to a target over 60 steps. We evaluate five objects of different shapes and sizes (milk carton, peanut-butter jar, soda can, pasta box, and water bottle), each with five targets.

We report *task progress*, defined as the fraction of the initial object-to-target distance closed by the end of the episode. Performance is measured across planning budgets of up to 100 s for Pushing and 200 s for Pile Manipulation and Picking, on a single L40S GPU. Additional task details are provided in Appendix E.1.

Planning methods. *Noise-space planning.* *Latent Noise Optimization (LNO)* optimizes the initial Gaussian noise by backpropagating the terminal reward through the generated trajectory. As a gradient-free alternative, *Noise CEM* applies the cross-entropy method in the same latent space: each iteration samples candidate noise vectors, generates and evaluates their trajectories, and refits the sampling distribution to the highest-reward candidates. Both methods exploit the learned joint distribution over actuator and object motion. Generated actuator trajectories are executed using a simple tracking controller.



Figure 4 Planning performance. Task progress versus planning budget, measured after open-loop execution in simulation. Results are averaged over 50 scenarios for Pushing and Pile Manipulation and 25 problems for Picking. Reported values are interpolated at each task's planning budget (dashed lines).

Action-space planning. *Action CEM* instead optimizes raw per-frame end-effector displacements, evaluating each candidate using an action-conditioned world model. We apply Action CEM to both GPWM and PointWorld (Huang et al., 2026). Crucially, action-space and noise-space planning use the *same GPWM checkpoint*, isolating the effect of the search space rather than differences in model accuracy. We vary planning budgets through the number of samples, optimization iterations, or CEM rounds. Implementation details are provided in Appendix E.3.

Generative interaction priors enable contact-rich planning. Figure 4 shows that searching GPWM's noise space substantially outperforms direct action-space optimization, even when both use the same world model. On Pushing and Picking, noise-space planners achieve 0.48–0.57 and 0.40–0.56 task progress, respectively, while action-space planners remain at or below 0.01. These tasks require discovering structured sequences of contact: repositioning the rod before pushing, or establishing a grasp before lifting. Such behaviors are unlikely to emerge from random action proposals, whereas GPWM's learned interaction prior provides plausible trajectories that can be steered toward the goal.

Pile Manipulation presents a different regime: even random paddle contacts can produce task progress, making action-space search more competitive. Nevertheless, noise-space planning achieves 0.21–0.23 task progress, compared with 0.19 for the strongest action-space baseline.

Beyond discovering effective contacts, the generative prior also reduces the risk of exploiting world-model errors. Direct action optimization can select out-of-distribution trajectories that appear successful under the learned dynamics but fail when executed in simulation, as illustrated by Picking (Appendix E.5). Noise-space optimization instead searches through the model's learned distribution of interactions, favoring actuator motions whose predicted effects are more likely to transfer to the underlying simulator.

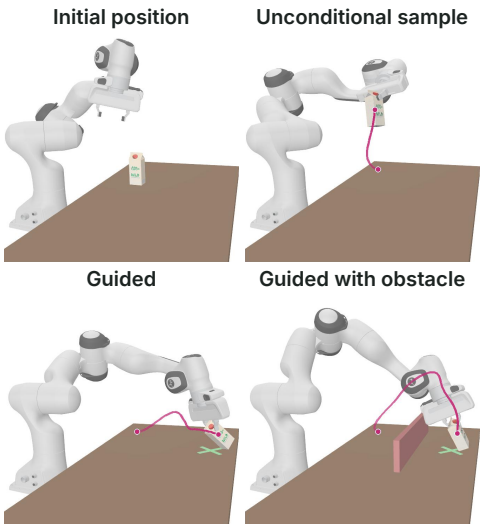


Figure 5 Planning with a generative interaction prior. Latent noise optimization guides a frozen GPWM toward an object-space goal (green ×). When a wall blocks the direct path, the generated actuator trajectory lifts the object over it.

Together, these results demonstrate that *accurate dynamics prediction alone is insufficient for effective contact-rich planning*: the learned distribution of actuator–object interactions provides a powerful search prior, enabling object-space goals to be translated into actuator trajectories without reference demonstrations or goal-specific policies.

Additional qualitative results, including baseline comparisons, generalization to real-world images, and planning examples, are available on our project page.

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 still trails domain-specific models on some 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 stable auto-regressive generation far beyond the training horizon, including coherent deformable-object prediction for more than $55\times$ the trained prediction window. Beyond prediction, GPWM learns a generative prior over coupled actuator–object motion that enables contact-rich planning from object-space goals. Searching this learned interaction prior substantially outperforms direct action-space optimization, even when using the same world model. Together, these results establish generative particle world models as a common framework for long-horizon physical prediction and goal-directed interaction in 3D.

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.

Bardienus P Duisterhof, Kaifeng Zhang, Adam Hung, Bowen Wen, Stan Birchfield, Yunzhu Li, Deva Ramanan, and Jeffrey Ichnowski. Pointzero: 3d point track completion for learning transferable 3d dynamics. *arXiv preprint arXiv:2609.19142*, 2026.

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.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. *URL https://kellerjordan. github. io/posts/muon*, 6(3):4, 2024.

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.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

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.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *Conference on Robot Learning*, pp. 1049–1065. PMLR, 2021.

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.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

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.

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.

He Zhu, Lusen Zhao, Kwan Man Cheng, Su Li, and Katerina Fragkiadaki. SkillWeaver: Agentic exploration over neural interaction skills for scalable robot data generation, 2026. URL https://arxiv.org/abs/2609.36171.

A ParticleWorlds Dataset Details

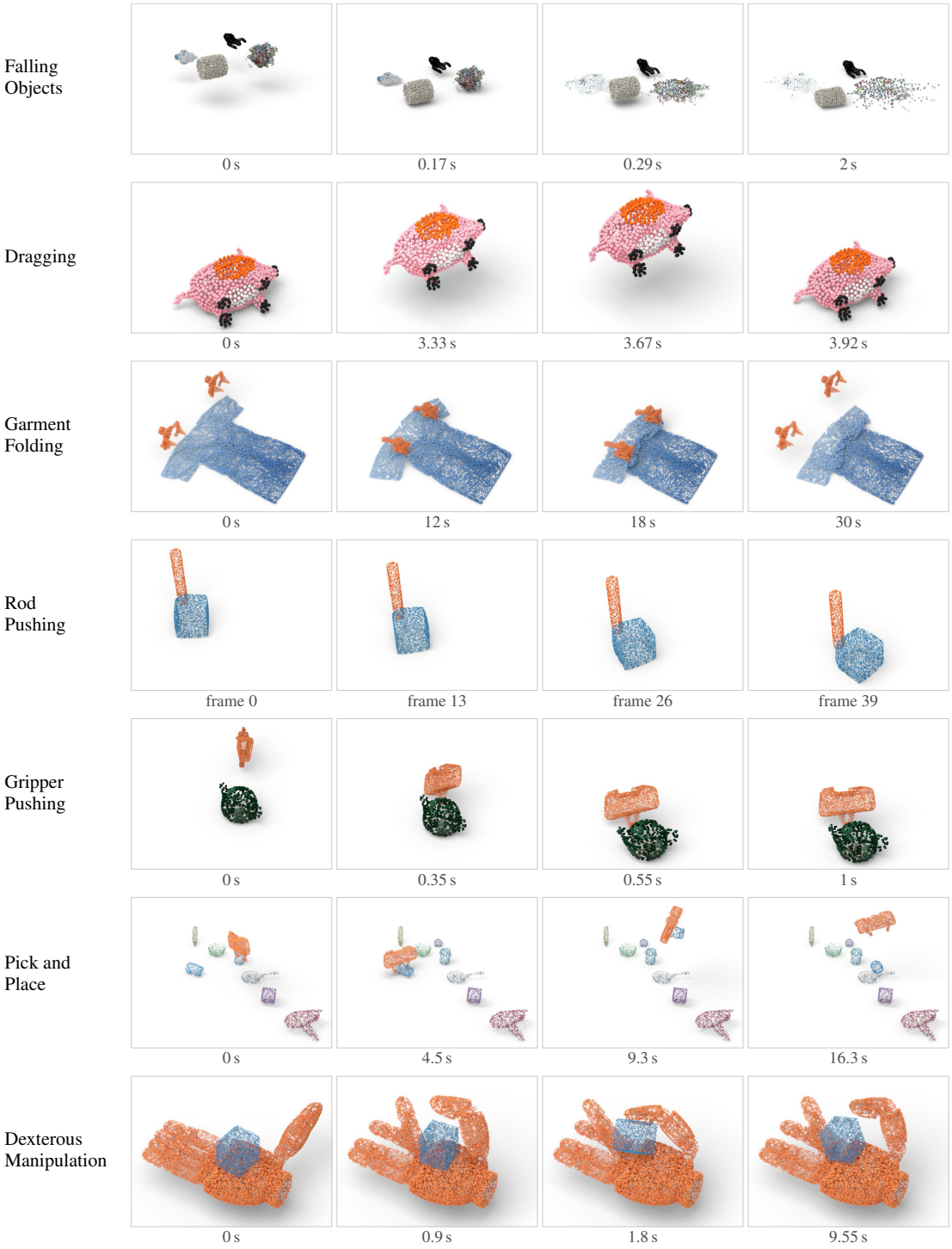


Figure 6 Examples from the ParticleWorlds domains. Each row shows one recorded episode at four moments, labeled with the time since its first frame; Pushing has one row for each of its two sources. From the top, four objects of three materials fall onto the table, a piggy bank is lifted by a patch on 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 3 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 3 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 900 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 our recordings of scripted rod motion against a block, with 40 states and 256 points per episode. Gripper Pushing uses our reinforcement-learning policy recordings of successful object pushes. Pick and Place recordings come from SkillWeaver (Zhu et al., 2026). 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 4. 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 4 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 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 GPWM models are trained with Muon (Jordan et al., 2024) on the weight matrices (learning rate 3×10^{-4}, with the update scaled to match AdamW's RMS (Liu et al., 2025)) and AdamW on all remaining parameters (learning rate 10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$), with decoupled weight decay of 10^{-3} for both. Both learning rates follow a 5,000-step linear warmup and a cosine decay to 1% of their peak value. We clip gradients to a global norm of 1.0, keep an exponential moving average of the weights with decay 0.9999, and train in bf16 mixed precision. We train each specialist on two NVIDIA RTX PRO 6000 GPUs for 200k iterations, and the unified model on four for 600k iterations. Note that while

the unified model has seen more training iterations, these update steps are split across 6 domains. So the unified model gets only 100k iterations per domain, half the amount that the specialist models see.

