Learning Equivariant Neural-Augmented Object Dynamics from Few Interactions

Anonymous Author(s)
Affiliation
Address
`email`

Abstract: Learning data-efficient object dynamics models for robotic manipulation is challenging, especially for deformable bodies. Popular approaches model objects as 3D graphs and learn particle displacements using graph neural networks; however, they often require thousands of interactions. Even so, these models fail to adhere to real world physics by violating interpenetration constraints and not maintaining object shape over time. We introduce PIEGraph, a neural-augmented dynamics model capable of learning physically-grounded object dynamics for rigid and deformable bodies from few interactions. PIEGraph is a hierarchical framework built using two key layers: (1) a **P**hysically **I**nformed prior implemented as a spring mass system to model physically feasible particle motions over time, and (2) an action-conditioned **E**quivariant **Graph** Neural Network that exploits symmetries in particle motion and guides the physics prior. We demonstrate the ability to learn object dynamics for robotic planning on ropes, cloth, stuffed animals, and rigid bodies using only a few minutes of human interaction data.

Keywords: dynamics, deformable, equivariant

1 Introduction

Humans have an innate ability to reason about the effect of our actions. We understand that pushing a cup of water from the top causes it to topple over, or pushing a rope on the table causes it to deform over time. This reasoning allows us to generate goal-directed behavior remarkably efficiently, and embedding this capability in robots is a promising route to achieving the same results.

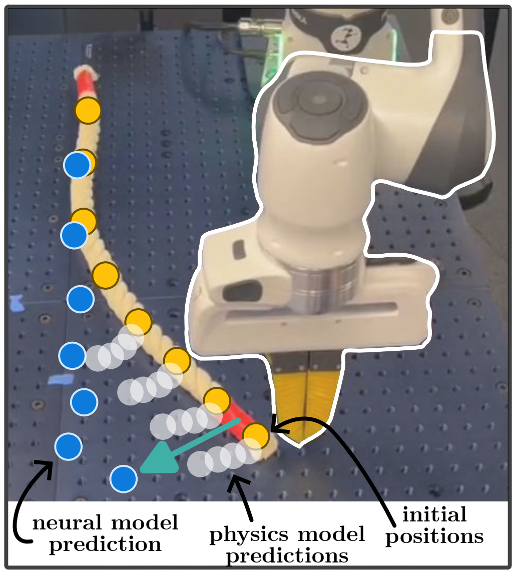


Figure 1: **General Overview.** We guide physics models toward particle-based neural outputs to guarantee physical plausibility and realistic object motion over long horizons.

An action-conditioned dynamics model answers the following question: Given some state of the world, along with some desired interaction, what is the next feasible state? We introduce PIEGraph, a **P**hysically **I**nformed particle dynamics model that utilizes an **E**quivariant **Graph** Neural Network (EGNN) to learn object dynamics. PIEGraph is a neural-augmented dynamics model, meaning that its lowest level uses numerical methods implemented as spring mass systems to reason about particle motion over time. Although these methods maintain physical plausibility like object shape and collisions, they tend to be misaligned since they are not a true representation of the real world; therefore, we guide their particles' motions (as in Figure 1) using a novel action-conditioned equivariant graph neural network trained using only a few minutes of real-world interaction data.

2 Related Works

Particle simulation. A common dynamics modeling pipeline is to capture and represent a scene using simulators then performing system identification on its physical parameters [1, 2, 3, 4, 5, 6]. System identification often requires a complicated multi-step optimization process over sparse differentiable and non-differentiable physical parameters. Our method performs no such optimization, using the simulator only to guarantee physical feasibility constraints, such as object shape, particle connections, collisions with the ground, and gravity.

Additionally, some works [2] focus on constructing "digital twins" using reconstruction methods like Gaussian Splatting [7] and modeling them as particles. The results tend to be visually realistic; however, these works focus more on 3D tracking. Their application to robotics is often left as a secondary downstream application with little quantitative results. Our focus is on physical realism as it pertains to robotic planning, and we find that constructing scenes using initial segmented point clouds and learning the dynamics over that is sufficient for our experiments.

