Point Cloud Sequence Encoding for Material-conditioned Graph Network Simulators

Philipp Dahlinger[1*] **Balázs Gyenes**[1] **Niklas Freymuth**[1] **Luca Geminiani**[2]

Tobias Würth[3] **Johannes Mitsch**[3] **Nadja Klein**[2] **Luise Kärger**[3]

Gerhard Neumann[1]

[1]Autonomous Learning Robots, [2]Methods for Big Data, [3]Institute of Vehicle System Technology
Karlsruhe Institute of Technology, Karlsruhe

Abstract

Graph Network Simulators (GNSs) have emerged as powerful surrogates for complex physics-based simulation, offering inherent differentiability and orders-of-magnitude speedups over traditional solvers. However, GNSs typically assume access to the underlying material parameters, such as stiffness or viscosity, severely limiting their utility in realistic experimental settings. While recent meta-learning approaches address the parameter dependency by inferring properties from mesh trajectories, reconstructing a mesh from an observed scene is challenging. In this work, we introduce Point Cloud Encoding for Accurate Context Handling (*PEACH*), a novel framework that applies in-context learning on point clouds to adapt a learned simulator to unseen physical properties during inference. Our approach relies on a novel spatio-temporal point cloud sequence encoder, as well as two forms of auxiliary supervision to help improve simulation fidelity. We demonstrate that *PEACH* is capable of accurate zero-shot sim-to-real transfer on a challenging, dynamic scene. Experiments on simulation scenes show that *PEACH* even outperforms mesh-based baselines on prediction accuracy, while being much more practical for real-world deployment.

1 Introduction

Physics simulation is central to engineering disciplines ranging from structural mechanics to robotics, yet classical solvers based on the finite element method are computationally expensive [1, 2]. Graph Network Simulators (GNSs) have emerged as fast and differentiable surrogates for mesh-based physics, offering orders-of-magnitude speedups over classical solvers [3–7]. However, standard GNSs assume known process conditions, resulting in a parameter estimation problem when physical parameters change. Classical system identification can recover such parameters, but doing so per-instance multiplies the experimental and computational cost in ways that scale poorly [8–13]. Test-time gradient inversion through a differentiable simulator is one response, but reintroduces a per-instance optimization loop and there is no guarantee that the loss objective for inversion is well-conditioned [14, 15]. In-context conditioning instead amortizes identification by encoding a set of context trajectories into a latent material descriptor, enabling rapid adaptation to new material configurations at inference time [16]. In this work, we formulate adaptive physical simulation as a real-to-sim in-context learning problem, in which a novel spatio-temporal encoder infers a latent

*Correspondence to philipp.dahlinger@kit.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

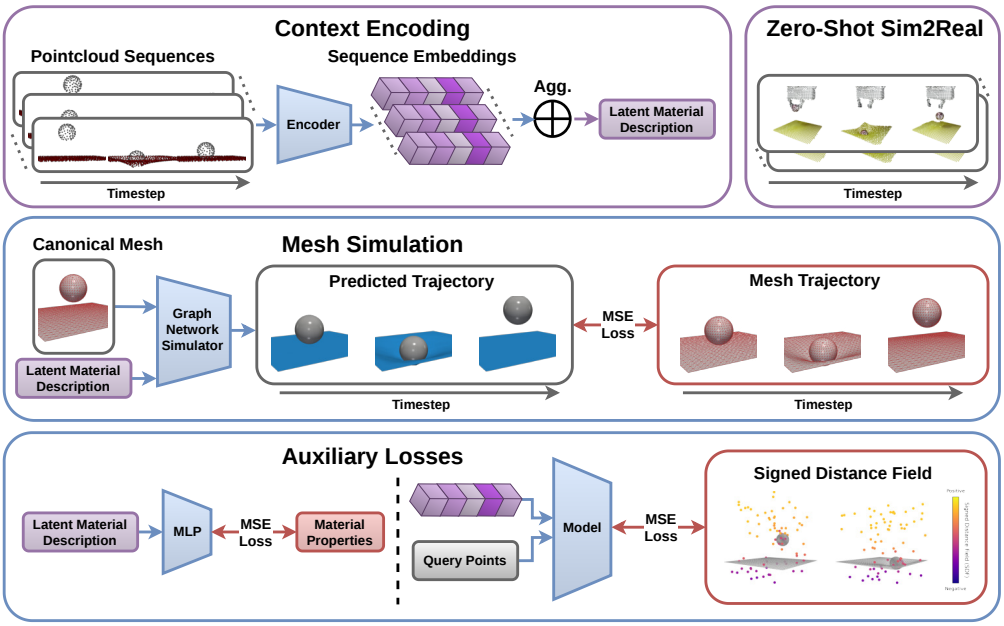


Figure 1: Overview of the Point Cloud Encoding for Accurate Context Handling (*PEACH*) framework. Point cloud sequences from several simulated or real trajectories are encoded into a single latent material description via our novel spatio-temporal encoder architecture. The aggregated latent vector conditions a mesh-based simulator, which predicts complete trajectories from an initial mesh. To improve accuracy, we apply auxiliary losses based on the ground truth physical properties and the Signed Distance Field (SDF). During inference, only the context point cloud sequences and an initial geometry are needed to simulate behavior with novel, unseen material properties.

description of the unknown physical properties from point cloud observations and provides it as context to the simulator.

Operating on meshes instead of points [17, 18] helps GNSs generate stable and more accurate predictions over long sequences. However, mesh trajectories of arbitrary deforming objects are not directly observable with standard sensing; reconstructing them requires either invasive marker setups or template registration that itself depends on geometric and constitutive priors [19, 20]. Point clouds, in contrast, are produced directly by depth cameras and LiDAR and capture 3D geometry of deforming objects. Because they rely on geometric structure rather than surface appearance, sim-to-real transfer is less constrained by rendering fidelity [21–24]. While point cloud observations can easily be generated from meshes, point clouds from the real world do not contain correspondences between points from subsequent time steps. This complicates the estimation of physical parameters, since the motion of individual points contains the most information about those parameters. Unlike in mesh-based approaches, these correspondences must be inferred indirectly.