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). PointWorld is a deterministic point-cloud dynamics model with a PTv3 backbone. We train it from scratch with its released code adapted to our setting: it operates on complete 3D point clouds and is applied recurrently to produce the autoregressive rollouts used in our evaluation. To match our 100 M-parameter specialists, we use the released small variant with wider channels (about 100 M parameters), and train each run on the same episodes, splits, start frames and number of action-conditioned training windows as our corresponding specialist. PointWorld voxel-downsamples each point cloud before its backbone, by default on a 15 mm grid; on several of our domains this discards much of the objects' geometry, so we use a 1 mm grid instead, and at evaluation each surviving point's prediction is copied to the points merged into its voxel. We drop the input features our data does not provide, such as color and surface normals: each scene point carries only its positions over time and its distance to the robot, and each actuator point its positions, velocity and acceleration. We disable the training-time sphere crop, which removes scene points far from the actuator, so the model sees the whole scene, and fit its normalization statistics on 20,000 training windows of each domain. The released optimizer and augmentations are otherwise unchanged.
PointWorld receives the actuator's points at every frame and predicts only the scene, the actuator's future motion is an input and is never predicted. It observes a single context frame, where our models receive 4, and we extend its prediction horizon from the released 10 frames to 16 to match our models. For cloth, training windows are drawn with the same motion weighting as our loader. Longer evaluation windows are rolled out autoregressively in 16-frame chunks, each anchored on the model's own previous prediction.

GNS (Sanchez-Gonzalez et al., 2020). We use GNS only for passive dynamics (Free Fall) where no actuator drives the scene; it is not run on the action-conditioned settings where PGND serves as the graph/grid-based baseline. We train it on the same episodes, splits, and start frames as our models, 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 it out autoregressively at every step to obtain a full-length prediction. GNS carries a 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.

PGND (Zhang et al., 2025). We train PGND with 4 given and 16 predicted frames at each domain's native frame rate, rolling the model out one frame at a time with each prediction fed back as input. 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 $\sim$100 material points is driven by a commanded velocity, so there we use PGND's sphere collider, placed on the patch at the start of each drag

and then moved only by the commanded velocity, independent of the material. PGND does not condition on physics properties; on Dragging, where our model receives each object's material, we give PGND the same information as extra per-particle input channels. PGND simulates inside a fixed grid. Rather than excluding episodes that do not fit, we enlarge the grid or rescale the scene so that every training episode fits, and for cluttered Pick and Place we recenter a fixed $1.4 \times 1.0 \times 1.4$ m grid on the gripper for each training window, freezing objects that leave it. In every domain the network's neighborhood radius covers the same 20 cm as PGND's released recipes. Otherwise, we train PGND with the same episodes, splits, and start frames as our own loader, and match each run to our corresponding specialist in gradient steps and windows per step.

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 5 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 5 Test clips. The 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	20	garments	898	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

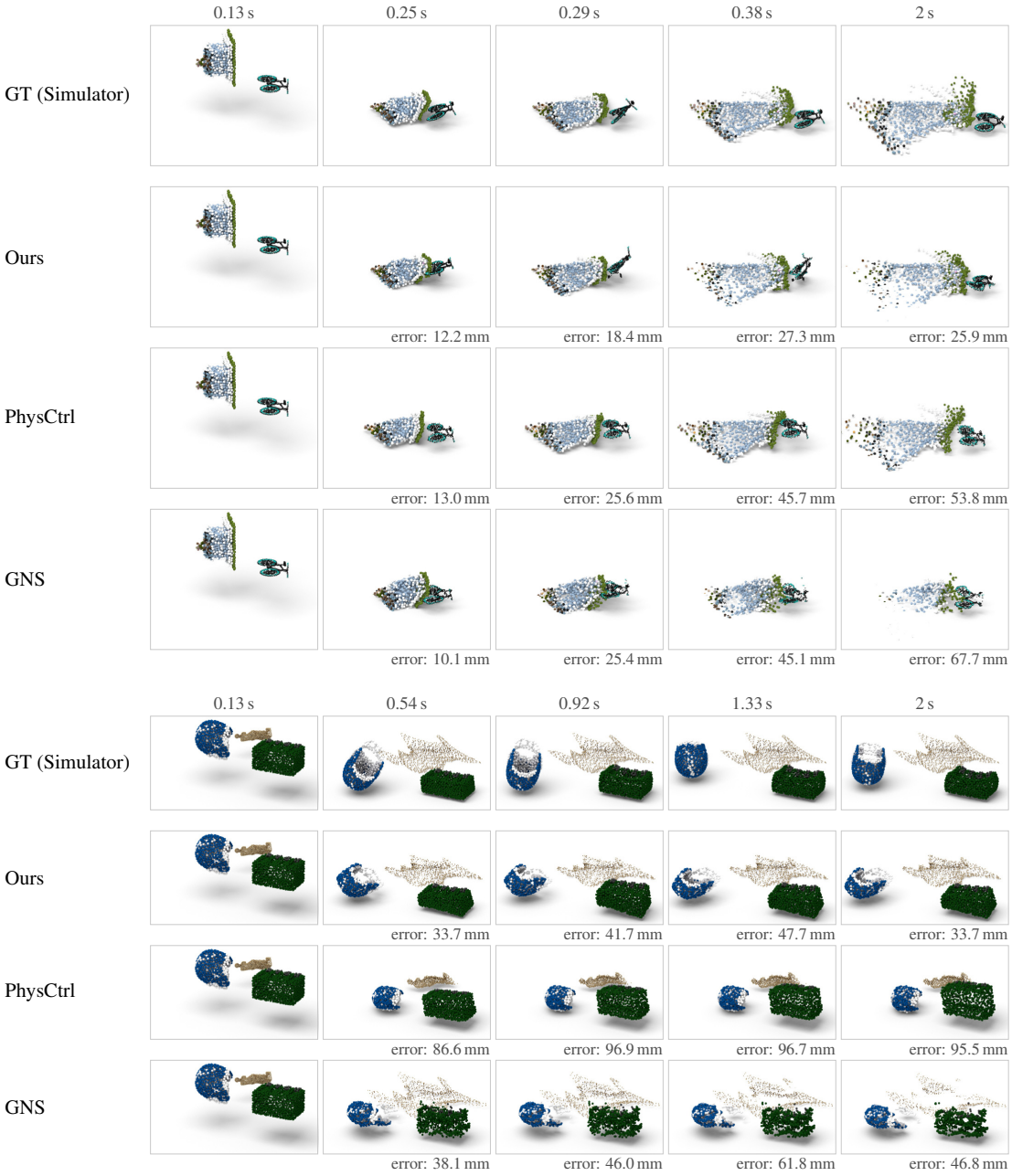


Figure 7 Free Fall. Two test clips with unseen shapes. Rows show the simulator's recording and three models, and columns show the last given frame, three moments of the fall and the last predicted frame, labeled with the time since release. Ours has the lowest clip error on 172 of the 200 Free Fall test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