Neural-based simulation. Other works [8, 9, 10, 11] model object dynamics using particle-based learning approaches, predominantly through graph-based networks like PropNet [12]. These works learn deformable object dynamics for ropes [9, 13], cloth [14], granular piles [9], and stuffed animals [8]. A closer inspections reveals that they need thousands of interactions or tens of minutes of data to learn. This is not hard to collect when learning in simulation, but then the sim2real gap must be resolved either through a residual model [11] learned from real data or system identification on a material-adaptive [9] dynamics model. We overcome these shortcomings by learning a model which exploits symmetries in object motion and deformation over time using E(n) Graph Neural Networks [15]. Such an architecture allows us to learn accurate models using only a fraction of data seen in previous works, often requiring only 100 human interactions or about 5 minutes. We also demonstrate cases where roughly a minute of human interaction data is sufficient for dynamics learning in Section 8.1.

3 Neural-Augmented Particle Dynamics

Problem Formulation. We want to learn an object-centric dynamics model

$$\mathbf{X}_{t+\Delta t} = f(\mathbf{X}_t, \mathbf{u}_t; \theta) \tag{1}$$

where t is the current timestep, Δt is the duration of the action, f is the dynamics model, $\mathbf{X}_t$ is the object state, $\mathbf{u}_t$ is the action specified by 3D coordinates of start and end points of contact with the object, and θ is a set of learnable parameters. **PIEGraph**—**P**hysically **I**nformed **E**quivariant **Graph** Neural Network—consists of two layers: (1) a particle-based **physics model** (specifically a spring-mass system), and (2) an action-conditioned **neural model** (an E(n) GNN). The neural model predicts object state directly at the end of an action, providing "global" positions to guide the physics model. The physics model in turn maintains particle-level physical consistency in its predictions, and together they predict object state at the end of an action with high spatial and physical accuracy.

Physics Model. Let ϕ_{SMS} be the function $\hat{\mathbf{X}}^{\mathsf{SMS}}_{t+1} = \phi_{\mathsf{SMS}}(\mathbf{X}^{\mathsf{SMS}}_t, \mathbf{F}_t)$ which takes as input a spring mass system state $\mathbf{X}^{\mathsf{SMS}}_t$ and force $\mathbf{F}_t$ to predict the next state at time $t+1$. We obtain an initial $\mathbf{X}^{\mathsf{SMS}}$ from a fused point cloud of the object at $t = 0$. $\mathbf{F}_t$ is calculated from the following optimization problem:

$$\mathbf{F}^*_t = \arg\min_{\mathbf{F}} \sum_{i}^{N} \left(\mathbf{x}^{\mathsf{setpoint}}_i - \mathbf{x}^{\mathsf{SMS}}_{i,t} \right) \tag{2}$$

where $\mathbf{x}^{\mathsf{SMS}}_{i,t}$ is the position of particle i at time t, and $\mathbf{F} \in \mathbb{R}^{3\times n}$ are forces applied to each particle that minimizes the particle's distance (Figure 2.D and Figure 3.H) to some setpoint $\mathbf{X}^{\mathsf{setpoint}}$

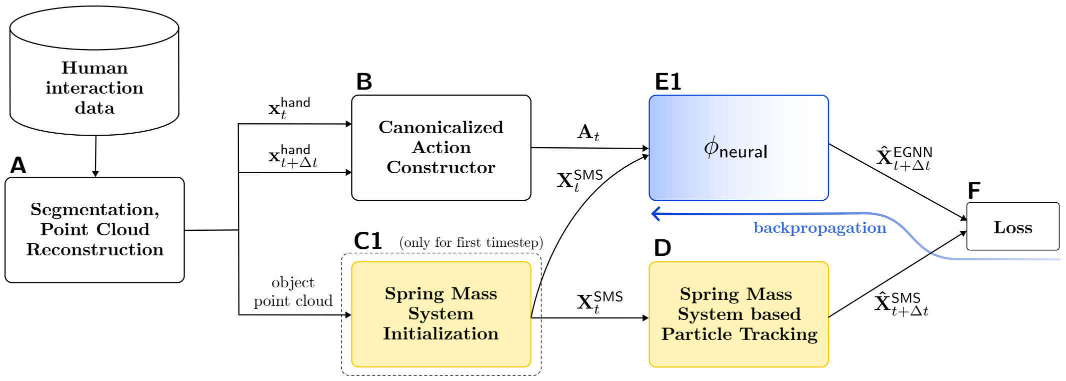

Human interaction data
A
Segmentation, Point Cloud Reconstruction
$\mathbf{x}_t^{\text{hand}}$
$\mathbf{x}_{t+\Delta t}^{\text{hand}}$
object point cloud
B
Canonicalized Action Constructor
C1 (only for first timestep)
Spring Mass System Initialization
$\mathbf{A}_t$
$\mathbf{X}_t^{\text{SMS}}$
E1
ϕ_{neural}
backpropagation
D
Spring Mass System based Particle Tracking
$\hat{\mathbf{X}}_{t+\Delta t}^{\text{EGNN}}$
$\hat{\mathbf{X}}_{t+\Delta t}^{\text{SMS}}$
F
Loss

Figure 2: **Illustrative system diagram—training:** We train an action-conditioned equivariant graph dynamics model (**E1**) from human interaction data captured as an RGBD video. We initialize (**C1**) a spring mass system from an object point cloud at time $t = 0$ and track it (**D**) over multiple actions. We track the human's hands at each time step to construct a representation of the action (**B**).

