Learning Physics-Guided Residual Dynamics for Deformable Object Simulation

Shivansh Patel[1] Kaifeng Zhang[2*] Sanjay Pokkali[1*] Svetlana Lazebnik[1] Yunzhu Li[2]

[1]UIUC [2]Columbia University

***Abstract*—Simulating deformable objects is essential for a wide range of robotic manipulation applications, yet accurately predicting their dynamics remains challenging. We propose Physics-Guided Residual Dynamics (PGRD), a hybrid simulation framework that combines the advantages of physics-based and learning-based approaches. Specifically, PGRD combines an optimizable spring–mass simulator as a backbone with a learned neural network that predicts residual corrections to the physics-based predictions. We adopt a velocity-based formulation to ensure stable simulation and a sliding-window transformer architecture to capture temporal dependencies. We show that PGRD produces more accurate results than both purely physics-based and learning-based methods on a set of diverse real-world deformable objects. We further demonstrate the utility of PGRD in two applications: manipulation planning via Model Predictive Control, including a language-conditioned setting with a generated goal image; and interactive simulation via action-conditioned video prediction by 3D Gaussian Splatting. Project page: https://pgrd-robot.github.io/**

I. Introduction

Accurate simulation of deformable objects is a persistent challenge in robotics and computer vision. Such objects can change shape significantly under external forces, undergoing stretching, bending, crumpling, and twisting. Simulation difficulty stems from material properties, such as heterogeneous elasticity and damping, but is also complicated by factors, such as self-collisions and friction, that come into play during contact-rich interactions.

The literature contains two dominant paradigms for deformable object simulation: physics-based methods [4, 73, 57] and learning-based methods [87, 10, 38] (see Section II for a more detailed survey of related works). Physics-based methods use equations to describe how materials deform and respond to forces, yielding physically plausible and interpretable simulations. However, these methods require precise material parameters that are difficult to obtain in practice [46, 89, 58]. The complex mathematics of these simulators typically restricts practitioners to gradient-free optimization [46], which can tune only a handful of parameters before becoming intractable [89, 98]. Even with optimal parameters, physics-based models are inherently limited by discretization (coarse meshes miss fine-scale deformations [74, 52]) and modeling assumptions (linear elasticity fails under large strains [80, 53]). Learning-based methods sidestep these limitations by fitting observations that are challenging to model analytically. However, purely data-driven models are data-hungry [91] and suffer from poor generalization to unseen scenarios [13, 76]. They often lack the physical consistency required for real-world deployment [13] and, without the inductive bias provided by physical priors, they may learn spurious correlations [79].

To combine the interpretability and generalizability of physics-based models with the flexibility of learning-based models, we propose a hybrid simulation framework of Physics-Guided Residual Dynamics (PGRD). As shown in Fig. 1 and explained in Section III, PGRD uses a spring-mass simulator as its physics backbone and a neural network to predict residual corrections to that model. PGRD follows a two-stage training procedure: first, we optimize the simulator's parameters to match real-world observations; and second, we train the neural network to compensate for the discrepancies between the physics model and the data. Note that making this two-stage approach work is nontrivial, as directly injecting learned position corrections into simulated states can destabilize the dynamics and cause error accumulation over time. Motivated by impulse-based simulation that evolves states through velocity updates rather than position [54, 72, 72], we predict residual velocities and integrate them forward in time, yielding position corrections that respect the underlying dynamics. This velocity-based formulation enables smooth, stable training and simulation. To model temporal dependencies, we incorporate a sliding-window transformer that refines velocity corrections across multiple timesteps. This temporal history enables the network to capture dynamic phenomena such as momentum,

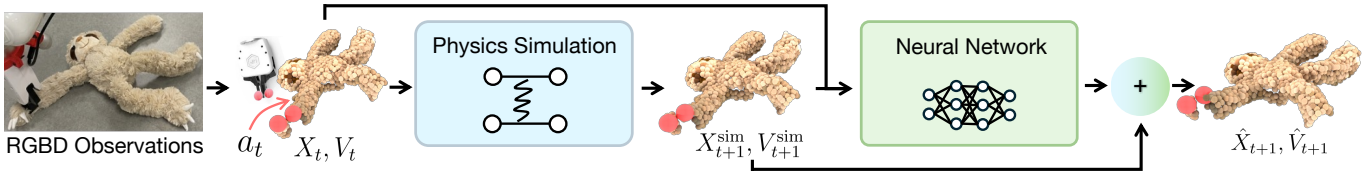

Fig. 1: **Physics-Guided Residual Dynamics Overview.** Given RGBD observations, we extract surface points to instantiate the simulation. Red dots indicate the point where gripper held the object. The framework rolls out an optimized spring-mass physics backbone from the current object state X_t, V_t and robot actions a_t to produce simulation prediction $X^{\text{sim}}_{t+1}, V^{\text{sim}}_{t+1}$. Subsequently, our residual dynamics network predicts per particle residual velocities, which are added to the simulator velocities and time integrated to obtain the final positions $\hat{X}_{t+1}, \hat{V}_{t+1}$.

while a gating mechanism ensures stable training.

Compared to purely physics-based or learning-based simulation methods, PGRD offers several advantages. First, the physics backbone provides priors that can improve generalization to new scenarios. Second, it is expected to require less training data than purely learning-based approaches, as the network only needs to predict corrections rather than the entire dynamics from scratch. Third, it maintains computational efficiency while achieving higher accuracy than physics-based methods alone. To further reduce compounding errors over long horizons, we train PGRD with multi-step rollouts: during training, the model is recursively applied to its own predicted states rather than being reset to ground truth at every step. This exposes the residual model to the same distribution shift encountered at test time and is important for stable long-horizon simulation.

In Section IV, we validate PGRD on a diverse set of real-world deformable objects, including rope, paper, soft toys, flag, and duster. Across these objects, we evaluate simulation quality against tracked 3D point trajectories from real-world executions, covering both prehensile and non-prehensile manipulation. Our experiments demonstrate that PGRD consistently outperforms both purely physics-based and purely learning-based approaches in tracking accuracy. Notably, only PGRD performs well on the duster, which is a heterogeneous object with a rigid stem and soft feathers. Here, uniform material assumptions in most physics-based deformable simulators fail and learning-based methods struggle with the complexity. Besides standard tracking evaluation, we also show the advantage of PGRD for action-conditioned video prediction, which we perform by attaching 3D Gaussians to the simulated particles.

Beyond evaluation, we demonstrate two applications of PGRD in Section V. First, PGRD can serve as a forward model for *manipulation planning* via Model Predictive Path Integral (MPPI) control, where candidate action sequences are rolled out and ranked by Chamfer Distance to a target configuration. We demonstrate this on challenging tasks such as cable rerouting through a narrow slot, where PGRD succeeds in 8 out of 10 trials compared to 2 out of 10 for the tuned spring-mass baseline. We further extend planning to a language-conditioned setting, where a goal image generated from a language command replaces the need for a pre-collected target point cloud. Second, PGRD enables *interactive photorealistic simulation* in which users issue manipulation commands and the predicted particle states drive 3D Gaussians to render photorealistic views of the resulting deformation.

In summary, our contributions are as follows. (1) We propose Physics-Guided Residual Dynamics, a hybrid simulation framework that combines an optimizable spring-mass model with learned residual corrections to accurately model deformable object dynamics. (2) We introduce a two-stage training procedure that first optimizes physics parameters using black-box optimization and then trains a neural network to predict residual corrections. (3) We conduct extensive experiments on diverse real-world deformable objects, demonstrating that our approach achieves superior performance compared to existing physics-based and learning-based methods.

II. Related Works

Physics-Based Simulation for Deformable Objects. Traditional physics-based simulation methods rely on analytical models to discretize deformable objects and numerical solvers for the equations of motion of the models. For deformable objects, spring-mass models [4, 45, 41] represent one of the most intuitive and efficient approaches, where objects are modeled as networks of point masses interconnected by springs. Consequently, this representation has been extensively employed to model the dynamics of diverse deformable objects in both computer graphics and robotics [4, 45, 41, 96, 46, 95, 34]. Our work leverages the spring-mass model as a backbone precisely because of its efficiency and interpretability. While more complex approaches like Finite Element Methods (FEM) [71, 56], Position-Based Dynamics (PBD) [57, 51], and Material Point Methods (MPM) [73, 33] offer higher physical fidelity, they incur high computational costs and complexity. We include MPM as a baseline in our experiments to validate this tradeoff.

Despite their physical foundations, the above-mentioned methods require precise material parameters that are difficult to obtain in practice. A recent line of work addresses this challenge by reformulating physics simulators to be compatible with automatic differentiation [41, 75, 28, 9, 15, 21, 26, 32, 63, 66, 48, 44, 19, 34]. These methods identify optimal physics parameters by backpropagating through the simulation process, solving inverse problems to find material properties that best match observed data. While such parameter identification can improve simulation accuracy, it imposes a strong requirement the simulator must be fully differentiable. This is often impractical in contact-rich robotics tasks where discontinuities from collisions [8, 42, 96], friction mode transitions [42, 36, 24], and non-smooth contact geometry [81, 90, 42] lead to exploding or vanishing gradients. In practice, even state-of-the-art differentiable simulation methods suffer from these fundamental limitations. We compare against PhysTwin [34], a recent state-of-the-art differentiable spring-mass approach, across all of our tasks, and show that it additionally struggles on heterogeneous deformable objects. Our method does not require the simulator to be differentiable, enabling broader applicability to realistic manipulation scenarios.

Learning-Based Simulation for Deformable Objects. Learning has emerged as a powerful alternative to traditional physics-based simulation, particularly excelling in scenarios requiring real-time performance with complex nonlinear dynamics [87, 10, 38, 97, 86, 6, 1, 29, 12, 17, 49, 84, 85, 47, 30]. These approaches leverage data-driven models, typically neural networks, to learn deformation patterns directly from observation data, bypassing the need for explicit material parameters or complex numerical solvers [39, 55].

Graph Neural Networks (GNNs) have become prominent in this domain due to their natural ability to represent meshes as graphs. GNNs excel at capturing long-range interactions and complex dependencies through message passing [92, 94,

67, 61, 39, 77, 40], allowing them to effectively simulate internal and external forces causing deformation [5, 37]. Notably, GBND [94] establishes a topology over sparse particles, enabling the GNN to efficiently learn and propagate the dynamics across the object. We compare against GBND in our experiments. However, GNN-based methods face significant challenges: the effectiveness of message passing is sensitive to graph structure and is vulnerable to partial observations, while insufficient steps fail to capture global information and excessive steps cause oversmoothing [78, 3].