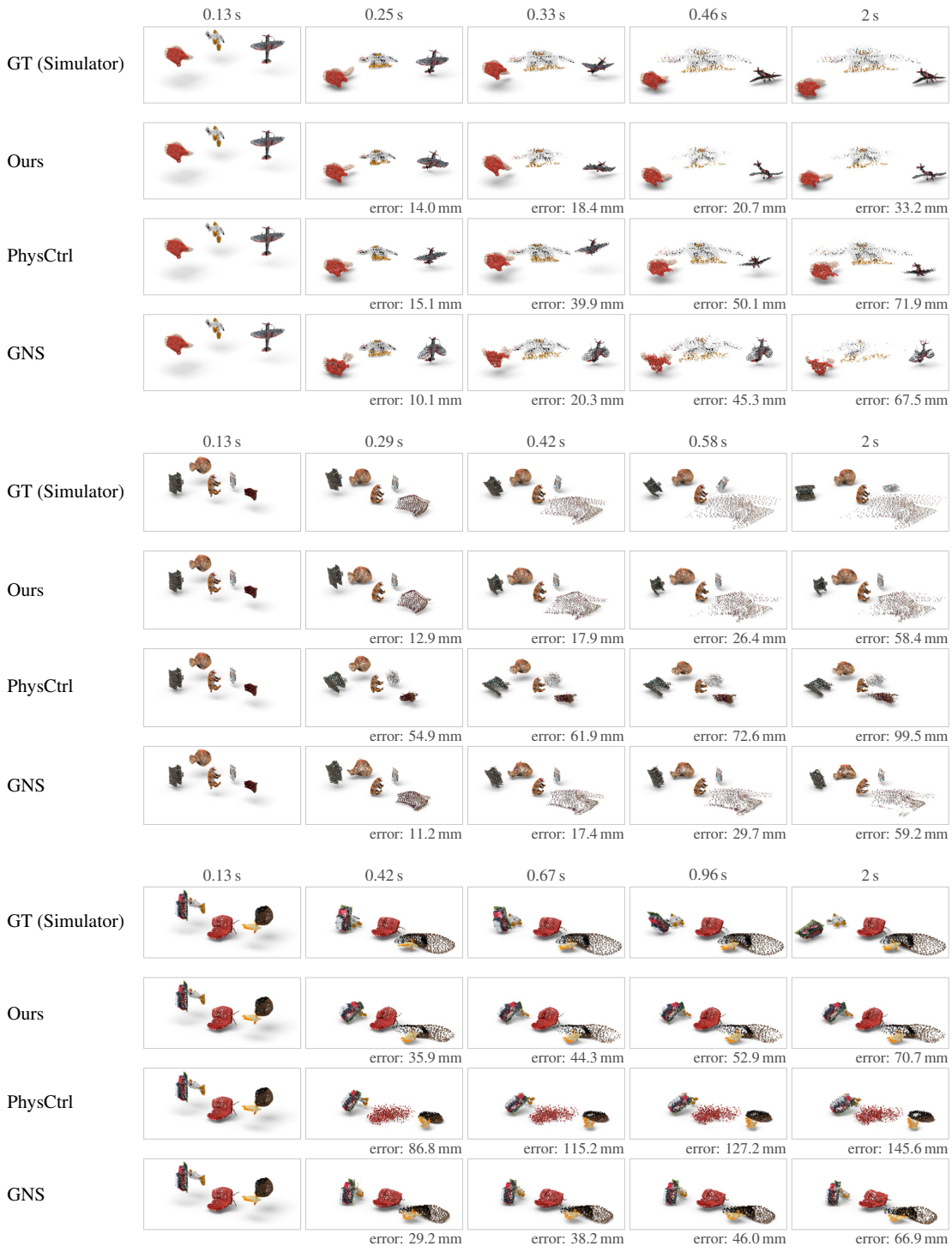


Figure 8 Free Fall, continued.

D.2 Dragging

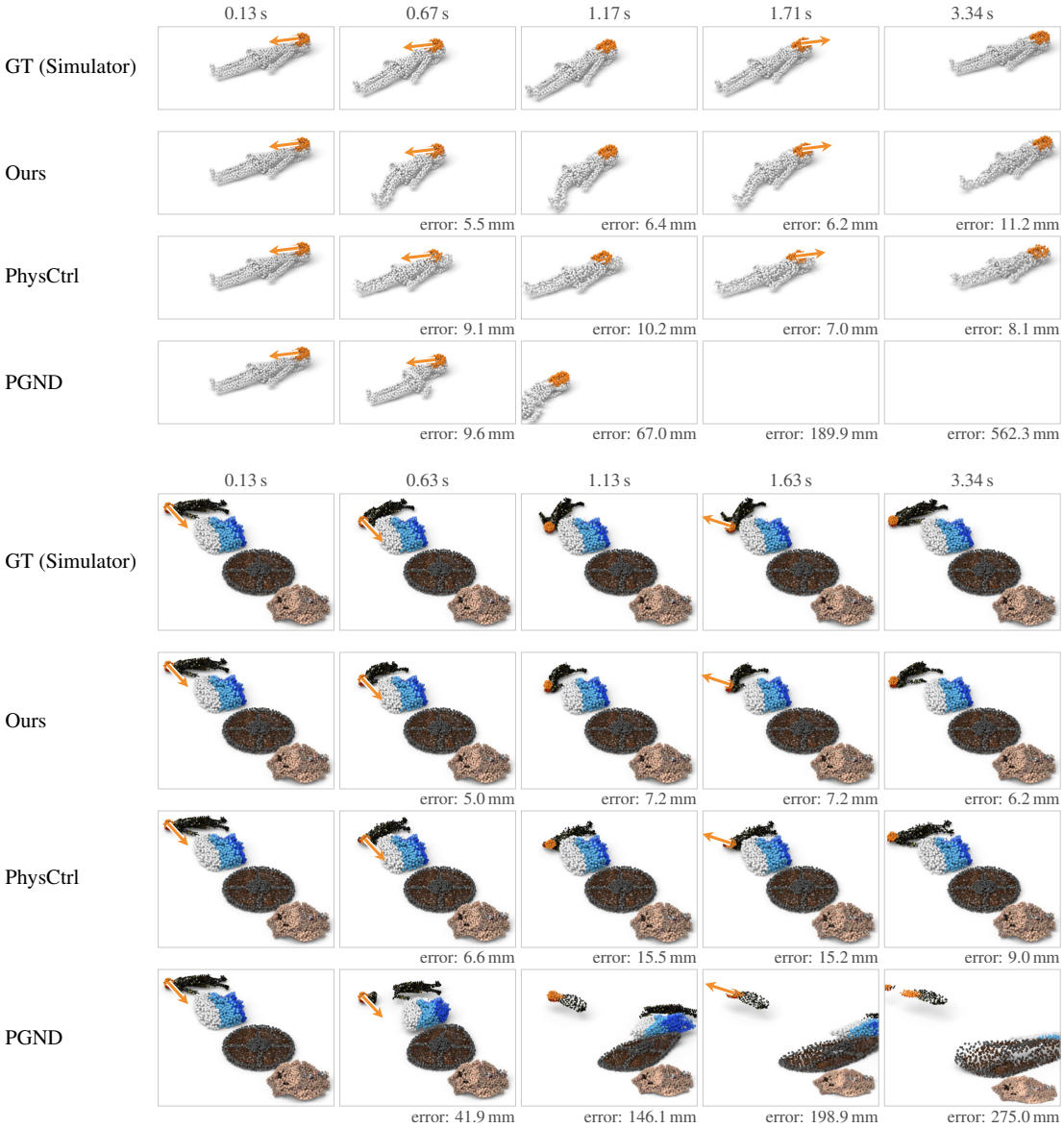


Figure 9 Dragging. Two test clips with unseen shapes, laid out as in Figure 7. The orange arrow shows the direction in which the dragged patch (orange) is pulled at that moment, from the recorded command that every model receives. Ours has the lowest clip error on 81 of the 100 Dragging test clips.

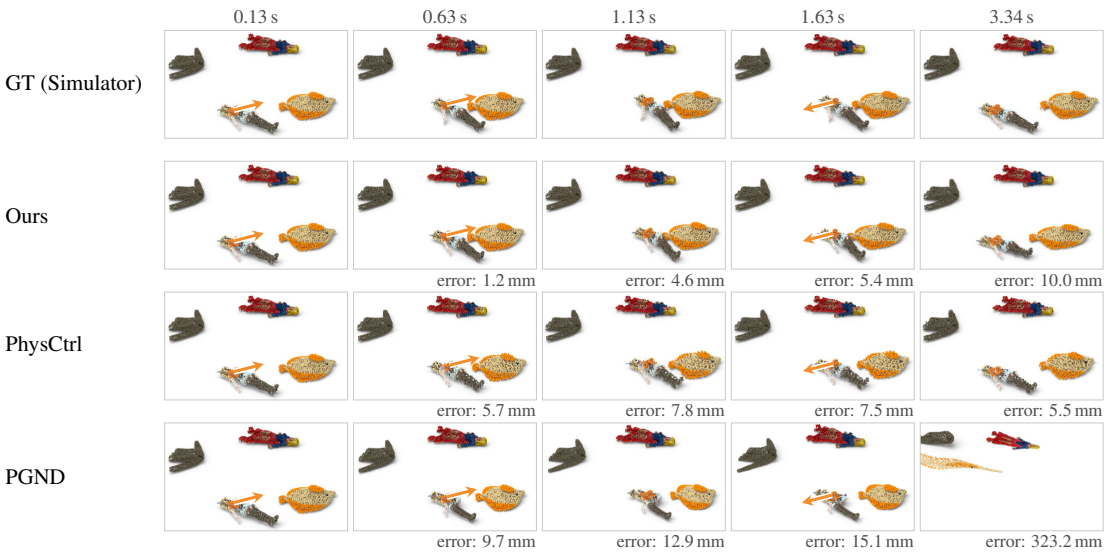


Figure 10 Dragging, continued. On this clip PhysCtrl has the lowest error.

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.