(described below). These forces are modeled as PID controllers $\mathbf{f}_{i,t} = K_p e_i(t) + K_I \int_0^t e_i(t) + K_D \frac{de_i(t)}{dt}$, where $e_i(t) = (\mathbf{x}_i^{\text{setpoint}} - \mathbf{x}_{i,t}^{\text{SMS}})$, and the controllers are applied for multiple iterations such that $\mathbf{X}^{\text{SMS}}$ converges on the setpoint.

Neural model. Let ϕ_{neural} be the function $\hat{\mathbf{X}}_{t+\Delta t}^{\text{EGNN}} = \phi_{\text{neural}}(\mathbf{X}_t^{\text{SMS}}, \mathbf{A}_t; \theta)$ (Figure 2.E1 and Figure 2.E2) which takes as input a spring mass system state $\mathbf{X}_t^{\text{SMS}}$ (described below), an action $\mathbf{A}_t$, and learnable parameters θ. Concretely, ϕ_{neural} is implemented as an E(n) GNN [15]. Rather than use the raw action $\mathbf{u}_t$, which is the start and end points of an end-effector, we opt for a canonicalization approach to compute $\mathbf{A}_t$, which we show in Section 5 performs better. By decomposing an action $\mathbf{u}_t$ into a start end-effector state $\mathbf{s}_t$ and end end-effector state $\mathbf{e}_t$, we define a transformation invariant action space (Figure 2.B) that is canonicalized to an object using

$$\mathbf{a}_{i,t} = R^{-(\text{atan2}(\mathbf{e}_t - \mathbf{s}_t)+2\pi)}(\mathbf{x}_{i,t} - \mathbf{e}_t), \forall \mathbf{x}_{i,t} \in \mathbf{X}$$

where $\mathbf{a}_{i,t}$ is a transform invariant action applied to particle i. This process is described visually in Figure 6, and a detailed proof of its transformation invariance is in Section 8.3.

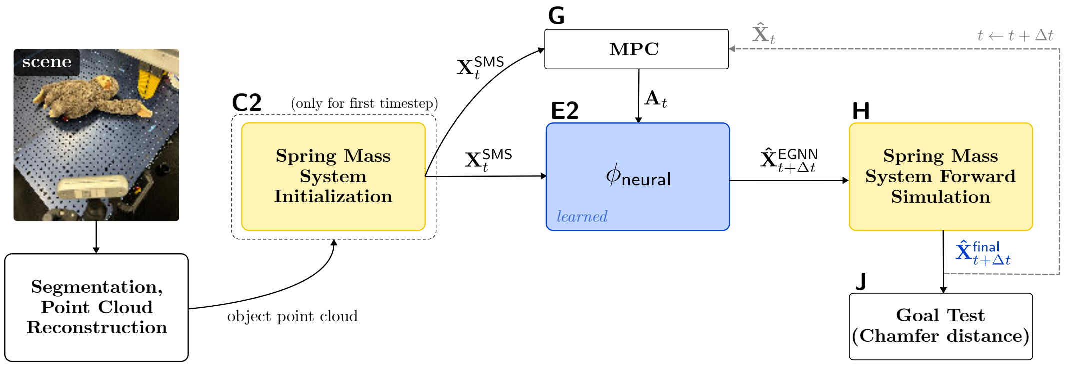

scene
Segmentation, Point Cloud Reconstruction
object point cloud
C2 (only for first timestep)
Spring Mass System Initialization
$\mathbf{X}_t^{\text{SMS}}$
$\mathbf{X}_t^{\text{SMS}}$
G
MPC
$\hat{\mathbf{X}}_t$
$t \leftarrow t + \Delta t$
$\mathbf{A}_t$
E2
ϕ_{neural}
learned
$\hat{\mathbf{X}}_{t+\Delta t}^{\text{EGNN}}$
H
Spring Mass System Forward Simulation
$\hat{\mathbf{X}}_{t+\Delta t}^{\text{final}}$
J
Goal Test (Chamfer distance)

Figure 3: **Illustrative system diagram—planning.** We use a learned equivariant action-conditioned graph dynamics model (**E2**) to guide (**H**) a spring mass system constructed (**C2**) from an initial point cloud of an object at time $t = 0$. This guidance process is used to plan (**G**) for robot actions that reach a specified goal configuration (**J**) implemented as a point cloud.