Recent learning-based methods have explored architectural improvements, including transformers for handling dense particles [68, 82], recurrent neural networks for temporal consistency [50], and attention mechanisms for improved long-range dependency modeling [67]. These methods have demonstrated success across diverse materials, including cloth [41, 43, 61], fluids [67, 39], plasticine [31, 70, 69], and granular materials [39]. A recent state-of-the-art method is Particle-Grid Neural Dynamics [93] (PGND), which combines particle representations with spatial grids to learn dynamics while maintaining spatial continuity, and serves as a strong baseline in our experiments. On the down side, approaches like PGND require substantial training data and may struggle with generalization to unseen scenarios or material properties that are significantly different from the training distributions. Our work addresses these limitations by grounding the neural network in physical priors, allowing it to focus on learning only residual corrections rather than the full dynamics from scratch.

Residual Dynamics. The idea of residual learning has been previously employed to model the dynamics of robots [22, 23, 7, 14]. In these works, the robot's model is known, and the residuals primarily reflect simple modeling errors, such as PID gains or backlash. These discrepancies are relatively easy to capture because they often manifest as consistent deviations from the nominal model. In contrast, we model the dynamics of deformable objects, where the residual must compensate for complex, high-dimensional phenomena, including nonlinear stiffness, spatially varying material properties, and contact-rich interactions. The most closely related work in spirit is [2], which models the environment's residual dynamics rather than the robot's. However, it is limited to low-dimensional toy problems such as planar pushing, and its simple architecture does not scale to the high-dimensional particle cloud representations required for deformable object manipulation, making a direct comparison infeasible. To the best of our knowledge, PGRD is the first work to extend the residual paradigm to modeling high-dimensional, complex states of deformable objects.

III. Method

This section presents the details of our Physics-Guided Residual Dynamics method, which is illustrated in Fig. 1. We first explain our physics backbone, followed by the residual dynamics framework.

A. Physics Backbone

Given RGBD observations, we extract surface points to instantiate the simulation. We represent deformable objects using a spring-mass model, wherein the object is discretized into a graph structure consisting of point masses (nodes) interconnected by springs (edges). Each node i has a position $\mathbf{x}_i \in \mathbb{R}^3$ and velocity $\mathbf{v}_i \in \mathbb{R}^3$ that evolve over time according to Newtonian mechanics with a semi-implicit Euler solver. This representation provides computational efficiency and straightforward implementation while capturing essential elastic deformation behaviors. To model manipulation, we apply constraints to particles near the gripper: for prehensile manipulation, they move rigidly with the gripper, whereas for non-prehensile manipulation, the gripper acts as a spherical collision shape that pushes penetrating particles to its surface.

The fidelity of the spring-mass simulator depends critically on five physical parameters collectively denoted as θ: 1) *Stiffness*, which defines the elastic resistance of springs; 2) *Damping*, which controls energy dissipation during motion; 3) *Threshold*, which determines the maximum distance for creating springs between nodes; 4) *Max springs per node*, which limits the connectivity of each mass point for simulation stability; and 5) *Ground friction*, which governs contact interactions with the environment. These parameters collectively encode the material properties and environmental conditions that govern the dynamics of the simulated object.

Given a batch of trajectories sampled from the dataset, each with initial state $(\mathbf{X}_0, \mathbf{V}_0)$ and action sequences $\mathcal{A} = \{a_t\}_{t=0}^{T-1}$, we roll out the simulator for T time steps with actions $\{a\}_{t=0}^{T-1}$ under candidate parameters θ to obtain predicted point cloud positions $\{\hat{\mathbf{X}}_t\}_{t=1}^{T}$. At each time step t, we compute a pointwise mean squared error between predicted and ground truth configurations. We average this error over the trajectory horizon to obtain the optimization objective. We minimize this objective using Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [25], which is particularly effective for our problem as it optimizes parameters without requiring gradients. We use CMA-ES because contacts and collisions cause sudden changes in forces, preventing reliable gradient computation in the simulator.

For volumetric objects such as soft toys, modeling only the surface shell often leads to unrealistic collapse, as the lack of internal support points causes the object to flatten under gravity or compression. To address this, we augment the spring-mass model with internal points that provide structural volume. Starting from our multi-view observations, we first reconstruct the complete surface point cloud using RaySt3R [16] and extract a watertight mesh via marching cubes. We then uniformly sample particles within this mesh to populate the interior, ensuring the spring-mass model preserves the object's volume and structural integrity during simulation.

B. Residual Dynamics Framework

While our optimized spring-mass backbone provides a physically grounded baseline, it inherently struggles to capture complex behaviors such as nonlinear stiffness, non-uniform

contact friction, and heterogeneous material properties. Instead of attempting to model these intricacies analytically, we introduce a learned residual module to predict the discrepancy between the physics model and observations.

Formally, at each time step t and step size dt, we use the physics backbone to generate an immediate prediction $(X_{t+1}^{\text{sim}}, V_{t+1}^{\text{sim}})$. We then employ a neural network to predict a per-particle residual velocity $\Delta V_{t+1}^{\text{residual}}$ based on the current simulator state and history. An alternative design would be to predict residual positions directly. However, we found this formulation to be unstable in practice: directly correcting positions led to occasional simulation blow-ups, where object state becomes NaNs. Intuitively, directly predicting positions can create a mismatch between the corrected positions and the velocities, destabilizing the simulation. We therefore adopt a velocity-based residual formulation, which ensures smooth integration and avoids the instability often associated with direct position corrections. The final velocities are obtained by adding the learned residuals to the simulator's predictions:

$$\hat{V}_{t+1} = V_{t+1}^{\text{sim}} + \Delta V_{t+1}^{\text{residual}}.$$

These corrected velocities are then time-integrated to obtain the final position updates:

$$\hat{X}_{t+1} = X_{t+1}^{\text{sim}} + \Delta V_{t+1}^{\text{residual}} \cdot dt.$$

For particles that are rigidly held by grippers, we enforce a boundary condition by zeroing out their residuals. This formulation effectively bridges the gap between the optimized spring-mass model and the observed real-world dynamics.

We train the residual network with a supervised objective that minimizes the mean squared error between predicted and ground truth particle positions over a rollout:

$$\mathcal{L}_{\text{res}} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\left\|\hat{X}_t - X_t^*\right\|_2^2, \tag{1}$$

where $\hat{X}_t$ denotes the predicted particle positions at rollout step t, X_t^* denotes the corresponding ground truth positions, and N is the number of particles. To improve robustness to error accumulation over long horizons, we apply this loss over multi-step rollouts rather than a single step. During training, we do not reset the simulator to the ground truth state at every step. Instead, we feed the hybrid simulator's predicted state at time t back as the input for time $t+1$. This exposes the network to its own past predictions and allows it to learn how to recover from drift.

C. Network Architecture

To parameterize the residual velocity, we design a network consisting of three primary components: encoder, decoder, and temporal aggregator. The network architecture is shown in Fig. 2. While individual components draw on established design choices, our contribution lies in their integration into a single residual-dynamics architecture. We ablate the architecture components in App. A and provide the complete hyperparameters for the network components in App. B.

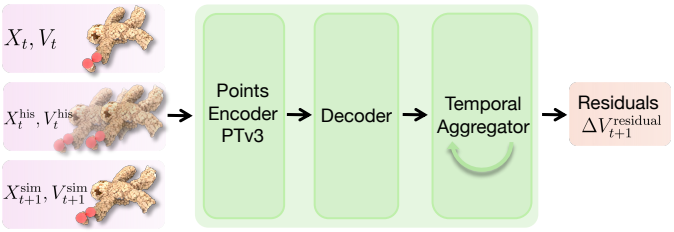

Fig. 2: **Network Architecture.** The points encoder extracts spatiotemporal features, which the decoder projects to estimate initial residual velocity. Finally, the temporal aggregator refines these estimates using a recurrent history to output the final residuals $\Delta V_{t+1}^{\text{residual}}$.

Encoder. We employ a Point Transformer V3 (PTv3) [83] architecture to extract per-particle features. At time step t, we construct input features by concatenating the current state (X_t, V_t), history from previous time steps $(X_t^{\text{his}}, V_t^{\text{his}})$, and the simulator's immediate predictions $(X_{t+1}^{\text{sim}}, V_{t+1}^{\text{sim}})$. The encoder processes these inputs through multiple layers of patch-based self-attention, where each particle attends to others within its local neighborhood. We demonstrate experimentally that the attention mechanism computes position-dependent features for each particle based on its local geometric configuration.

Decoder. To decode per-particle residuals, we utilize a Neural Radiance Field (NeRF) style architecture. We apply Fourier positional encoding to the particle positions and concatenate the resulting embeddings with the features extracted by the encoder. These concatenated features are passed through an MLP decoder to produce an initial velocity correction estimate $\Delta V_{t+1}^{\text{initial}}$. Empirically, we found that including the decoder improves performance compared with directly using PTv3 outputs as velocity estimates, as discussed in the decoder ablation in App. A-B.

Temporal Aggregator. To incorporate dynamics from the previous predictions, we refine the base predictions using a sliding-window transformer. At each step t, we update a temporal buffer with the current initial correction $\Delta V_{t+1}^{\text{initial}}$. If the buffer contains fewer than W frames, we pad it by replicating the most recent frame. The sequence is projected to a latent embedding, augmented with sinusoidal positional encodings, and processed by the transformer. We extract the output feature vector $\mathbf{h}_t$ corresponding to the final time step and apply two parallel learned projections:

$$\boldsymbol{\delta}_t = \tanh(\text{Proj}_\delta(\mathbf{h}_t)), \quad g_t = \sigma(\text{Proj}_g(\mathbf{h}_t)),$$

where Proj denotes a linear layer, $\tanh$ produces a bounded temporal offset $\boldsymbol{\delta}_t$, and the sigmoid function σ generates gating weights g_t. The final residual velocity is computed as:

$$\Delta V_{t+1}^{\text{residual}} = 0.1 \cdot [(1 - g_t) \odot \Delta V_{t+1}^{\text{initial}} + g_t \odot \boldsymbol{\delta}_t],$$

where $\odot$ denotes element-wise multiplication. This gating mechanism allows the network to adaptively blend the local base prediction with the temporally refined correction based on the transformer's context. This design is similar in spirit to gated recurrent models [27, 11], in that it uses temporal context to decide how much of the current prediction to preserve and

how much to replace with a history-based update.