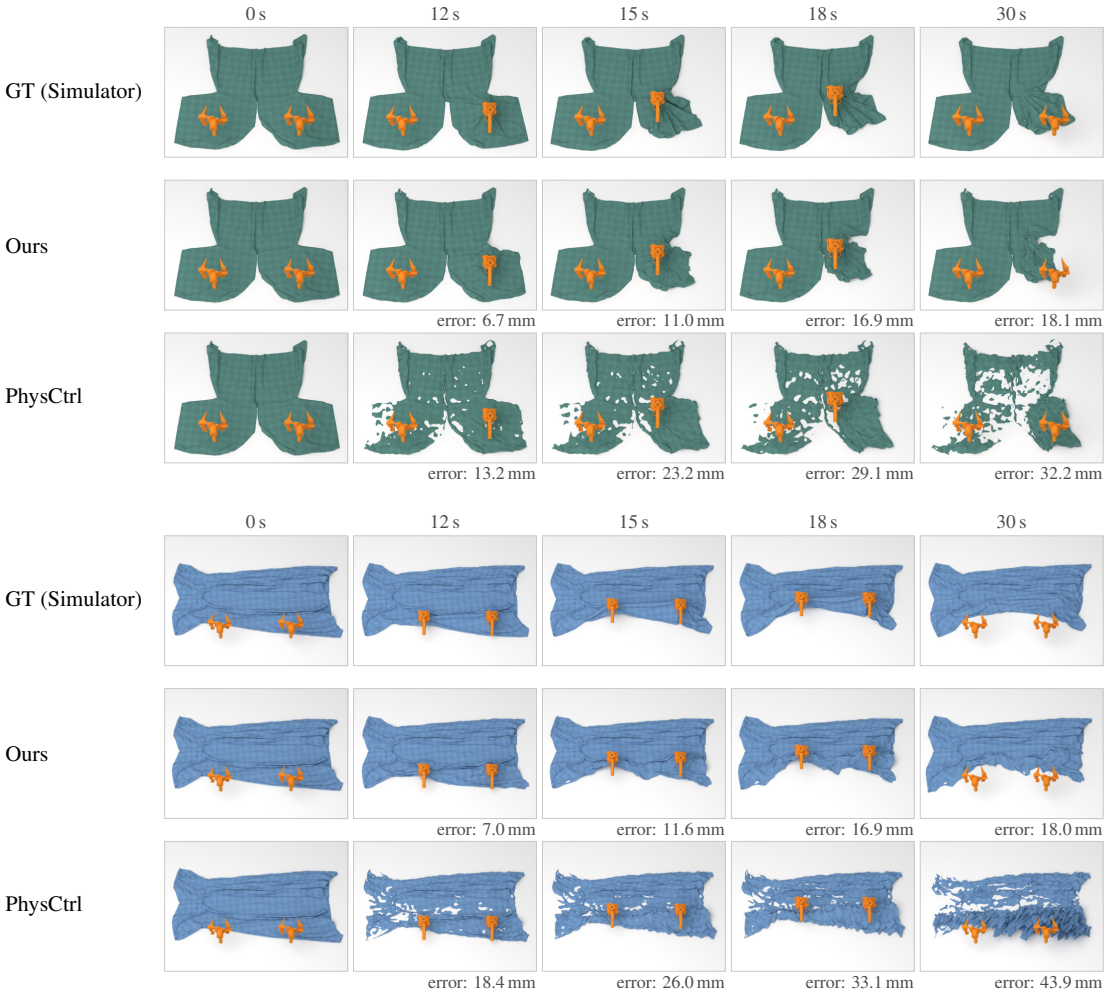


Figure 11 Garment Folding, rendered back to the simulation mesh. A jacket and a T-shirt not seen in training, 0 to 30 s after the given history frames. Rows show the simulator and each model's particles driving the same mesh, with the grippers (orange) at their recorded poses. The error under each panel 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

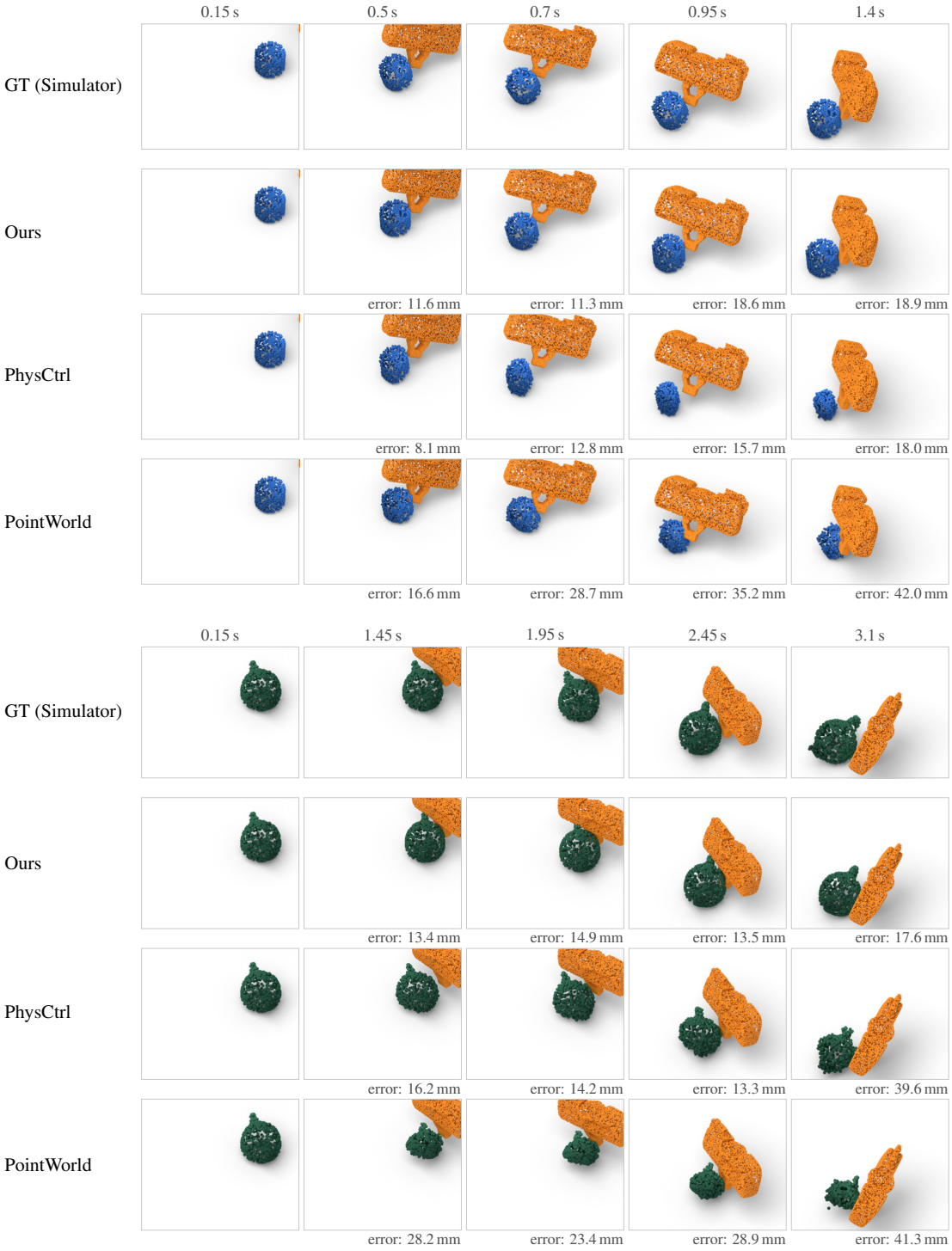


Figure 13 Pushing. Test clips with objects held out from training, laid out as in Figure 7. The gripper (orange) is given to every model and drawn at its recorded pose. Ours has the lowest clip error on 165 of the 200 Pushing test clips.

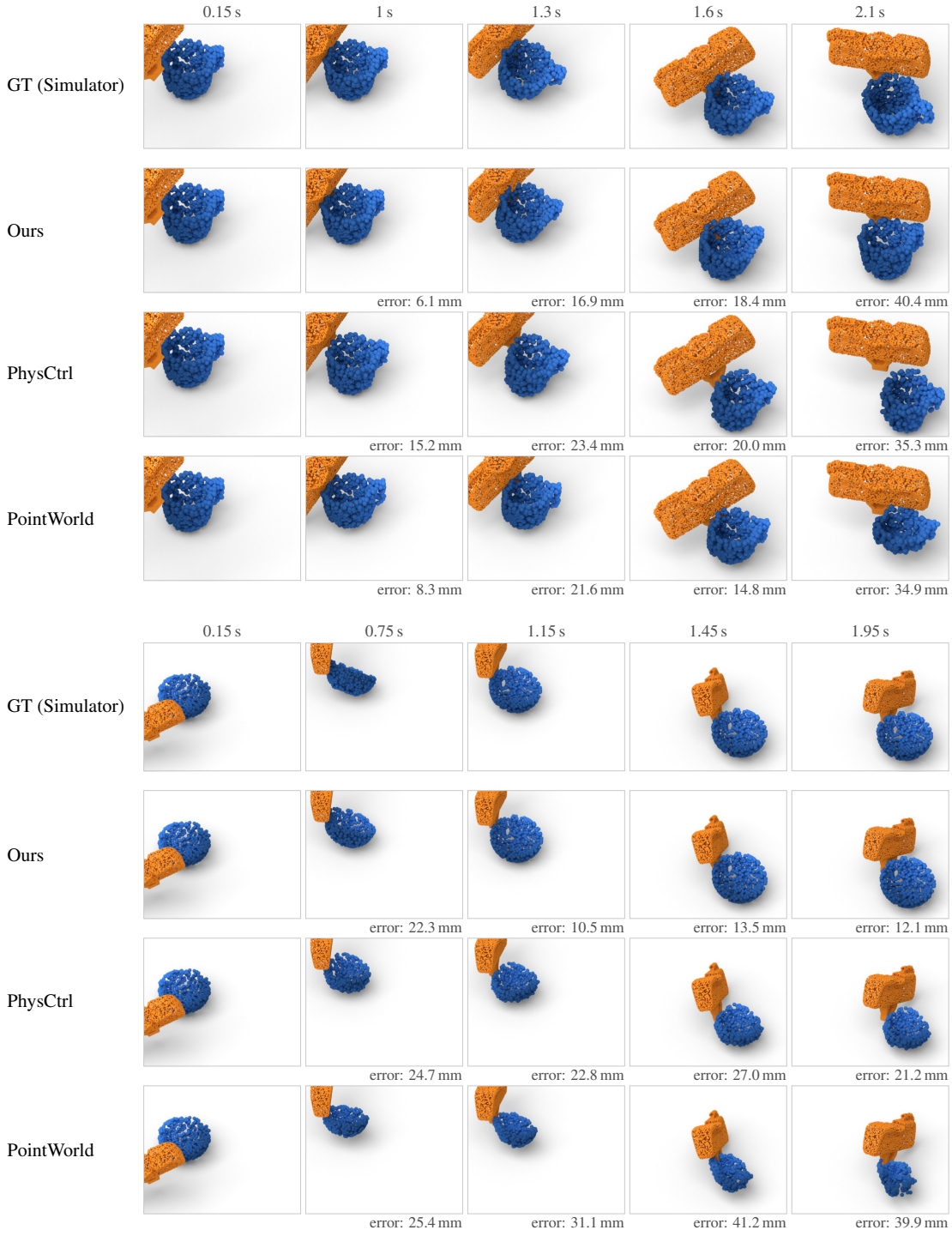


Figure 14 Pushing, continued.