To address these challenges, we introduce Point Cloud Encoding for Accurate Context Handling (*PEACH*), an in-context framework that conditions a graph network simulator on a small set of observed point cloud sequences. This framework is able to predict mesh-level dynamics for unseen materials, without any test-time parameter optimization. *PEACH* extends the conditional neural process framework [25] to spatio-temporal observations: a spatio-temporal encoder maps each point cloud sequence to a latent vector, and the resulting set is aggregated into a single material descriptor that conditions the simulator. Our novel encoder architecture extends the common point-patch paradigm [26–28] by treating a sequence as a single point cloud embedded in 4D space-time. By relying on proximity in 4D space-time, the network can extract physical properties from the observation without explicit point-to-point correspondences. While this architecture is already competitive, we found in preliminary experiments that additional auxiliary losses can further improve the results. We investigate two auxiliary losses applied during training only: a regression head that grounds the latent embedding in the ground truth physical parameters, and a Signed Distance Field (SDF) reconstruction loss that encourages the encoder to retain fine-grained geometric information

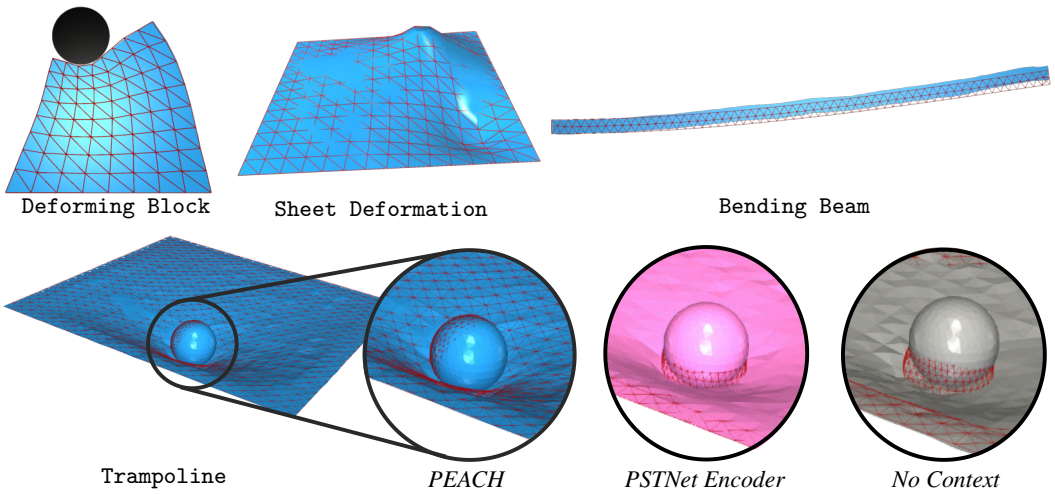


Figure 2: Overview of the four simulation scenes. The environments cover a range of deformable object types and material models, from purely elastic to viscoelastic constitutive behaviors, with varying material properties across tasks. Qualitative predictions of *PEACH* and selected baselines are shown in blue, pink, and gray, respectively, with ground-truth meshes overlaid in red.

in the earlier layers. For inference, the model receives an initial mesh of the canonical geometry, which is typically available from CAD or a one-time scan, and 1 to 8 point cloud sequences from the target process. *PEACH* predicts the remaining mesh simulation using the implicit physical properties defined by the context.

To summarize, our contributions are as follows: (i) We propose *PEACH*, an end-to-end architecture that efficiently and accurately simulates a mesh-based trajectory conditioned on a handful of observed point clouds, without test-time optimization. As a key benefit, our novel spatio-temporal point cloud encoder is applicable to any operation on point cloud sequences due to its generality. (ii) We demonstrate *PEACH*'s sim-to-real capability through zero-shot, rapid adaptation of a real-world scene in which a robot drops a ball onto a rubber trampoline, a setting beyond the reach of mesh-based methods. We additionally validate on four simulation scenes featuring deformable objects and outperform baselines with access to the ground truth context mesh. (iii) Through parameter studies, we provide insights that underscore the necessity of auxiliary losses for parameter regression and SDF. Furthermore, we visualize the latent space and show statistically that it converges with increasing context size at a rate close to averaging independent observations, and that distances between tasks in the latent space strongly correlate with distances between their physical parameters.

2 Related Work

System Identification. Physical parameters such as stiffness or relaxation behavior are rarely known precisely for real-world scenes, which limits the use of Partial Differential Equation (PDE) simulators in many engineering applications [29–33]. System identification [8, 9] recovers such parameters from indirect observations. In applied mechanics, related approaches include finite element model updating and the Virtual Fields Method [34–36]. Recent approaches pair learned models with physics engines or differentiable simulators for property estimation and control [11–13], and train surrogate models for PDE-constrained parameter inference [14, 15, 37].

A different line of robotic real-to-sim work recovers deformable-object parameters via a per-instance test-time optimization. These works range from iterative FEM-comparison [38] to differentiable rendering and simulation [39] to differentiable-simulator system identification on point clouds [40, 37]. Each instance requires a fresh optimization, which is typically done iteratively. The objective is non-convex and exhibits local minima and sloppy parameter directions [41, 42]. Direct-inversion alternatives sidestep the non-convexity but require dense full-field strain measurements [34] that are unavailable in typical real-world settings. *PEACH* amortizes identification into a learned encoder, replacing per-instance optimization with a single forward pass, and we compare against a test-time optimization baseline in Section 4.