IV. Experiments

A. Data Collection

We evaluate PGRD on six objects shown in Fig. 3 spanning a range of shapes, material properties, and manipulation types.

Fig. 3: **Experimental Objects** (see text).

a) Rope. A mostly one-dimensional object grasped at one end and manipulated by dragging across the table or lifting and lowering motions.

b) Paper. A thin two-dimensional sheet grasped at the top and manipulated by suspending it in the air while performing waving motions, exhibiting bending and twisting.

c) Flag. A cloth flag, demonstrating large area deformation dynamics that are challenging to model due to the waving motion. We chose it because its top stick prevents the cloth's surface from collapsing. We model the stick as a rigid constraint, assuming that all points along the stick follow the gripper trajectory.

d) Sloth Toy. A three-dimensional volumetric object with long, highly flexible limbs that are challenging to model.

e) Duster. A heterogeneous object with a rigid stem and highly deformable feathers. This is especially challenging because it requires different dynamics predictions for the rigid stem compared to the flexible feathers.

f) Teddy Toy. A volumetric object with higher rigidity than the soft toy. We deform it by poking rather than holding it, representing non-prehensile manipulation.

We collect data using UFACTORY xArm 7 following the procedure from [93]. Four Intel RealSense D455 cameras capture synchronized multi-view observations while the object is manipulated. We segment the object with Grounded SAM 2 [65] and track image points with CoTracker [35]. These 2D tracks are lifted to 3D using depth from all camera views, and an iterative rollout procedure is used to extract persistent correspondences across frames. The resulting dataset consists of temporally consistent 3D point cloud trajectories of the object together with the robot gripper positions over time, obtained from robot proprioception and used as the action sequence. Additional details are provided in App. D.

For training, we use 100 episodes, and for validation, we use 20 episodes. Each episode spans 4 seconds, but consecutive episodes are generated with a temporal stride of 2 seconds, so neighboring episodes overlap by 2 seconds. As a result, the 100 training episodes correspond to 202 seconds of unique data, or approximately 3.37 minutes, while the 20 validation episodes correspond to 42 seconds, or 0.70 minutes. In total, the dataset contains 120 episodes extracted from 244 seconds of interaction, which is approximately 4.07 minutes of recorded data. The entire data processing is offline and happens before training. We train PGRD with 5-step rollout, but evaluate it over the full evaluation episode length of 37 steps. Hence, this evaluation paradigm tests generalization capability over horizons not seen during training.

B. Evaluation Tasks and Metrics

We evaluate PGRD in two settings. First, we measure how accurately it predicts object motion in 3D using tracking metrics. Second, we evaluate whether the same predicted trajectories produce visually accurate future observations in image space. In both settings, performance is measured over complete validation episode rollouts rather than isolated timesteps. For the visual-space evaluation, we convert the predicted 3D trajectories into 2D video frames using 3D Gaussian Splatting. We initialize a 3DGS representation from the first frame and then update the Gaussians using the predicted particle motion while keeping appearance parameters fixed. This yields action-conditioned video predictions that can be compared directly against ground-truth frames using visual metrics.

Tracking Metrics. We assess rollout accuracy in 3D using three metrics:

- **Mean Distance Error (MDE)** measures the average distance (in cm) between corresponding points in predicted and ground truth configurations, providing a direct assessment of positional accuracy.
- **Chamfer Distance (CD)** computes the bidirectional nearest neighbor distance (in cm) between point clouds, capturing coverage and precision without requiring correspondences.
- **Earth Mover's Distance (EMD)** quantifies the minimum cost (in cm) of transforming one point cloud into another, providing a holistic measure of distributional similarity.

Visual Metrics. To evaluate action-conditioned video prediction, we render future frames. We then compare these rendered frames against the ground truth using three metrics:

- **J-Score (IoU)** measures intersection over union of predicted and ground truth masks, quantifying spatial overlap.
- **F Score** evaluates contour accuracy, determining how well the boundary of the predicted object mask matches the boundary of the ground truth.
- **Learned Perceptual Image Patch Similarity (LPIPS)** assesses perceptual similarity using features extracted from predicted and ground truth images, capturing visual differences closer to human perception.

C. Baselines

We compare PGRD against five baselines: two analytical physics-based simulations and two learning-based approaches. **Spring-Mass Model [34]:** This is the same as the physics backbone we use for our model. Improvements over this baseline demonstrate that the learned residual model effectively captures the dynamics missed by the physics simulator alone. **Material Point Method (MPM) [73] :** MPM combines Eulerian and Lagrangian representations to handle material behaviors and topological changes. This hybrid approach discretizes

Method	Metric	Rope	Paper	Flag	Sloth Toy	Duster	Teddy Toy
Spring-Mass [34]	MDE ↓	$4.4_{\pm 2.0}$	$2.3_{\pm 2.0}$	$5.7_{\pm 2.5}$	$3.2_{\pm 0.8}$	$3.8_{\pm 2.0}$	$5.8_{\pm 3.4}$
MPM [73]		$7.3_{\pm 2.5}$	$15.5_{\pm 4.2}$	$23.1_{\pm 7.0}$	$7.4_{\pm 1.8}$	$6.0_{\pm 2.6}$	–
GBND [94]		$5.5_{\pm 1.7}$	$3.0_{\pm 1.4}$	$30.9_{\pm 6.2}$	$7.7_{\pm 2.6}$	$5.1_{\pm 2.2}$	$1.5_{\pm 0.4}$
PGND [93]		$3.3_{\pm 1.8}$	$2.1_{\pm 0.5}$	$3.2_{\pm 1.4}$	$4.0_{\pm 1.3}$	$3.8_{\pm 0.1}$	$1.6_{\pm 1.3}$
Diff. Spring-Mass [34]		$3.5_{\pm 1.7}$	$1.9_{\pm 1.6}$	$4.6_{\pm 2.2}$	$2.9_{\pm 0.7}$	$3.5_{\pm 1.6}$	$4.6_{\pm 2.7}$
Ours		$\mathbf{2.6_{\pm 1.31}}$	$\mathbf{1.7_{\pm 0.7}}$	$\mathbf{2.8_{\pm 1.2}}$	$\mathbf{2.7_{\pm 0.4}}$	$\mathbf{2.9_{\pm 1.8}}$	$\mathbf{1.3_{\pm 0.3}}$
Spring-Mass [34]	CD ↓	$4.5_{\pm 2.4}$	$1.7_{\pm 1.2}$	$9.1_{\pm 4.7}$	$2.9_{\pm 0.7}$	$4.7_{\pm 2.5}$	$5.1_{\pm 3.0}$
MPM [73]		$6.1_{\pm 2.4}$	$14.8_{\pm 6.5}$	$23.6_{\pm 10.4}$	$7.1_{\pm 1.7}$	$5.2_{\pm 1.7}$	–
GBND [94]		$6.6_{\pm 2.4}$	$5.0_{\pm 1.7}$	$49.1_{\pm 13.0}$	$6.5_{\pm 1.8}$	$6.3_{\pm 2.8}$	$2.7_{\pm 0.3}$
PGND [93]		$2.7_{\pm 1.4}$	$2.2_{\pm 0.9}$	$4.3_{\pm 2.4}$	$3.2_{\pm 0.8}$	$4.0_{\pm 0.1}$	$1.6_{\pm 1.0}$
Diff. Spring-Mass [34]		$3.4_{\pm 1.8}$	$1.5_{\pm 1.0}$	$6.8_{\pm 3.6}$	$2.8_{\pm 0.7}$	$4.3_{\pm 2.1}$	$3.8_{\pm 2.3}$
Ours		$\mathbf{2.3_{\pm 1.3}}$	$\mathbf{1.3_{\pm 0.6}}$	$\mathbf{4.3_{\pm 2.2}}$	$\mathbf{2.6_{\pm 0.3}}$	$\mathbf{3.8_{\pm 2.5}}$	$\mathbf{1.3_{\pm 0.2}}$
Spring-Mass [34]	EMD ↓	$2.4_{\pm 1.3}$	$1.8_{\pm 1.2}$	$4.9_{\pm 2.6}$	$1.5_{\pm 0.4}$	$1.9_{\pm 1.2}$	$2.9_{\pm 1.8}$
MPM [73]		$3.9_{\pm 1.6}$	$11.4_{\pm 5.5}$	$19.4_{\pm 7.8}$	$3.9_{\pm 0.9}$	$2.8_{\pm 1.2}$	–
GBND [94]		$3.0_{\pm 1.3}$	$2.1_{\pm 0.9}$	$24.5_{\pm 6.3}$	$3.1_{\pm 1.1}$	$2.8_{\pm 1.4}$	$0.8_{\pm 0.2}$
PGND [93]		$1.4_{\pm 0.6}$	$1.5_{\pm 0.5}$	$2.3_{\pm 1.3}$	$1.6_{\pm 0.4}$	$1.6_{\pm 0.4}$	$0.8_{\pm 0.6}$
Diff. Spring-Mass [34]		$1.9_{\pm 1.0}$	$1.6_{\pm 1.0}$	$3.9_{\pm 2.1}$	$1.6_{\pm 0.4}$	$1.6_{\pm 1.0}$	$2.3_{\pm 1.3}$
Ours		$\mathbf{1.2_{\pm 0.6}}$	$\mathbf{1.4_{\pm 0.7}}$	$\mathbf{2.2_{\pm 1.2}}$	$\mathbf{1.4_{\pm 0.2}}$	$\mathbf{1.4_{\pm 1.1}}$	$\mathbf{0.6_{\pm 0.1}}$

TABLE I. **Tracking accuracy across diverse objects.** Physics-Guided Residual Dynamics achieves the lowest tracking error across all objects and metrics, outperforming both physics-based methods and learning-based approaches.

materials into particles while using a background grid for computing spatial derivatives and enforcing conservation laws. MPM is particularly effective for materials exhibiting both solid and fluid-like behaviors but requires careful parameter tuning and can be computationally expensive.