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

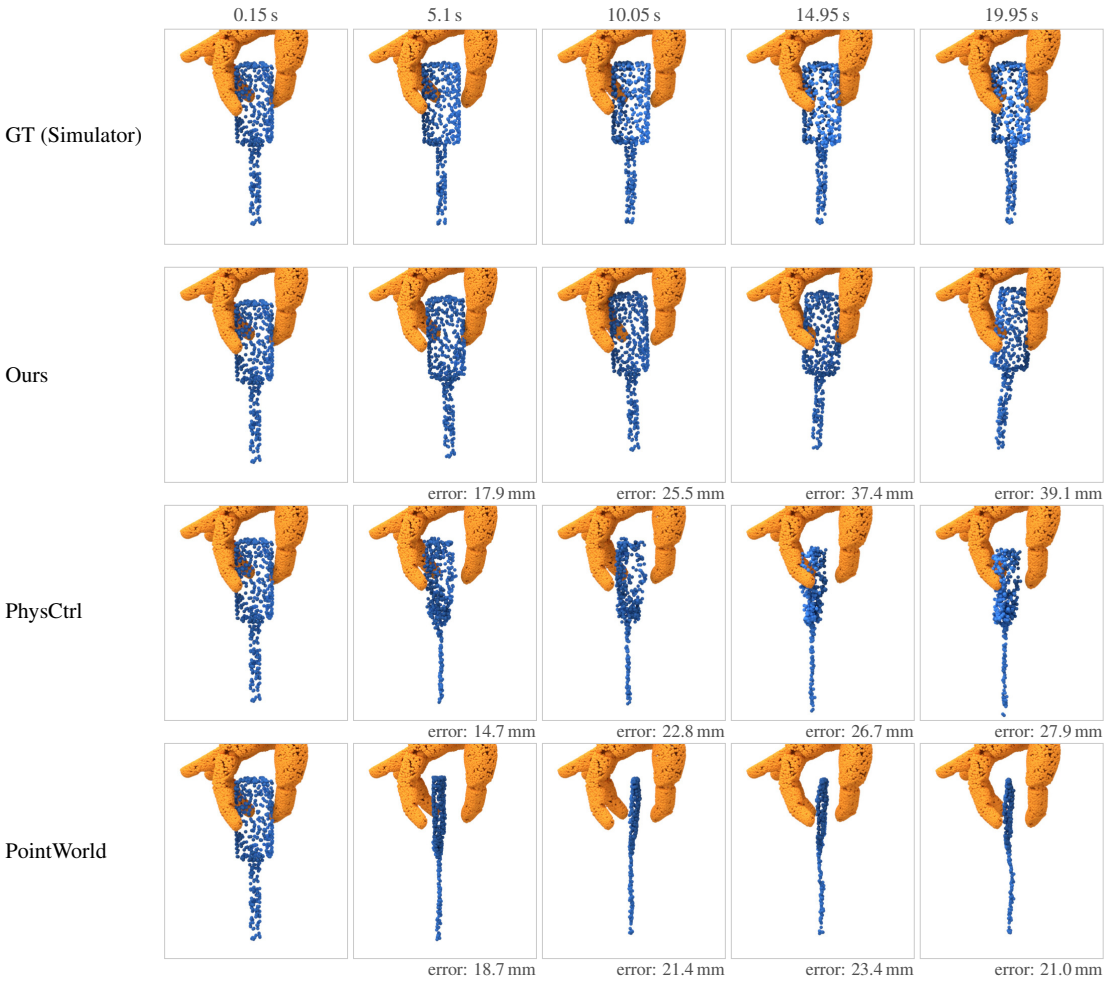


Figure 17 Dexterous Hands. A held-out screwdriver turned by the Sharpa hand. On this clip PhysCtrl and PointWorld have lower mean error than ours, but both move every point freely and thin the screwdriver.

E Planning

E.1 Planning Tasks

In every task an actuator must bring a passive object to a target within a fixed number of steps. The reward is sparse and defined only on the final frame: the negative distance from the object's centroid to its target. Each planner produces a single open-loop plan for the full horizon inside its world model; the plan is then executed in the task's simulator from the same start state, and the simulator never enters planning.

- **Pushing.** A rod pushes a rigid block toward a target on a tabletop. Targets typically require the rod to move around the block before pushing, so random rod motions rarely produce progress. Horizon 60 steps (3× the model window), 50 scenarios, executed in ManiSkill.
- **Pile Manipulation.** A paddle moves the centroid of a granular pile toward a target in the plane. World models are trained on 6,000 trajectories of 300 frames from a scripted policy that randomly pushes and presses the pile. Horizon 150 steps (7.5× the model window), 50 scenarios, executed in Genesis.
- **Picking.** A parallel-jaw gripper must grasp a tabletop object (milk carton, peanut-butter jar, soda can, pasta box, or water bottle) and carry it to a target 15 cm above and up to 15 cm to the side of its start (5 objects × 5 targets). Horizon 60 steps, executed with a floating Franka hand in Isaac Lab.

Figure 18 shows one scenario of each task and the executed plan found by LNO.

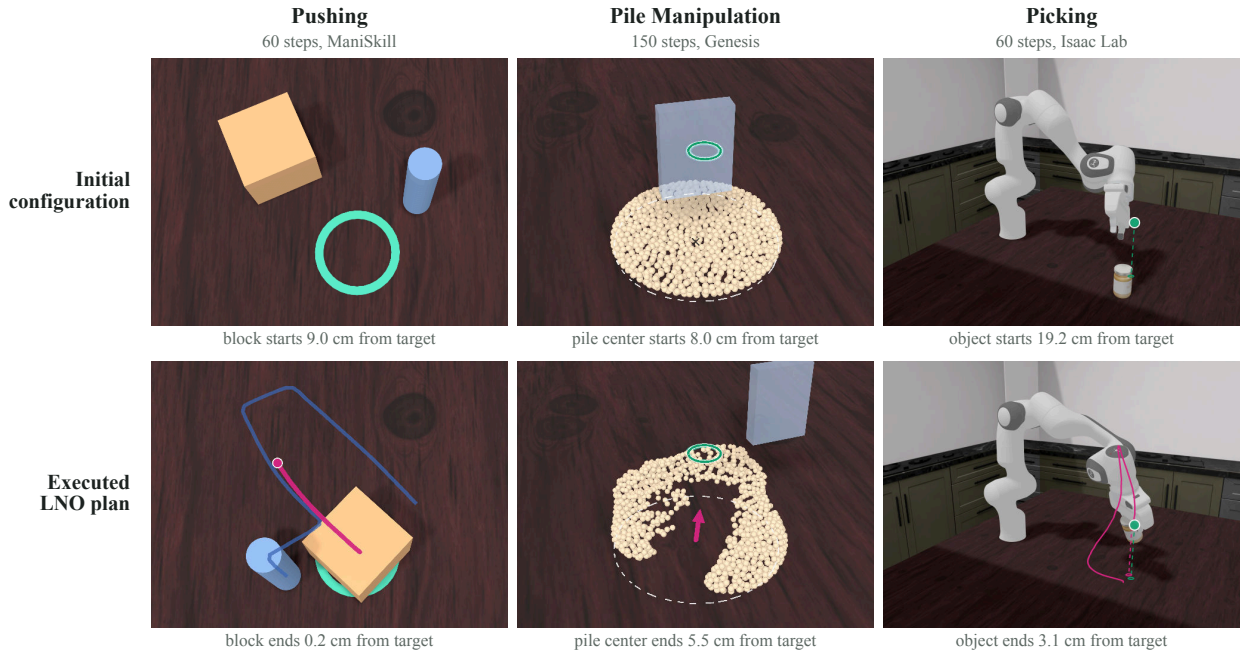


Figure 18 Planning tasks. One scenario per task: the initial configuration with its target (top) and the outcome of the LNO plan executed open-loop in the simulator (bottom). The rod moves around the block before pushing it; the paddle sweeps part of the pile onto the target; the gripper grasps the jar before carrying it to the target.

Metric and budget. Task progress is $(d_0 - d_T)/d_0$, where d_0 and d_T are the object-to-target distances at the start and at the end of the executed episode; it is averaged over scenarios. The planning budget is the wall-clock time of the whole planning call on one L40S GPU. Each planner is run at a ladder of settings (Table 6), and the value reported at a task's budget is linearly interpolated between the two settings that bracket it.

E.2 World Models

GPWM. Each task uses a task-specific GPWM (100M parameters, 200k training steps) trained on that task's data. During training, the actuator's given motion is withheld in a random half of the windows and replaced by a learned token, so the model learns both to generate the actuator jointly with the scene and to follow a given actuator motion. Planning uses the same checkpoint in both modes: noise-space planners use the generative mode, and Action CEM uses the action-conditioned mode, in which the actuator's motion is given as per-frame rigid displacements. Rollouts over the full horizon use the streaming sampler of Section 3.4 with 6 denoising steps.