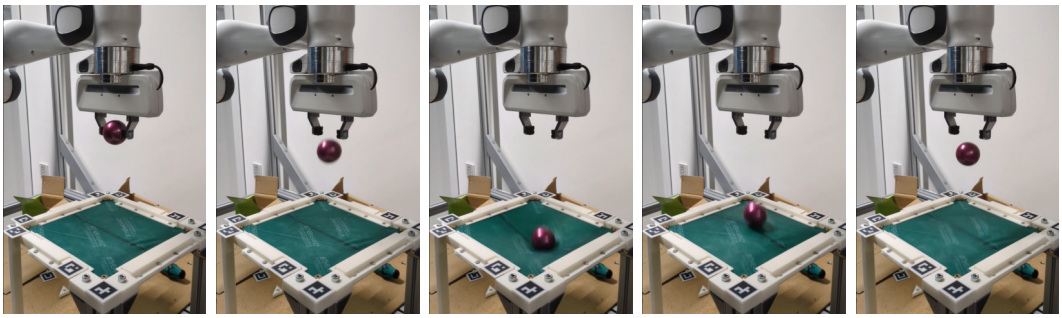


Figure 3: Real-world `Trampoline` setup. A robot arm releases a steel ball above an elastic membrane stretched across a 3D-printed frame, with ArUco markers at the corners for calibration.

Adapting Learned Simulators to New Materials. Graph network simulators have emerged as fast, differentiable surrogates for mesh- and particle-based physics, enabling scalable simulation of physical systems of rigid bodies [3, 43], deformable solids [4], and PDE-like dynamics on unstructured meshes [5]. Recent extensions aim to capture long-range interactions in nonlinear solid mechanics via hierarchical architectures [44, 45, 7]. These methods assume known process conditions and thus inherit the parameter-estimation problem whenever materials change. Recent work therefore conditions learned simulators on material parameters either via parameter-efficient fine-tuning [46] or via meta-learning over a context set of trajectories [47, 48].

In particular, MaNGO [16] encodes a context set of mesh trajectories into a latent material descriptor that conditions a trajectory-level GNS, enabling adaptation to unseen materials in a single forward pass without test-time optimization. This is achieved via a Conditional Neural Process (CNP) formulation for in-context learning [25]. However, mesh trajectories of arbitrarily deforming objects are not directly observable with real-world sensors, limiting MaNGO to synthetic data regimes. Instead, we condition on point cloud sequences, enabling adaptive physical simulation of real scenes with readily available 3D cameras. Recent advances in foundation models for stereo reconstruction further support this approach by enabling the generation of cleaner and less noisy point clouds [49].

Point Cloud Sequence Observations for Learned Simulators. While permutation-invariant set encoders for individual point clouds [50, 51, 26–28] are well-established, processing point cloud sequences requires architectures that handle spatial irregularity and frame-to-frame motion without explicit point correspondences [52, 53]. Within learned simulation, point clouds have been used either to ground mesh-based GNSs in sensor observations [6], or as the basis of particle-based dynamics models that operate directly in observation space for deformable-object manipulation [54, 17]. In both cases, the latent physical properties are implicitly learned in the dynamics model, making adapting to new conditions less data efficient. Instant Policy [55] instead conditions imitation learning policies on point cloud demonstrations via amortized in-context learning. The method shares PEACH's conditioning paradigm but applies to robot policies rather than material parameters and operates without a downstream physics simulator.

3 Point Cloud Encoding for Accurate Context Handling

Problem Setting. We consider a family of physical systems that share the same governing equations but differ in their physical properties, such as stiffness, damping, or thickness. The goal is to learn a simulator that mimics a Finite Element Method (FEM) solver and predicts a sequence of mesh states $\hat{\mathcal{G}}_{1:T}$ at discrete frames $1, \ldots, T$, without explicit knowledge of the physical properties ρ.

This setup is common in industrial applications, where mesh-based representations are commonly available or can be obtained through established reconstruction pipelines. In our work, each *task* corresponds to a fixed configuration of physical properties ρ, where the initial conditions such as mesh geometry or external forces may vary. For each initial condition, we generate a ground truth mesh trajectory using an FEM solver. By ray-tracing the mesh states with a virtual camera, we subsequently create an observed point cloud trajectory for each mesh trajectory. At inference, we wish to adapt our learned simulator to the given task, conditioned only on a small context of observed point cloud trajectories.

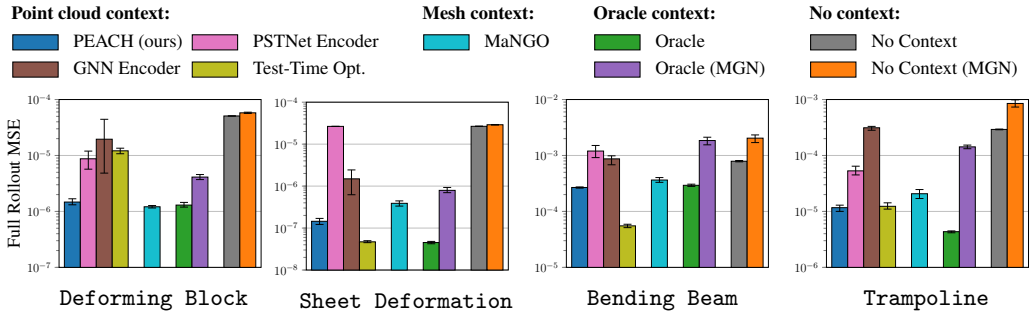


Figure 4: Simulation accuracy (MSE) on simulation datasets using 8 context trajectories. *PEACH* consistently outperforms other methods relying on point cloud observations for simulator adaptation. *PEACH* performs comparably to *MaNGO*, which uses privileged information in the form of meshes, and sometimes achieves performance competitive with the Oracle models.