Graph Based Neural Dynamics (GBND) [94] : This is a learned approach that employs Graph Neural Networks to learn object dynamics directly from data without explicit material parameters. The object is represented as a graph where message passing mechanisms propagate information between nodes to capture long range interactions.

Particle-Grid Neural Dynamics (PGND) [93] : This represents the state-of-the-art in learning-based methods, employing a hybrid representation of particles and spatial grids inspired by MPM to model deformable object dynamics. In this method, particles capture the object geometry while the spatial grid discretizes the 3D domain to ensure spatial continuity and improve learning efficiency.

Differentiable (Diff.) Spring-Mass [34]: This extends the spring-mass baseline with a two-stage optimization. It first estimates global parameters as spring-mass baseline, then refines them through gradient-based optimization using a differentiable simulator to learn spatially varying physical parameters. This baseline is motivated by PhysTwin [34], but we implement a variant that builds one canonical spring graph and registers it to each episode's initial observations before fitting the physical parameters across episodes. While this yields a more expressive spring-mass model, the dynamics remain constrained by the underlying spring-mass formulation.

D. Results

As explained in Section IV-B, we evaluate PGRD in two ways: 3D tracking accuracy and action-conditioned video prediction.

Dynamics and Tracking Accuracy. We first analyze our model's ability to track and simulate object states over time, with results summarized in Tab. I. PGRD consistently achieves the lowest error across all objects and metrics, demonstrating that learned residual corrections on physics backbones improve accuracy beyond either pure physics or pure learning approaches. Among physics methods, the optimized spring-mass model generally outperforms MPM, particularly for objects with coherent structure like the rope and sloth toy. Among learning-based methods, PGND substantially outperforms GBND across all objects, indicating that particle representations with physics priors are more effective than graph neural approaches for deformable object simulation. Building on the optimized spring-mass model, Diff. Spring-Mass improves over the optimized spring-mass baseline on most objects, showing that differentiable parameter refinement provides a stronger physics-based model. However, it still falls short of PGRD, whose learned residual corrections can capture dynamics beyond the spring-mass formulation.

Qualitatively, Fig. 4 reveals consistent patterns in how different approaches handle deformation. The optimized spring-mass model captures overall motion but lacks the flexibility to represent local variations in material properties, often appearing too stiff. In our implementation, the MPM baseline often struggles to maintain structural cohesion, with object particles

Fig. 4: **Qualitative comparison of PGRD on dynamics prediction.** Columns shaded in green show the input state with gripper positions (red spheres) and the ground truth object configuration after taking the action. The remaining columns show predictions from each method. PGRD accurately captures deformations across all objects. For the duster, only PGRD maintains stem rigidity while allowing deformation for the feathers. The MPM baseline is not evaluated on the Teddy Toy because it does not model the non-prehensile manipulation used in this experiment. For the Teddy Toy, PGND uses sampled gripper surface points for simulation, which are visualized as red spheres, making this example denser than the others.

separating rather than behaving as connected solids. Learning-based methods show complementary strengths: PGND generates smooth predictions but accumulates drift over time, while GBND struggles with graph topology, producing unphysical artifacts. Diff. Spring-Mass consistently matches the observed deformation better than the spring-mass model itself, suggesting that differentiable parameter optimization improves the fitted physical response. However, because it is still constrained by the spring-mass formulation, it remains less flexible than our residual model. PGRD leverages its physics backbone for stability while using learned residuals to correct local dynamics, maintaining both plausibility and accuracy throughout the rollout.

The qualitative results in Fig. 4 provide further insights into the limitations of different approaches. In our implementation, the MPM baseline performs poorly on thin objects like paper and flag, causing them to break apart and lose structure. We hypothesize that this behavior arises from the interaction between the grid discretization and the sparse particle representation used for these thin objects. The duster, with its rigid handle and soft feathers, reveals distinct failure modes across all baselines: the optimized spring-mass model cannot assign spatially varying stiffness and causes the rigid stem to bend unnaturally; MPM fails to maintain volume with material points collapsing toward the floor; GBND predicts erroneous upward motion for all feather particles; and PGND incorrectly compresses the rigid stem. Only PGRD simulates the duster correctly. For the teddy, under non-prehensile manipulation, the object slips out of the optimized spring-mass model. The differentiable spring-mass model cannot accurately model the spatially varying parameters for the duster. We do not report MPM results for the Teddy Toy because the evaluated MPM baseline does not model the non-prehensile manipulation required for this task. GBND predicts minimal particle

Method	Metric	Rope	Paper	Flag	Sloth Toy	Duster	Teddy Toy
Spring-Mass [34]	$\mathcal{J}$-Score / IoU ($\times 10$) ↑	$3.17_{\pm 1.51}$	$6.82_{\pm 1.47}$	$4.60_{\pm 1.95}$	$5.30_{\pm 0.87}$	$6.66_{\pm 0.82}$	$5.43_{\pm 1.99}$
MPM [73]		$2.44_{\pm 1.45}$	$4.51_{\pm 0.94}$	$3.78_{\pm 1.12}$	$5.22_{\pm 0.88}$	$5.92_{\pm 0.87}$	–
GBND [94]		$0.53_{\pm 0.55}$	$5.57_{\pm 1.40}$	$5.08_{\pm 1.42}$	$3.70_{\pm 1.49}$	$5.13_{\pm 1.04}$	$7.52_{\pm 0.66}$
PGND [93]		$3.18_{\pm 2.13}$	$7.08_{\pm 0.90}$	$5.47_{\pm 1.66}$	$5.86_{\pm 0.93}$	$6.69_{\pm 0.46}$	$7.13_{\pm 1.02}$
Diff. Spring-Mass [34]		$3.45_{\pm 1.43}$	$7.10_{\pm 1.37}$	$4.98_{\pm 1.82}$	$5.58_{\pm 0.82}$	$6.70_{\pm 0.78}$	$5.82_{\pm 1.85}$
Ours		$\mathbf{3.95_{\pm 1.69}}$	$\mathbf{7.53_{\pm 0.82}}$	$\mathbf{5.54_{\pm 1.89}}$	$\mathbf{6.21_{\pm 0.78}}$	$\mathbf{6.77_{\pm 0.83}}$	$\mathbf{7.82_{\pm 1.14}}$
Spring-Mass [34]	$\mathcal{F}$-Score ($\times 10$) ↑	$6.59_{\pm 1.60}$	$4.71_{\pm 2.61}$	$2.08_{\pm 1.81}$	$5.76_{\pm 1.12}$	$5.32_{\pm 0.97}$	$4.73_{\pm 2.72}$
MPM [73]		$4.92_{\pm 2.45}$	$4.73_{\pm 0.82}$	$2.49_{\pm 0.97}$	$5.13_{\pm 0.86}$	$5.00_{\pm 1.08}$	–
GBND [94]		$1.95_{\pm 1.91}$	$4.95_{\pm 1.35}$	$2.54_{\pm 2.04}$	$3.59_{\pm 1.15}$	$4.11_{\pm 1.09}$	$7.71_{\pm 0.96}$
PGND [93]		$5.87_{\pm 2.96}$	$5.39_{\pm 1.65}$	$2.65_{\pm 1.98}$	$5.53_{\pm 1.11}$	$5.29_{\pm 0.97}$	$7.27_{\pm 1.54}$
Diff. Spring-Mass [34]		$6.95_{\pm 1.50}$	$4.98_{\pm 2.45}$	$2.22_{\pm 1.70}$	$6.03_{\pm 1.05}$	$5.45_{\pm 0.91}$	$5.05_{\pm 2.50}$
Ours		$\mathbf{7.38_{\pm 1.85}}$	$\mathbf{5.91_{\pm 1.76}}$	$\mathbf{2.68_{\pm 2.14}}$	$\mathbf{6.30_{\pm 1.07}}$	$\mathbf{5.51_{\pm 1.02}}$	$\mathbf{8.03_{\pm 1.75}}$
Spring-Mass [34]	LPIPS ($\times 100$) ↓	$2.6_{\pm 0.9}$	$5.6_{\pm 2.2}$	$9.9_{\pm 3.0}$	$5.9_{\pm 2.0}$	$7.0_{\pm 1.9}$	$3.6_{\pm 1.7}$
MPM [73]		$3.8_{\pm 1.4}$	$6.5_{\pm 1.4}$	$9.1_{\pm 2.5}$	$7.7_{\pm 1.7}$	$6.8_{\pm 1.7}$	–
GBND [94]		$5.2_{\pm 1.7}$	$5.7_{\pm 1.3}$	$8.8_{\pm 1.3}$	$10.5_{\pm 2.0}$	$6.4_{\pm 1.3}$	$2.2_{\pm 0.5}$
PGND [93]		$2.6_{\pm 1.3}$	$5.2_{\pm 1.1}$	$8.9_{\pm 1.9}$	$6.5_{\pm 1.8}$	$6.6_{\pm 0.9}$	$2.0_{\pm 0.8}$
Diff. Spring-Mass [34]		$2.35_{\pm 0.85}$	$5.05_{\pm 2.00}$	$8.95_{\pm 2.70}$	$5.45_{\pm 1.85}$	$6.35_{\pm 1.75}$	$3.25_{\pm 1.55}$
Ours		$\mathbf{2.3_{\pm 1.1}}$	$\mathbf{4.9_{\pm 1.4}}$	$\mathbf{8.5_{\pm 2.9}}$	$\mathbf{5.4_{\pm 2.1}}$	$\mathbf{5.9_{\pm 1.9}}$	$\mathbf{1.7_{\pm 0.9}}$

TABLE II. **Action-conditioned video prediction with dynamics prediction and 3DGS.** PGRD produces the most accurate visual predictions, demonstrating superior rendering quality.

displacement, producing an overly stiff response. PGND shows the gripper's entire surface in our visualization, but the gripper fingers themselves are not visible because they have pushed into the teddy's interior. Only our PGRD successfully captures the deformation, properly maintaining contact while allowing realistic compression.