PointWorld. Our PointWorld (Huang et al., 2026) baseline is a deterministic, recurrent PTv3 dynamics model, obtained by adapting PointWorld to operate on complete 3D point clouds and to be applied recurrently (Appendix B.3). It is trained on the same data, for a number of action-conditioned training windows matched to GPWM's, conditions on the actuator's motion, and predicts the motion of the rest of the scene; longer horizons are rolled out by chaining 16-frame predictions, each starting from the previous prediction.

E.3 Planners

Cross-entropy method. Noise CEM and both Action CEM variants use the same cross-entropy method (**CEM**) and differ only in what it searches over. Each round draws K candidates from a Gaussian, rolls all of them out over the full horizon, and refits the Gaussian to the best eighth of the candidates. We use the elite-retention and variance-floor modifications of iCEM (Pinneri et al., 2021). The best-scoring candidate found over all rounds is the plan that is executed.

Noise CEM. The CEM variable is the noise of GPWM's sampler. We use a stochastic sampler, which draws fresh noise at every denoising step, so the searched noise is the initial draw together with these per-step draws. With a deterministic (ODE) sampler only the initial noise would remain, and noise-space planning applies to either. Every candidate is a complete generated trajectory, and the plan is its actuator motion.

Latent noise optimization. LNO optimizes the same noise by gradient ascent on the reward, with the weights of GPWM frozen (Fig. 19). The streaming sampler strides the model window across the full horizon of T frames, and the terminal reward $r = -\|c(\hat{\mathbf{s}}_T) - g\|_2$ on the predicted object centroid is backpropagated through every model call. LNO first draws a pool of at least 32 random samples and keeps the best P; each is then refined with $I = 5$ AdamW steps (learning rate 0.1, gradient clipping at 1.0, and a small penalty that keeps the noise statistically Gaussian). The member with the highest predicted reward is executed.

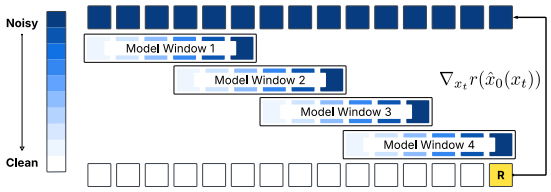


Figure 19 Guidance step with streaming lookahead. For a gradient step on noisy sample x_t, we get an estimate of the clean trajectory using streaming lookahead. Backpropagating through the lookahead allows guidance to optimize for long horizon tasks with sparse rewards only on the final frame.

Action CEM. Both world models are searched in the same action space: the actuator's per-frame motion along the degrees of freedom it uses in the training data. This is the rod's planar translation for Pushing, the paddle's translation and yaw for Pile Manipulation, and the hand's 6-DoF motion plus gripper opening for Picking (the two fingers share one command). Candidates are sampled independently per frame, and to keep motions in distribution every step is clipped to the 99th percentile of the recorded steps, both during planning and at execution.

Scaling the planning budget. A planner can spend more time in several ways: more candidates per CEM round (K), more CEM rounds (R), more LNO members (P), or more gradient steps per LNO member (I). For each planner we sweep the setting that we found empirically to give the best performance for its planning time, and keep the others fixed (Table 6). Noise CEM and Action CEM with GPWM sweep K, with R fixed at three rounds, so the two search spaces are compared at identical settings. LNO sweeps P with $I = 5$; for low planning budgets we also decrease the size of the initial screening pool. PointWorld sweeps R.

Table 6 Planner settings swept by each curve in Fig. 4; settings not listed are fixed. K: CEM candidates per round; R: CEM rounds; P: LNO members refined with 5 gradient steps each. LNO's smallest budgets decrease the screening pool and take no gradient steps.

Planner	Pushing (100 s)	Pile Manipulation (200 s)	Picking (200 s)
LNO	pool 4–32; P 1–48	pool 4–32; P 1–6	pool 4–32; P 1–12
Noise CEM	K 4–192	K 4–64	K 4–96
Action CEM (GPWM)	K 4–192	K 4–64	K 4–96
Action CEM (PointWorld)	K=64, R 3–48	K=64, R 3–36	K=64, R 3–15

E.4 Execution

For noise-space planning, the plan is the actuator motion that GPWM generates. At every frame we fit a rigid pose to the predicted actuator particles (Kabsch alignment to the actuator's initial geometry), and the simulator tracks these poses open-loop: the rod and the paddle by pose control, and the Franka hand as a floating base together with its gripper opening. For action-space planning, with either world model, the planned per-frame actions are executed directly.

E.5 Additional Analysis of Planning Results

Predicted versus executed progress. Table 7 compares each planner's progress as predicted by its world model with the progress it achieves in simulation. Noise-space plans lose part of their predicted progress on Pushing and Picking but keep most of it, and on Pile Manipulation their predictions carry over almost exactly. Action-space plans predict progress on every task and achieve almost none, with the exception of PointWorld on Pile Manipulation. Figure 21 shows each planner's predicted and simulated outcomes on Pile Manipulation.

Table 7 Predicted and executed task progress (%), at each task's planning budget: progress in the world model's own rollout of the chosen plan → progress after open-loop execution in simulation.

Planner	Pushing	Pile Manipulation	Picking
LNO	89 → 57	27 → 23	89 → 56
Noise CEM	94 → 48	21 → 21	70 → 40
Action CEM (GPWM)	23 → 0	7 → 1	78 → 0
Action CEM (PointWorld)	37 → 1	29 → 19	98 → 0

Pushing and Picking. Action-space search in either world model does not find the contact sequence these tasks require, and Table 7 shows that it fails differently on the two tasks. On Pushing, the search fails to find progress even inside the world models: its chosen plans are predicted to make only 23–37% progress, against 89–94% for noise-space plans. Under a sparse terminal reward, random per-frame motions that bump the block without pushing it toward the target give the search little to improve on, and the rod rarely moves around the block. On Picking, the search does find plans the world models rate highly (78–98% predicted progress), but none of that progress occurs in simulation. No progress is possible before a grasp, and a grasp is hard to discover in the high-dimensional space of hand and gripper motions. The search instead settles on plans whose predicted object motion comes from exploiting errors in the predicted dynamics. Noise-space plans in GPWM move the rod behind the block before pushing and grasp before lifting; these contact sequences come from the learned interaction prior, not from raw search over actions (Figs. 20 and 22). PointWorld's predictions for its chosen Picking plans show the object rising to the target while the gripper stays 3–12 cm away from it (Fig. 23).

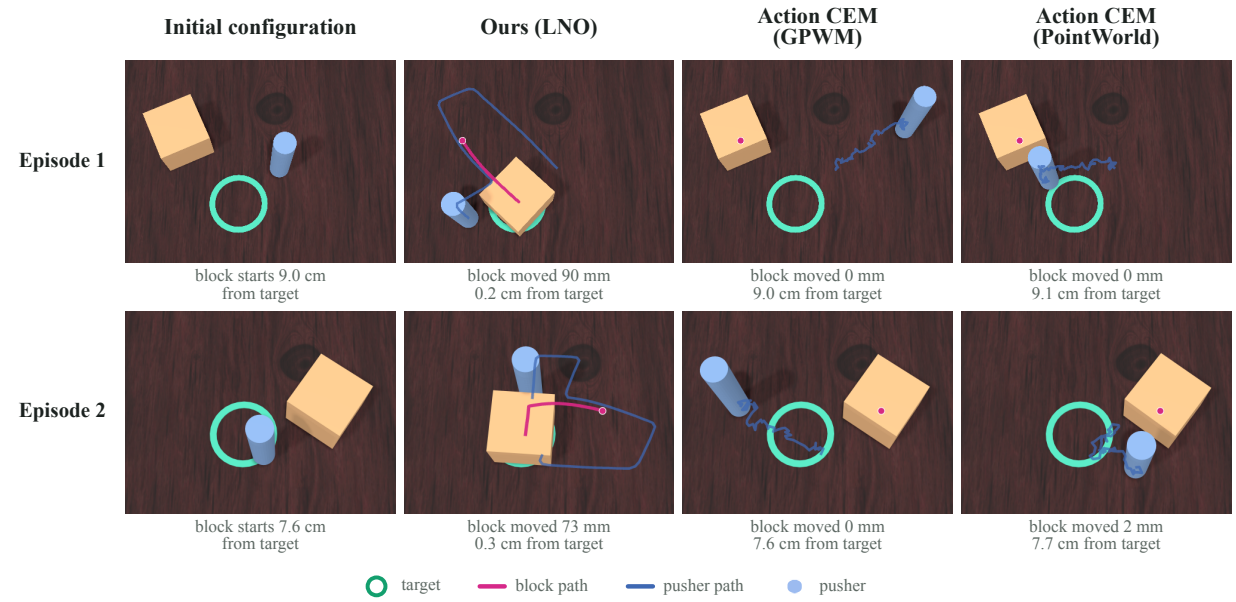


Figure 20 Pushing. Executed plans at the 100 s budget on two scenarios. LNO in GPWM moves the rod around the block and pushes it onto the target. Action CEM, with GPWM or PointWorld, produces jittering rod motions that leave the block where it is.