Framework Overview. During training, we sample a subset of C point cloud trajectories from one task, where each trajectory $\mathcal{P}_i \in \mathbb{R}^{T\times N\times 3}$ consists of T frames of N points in 3D space. We encode each trajectory using a point cloud sequence encoder to obtain latent codes $\{\mathbf{r}_1, \ldots, \mathbf{r}_C\}$, where each $\mathbf{r}_c$ is a $d_{\mathbf{r}}$-dimensional vector describing the physical properties from this trajectory in latent space. The per-sequence latent codes are further aggregated into a single task-level latent embedding $\mathbf{r}$ of the physical properties via a learned softmax aggregation, enabling the model to handle a variable number of context trajectories. Concretely, given per-sequence latent codes $\{\mathbf{r}_1, \ldots, \mathbf{r}_C\}$, the aggregated embedding is computed as

$$\mathbf{r} = \sum_{c=1}^{C} \boldsymbol{\alpha}_c \odot \mathbf{r}_c, \quad \boldsymbol{\alpha}_c = \frac{\exp(\beta \mathbf{r}_c)}{\sum_{c'=1}^{C} \exp(\beta \mathbf{r}_{c'})}, \tag{1}$$

where β is a learnable inverse-temperature parameter and the softmax is applied element-wise across the context dimension. Together with an initial mesh state $\mathcal{G}_0$ sampled from the same task but independent of the context trajectories, $\mathbf{r}$ is passed to a differentiable mesh-based simulator to predict the remaining trajectory $\hat{\mathcal{G}}_{1:T}$, supervised against the ground truth $\mathcal{G}_{1:T}$. The simulator is realized as a GNS. Gradients are propagated end-to-end through both the simulator and the point cloud encoder, ensuring that the learned physical representation is optimized for predictive accuracy.

At inference time, the material parameters ρ are unknown and only the mesh state $\mathcal{G}_0$ of the initial configuration is available. The model must therefore rely entirely on the learned context encoding to estimate the material properties from a given sequence of context point clouds and simulate the resulting deformation. As we show in our experiments, the model is capable of generalizing to unseen physical properties in simulation and even to real data. Note that no real-world data is used for training, as neither ground truth mesh trajectories nor the true physical properties ρ are available. We next describe the graph network simulator used as the dynamics backbone of our framework.

Graph Network Simulators. Unlike step-based simulators [4, 7], which accumulate errors during rollout, trajectory-based simulators [16, 48, 56] predict the entire remaining trajectory in a single forward pass. *PEACH* utilizes the trajectory-based Meta Neural Graph Operator (*MaNGO*) simulator [16], which represents the full trajectory as a single static graph with temporally indexed node and edge features. Each layer interleaves spatial message passing within a time step with a 1D residual Convolutional Neural Network (CNN) over time, and the latent material description is broadcast to all nodes and time steps, so that it conditions the entire predicted trajectory. The simulator is trained with per-timestep Mean Squared Error (MSE) losses over the full rollout (Appendix B.3).

Spatio-Temporal Point Cloud Encoder. The core challenge of the point cloud encoder is to compress a large number of spatially dense but informationally sparse points into a compact and expressive latent representation. Furthermore, there are numerous ways to treat a set of points across both space and time coordinates. While typical point-patch style encoders [26–28] are effective for tokenizing individual point clouds, material properties reveal themselves most clearly in the dynamics of the scene rather than in any single time step. Therefore, we adapt this paradigm to point cloud sequences and develop a novel encoder architecture by considering points in 4D space-time.

Space-time coordinates are computed by normalizing the sequence duration and applying a scaling factor of τ. Setting $\tau \to 0$ has the effect of ignoring the time axis, while $\tau \to \infty$ recovers the case where each time step is encoded independently of all others. We sample M patch centers using Farthest Point Sampling (FPS) [51] and extract patches using K-Nearest Neighbors (KNN), both over space-time coordinates. We compute the features of each point using a Fourier feature projection [57, 58] of its space-time coordinates $\gamma : \mathbb{R} \to \mathbb{R}^{2L}$ with $\gamma_k(x) = [\sin(2\pi x/\lambda_k), \cos(2\pi x/\lambda_k)]^\top$ for $k = 1, \ldots, L$, where L is the number of frequency bands and the wavelengths $\lambda_k = \lambda_{\text{max}}(\lambda_{\text{min}}/\lambda_{\text{max}})^{(k-1)/(L-1)}$ are spaced geometrically between λ_{max} and λ_{min}. The Fourier-encoded point features are concatenated with an additional object type feature, describing the object to which each point belongs. We use the PointMAE tokenizer [26] to get a patch embedding $\mathbf{s} \in \mathbb{R}^d$ from the point features of each patch, which is similar to PointNet [50] and uses max-pooling to aggregate over the patch. The point features of the patch center are passed through another Multilayer Perceptron (MLP) to yield a patch positional embedding $\mathbf{c} \in \mathbb{R}^d$, which is added to the patch embeddings to obtain tokens $\mathbf{t} = \mathbf{s} + \mathbf{c}$.

The M tokens $\{\mathbf{t}_1, \ldots, \mathbf{t}_M\}$ are passed into a transformer encoder, which refines them via self-attention. A final attention pooling layer compresses them into a single output token $\mathbf{r}_i$ for the point cloud sequence $\mathcal{P}_i$. In contrast to simple mean or max pooling, the attention pooling layer uses a learned query token to attend over the transformer outputs, allowing the model to selectively focus on the most informative spatio-temporal regions. We found that a lightweight architecture with a single transformer layer suffices and outperforms deeper variants, suggesting that the tokenizer already captures most of the relevant structure and only light global reasoning is needed on top.

Auxiliary Losses. To encourage the latent representation to capture physically meaningful material properties and stabilize training, we introduce two auxiliary training objectives. As in *MaNGO*, we supervise the trajectory-level latent material description $\mathbf{r}$ using ground-truth physical properties through a lightweight decoder $\hat{\rho} = \text{MLP}(\mathbf{r})$. The predicted $\hat{\rho}$ can be interpreted as an explicit system identification of the current task.

We additionally supervise the patch tokenizer using an SDF objective inspired by Vosylius and Johns [55]. An SDF $f : \mathbb{R}^3 \to \mathbb{R}$ assigns to each point $\mathbf{p}$ its signed distance to the nearest mesh surface, with positive values outside and negative values inside. For 2D sheet-like geometries we treat the underside as interior. The distances are scaled and passed through a $\tanh$ nonlinearity to obtain normalized targets $s = \tanh(\alpha\, f(\mathbf{p}))$, which act similarly to truncation [59] and smoothly approach an occupancy function [60] as $\alpha \to \infty$. For each trajectory and time step, we sample query points as visualized in the bottom right of Figure 1. For each query point, we find its K_{SDF} nearest patch tokens in 4D space-time and Fourier-encode their relative positions. These encodings are concatenated with the token embeddings and processed by an attention-based MLP that predicts a scalar weight per neighbor, biased by the normalized distance to emphasize spatio-temporal locality. A weighted aggregation of the features is decoded by a small MLP into a predicted signed distance, which is supervised against the ground-truth s using a smooth ℓ_1 loss.