Action Conditioned Video Prediction. Given an initial observation and a sequence of actions, we evaluate whether accurate dynamics predictions translate to realistic future frame generation. We use 3D Gaussian Splatting as the rendering module, where Gaussians are fit to the observed point cloud at the initial timestep and parameterized by position, covariance, opacity, and color. As PGRD predicts future point cloud configurations, we update the Gaussian positions and covariances while keeping opacity and color fixed, allowing the rendered frames to reflect the predicted deformations while preserving appearance consistency with the initial observation. At each timestep, we project the updated Gaussians into the camera view and alpha-blend them in depth order to produce the output image. Fig. 5 shows an example pair consisting of a frame rendered from the dynamics predicted by PGRD and the corresponding ground truth frame, which are used to compute the visual metrics. Among the physics-based methods, Diff. Spring-Mass generally improves over the optimized spring-mass baseline, indicating that differentiable parameter refinement translates to more accurate rendered masks and appearances. Compared with the learning-based method PGND, its performance is mixed: it is better on some objects and metrics, but worse on others. Overall, PGRD consistently achieves the best $\mathcal{J}$-Score and $\mathcal{F}$-Score across all objects, demonstrating better spatial overlap and foreground detection compared to both physics-based and learning-based baselines. PGRD also maintains the lowest LPIPS scores, indicating that the physics backbone enables the residual model to preserve object appearance during rendering.

V. Applications

A. Planning using PGRD

In our planning experiments, we use PGRD as a forward model for model-based control: given a state s_t and an action a_t, it predicts the next state s_{t+1}, where actions specify

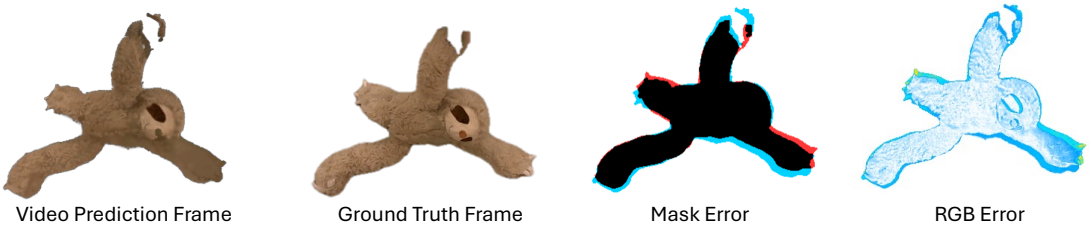

Fig. 5: **Example pair used to compute the visual metrics.** We evaluate only the object of interest, ignoring the background. The mask error visualization shows true-positive object pixels in black, false negatives in red, and false positives in cyan. The RGB error heat map shows the per-pixel appearance difference within the evaluated object region, with stronger colors indicating larger RGB error. Part of the upper arm is missing because the sloth was grasped by the gripper, which is now removed.

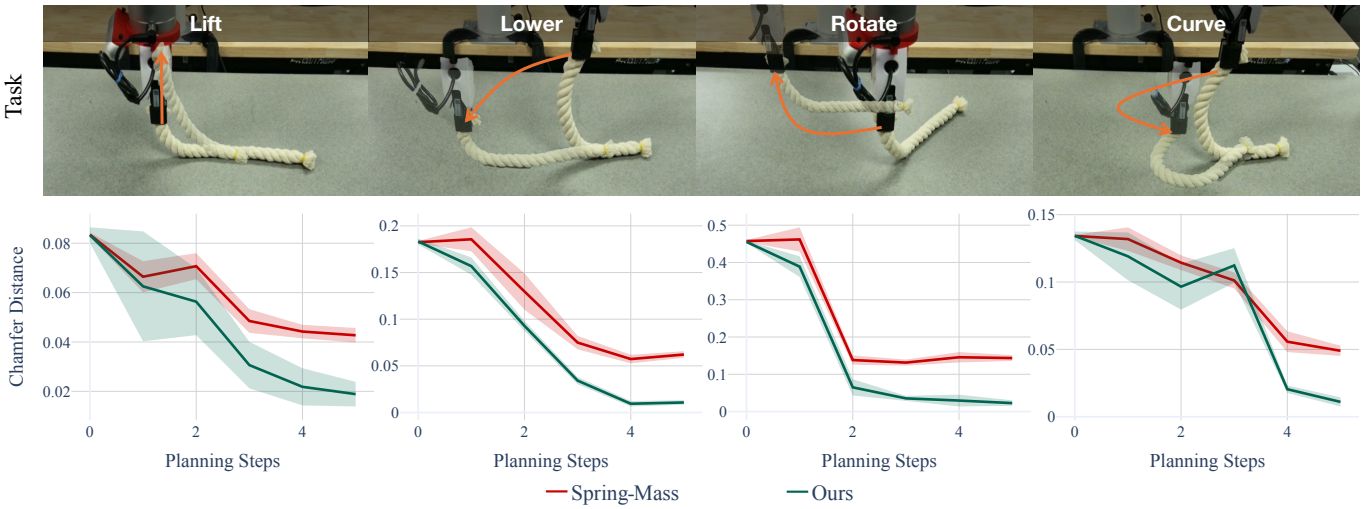

Fig. 6: **Planning results on rope manipulation tasks.** Top row shows start and end configurations for four tasks. The bottom row shows the Chamfer Distance to the target during MPPI planning. PGRD (green) converges faster and achieves lower final error than the optimized spring-mass baseline (red) across all tasks.

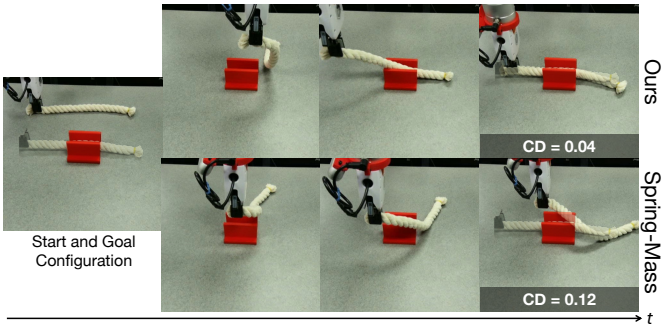

Fig. 7: **Cable Rerouting Execution.** Comparison of PGRD (top) and the Spring-Mass backbone (bottom) for rope rerouting through a slot. PGRD navigates the rope through the opening while the Spring-Mass backbone exhibits repeated collisions with the slot edges.

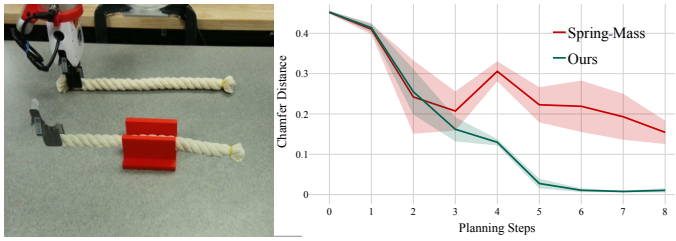

Fig. 8: **Cable rerouting through narrow slot.** Left: distribution of start and end configurations. Right: Chamfer Distance during MPPI planning steps. PGRD achieves lower distance than the spring-mass baseline, successfully threading the rope through the slot in 8 out of 10 trials compared to 2 out of 10 for the baseline.

gripper positions that interact with the object. To plan over this forward model, we integrate PGRD with Model Predictive Path Integral (MPPI) control [20]. At each timestep, MPPI samples a batch of candidate action sequences $\{\tau_i\}$ spanning a planning horizon H, rolls out PGRD to predict future states, and updates the action distribution toward lower-cost regions. Following standard MPC practice, only the first action is executed before replanning with the newly observed state. We use the Chamfer Distance (CD) between the predicted and a pre-collected target point cloud as the planning objective. The parameters for planning experiments is provided in App. E.

Standard Manipulation Tasks. We evaluate planning on the rope with four start and goal configurations, as shown in Fig. 6 (top): (1) lifting the rope up, (2) lowering it, (3) rotating it such that one end remains roughly stationary while the other moves, and (4) deforming it into a curved shape. We compare PGRD against the spring-mass backbone, providing both methods with the same MPPI planning budget. We run 10 trials for each task using the same start and end configurations. As shown in Fig. 6, PGRD's CD decreases smoothly and saturates at a value close to the target, whereas the spring-mass model converges more slowly and sometimes fails to reach a comparable final error. Additional results on the sloth toy are provided in App. C.

Cable Rerouting. We further test PGRD on a more challenging task: cable rerouting through a narrow slot. This requires the robot to move the rope and thread it through the slot. For these experiments, the residual dynamics model is not trained on examples containing rope-slot interaction data. Hence, this also shows the generalization capability of PGRD. We run 10 trials with different start and end configurations. The target configuration is provided to the planner.

As demonstrated in Fig. 7, PGRD successfully navigates the rope through the slot. The learned residuals compensate for inaccuracies in the physics backbone, enabling precise trajectory execution that avoids collisions with the edges. In contrast, the spring-mass backbone struggles with these fine dynamics. Small errors cause the rope to collide with the edges, forcing the planner to retry the motion multiple times before completing the task.