Tracking and Data Collection. From a video sequence of human-object interactions, we collect 4 RGBD images per frame at different view points for 100 interactions (about 5 minutes of data) at 5Hz. We follow a routine postprocessing pipeline using SAM [16] and MediaPipe Hands [17] to capture object segmentations and hand positions. At time $t = 0$, we construct a spring mass system $\mathbf{X}_0^{\text{SMS}}$ (Figure 2.C1 and Figure 2.C2) from an initial point cloud downsampled to 50 points and track the object's movement by applying external forces to each particle. Here, $\mathbf{x}_i^{\text{setpoint}}$ from Equation 2 is the closest point cloud point to $\mathbf{x}_{i,t}^{\text{SMS}}$ at time t, as seen in Figure 2.D.

Training. The neural function ϕ_{neural} is trained using MSE loss between the predicted particle positions $\hat{\mathbf{X}}^{\text{EGNN}}_{t+\Delta t}$ and tracked particle positions $\hat{\mathbf{X}}^{\text{SMS}}_{t+\Delta t}$ as well as a shape loss (2.F) that helps regularize object shape over time

$$L = ||\hat{\mathbf{X}}^{\text{EGNN}}_{t+\Delta t} - \hat{\mathbf{X}}^{\text{SMS}}_{t+\Delta t}||_2^2 + \sum_{s \in \mathcal{N}(r)} ||(\hat{\mathbf{x}}^{\text{EGNN},r}_{t+\Delta t} - \hat{\mathbf{x}}^{\text{EGNN},s}_{t+\Delta t}) - (\hat{\mathbf{x}}^{\text{SMS},r}_{t+\Delta t} - \hat{\mathbf{x}}^{\text{SMS},s}_{t+\Delta t})||_2^2 \tag{3}$$

where s are the nodes in the neighborhood $\mathcal{N}$ of node r.

Hierarchical Dynamics. ϕ_{neural}, though learned from real world demonstrations, has no physical feasibility guarantees, meaning that the shape of the object is not preserved, nor does it guarantee predictions free from intersecting with the ground. Even by providing shape losses (described in **Training**), the object shape may still not preserved over time, as seen in Section 5; therefore, we use $\hat{\mathbf{X}}^{\text{EGNN}}_{t+\Delta t}$ to guide (Figure 3.H) $\hat{\mathbf{X}}^{\text{SMS}}_{t}$ during planning by setting $\hat{\mathbf{X}}^{\text{EGNN}}_{t+\Delta t} = \mathbf{X}^{\text{setpoint}}$ from Equation 2. By modeling the optimization process this way, we guide the spring mass system to the predicted particle states without breaking the physical constraints imposed by the spring mass system like spring connections, rest lengths, or collisions with the ground. The hierarchical model prediction pipeline can therefore be explained as the following optimization process:

$$\mathbf{X}_{t+\Delta t} = \min_{\mathbf{F}} \sum_{i}^{N} (\hat{\mathbf{x}}^{\text{EGNN}}_{i,t+\Delta t} - \hat{\mathbf{x}}^{\text{SMS}}_{i,t}) \tag{4}$$

Planning. We use the Cross Entropy Method [18] to plan for optimal action sequences. We sample and simulate 1000 concurrent action trajectories and propagate the top 3 performers for 20 iterations based on the distances from the predicted particle states, using our dynamics model, to some desired goal configuration (Figure 3.J), implemented as a fused point cloud.

4 Experiments

All experiments are conducted using a Franka Research 3 with a Finray Gripper mounted on a 35 by 40 inch tabletop. We collect realworld datasets of 100 human-object interactions for 5 total objects, namely Tblock, Stiff Rope, Bendy Rope, Sloth, and Cloth using 4 D455 Realsense cameras.