Data Augmentation. To bridge the gap between simulated training data and real-world sensor inputs, we perturb simulated point cloud sequences during training to mimic typical artifacts observed in real-world data. These augmentations include multi-scale Gaussian noise to model sensor uncertainty and structured region dropout [61] to simulate occlusions or missing observations. To account for sensor failures and reconstruction errors, we additionally inject sporadic point artifacts by sampling a small number of points uniformly from the scene's bounding box.

4 Experiments and Results

Scenes. We evaluate the sim-to-real capability of *PEACH* on a novel scene called `Trampoline` that involves dynamic motion, complex contact interactions, and deformable objects. Additionally, we introduce the novel simulation-only dataset `Bending Beam` inspired by Würth et al. [7], and adopt two existing datasets, `Deforming Block` and `Sheet Deformation` [16][2]. These scenes span 2D and 3D domains, and are inspired by simulation-assisted process optimization in industrial contexts. They feature varying physical parameters such as Young's modulus (all except `Deforming Block`), Poisson's ratio (all except `Sheet Deformation`), membrane thickness, shear relaxation ratio (`Trampoline`), and relaxation time (`Bending Beam` and `Trampoline`). Rollout length varies by scene from 25 (`Trampoline`) to 100 (`Bending Beam`). Although the `Trampoline` scene has a

[2] Called `Deformable Plate` (*easy*) and `Planar Bending`, respectively, in Dahlinger et al. [16].

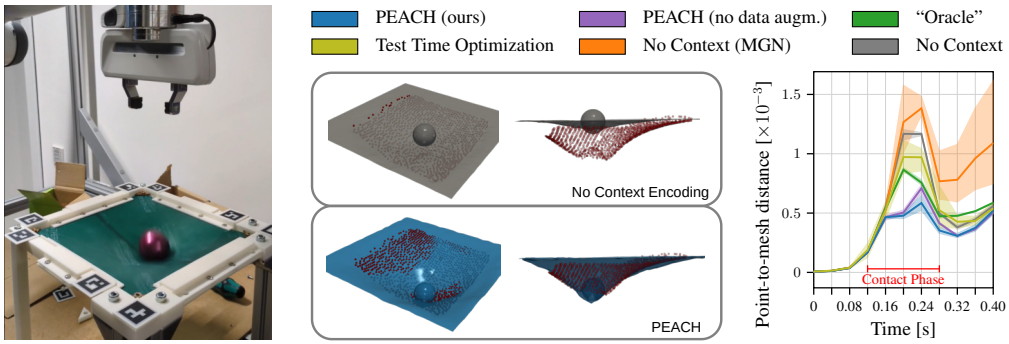


Figure 5: Real-world `Trampoline` scene. **Left:** Physical setup with a ball bouncing on a latex sheet. **Center:** Qualitative comparison at t=0.2 s (maximal sheet stretching) between *PEACH* and a context-free simulator. The red point cloud shows the ground truth observations. The bottom of the sheet is occluded by the ball and thus absent from the point cloud. **Right:** Mean point-to-mesh distance over all real-world trials, plotted over time, with ball-sheet contact occurring in $[0.12, 0.28]$ s.

shorter horizon, it has the most complex geometry, with roughly 10 times more triangles than other scenes. A visual overview of the scenes is provided in Figure 2, with further details in Appendix C.

In simulation, we obtain per-step point clouds by ray casting from a static camera and applying FPS [51] to retain 512 points; due to the finer geometry `Trampoline` uses 1024 points. To reduce the point cloud sequence length, we drop every 2nd time step. In `Trampoline` and `Deforming Block`, collider points are explicitly marked. For the real-world `Trampoline` dataset, we use a robot to drop balls of varying density onto latex sheets of different thicknesses and record depth images with a ZED Mini Stereo Camera (Stereolabs). Neither meshes nor ground truth physical parameters are known for these recordings. For the "*Oracle*" baseline, measurable material properties are determined experimentally while the remaining parameters are chosen based on informed estimates from literature values. Appendix D details the real-world data.

Baselines. We compare *PEACH* against methods using 4 different context types, namely point cloud contexts, mesh contexts, oracle, and no context. *Oracle* methods receive the ground truth physical properties at every time step as additional node features. *No Context* methods predict trajectories solely from the initial state without additional task information. Methods using point cloud contexts infer the task from the same context point cloud sequences as *PEACH*. *PSTNet Encoder* [53] and *GNN Encoder* differ from *PEACH* only in the encoder architecture. *PSTNet Encoder* captures local geometric evolution via space-time tubes anchored at FPS-sampled reference points, applying decoupled spatial and temporal convolutions hierarchically across the point cloud sequence. *GNN Encoder* adapts the graph network paradigm from mesh-based simulation to point cloud sequences. It tokenizes each timestep's point cloud into local patch embeddings, constructs a spatio-temporal graph connecting tokens within and across adjacent frames, and processes it via message passing with a final readout into a trajectory-level latent code. *Test-Time Optimization* instead follows the classical system identification paradigm. Starting from the mean training parameters, it optimizes the physical parameters of the trained *Oracle* model by gradient descent through the simulator, minimizing the distance between the observed context point clouds and the predicted meshes.

MaNGO [16] instead computes the latent embedding using privileged mesh contexts. Since such contexts are not readily available in real-world settings, *MaNGO* is not directly comparable to our method, but demonstrates the impact of replacing explicit mesh observations with point clouds in simulated tasks. All baselines that predict a latent embedding also receive auxiliary supervision via ground truth physical properties.

Lastly, we study the impact of the simulator component by replacing the MaNGO simulator with the MeshGraphNet (MGN) simulator [4], a mesh-based autoregressive simulator trained for one-step prediction. We instantiate the MGN baseline in *Oracle (MGN)* and *No Context (MGN)* variants. While it is possible to integrate MGN into the *PEACH* framework, early experiments indicated that it was outperformed by the MaNGO simulator. Appendix B details all baselines.