Quantitatively, Fig. 8 shows the distributions of start and end configurations overlaid (left) and comparisons of chamfer distances (right). PGRD achieves significantly lower CD than the optimized spring-mass. We also evaluate success rates for this

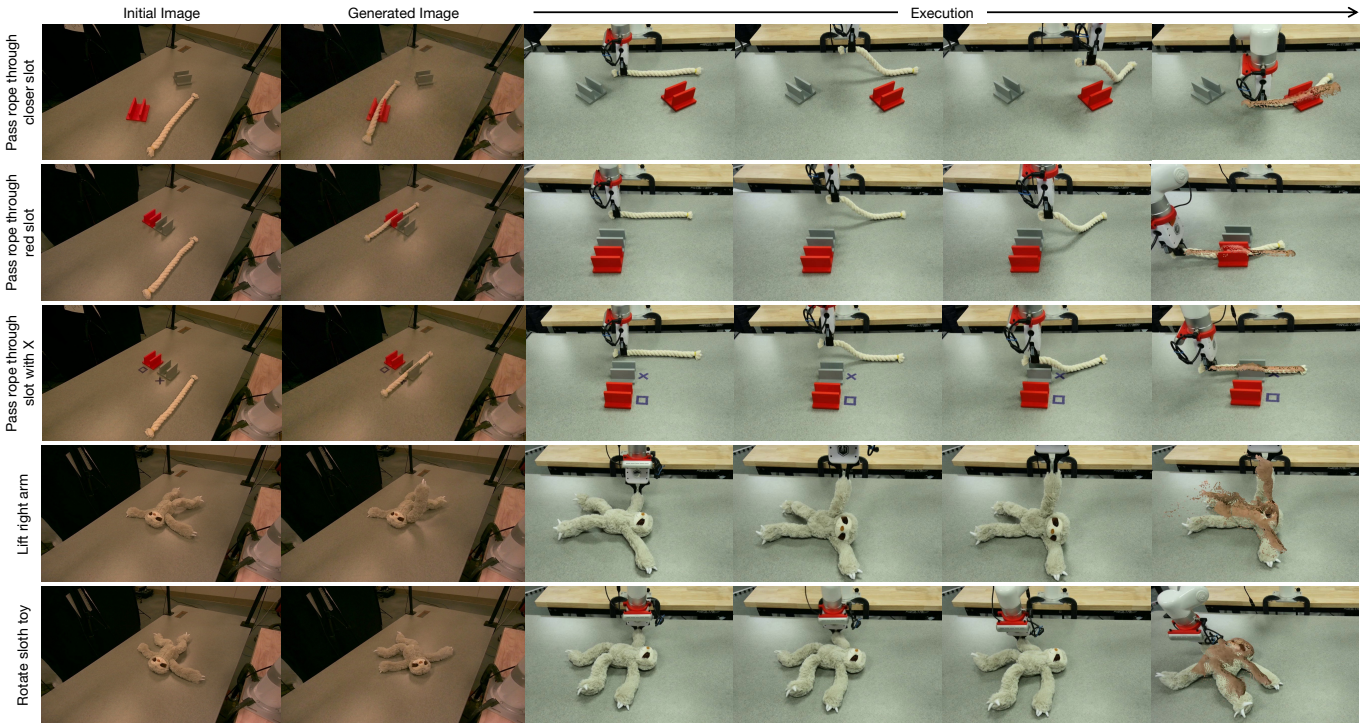

Fig. 9: **Execution of PGRD with generated goals.** The first column shows the initial image, and the second column shows the language-conditioned generated goal image. Subsequent columns show robot execution from a higher-quality demo camera. The target point cloud from the goal image is overlaid on the last image. PGRD successfully executes a range of tasks, including passing a rope through a specified slot and moving a sloth toy to target configurations.

task. A trial is considered successful if the rope passes entirely through the slot and the gripper remains below the slot height in the final configuration. Under this criterion, PGRD succeeds in 8/10 trials, whereas the spring-mass model succeeds in only 2/10 trials. These results demonstrate that the improved model enables more reliable execution of challenging manipulation tasks.

Language-Conditioned Planning. The planning experiments above require a target point cloud, so it must be collected beforehand. To relax this requirement, we integrate PGRD with a language-conditioned goal-generation pipeline, enabling the planner to operate solely on language commands.

Given an initial RGB image and a language command (e.g., "pass rope through red slot"), we use Nano Banana Pro [64] to generate a goal image. Note that here we use a single camera rather than the four in the standard planning experiments, since the goal image can be generated more reliably for a single viewpoint. To estimate depth from the generated image, we use Depth Anything V2 [88]. Following Patel *et al.* [60], we resolve the scale and shift ambiguity inherent to monocular depth estimation by fitting an affine transform to align the predicted depth to the initial depth map from the depth camera. We restrict the alignment to background table points, which remain consistent in both images. Since the object appears in a different configuration in the goal image than in the initial image, including object points for this alignment would degrade the quality. The resulting depth map is unprojected into 3D with known camera parameters to obtain the target point cloud, which is passed directly to the MPPI planner.

To determine where to grasp the object, we use AnyGrasp [18] to generate grasp candidates on the initial point cloud. For the rope, we select the candidate closest to the right end. For the sloth toy, the language command implies a specific region to manipulate, which we identify by extracting DINOv2 [59] features from both the initial and goal images to establish dense correspondences. We locate the object point with the largest displacement between the two images and select the AnyGrasp candidate closest to that point.

Fig. 9 shows real-world execution for multiple tasks. Beyond removing the need to physically collect the goal configuration, this pipeline opens up a richer space of goals, including spatial references such as "pass rope through closer slot" and "lift right arm" of the sloth toy.

B. Interactive Photo-Realistic Simulation.

As shown in Fig. 10, PGRD supports interactive, photo-realistic simulation by combining its predicted dynamics with 3D Gaussian Splatting. Users issue manipulation commands via keyboard input; at each step, the physics backbone propagates the object state, the learned residual network applies per-particle corrections, and the updated Gaussians are rendered to produce a photo-realistic view of the resulting deformation. Because only Gaussian positions are updated while appearance attributes remain fixed, the rendering maintains visual consistency with the initial observation throughout the interaction.

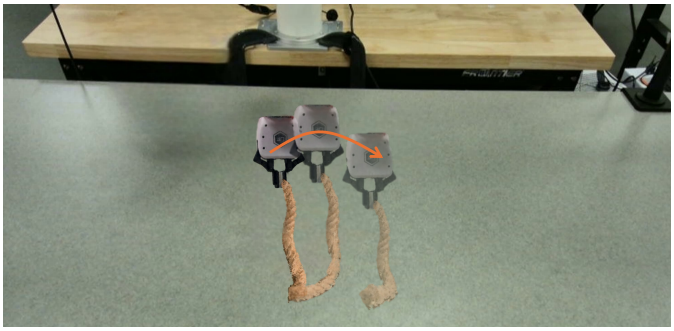

Fig. 10: **Interactive rope manipulation with PGRD rendered using 3DGS.** The visualization overlays three timesteps, with later configurations in lighter opacity. As PGRD predicts the dynamics, the 3D Gaussians attached to the particles are updated to render images that maintain visual consistency.

VI. Conclusions

This paper presented Physics-Guided Residual Dynamics (PGRD), a hybrid simulation framework that bridges physics-based and learning-based approaches for deformable object simulation. By combining an optimized spring-mass backbone with learned velocity residuals, PGRD achieves superior accuracy while maintaining physical plausibility. Our velocity-based residuals, combined with a sliding-window transformer for temporal aggregation, yield stable predictions, avoiding the instability issues plaguing naive residual learning. Extensive experiments across real-world objects demonstrate that PGRD consistently outperforms a variety of SOTA methods. PGRD handles challenging scenarios involving heterogeneous materials, volumetric deformations, and contact-rich interactions where existing methods struggle. Beyond tracking, PGRD also shows promise for applications like action-conditioned video prediction, planning, and interactive simulation.

References

[1] Bo Ai, Stephen Tian, Haochen Shi, Yixuan Wang, Tobias Pfaff, Cheston Tan, Henrik I Christensen, Hao Su, Jiajun Wu, and Yunzhu Li. A review of learning-based dynamics models for robotic manipulation. *Science Robotics*, 10(106):eadt1497, 2025.

[2] Anurag Ajay, Jiajun Wu, Nima Fazeli, Maria Bauza, Leslie P Kaelbling, Joshua B Tenenbaum, and Alberto Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3066–3073. IEEE, 2018.

[3] Hugo Attali, Davide Buscaldi, and Nathalie Pernelle. Rewiring techniques to mitigate oversquashing and oversmoothing in gnns: A survey. *arXiv preprint arXiv:2411.17429*, 2024.

[4] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, pages 43 – 54. SIGGRAPH, July 1998.

[5] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[6] Dominik Bauer, Zhenjia Xu, and Shuran Song. Doughnet: A visual predictive model for topological manipulation of deformable objects. In *European Conference on Computer Vision*, pages 92–108. Springer, 2024.

[7] Leonard Bauersfeld, Elia Kaufmann, Philipp Foehn, Sihao Sun, and Davide Scaramuzza. Neurobem: Hybrid aerodynamic quadrotor model. *arXiv preprint arXiv:2106.08015*, 2021.

[8] Onur Beker, Nico Gürtler, Ji Shi, A René Geist, Amirreza Razmjoo, Georg Martius, and Sylvain Calinon. A smooth analytical formulation of collision detection and rigid body dynamics with contact. *arXiv preprint arXiv:2503.11736*, 2025.

[9] Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *international conference on machine learning*, pages 2402–2411. PMLR, 2020.

[10] Zhenfang Chen, Kexin Yi, Yunzhu Li, Mingyu Ding, Antonio Torralba, Joshua B Tenenbaum, and Chuang Gan. Comphy: Compositional physical reasoning of objects and events from videos. *arXiv preprint arXiv:2205.01089*, 2022.

[11] Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1724–1734, 2014.

[12] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[13] Loris Di Natale, Bratislav Svetozarevic, Philipp Heer, and Colin N Jones. Physically consistent neural networks for building thermal modeling: theory and analysis. *Applied Energy*, 325:119806, 2022.

[14] Runpei Dong, Ziyan Li, Xialin He, and Saurabh Gupta. Learning humanoid end-effector control for open-vocabulary visual loco-manipulation. *arXiv preprint arXiv:2602.16705*, 2026.

[15] Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Diffpd: Differentiable projective dynamics. *ACM Transactions on Graphics (ToG)*, 41(2):1–21, 2021.

[16] Bardienus P. Duisterhof, Jan Oberst, Bowen Wen, Stan Birchfield, Deva Ramanan, and Jeffrey Ichnowski. Rayst3r: Predicting novel depth maps for zero-shot object completion, 2025. URL https://arxiv.org/abs/2506.05285.

[17] Ben Evans, Abitha Thankaraj, and Lerrel Pinto. Context

is everything: Implicit identification for dynamics adaptation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2642–2648. IEEE, 2022.

[18] Hao-Shu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 39(5):3929–3945, 2023.

[19] Junpeng Gao, Mike Y Michelis, Andrew Spielberg, and Robert K Katzschmann. Sim-to-real of soft robots with learned residual physics. *IEEE Robotics and Automation Letters*, 2024.

[20] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[21] Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

[22] Florian Golemo, Adrien Ali Taiga, Aaron Courville, and Pierre-Yves Oudeyer. Sim-to-real transfer with neural-augmented robot simulation. In *Conference on Robot Learning*, pages 817–828. PMLR, 2018.

[23] Joshua Gruenstein, Tao Chen, Neel Doshi, and Pulkit Agrawal. Residual model learning for microrobot control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7219–7226. IEEE, 2021.

[24] Mathew Halm and Michael Posa. Set-valued rigid body dynamics for simultaneous frictional impact. *arXiv preprint arXiv:2103.15714*, 2021.