Baselines. Our neural-augmented dynamics model seamlessly combines the physical feasibility of explicit simulation and the simplicity of data-driven neural-based methods; therefore, we showcase it's combined performance by comparing it to other neural-based, neural-augmented, and simulation-only baselines. **Ours(NG)** is an ablation baseline of our approach, where **NG** stands for No Guidance. It uses an E(n) GNN along with our canonicalized action approach introduced in Section 3, but it does not guide the underlying spring mass system. **SMS(NO)** is another ablation baseline of our approach, where NO stands for Non-Optimized. It uses a spring mass system with non-optimized physical parameters with an impulse-based collision handler to model action trajectories. **EGNN+G** is a baseline that models actions as particles instead of our action space defined in 3. The output of this model is used to guide the underlying spring mass system. **SMS(O)** is a spring mass system with optimized (O) spring stiffness and damping coefficients using first-order gradient descent on a simulator implemented in Warp [19] using the same training data as our method.

5 Results

H	**Tblock**	**Stiff Rope**	**Bendy Rope**	**Sloth**	**Cloth**
1	(4.1, **1.3**)	(5.1, **2.5**)	(5.5, **2.4**)	(10.9, **6.4**)	(7.2, **2.6**)
2	(9.9, **2.8**)	(11.7, **6.0**)	(13.3, **5.5**)	(26.3, **14.5**)	(15.7, **5.4**)
4	(23.2, **5.7**)	(27.1, **12.8**)	(31.9, **13.8**)	(63.9, **33.7**)	(34.6, **15.7**)

Table 1: **Dynamics Results.** We present a custom chamfer distance and shape loss metric (**CD+S**) for our neural model without and with guidance respectively — (Ours(NG), Ours) — for tblock, stiff rope, bendy rope, sloth, and cloth for horizon lengths (**H**) $1, 2, 4$.

Dynamics Results. The prediction performance of our hierarchical dynamics model is given in Table 1 for tblock, stiff rope, bendy rope, sloth, and cloth. Across 20 pushes, we calculate the average value of a custom loss metric, namely CD+S, for multiple dynamics prediction horizons. CD+S is the summation of two losses, namely Chamfer Distance and Shape Loss defined in the second half of Equation 3. We see that even with the additional shape loss in our training regime, Ours(NG) has much worse autoregressive accumulation of errors.