Training. We train each method end-to-end. All in-context methods are trained using batches of 1 to 8 randomly sampled context trajectories to improve adaptation capability. We repeat each experiment

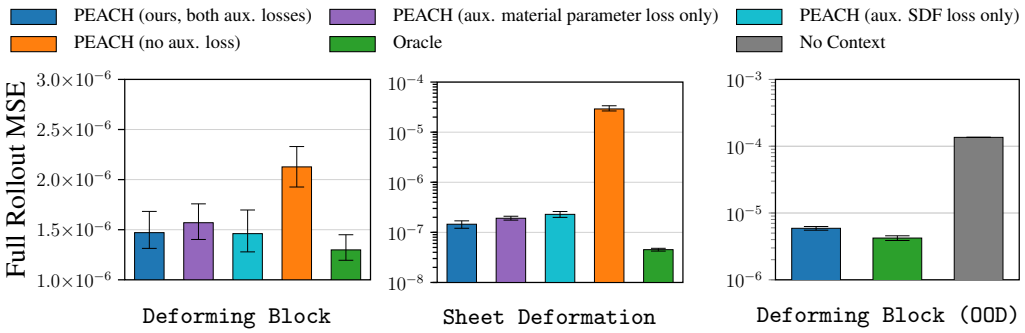


Figure 6: **Left, Center:** Ablation study of auxiliary losses in PEACH on the `Deforming Block` and `Sheet Deformation` environments, respectively. **Right:** Performance of PEACH on out-of-distribution test tasks for `Deforming Block`.

across 4 random seeds using the AdamW optimizer [62] with a learning rate of 5.0×10^{-5}. All methods were trained on either an A100 or H100 GPU for 2 days, with the exception of `Bending Beam`, which converged after 1 day. Appendix B.5 lists hyperparameters, and Appendix B.4 provides parameter counts, inference runtimes, and an analysis of how the encoding time scales with context size, point count, and trajectory length.

Simulation Results. We evaluate the simulation fidelity using the MSE averaged across all steps of a complete rollout, on a test split whose physical properties and initial conditions are unseen during training. Figure 4 shows that *PEACH* outperforms the other learned point cloud encoders by a substantial margin. It also outperforms the mesh-based *MaNGO* on three out of four environments despite its privileged mesh context, which we attribute to the limited capacity of the shallow CNN and Deep Sets encoder of *MaNGO*. *Test-Time Optimization* is considerably less accurate than *PEACH* on `Deforming Block` and on par on `Trampoline`, but matches the *Oracle* on `Sheet Deformation` and achieves the lowest error on `Bending Beam`, at about 140 times the inference time of *PEACH* (Appendix B.4). On `Bending Beam`, both *Test-Time Optimization* and *PEACH* surpass the *Oracle*. We attribute this to approximation errors of the learned simulator. Since both methods infer their conditioning from observed deformations, they can compensate for such errors by representing an effective stiffness that reproduces the observed bending, even if it deviates from the true material parameters, whereas the *Oracle* is bound to the ground truth parameters. *Test-Time Optimization* performs this adaptation explicitly for each task, which explains its larger gain on the clean simulated observations. Replacing the simulator with the step-based MGN generally reduces accuracy, even with oracle information. Figure 2 shows qualitative predictions of *PEACH* across all four scenes, and Appendix G provides additional qualitative results.

Real World Results. Figure 3 shows an example trajectory of our real-world `Trampoline` setup. A robot arm releases a ball of unknown size and weight above an elastic membrane of varying thickness. Figure 5 demonstrates that *PEACH* is able to adapt to contexts from this real world setup without ever having seen real-world point clouds during training. Qualitatively, *PEACH* adapts to the interaction dynamics determined by the ball's size and weight and the rubber sheet's thickness, accurately capturing the sheet deformation at peak elongation. In contrast, the *No Context Encoding* baseline regresses to the mean, predicting only minimal deformation regardless of the physical parameters.

Quantitatively, *PEACH* outperforms all baselines, including the *Oracle* with material parameters measured directly from the physical setup and *Test-Time Optimization*, whose accuracy degrades on the noisy and partially occluded real-world observations. All methods perform well prior to ball-sheet contact, having learned to accurately predict free-fall dynamics, but baselines deteriorate significantly during the contact phase. Toward the end of the trajectory, prediction error increases across all methods, which we attribute to the inherently chaotic post-bounce behavior of the ball, whose direction after impact is difficult to predict deterministically. Combined, these results indicate our method's ability to accurately simulate real-world scenes from only point cloud observations.

Parameter Study. We test the extrapolation capability of *PEACH* using an additional, out-of-distribution set of `Deforming Block` tasks in simulation. As illustrated in Figure 10 of the Appendix, test tasks are drawn from the tails of the Poisson ratio distribution, ensuring that the evaluated material properties lie outside the training range. Figure 6 (right) shows that *PEACH* remains close to the

oracle performance even under this distribution shift, demonstrating that the learned latent material representation generalizes beyond the training distribution. A general analysis of performance as a function of context size is provided in Figure 12 in Appendix E.

Ablations. Figure 6 (left, center) presents an ablation of the auxiliary losses on `Deforming Block` and `Sheet Deformation`. Removing both auxiliary losses leads to a substantial performance drop, confirming that auxiliary supervision is critical for learning a useful latent material representation. While *PEACH* benefits from combining both losses, training with only the SDF auxiliary loss remains competitive. Since SDF labels can be computed purely from mesh geometry, this is a practical option for settings where ground truth physical properties are unavailable. To test robustness to sensor artifacts, we corrupt the real-world context point clouds by removing 1%, 9%, or 25% of the points in small patches and adding the same number of spurious points (Figure 11 in Appendix D.1). *PEACH* remains nearly unaffected by all corruption levels and stays more accurate than the *Oracle* during the contact phase. Without data augmentation, its accuracy drops slightly on clean observations (Figure 5, right) and considerably under high corruption, approaching the *No Context* baseline at peak contact.