[25] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[26] Eric Heiden, Miles Macklin, Yashraj Narang, Dieter Fox, Animesh Garg, and Fabio Ramos. Disect: A differentiable simulation engine for autonomous robotic cutting. *arXiv preprint arXiv:2105.12244*, 2021.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[28] Philipp Holl, Vladlen Koltun, and Nils Thuerey. Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457*, 2020.

[29] 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. *arXiv preprint arXiv:2506.23126*, 2025.

[30] 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.

[31] Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *arXiv preprint arXiv:2104.03311*, 2021.

[32] Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. *arXiv preprint arXiv:2104.02646*, 2021.

[33] Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)*, 34(4): 1–10, 2015.

[34] Hanxiao Jiang, Hao-Yu Hsu, Kaifeng Zhang, Hsin-Ni Yu, Shenlong Wang, and Yunzhu Li. Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos. *arXiv preprint arXiv:2503.17973*, 2025.

[35] Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6013–6022, 2025.

[36] Hajun Kim, Dongyun Kang, Min-Gyu Kim, Gijeong Kim, and Hae-Won Park. Online friction coefficient identification for legged robots on slippery terrain using smoothed contact gradients. *IEEE Robotics and Automation Letters*, 2025.

[37] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International conference on machine learning*, pages 2688–2697. Pmlr, 2018.

[38] Thomas Kipf, Elise Van der Pol, and Max Welling. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019.

[39] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018.

[40] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation networks for model-based control under partial observation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1205–1211. IEEE, 2019.

[41] Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. *Advances in neural information processing systems*, 32, 2019.

[42] Quentin Le Lidec, Louis Montaut, Yann de Mont-Marin, Fabian Schramm, and Justin Carpentier. End-to-end and highly-efficient differentiable simulation for robotics. *arXiv preprint arXiv:2409.07107*, 2024.

[43] Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, pages 256–266. PMLR, 2022.

[44] Min Liu, Gang Yang, Siyuan Luo, and Lin Shao. Softmac: Differentiable soft body simulation with forecast-based contact model and two-way coupling with articulated rigid bodies and clothes. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12008–12015. IEEE, 2024.

[45] Tiantian Liu, Adam W Bargteil, James F O'Brien, and Ladislav Kavan. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)*, 32(6):1–7, 2013.

[46] Bryn Lloyd, Gábor Székely, and Matthias Harders. Identification of spring parameters for deformable object simulation. *IEEE transactions on visualization and computer graphics*, 13(5):1081–1094, 2007.

[47] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pages 800–809. IEEE, 2024.

[48] Pingchuan Ma, Tao Du, Joshua B Tenenbaum, Wojciech Matusik, and Chuang Gan. Risp: Rendering-invariant state predictor with differentiable simulation and rendering for cross-domain parameter estimation. *arXiv preprint arXiv:2205.05678*, 2022.

[49] Pingchuan Ma, Peter Yichen Chen, Bolei Deng, Joshua B Tenenbaum, Tao Du, Chuang Gan, and Wojciech Matusik. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *International Conference on Machine Learning*, pages 23279–23300. PMLR, 2023.

[50] Xiao Ma, David Hsu, and Wee Sun Lee. Learning latent graph dynamics for visual manipulation of deformable objects. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8266–8273. IEEE, 2022.

[51] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 33 (4):1–12, 2014.

[52] Dominique Madier. An introduction to the fundamentals of mesh generation in finite element analysis. Technical report, FEA Academy, Montreal, Canada, December 2023. URL https://www.fea-academy.com/pdf/FEA%20Academy%20-%20The%20Fundamentals%20of%20Mesh%20Generation%20in%20Finite%20Element%20Analysis.pdf. Published by FEA Academy.

[53] L Angela Mihai and Alain Goriely. How to characterize a nonlinear elastic material? a review on nonlinear constitutive parameters in isotropic finite elasticity. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2207):20170607, 2017.

[54] Brian Mirtich and John Canny. Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 181–ff, 1995.

[55] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li F Fei-Fei, Josh Tenenbaum, and Daniel L Yamins. Flexible neural representation for physics prediction. *Advances in neural information processing systems*, 31, 2018.

[56] Matthias Müller and Markus H Gross. Interactive virtual materials. In *Graphics interface*, volume 2004, pages 239–246, 2004.

[57] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2): 109–118, 2007.

[58] J Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. In *International conference on learning representations*, 2020.

[59] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[60] Shivansh Patel, Shraddhaa Mohan, Hanlin Mai, Unnat Jain, Svetlana Lazebnik, and Yunzhu Li. Robotic manipulation by imitating generated videos without physical demonstrations. *arXiv preprint arXiv:2507.00990*, 2025.

[61] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International conference on learning representations*, 2020.

[62] 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.

[63] Yiling Qiao, Junbang Liang, Vladlen Koltun, and Ming Lin. Differentiable simulation of soft multi-body systems. *Advances in Neural Information Processing Systems*, 34:17123–17135, 2021.

[64] Naina Raisinghani. Introducing nano banana pro. https://blog.google/innovation-and-ai/products/nano-banana-pro/, November 2025. Google DeepMind Blog. Accessed: 2026-03-29.

[65] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024.

[66] Junior Rojas, Eftychios Sifakis, and Ladislav Kavan. Differentiable implicit soft-body physics. *arXiv preprint arXiv:2102.05791*, 2021.

[67] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[68] Yidi Shao, Chen Change Loy, and Bo Dai. Transformer with implicit edges for particle-based physics simulation.

In *European conference on computer vision*, pages 549–564. Springer, 2022.

[69] Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. *arXiv preprint arXiv:2306.14447*, 2023.

[70] 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.

[71] Eftychios Sifakis and Jernej Barbic. Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *Acm siggraph 2012 courses*, pages 1–50. ACM, 2012.

[72] David Stewart and Jeffrey C Trinkle. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 162–169. IEEE, 2000.

[73] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013.

[74] Shijie Tan, Hongjun Zhou, and Jinjin Zheng. A hybrid model method for accurate surface deformation and incision based on fem and pbd. *Scientific Programming*, 2021(1):8343312, 2021.

[75] Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in neural information processing systems*, 33:6111–6122, 2020.

[76] Vaiva Vasiliauskaite and Nino Antulov-Fantulin. Generalization of neural network models for complex network dynamics. *Communications Physics*, 7(1):348, 2024.

[77] Changhao Wang, Yuyou Zhang, Xiang Zhang, Zheng Wu, Xinghao Zhu, Shiyu Jin, Te Tang, and Masayoshi Tomizuka. Offline-online learning of deformation model for cable manipulation with graph neural networks. *IEEE Robotics and Automation Letters*, 7(2):5544–5551, 2022.

[78] Xinjue Wang, Esa Ollila, and Sergiy A Vorobyov. Graph neural network sensitivity under probabilistic error model. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 2146–2150. IEEE, 2022.

[79] Zizhao Wang, Xuesu Xiao, Zifan Xu, Yuke Zhu, and Peter Stone. Causal dynamics learning for task-independent state abstraction. *arXiv preprint arXiv:2206.13452*, 2022.

[80] Robert J Webster III and Bryan A Jones. Design and kinematic modeling of constant curvature continuum robots: A review. *The International Journal of Robotics Research*, 29(13):1661–1683, 2010.

[81] Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C Karen Liu. Fast and feature-complete differentiable physics for articulated rigid bodies with contact. *arXiv preprint arXiv:2103.16021*, 2021.

[82] William F Whitney, Jacob Varley, Deepali Jain, Krzysztof Choromanski, Sumeet Singh, and Vikas Sindhwani. Modeling the real world with high-density visual particle dynamics. *arXiv preprint arXiv:2406.19800*, 2024.

[83] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4840–4851, 2024.

[84] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv:1910.13439*, 2019.

[85] Zhenjia Xu, Jiajun Wu, Andy Zeng, Joshua B Tenenbaum, and Shuran Song. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. *arXiv preprint arXiv:1906.03853*, 2019.

[86] Shangjie Xue, Shuo Cheng, Pujith Kachana, and Danfei Xu. Neural field dynamics model for granular object piles manipulation. In *Conference on Robot Learning*, pages 2821–2837. PMLR, 2023.

[87] Mengyuan Yan, Yilin Zhu, Ning Jin, and Jeannette Bohg. Self-supervised learning of state estimation for manipulating deformable linear objects. *IEEE robotics and automation letters*, 5(2):2372–2379, 2020.

[88] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 2024.

[89] Xintong Yang, Ze Ji, and Yu-Kun Lai. Differentiable physics-based system identification for robotic manipulation of elastoplastic materials. *The International Journal of Robotics Research*, page 02783649251334661, 2025.

[90] Xiaohan Ye, Kui Wu, Zherong Pan, and Taku Komura. Efficient differentiable contact model with long-range influence. *arXiv preprint arXiv:2509.20917*, 2025.

[91] Rose Yu and Rui Wang. Learning dynamical systems from data: An introduction to physics-guided deep learning. *Proceedings of the National Academy of Sciences*, 121(27):e2311808121, 2024.

[92] Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Adaptigraph: Material-adaptive graph-based neural dynamics for robotic manipulation. *arXiv preprint arXiv:2407.07889*, 2024.

[93] 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.

[94] Mingtong Zhang, Kaifeng Zhang, and Yunzhu Li. Dynamic 3d gaussian tracking for graph-based neural dynamics modeling. *arXiv preprint arXiv:2410.18912*, 2024.

[95] Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and simulation of elastic objects with

spring-mass 3d gaussians. In *European Conference on Computer Vision*, pages 407–423. Springer, 2024.

[96] Yaofeng Desmond Zhong, Jiequn Han, Biswadip Dey, and Georgia Olympia Brikis. Improving gradient computation for differentiable physics simulation with contacts. In *Learning for dynamics and control conference*, pages 128–141. PMLR, 2023.

[97] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

[98] Yicheng Zhu, David Yang, and Yangming Lee. Deformable and fragile object manipulation: A review and prospects. *Sensors*, 25(17):5430, 2025.