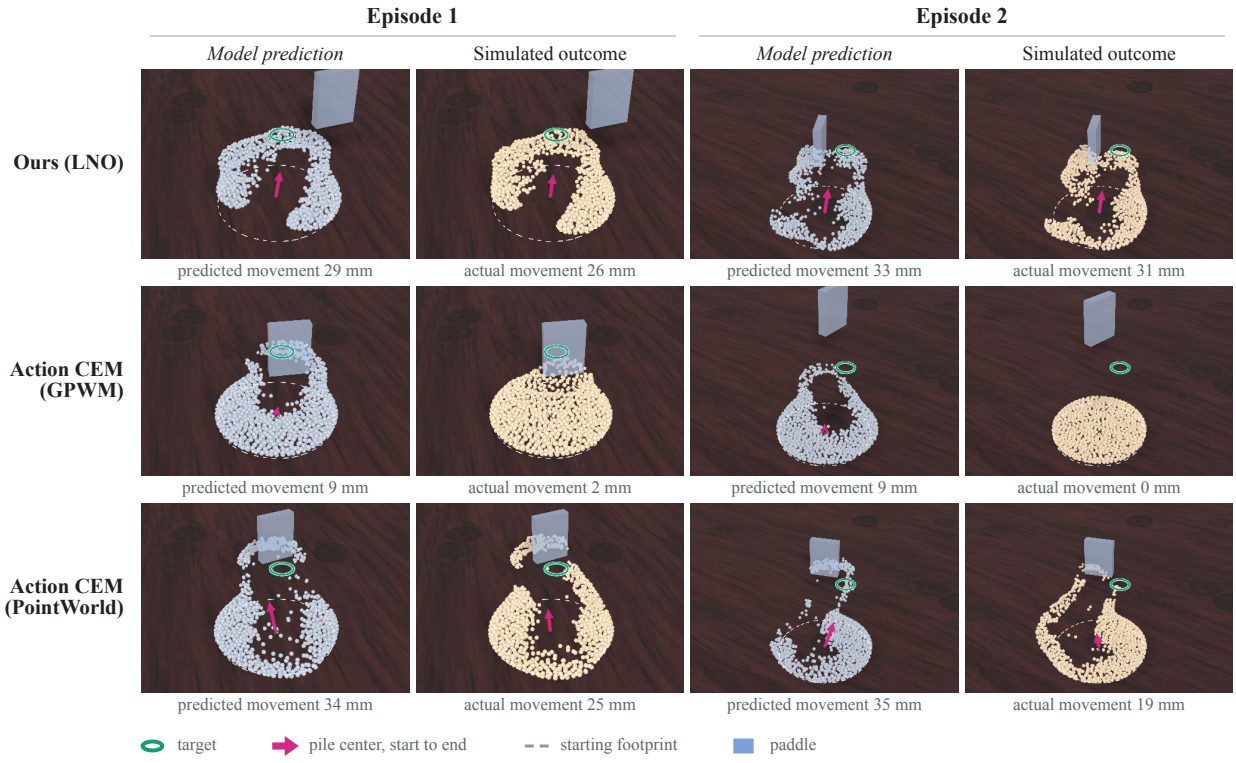


Figure 21 Pile Manipulation. Each planner's predicted outcome and the outcome of executing the same plan in Genesis, at the 200 s budget. LNO's predicted push carries over to the simulator almost unchanged.

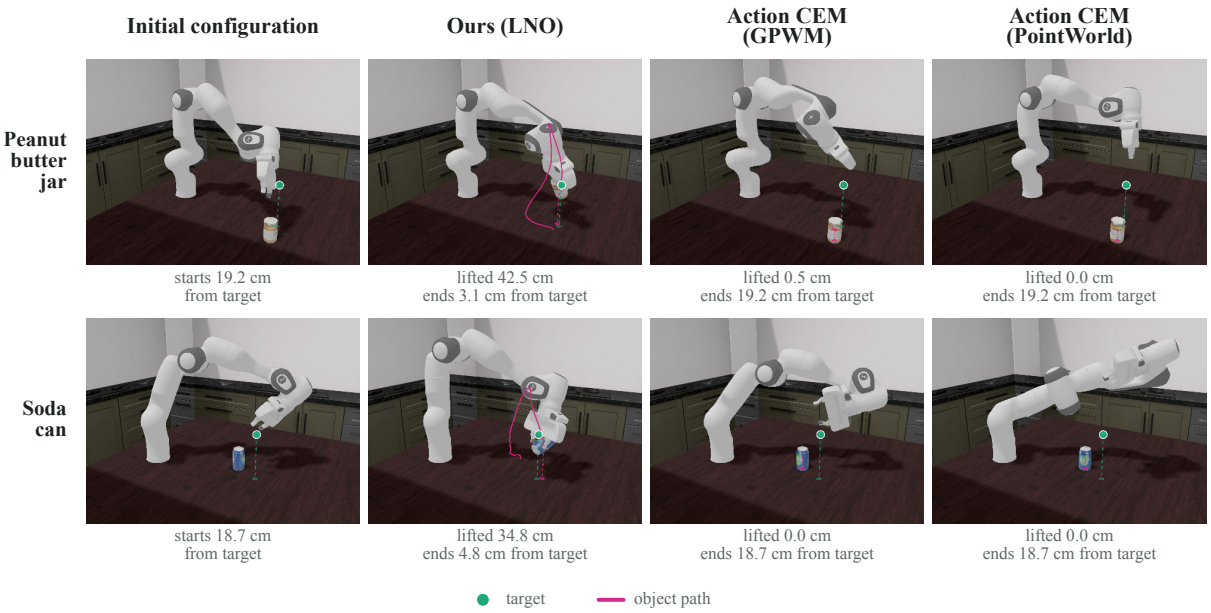


Figure 22 Picking. Executed plans at the 200 s budget for two objects. LNO in GPWM grasps and lifts each object to the target. Neither Action CEM plan establishes a grasp, and the objects are not lifted. The full Franka arm is rendered for visualization only; plans are executed with a floating hand.

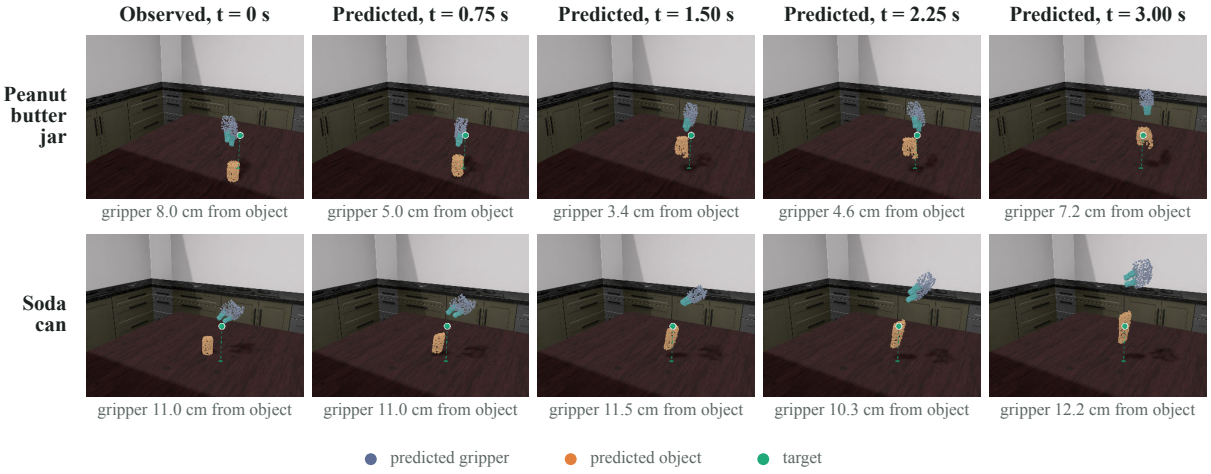


Figure 23 PointWorld's predictions for its Picking plans. The predicted object rises to the target while the predicted gripper stays 3–12 cm away from it; executed in simulation, the same plans leave the object on the table (Fig. 22, right).

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 24 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 25 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 26, 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 27 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.