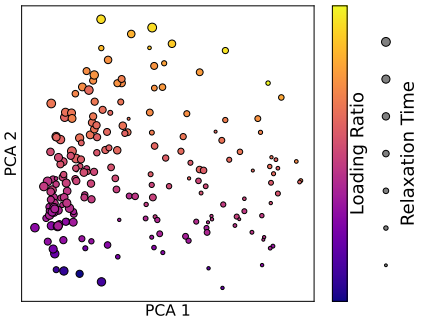


Figure 7: Visualization of the first and second PCA component of the latent space on the `Trampoline` dataset. Each point represents one simulation test task. Color corresponds to ratio of mass to product of Young's modulus and sheet thickness, i.e., loading ratio, which governs the degree of sheet deformation. Size corresponds to relaxation time.

Latent Space Analysis. Figure 7 visualizes the PCA projection of the latent material codes encoded by *PEACH* on the `Trampoline` dataset. Despite no explicit structure being imposed on the latent space, the projection reveals a clear organization: the first principal component separates tasks by relaxation time, while the second separates them by loading ratio. This emergent disentanglement of physically meaningful material properties suggests that *PEACH* learns structured representations purely from point cloud observations. Latent space visualizations for all environments are provided in Figure 15 of the Appendix. A quantitative analysis in Appendix F further shows that the variance of the latent description decreases with the context size at a rate close to averaging independent observations, and that distances between tasks in the latent space correlate strongly with distances between their physical parameters.

5 Conclusion

We present Point Cloud Encoding for Accurate Context Handling (*PEACH*), an in-context framework that conditions a graph network simulator on a handful of point cloud sequences to predict mesh-based dynamics for unseen materials. By conditioning on point clouds instead of, e.g., meshes, context data can be obtained easily from real-world scenes. This allows *PEACH* to simulate real-world scenes without explicit parameter estimation or test-time optimization, and we demonstrate zero-shot sim-to-real transfer on a challenging trampoline scene. Across four simulated datasets, *PEACH* outperforms all learned point cloud encoders and performs on par with the mesh-based *MaNGO*, while being about 140 times faster than test-time optimization. Crucial to our method are our novel spatio-temporal encoder that treats context sequences as point clouds in 4D space-time, and auxiliary supervision in the form of direct parameter regression and prediction of SDF values.

Limitations and future work. *PEACH* assumes access to an initial scene geometry and requires context trajectories that share physical properties with the target process. Generating an initial mesh from observation would relax the geometry assumption, while accumulating context online during, e.g., robot execution would handle material drift across a process. Additionally, *PEACH* currently predicts the full trajectory in a single forward pass, which may impose memory constraints for long sequences and high-resolution meshes. Predicting sparse keyframes and training a separate network to interpolate between them is a promising way to decouple memory cost from horizon length. Further extensions could include other sensor modalities such as temperature readings and touch sensors for more accurate parameter estimation. We discuss broader impact in Appendix A.

Acknowledgments

This work is part of the DFG AI Research Unit 5339 regarding the combination of physics-based simulation with AI-based methodologies for the fast maturation of manufacturing processes. The financial support by the German Research Foundation (DFG, *Deutsche Forschungsgemeinschaft*) is gratefully acknowledged.

The work has been financially supported by the German Research Foundation (DFG, *Deutsche Forschungsgemeinschaft*) as part of the SFB-1574 – 471687386 "Circular Factory", and by the European Research Council (ERC) under the European Union's Horizon Europe programme through the project SMARTI^3 (Grant Agreement No. 101171393).

The authors acknowledge support by the state of Baden-Württemberg through bwHPC, as well as the HoreKa supercomputer funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the German Federal Ministry of Education and Research.

The authors thank the Gauss Centre for Supercomputing e.V. (GCS) for providing computing time on the Supercomputer JUWELS [63] at the Jülich Supercomputing Centre via the John von Neumann Institute for Computing. JUWELS is supported by GCS through funding by the German Federal Ministry of Research, Technology and Space and the Ministry of Culture and Science of the State of North Rhine-Westphalia.

The authors gratefully acknowledge the computing time provided on the high-performance computer Lichtenberg II at TU Darmstadt, funded by the German Federal Ministry of Research, Technology, and Space (BMFTR), and the State of Hesse Ministry of Science and Research, Art and Culture (HMWK).

This work has been supported by the German Federal Ministry of Research, Technology, and Space (BMFTR) under the Robotics Institute Germany (RIG).

References

[1] Olek C Zienkiewicz and Robert Leroy Taylor. *The finite element method for solid and structural mechanics*. Elsevier, 2005.

[2] Eva Stanova, Gabriel Fedorko, Stanislav Kmet, Vieroslav Molnar, and Michal Fabian. Finite element analysis of spiral strands with different shapes subjected to axial loads. *Advances in engineering software*, 83:45–58, 2015.

[3] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

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

[5] Johannes Brandstetter, Daniel E Worrall, and Max Welling. Message passing neural pde solvers. In *International Conference on Learning Representations*, 2022.

[6] Jonas Linkerhägner, Niklas Freymuth, Paul Maria Scheikl, Franziska Mathis-Ullrich, and Gerhard Neumann. Grounding graph network simulators using physical sensor observations. In *The Eleventh International Conference on Learning Representations*, 2023.

[7] Tobias Würth, Niklas Freymuth, Gerhard Neumann, and Luise Kärger. Diffusion-based hierarchical graph neural networks for simulating nonlinear solid mechanics. *Advances in Neural Information Processing Systems*, 39, 2025.

[8] Karl Johan Åström and Peter Eykhoff. System identification—a survey. *Automatica*, 7(2): 123–162, 1971.

[9] Ljung Lennart. System identification: theory for the user. *PTR Prentice Hall, Upper Saddle River, NJ*, 28:540, 1999.

[10] Victor Isakov. *Inverse problems for partial differential equations*, volume 127. Springer, 2006.

[11] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. *Advances in neural information processing systems*, 28, 2015.

[12] 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.