Appendix

We structure the supplement into the following sections:

- A Ablation study of the main components of PGRD, including the encoder, decoder, temporal aggregator, gating, and internal particles for volumetric objects.
- B Network architecture details and hyperparameter settings for the encoder, decoder, and temporal aggregator.
- C Planning experiments on the sloth toy, showing PGRD's behavior on a volumetric object with long flexible limbs.
- D Data collection and processing pipeline for extracting temporally consistent particle trajectories from multi-view RGBD observations.
- E MPPI planning hyperparameters used for the simple tasks and the cable rerouting experiments.

A. Ablation Study

Ablation variant	Rope	Sloth Toy
PGRD	**2.6**	**2.7**
Replace PTv3 with PointNet++	3.0	3.2
Remove decoder	2.8	3.0
Remove temporal aggregator	2.8	3.1
Replace gating with addition	2.7	2.9
Remove internal particles	N/A	3.8

TABLE III. **Ablations of the main components of our model.** We report MDE. Replacing velocity with position residuals leads to divergence. Rope has no internal particles, so it is N/A.

We provide a detailed ablation study of the main components of PGRD. We consider both replacement ablations, which test natural alternatives to our design, and removal ablations, which isolate the role of individual components. All quantitative results are summarized in Tab. III. We report only MDE, and the other metrics follow the same trends.

A. Encoder choice

We use PTv3 as our encoder. We compare this design to a PointNet++ [62], which is a natural baseline because it is a standard point cloud architecture and also used in PGND. This comparison tests whether the stronger local feature extraction of PTv3 is important for residual prediction. The trend in Tab. III indicates that it is.

B. Decoder

In Sec. III-C, we describe a NeRF-style decoder that combines encoder features with positional encoding to predict per-particle residual velocities. A natural simplification is to remove this decoder and directly project the encoder features to residual velocities. This tests whether explicit position-conditioned decoding remains useful once the encoder features are already available. The results in Tab. III suggest that it does.

C. Temporal aggregator

The temporal aggregator refines the current residual estimate using a sliding window of previous predictions. This design is motivated by the fact that many deformation effects depend on recent motion history rather than on the current configuration alone. The corresponding ablation removes temporal refinement while keeping the encoder and decoder unchanged. The degradation in Tab. III supports the need for temporal context in predicting residual corrections.

D. Gated temporal refinement

Our temporal module uses gating to blend the decoder prediction with the temporally refined correction. A natural alternative is direct addition. This ablation isolates whether the gate is important beyond the temporal model itself. As shown in Tab. III, the gate is useful for turning temporal information into a stable refinement rather than an uncontrolled update.

E. Internal particles for volumetric objects

For volumetric objects, the spring mass backbone includes internal particles so the particles don't collapse. This component is relevant only to the sloth and the teddy toy. The corresponding ablation removes the internal particles while keeping the rest of the system unchanged. The resulting drop in Tab. III is consistent with the role of internal particles in preventing unrealistic collapse for volumetric objects.

B. Network Architecture Details

Our network network consists of three primary components: an encoder, a decoder, and a temporal aggregator. The hyperparameter values for each component are provided in Table IV. The encoder uses a Point Transformer V3 (PTv3) [83] architecture to extract per-particle features through patch-based self-attention, where the feature dimension determines the size of the latent representation. The input channels correspond to the concatenated position and velocity information from the current state, history timesteps, and the physics simulator predictions. The decoder employs a conditional NeRF-style MLP to map spatial locations to residual velocities, where hidden layers refers to the number of fully connected layers, hidden dimension specifies the width of each layer, and output channels indicates the dimensionality of the predicted residual velocity (3 for xyz components). The decoder takes as input the per-particle features from the encoder, concatenated with Fourier positional encodings of particle positions. The temporal aggregator employs a transformer encoder to refine predictions using temporal context. Here, the embedding dimension defines the size of the latent space for temporal features, attention heads specifies the number of parallel attention mechanisms, transformer layers indicates the depth of the transformer stack, and feedforward dimension determines the width of the intermediate feedforward network within

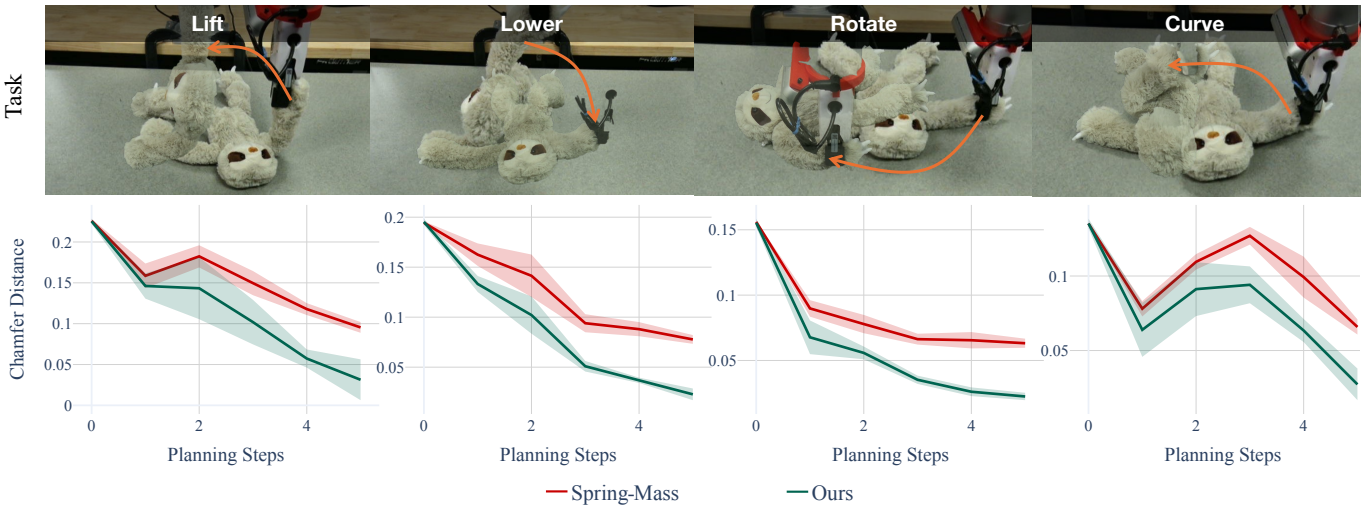

Fig. 11: **Planning results on sloth toy manipulation tasks.** Top row displays the initial and target configurations for four manipulation objectives. Bottom row plots the Chamfer Distance evolution throughout MPPI planning. PGRD (green) consistently achieves superior convergence compared to the tuned spring-mass baseline (red).

each transformer layer. Finally, the gating scale controls the magnitude of temporal corrections applied to the base velocity predictions, preventing large adjustments that could destabilize the simulation.

Parameter	**Value**
Encoder (PTv3)	
Feature dimension	64
Input channels	18
Decoder	
Hidden layers	2
Hidden dimension	64
Output channels	3
Temporal Aggregator	
Embedding dimension (d)	64
Attention heads (H)	4
Transformer layers (L)	2
Feedforward dimension (d_{ff})	128
Gating scale	0.1

TABLE IV. Network architecture hyperparameters.

C. Planning on Sloth Toy

We extend our planning experiments to the sloth toy, a volumetric object with long, flexible limbs. It requires accurate prediction of both volumetric deformation and limb dynamics, testing the framework's ability to handle three-dimensional objects. Similar to rope, we design four manipulation objectives with varying complexity, as visualized in Fig. 11 (top): (1) lifting the object, (2) lowering it, (3) rotating it, and (4) bending it into a curved configuration. Following the same experimental setup, we conduct 10 trials per objective with identical initial and target states, allocating equal computational budget to both PGRD and the spring-mass baseline. The results in Fig. 11 demonstrate that PGRD exhibits consistent convergence behavior, reaching target configurations with reasonable residual error. The spring-mass baseline shows slower error reduction and occasionally plateaus at suboptimal solutions, particularly for tasks involving complex limb deformation, where the learned residuals provide crucial corrections to the physics backbone predictions.

D. Data Collection and Processing

We mount four Intel RealSense D455 cameras around the workspace to capture synchronized RGBD observations at 30 Hz. The cameras are calibrated to a shared world coordinate frame using a checkerboard calibration procedure, enabling the fusion of observations across views. For each camera view, we apply Grounded SAM 2 to extract object masks. Using text prompts containing object descriptions, it detects and segments the object in the first frame of a sequence and propagates the mask across subsequent frames. We employ CoTracker [35] as the tracking model due to its stable tracking performance. It predicts a set of 2D trajectories, initialized from uniformly sampled grid locations within the first-frame object mask. For each pixel in the segmented region at time t, we compute its 2D displacement to time $t+1$ from the predicted trajectories, and convert it to a 2D velocity by dividing by the time step.

For each camera view, we lift the 2D pixel velocities to 3D using the corresponding depth measurements. Specifically, for a pixel at position (u, v) with depth d and 2D velocity (v_u, v_v), we compute the 3D velocity by back-projecting both the current and next pixel positions to 3D coordinates using camera intrinsics, then computing the difference. This yields per-pixel 3D velocities in the camera frame, which we transform to the world frame using the calibration parameters. We aggregate velocities across all camera views by averaging the 3D velocity estimates for overlapping spatial regions, thereby improving robustness to noise and partial observations from individual views.

Given the 3D point cloud with per-point velocities at each timestep, we extract temporally consistent particle trajectories using an iterative rollout procedure. Starting from frame 0, we initialize particles at the point cloud positions $\{x_i^0\}$. For

each subsequent timestep t, we propagate particles forward using their current velocities: $x_i^{t+1} = x_i^t + v_i^t \cdot \Delta t$. To update velocities at the new positions, we perform k-nearest neighbor search (with $k = 5$) between the propagated particle positions and the observed point cloud at time $t + 1$. Each particle's velocity is updated to the mean velocity of its k nearest neighbors in the observed cloud. This iterative process maintains correspondence across frames while being robust to tracking noise, yielding persistent point tracks suitable for training our dynamics model. The entire processing pipeline runs at approximately 2 Hz for sequences with 1000 particles.

E. MPPI Planning Parameters

We use different set of parameters for simple and cable rerouting experiments. They are provided in Tab. V.

MPPI Hyperparameter	**Simple tasks**	**Cable rerouting**
Number of sampled trajectories	8	15
Planning horizon length	10	15
Optimization iterations per step	3	6
Action noise covariance	0.1	0.1

TABLE V. **MPPI hyperparameters used for planning.** Separate settings are used for the simpler tasks and for cable rerouting.