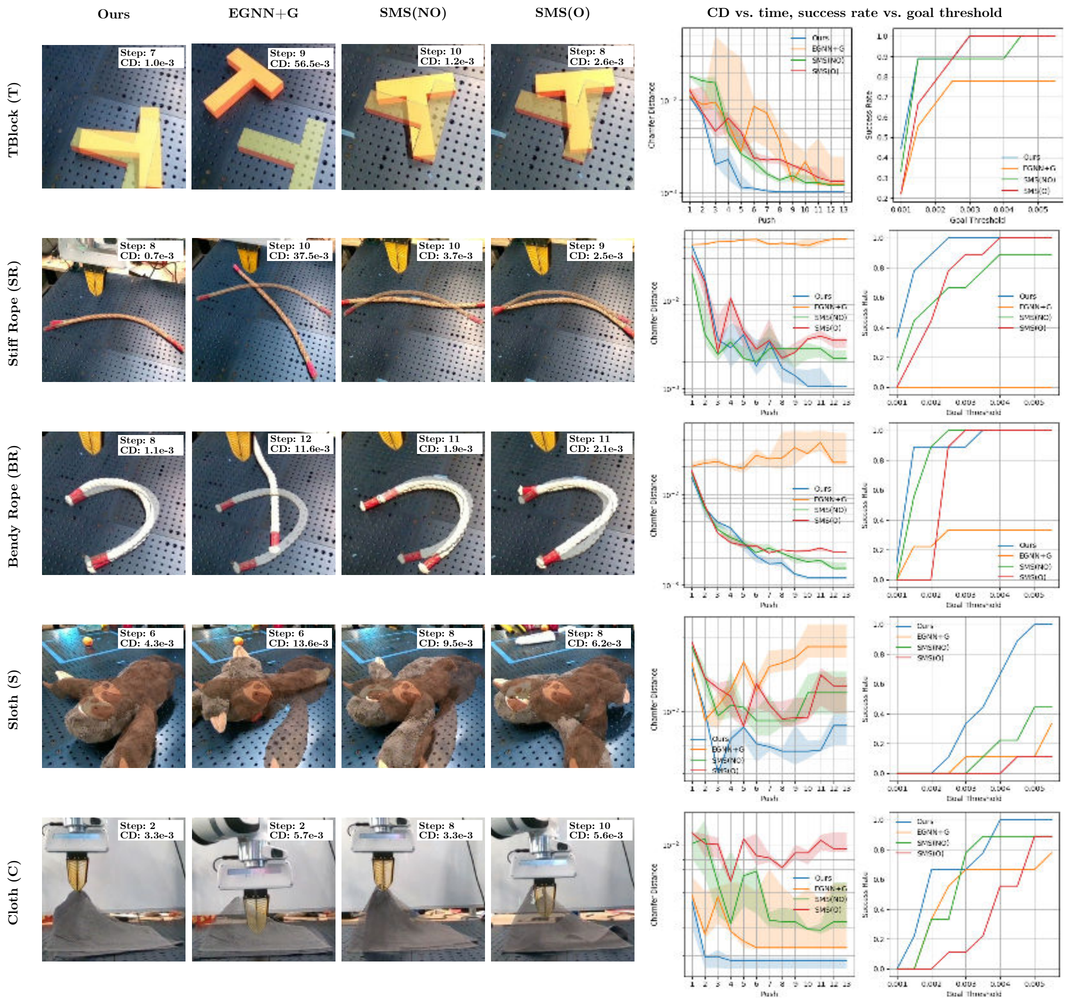


Figure 4: **Robot Planning Results.** For each object, we plan action sequences to reach a goal configuration implemented as a point cloud. We plan actions using MPC for 3 separate goals, and repeat each experiment 3 times. On the left, we hand selected qualitative planning results. On the right, we show quantitative planning results which visualize the chamfer distance over time and task success with varying goal thresholds. We also display the 40th and 60th percentiles as shaded regions to capture variance in performance.

Robotic Manipulation Results. For each object, we construct 3 goal configurations and plan up to 13 interactions. We plan with each model 3 times per goal with a total of 9 runs per model. Across all baselines, our model solves each task with fewer planning steps and much lower costs (Figure 4).

6 Conclusion

PIEGraph is a flexible, data-efficient, and physically grounded dynamics modeling framework. We demonstrate its efficacy to learn dynamics on a wide variety of objects from very little human interaction data, while still being able to minimize auto-regressive accumulation of errors. Our model is the first known method to augment existing particle simulators for general object manipulation using equivariant graph neural networks.

7 Limitations

Although we are able to learn object dynamics from relatively few human interactions, our neural-augmented dynamics model is mostly limited to nonprehensile manipulation tasks on single objects. This is due to the simplicity of our action representation as start and end end-effector positions. In future work, we aim to better model contact forces such that 3-dimensional interaction is better represented.

References

[1] K. M. Jatavallabhula, M. Macklin, F. Golemo, V. Voleti, L. Petrini, M. Weiss, B. Considine, J. Parent-Levesque, K. Xie, K. Erleben, L. Paull, F. Shkurti, D. Nowrouzezahrai, and S. Fidler. gradsim: Differentiable simulation for system identification and visuomotor control. *International Conference on Learning Representations (ICLR)*, 2021.

[2] H. Jiang, H.-Y. Hsu, K. Zhang, H.-N. Yu, S. Wang, and Y. Li. PhysTwin: Physics-informed reconstruction and simulation of deformable objects from videos. *ICCV*, 2025.

[3] F. Liu, E. Su, J. Lu, M. Li, and M. C. Yip. Robotic manipulation of deformable rope-like objects using differentiable compliant position-based dynamics. *IEEE Robotics and Automation Letters*, 8(7):3964–3971, 2023.

[4] M. Liu, G. Yang, S. Luo, and L. Shao. SoftMAC: Differentiable soft body simulation with forecast-based contact model and two-way coupling with articulated rigid bodies and clothes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[5] Y. Shuai, R. Yu, Y. Chen, Z. Jiang, X. Song, N. Wang, J. Zheng, J. Ma, M. Yang, Z. Wang, et al. Pugs: Zero-shot physical understanding with gaussian splatting. *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[6] L. Zhong, H.-X. Yu, J. Wu, and Y. Li. Reconstruction and simulation of elastic objects with spring-mass 3D gaussians. In *European Conference on Computer Vision*, pages 407–423. Springer, 2024.

[7] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[8] M. Zhang, K. Zhang, and Y. Li. Dynamic 3d gaussian tracking for graph-based neural dynamics modeling. In *8th Annual Conference on Robot Learning*, 2024.

[9] K. Zhang, B. Li, K. Hauser, and Y. Li. Adaptigraph: Material-adaptive graph-based neural dynamics for robotic manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[10] K. Zhang, B. Li, K. Hauser, and Y. Li. Particle-grid neural dynamics for learning deformable object models from rgb-d videos. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

[11] C. Wang, Y. Zhang, X. Zhang, Z. Wu, X. Zhu, S. Jin, T. Tang, and M. Tomizuka. Offline-online learning of deformation model for cable manipulation with graph neural networks. *IEEE Robotics and Automation Letters*, 7(2):5544–5551, 2022.