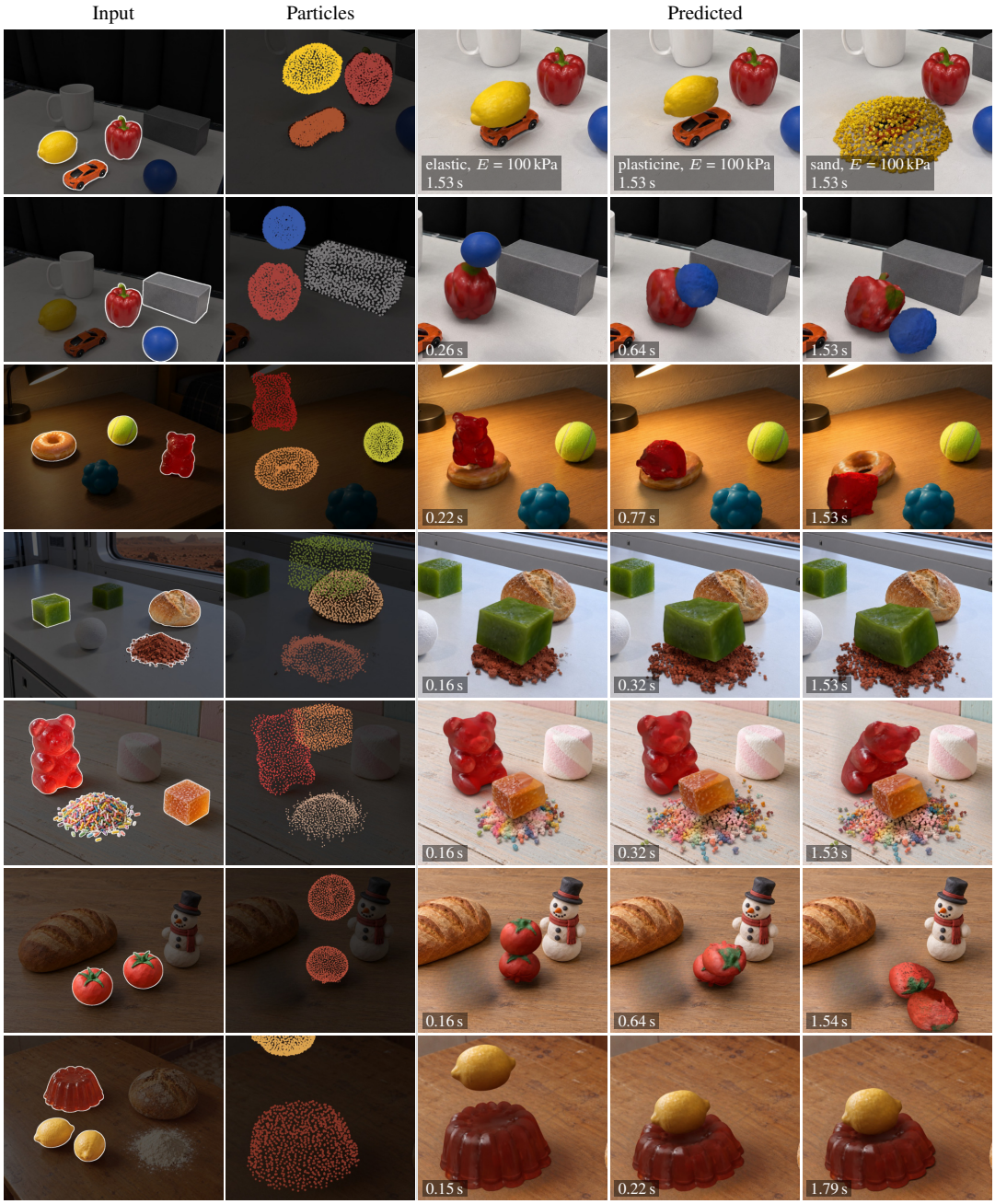


Figure 24 More drops from images. Each row shows the input image with the simulated objects outlined, their particles at the last input frame, and three predicted frames. Row 1, the lemon from the desk photograph falls onto the toy car as an elastic, a plasticine, and a sand object, shown 1.53 s after 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.

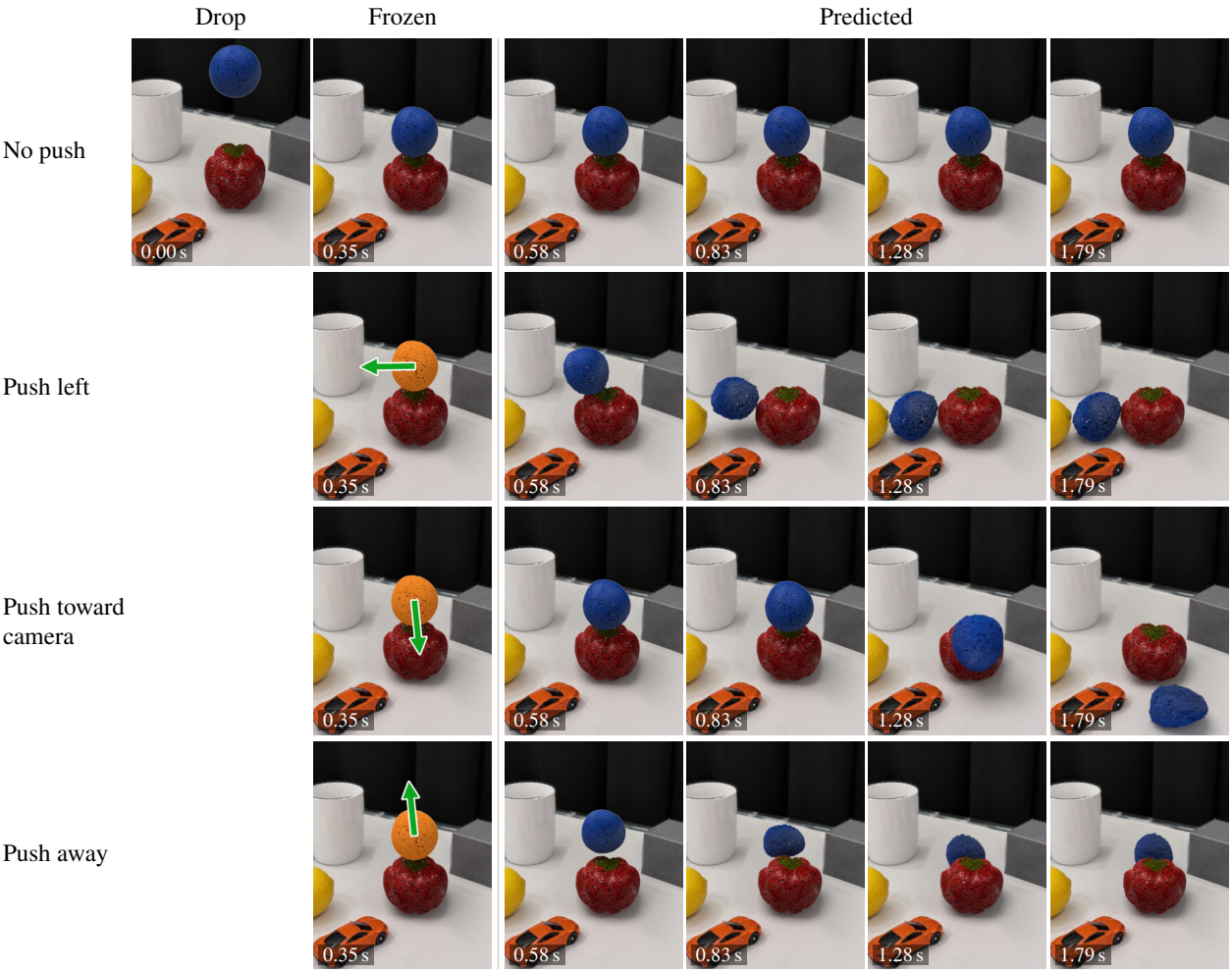


Figure 25 Changing the observed history. Every row continues the same drop of the blue ball onto the pepper. In the top row the ball is released, lands, and rests on top at 0.35 s, when we freeze the scene. Without an edit it stays there. In each row below, only the ball's four input frames at the freeze 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.

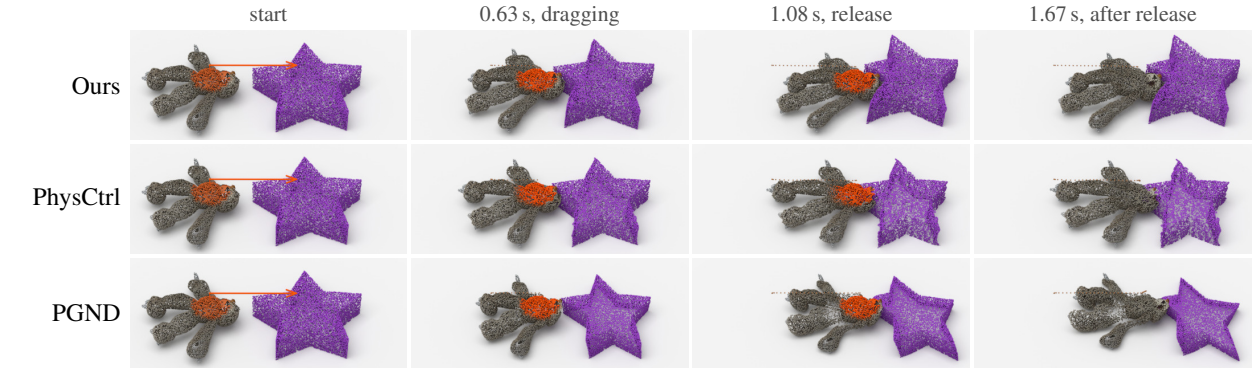


Figure 26 Dragging a sloth into a foam star. A sloth reconstructed from a photograph is dragged into a foam star, and every model gets the same start and drag. The arrow shows the commanded drag. Objects are drawn with dense points that follow each model's predicted particles.

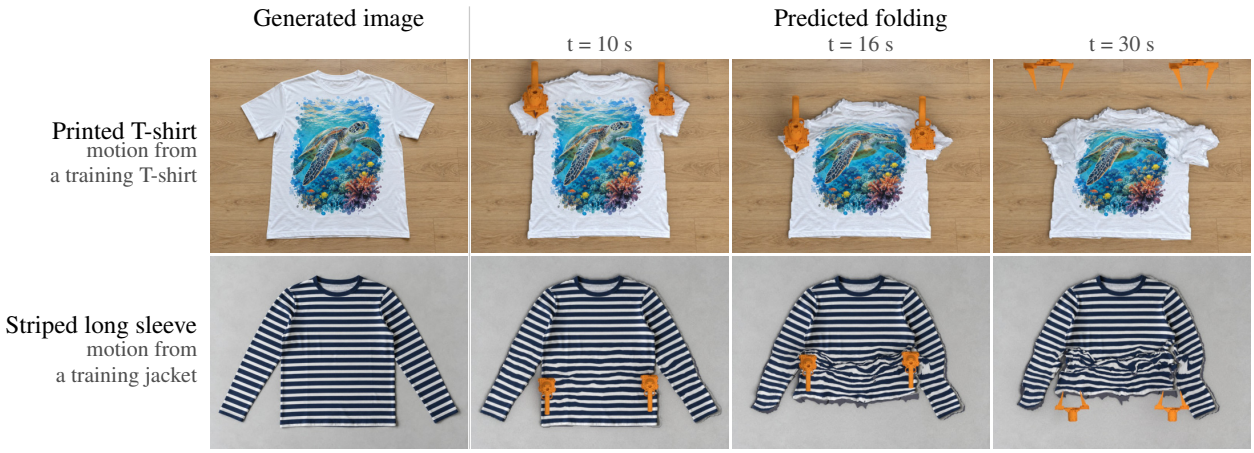


Figure 27 Folding garments from generated images. The grippers (orange) follow the motion recorded for a training garment, and their points are given to the model.