[12] Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake. Propagation networks for model-based control under partial observation. In *ICRA*, 2019.

[13] P. Mitrano, A. LaGrassa, O. Kroemer, and D. Berenson. Focused adaptation of dynamics models for deformable object manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5931–5937, 2023.

[14] Z. Huang, X. Lin, and D. Held. Mesh-based dynamics with occlusion reasoning for cloth manipulation. In *Robotics: Science and Systems (RSS)*, 2022.

[15] V. G. Satorras, E. Hoogeboom, and M. Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

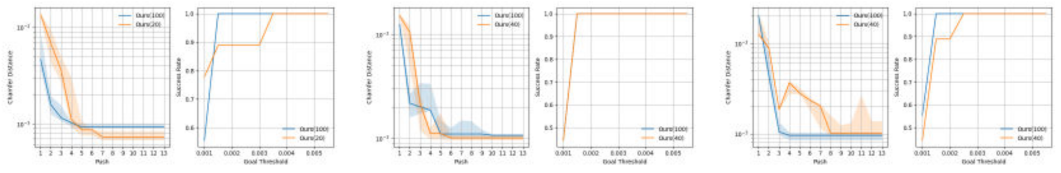
[16] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[17] A. Vakunov, C.-L. Chang, F. Zhang, G. Sung, M. Grundmann, and V. Bazarevsky. Mediapipe hands: On-device real-time hand tracking. In *CV4ARVR*, 2020. https://mixedreality.cs.cornell.edu/workshop.

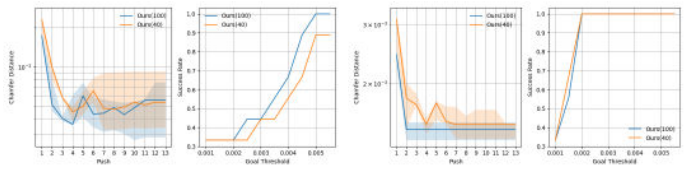
[18] L.-Y. Deng. The cross-entropy method: A unified approach to combinatorial optimization, monte-carlo simulation, and machine learning. *Technometrics*, 48(1):147–148, 2006.

[19] M. Macklin. Warp: A high-performance python framework for gpu simulation and graphics, March 2022. NVIDIA GPU Technology Conference (GTC).

8 Appendix



(a) TBlock(20) (b) Bendy Rope(40) (c) Stiff Rope(40)



(d) Sloth(40) (e) Cloth (40)

Figure 5: **Robot Planning with Different Data Fidelities.** We plan actions using MPC for 2 separate goals, and repeat each experiment 3 times. We compare various versions of our models, namely **Ours(N)** where **N** is the number of human interactions our model is trained on. For each object we show quantitative results which visualize the chamfer distance over time and task success with varying goal thresholds. We display the 40th and 60th percentiles as shaded regions to capture variance in performance.

8.1 Learning With Different Data Fidelities

The experiments described in Section 4 used models trained on 100 interactions, or about 5 minutes of data for each object. Here, we vary the data fidelity. As seen in Figure 5, there is a negligible performance decrease when reducing the interaction data of each object by more than half. In the case of the TBlock, using only 20 interactions, or about a minute of data, is still sufficient for learning our dynamics model for robotic planning.

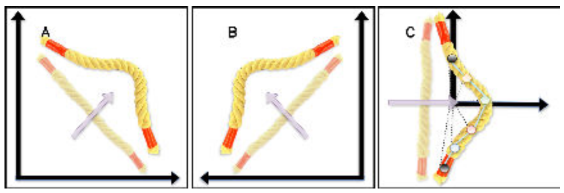


Figure 6: **(a)** We apply a linear push to an object (purple arrow), which results in a u-shaped deformation over time. **(b)** The same linear push can be applied to the object under some world transformation, meaning that actions in (a) and (b) should be invariant to transformations and canonicalizaed to the object. **(c)** We align both scenes (a) and (b) to the x axis, such that the actions occur along the x axis. To enforce object pose sensitivity, we calculate the difference between each object particle to the aligned action end position.

8.2 Simulated Dynamics Results

We compare our model to Propnet [12], a popular modeling architecture for particle dynamics. To cheaply obtain large amounts of interaction data, we train Propnet and our model on a t-block simulated environment implemented in Pymunk, where the tblock is constructed of 8 particles and

H	**T**
1	$(0.0073,\ \mathbf{0.0029})$
10	$(0.0299,\ \mathbf{0.013})$
100	$(0.0868,\ \mathbf{0.0682})$

Table 2: **Simulated Dynamics Results.** We present average particle distance losses for Propnet and our model respectively — (Propnet, Ours) — for a 2D Tblock (**T**) for horizon lengths (**H**) 1, 10, and 100. These results are averaged across a single episode of 500 timesteps.

the end-effector is a single point in image space. Our model is trained on a single episode of 500 timesteps ($\sim$ 50 seconds), while Propnet is trained on 30 episodes of 500 timesteps. As seen in Figure 2, our model accumulates less error over time while using 30 times less the amount of data.

8.3 Invariant Action Space

Let $\mathbf{x}$ be our input state, $\mathbf{s}$ be our initial end-effector position, and $\mathbf{e}$ be our final end-effector position. We need to develop an action that is invariant to translations and rotations such that the following statement is true:

$$f(\mathbf{x}, \mathbf{s}, \mathbf{e}) = f(\mathbf{R}^\theta \mathbf{x} + \mathbf{g}, \mathbf{R}^\theta \mathbf{s} + \mathbf{g}, \mathbf{R}^\theta \mathbf{e} + \mathbf{g}) = \mathbf{a}$$

8.3.1 Proof

Let's define our action like so:

$$\mathbf{a} = \mathbf{R}^{-(atan2(\mathbf{e}-\mathbf{s})+2\pi)}(\mathbf{x} - \mathbf{e})$$

We need to prove the following equivalence

$$\begin{aligned}\mathbf{a} &= \mathbf{R}^{-(atan2(\mathbf{e}-\mathbf{s})+2\pi)}(\mathbf{x} - \mathbf{e}) \\ &= \mathbf{R}^{-(atan2(\mathbf{R}^\theta \mathbf{e}+\mathbf{g}-(\mathbf{R}^\theta \mathbf{s}+\mathbf{g})+2\pi)}(\mathbf{R}^\theta \mathbf{x} + \mathbf{g} - (\mathbf{R}^\theta \mathbf{e} + \mathbf{g}))\end{aligned}$$

We begin by simplifying,

$$\mathbf{a} = \mathbf{R}^{-(atan2(\mathbf{R}^\theta(\mathbf{v}))+2\pi)}(\mathbf{R}^\theta(\mathbf{x} - \mathbf{e}))\textbf{, where } \mathbf{v} = \mathbf{e} - \mathbf{s}$$

We show that $atan2(\mathbf{R}^\theta(\mathbf{v})) = \theta + atan2(\mathbf{v})$ by first converting $\mathbf{v}$ into polar coordinates like so:

$$\mathbf{v} = r. \begin{pmatrix} cos(\phi) \\ sin(\phi) \end{pmatrix}\textbf{, where } \phi = atan2(\mathbf{v})$$

Apply $\mathbf{R}^\theta$,

$$\begin{aligned}\mathbf{R}^\theta \mathbf{v} &= r.\mathbf{R}^\theta \begin{pmatrix} cos(\phi) \\ sin(\phi) \end{pmatrix} \\ &= r \begin{pmatrix} cos(\theta), -sin(\theta) \\ sin(\theta), cos(\theta) \end{pmatrix} \begin{pmatrix} cos(\phi) \\ sin(\phi) \end{pmatrix} \\ &= r \begin{pmatrix} cos(\theta + \phi) \\ sin(\theta + \phi) \end{pmatrix}\end{aligned}$$

So,

$$\mathbf{R}^\theta \mathbf{v} = r. \begin{pmatrix} cos(\theta + atan2(\mathbf{v})) \\ sin(\theta + atan2(\mathbf{v})) \end{pmatrix}$$

Thus,

$$atan2(\mathbf{R}^\theta \mathbf{v}) = \theta + \phi$$

We can now rewrite our action as:

$$\mathbf{a} = \mathbf{R}^{-(\theta+\phi+2\pi)}(\mathbf{R}^{\theta}(\mathbf{x}-\mathbf{e})), \textbf{ where } \phi = atan2(\mathbf{e}-\mathbf{s})$$

and simplify,

$$\mathbf{a} = \mathbf{R}^{-(atan2(\mathbf{e}-\mathbf{s})+2\pi)}(\mathbf{x}-\mathbf{e})$$

■