[13] Rika Antonova, Jingyun Yang, Krishna Murthy Jatavallabhula, and Jeannette Bohg. Rethinking optimization with differentiable simulation from a global perspective. In *Conference on robot learning*, pages 276–286. PMLR, 2023.

[14] Qingqing Zhao, David B. Lindell, and Gordon Wetzstein. Learning to solve PDE-constrained inverse problems with graph networks. In *International Conference on Machine Learning*, pages 26895–26910. PMLR, 2022.

[15] Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse PDE problems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=VLw8ZyKfcm.

[16] Philipp Dahlinger, Tai Hoang, Denis Blessing, Niklas Freymuth, and Gerhard Neumann. Mango—adaptable graph network simulators via meta-learning. *Advances in Neural Information Processing Systems*, 39, 2025.

[17] Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, Proceedings of Machine Learning Research, pages 642–660. PMLR, 2023. URL https://proceedings.mlr.press/v229/shi23a.html.

[18] William F Whitney, Jake Varley, Deepali Jain, Krzysztof Marcin Choromanski, Sumeet Singh, and Vikas Sindhwani. Modeling the real world with high-density visual particle dynamics. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=pcPSGZFaCH.

[19] Elham Amin Mansour, Hehui Zheng, and Robert K. Katzschmann. Fast point cloud to mesh reconstruction for deformable object tracking. In Joaquim Filipe and Juha Röning, editors, *Robotics, Computer Vision and Intelligent Systems - 4th International Conference, ROBOVIS 2024, Rome, Italy, February 25-27, 2024, Proceedings*, Communications in Computer and Information Science, pages 391–409. Springer, 2024. doi: 10.1007/978-3-031-59057-3_25. URL https://doi.org/10.1007/978-3-031-59057-3_25.

[20] Matthew Loper, Naureen Mahmood, and Michael J. Black. Mosh: motion and shape capture from sparse markers. *ACM Trans. Graph.*, 33(6):220:1–220:13, 2014. doi: 10.1145/2661229.2661273. URL https://doi.org/10.1145/2661229.2661273.

[21] Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Hao Su, and Xiaolong Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=tJE1Yyi8fUX.

[22] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=gqCQxObVz2.

[23] Albert Wilcox, Mohamed Ghanem, Masoud Moghani, Pierre Barroso, Benjamin Joffe, and Animesh Garg. Adapt3r: Adaptive 3d scene representation for domain transfer in imitation learning. In *9th Annual Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=sUWOSP6SUJ.

[24] Xiaogang Jia, Qian Wang, Anrui Wang, Han A. Wang, Balázs Gyenes, Emiliyan Gospodinov, Xinkai Jiang, Ge Li, Hongyi Zhou, Weiran Liao, Xi Huang, Maximilian Beck, Moritz Reuss, Rudolf Lioutikov, and Gerhard Neumann. Pointmappolicy: Structured point cloud processing for multi-modal imitation learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=ZR2mdBrhJX.

[25] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. *International Conference on Machine Learning*, 2018.

[26] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European Conference on Computer Vision*, pages 604–621. Springer, 2022.

[27] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19313–19322, 2022.

[28] Guangyan Chen, Meiling Wang, Yi Yang, Kai Yu, Li Yuan, and Yufeng Yue. Pointgpt: Auto-regressively generative pre-training from point clouds. *Advances in Neural Information Processing Systems*, 36:29667–29679, 2023.

[29] Lev Davidovich Landau, LP Pitaevskii, Arnold Markovich Kosevich, and Evgenii Mikhailovich Lifshitz. *Theory of elasticity: volume 7*, volume 7. Elsevier, 2012.

[30] Jiri Blazek. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[31] Stjepan Salatovic, Sebastian Krumscheid, Florian Wittemann, and Luise Kärger. Reliable uncertainty quantification for fiber orientation in composite molding processes using multilevel polynomial surrogates. *Probabilistic Engineering Mechanics*, page 103806, 2025.

[32] Stéphane Avril, Sam Evans, et al. *Material parameter identification and inverse problems in soft tissue biomechanics*, volume 573. Springer, 2017.

[33] Boon Xian Chai, Maheshi Gunaratne, Mohammad Ravandi, Jinze Wang, Tharun Dharmawickrema, Adriano Di Pietro, Jiong Jin, and Dimitrios Georgakopoulos. Smart industrial internet of things framework for composites manufacturing. *Sensors*, 24(15):4852, 2024.

[34] Stéphane Avril, Marc Bonnet, Anne-Sophie Bretelle, Michel Grédiac, François Hild, Patrick Ienny, Félix Latourte, Didier Lemosse, Stéphane Pagano, Emmanuel Pagnacco, et al. Overview of identification methods of mechanical parameters based on full-field measurements. *Experimental Mechanics*, 48(4):381–402, 2008.

[35] Fabrice Pierron, Stéphane Avril, and V The Tran. Extension of the virtual fields method to elasto-plastic material identification with cyclic loads and kinematic hardening. *International Journal of Solids and Structures*, 47(22-23):2993–3010, 2010.

[36] Marc Bonnet and Andrei Constantinescu. Inverse problems in elasticity. *Inverse problems*, 21 (2):R1–R50, 2005.

[37] Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Adaptigraph: Material-adaptive graph-based neural dynamics for robotic manipulation. In Dana Kulic, Gentiane Venture, Kostas E. Bekris, and Enrique Coronado, editors, *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, 2024. doi: 10.15607/RSS.2024.XX.010. URL https://doi.org/10.15607/RSS.2024.XX.010.

[38] Barbara Frank, Ruediger Schmedding, Cyrill Stachniss, Matthias Teschner, and Wolfram Burgard. Learning the elasticity parameters of deformable objects with a manipulation robot. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan*, pages 1877–1883. IEEE, 2010. doi: 10.1109/IROS.2010.5653949. URL https://doi.org/10.1109/IROS.2010.5653949.

[39] Priya Sundaresan, Rika Antonova, and Jeannette Bohg. Diffcloud: Real-to-sim from point clouds with differentiable simulation and rendering of deformable objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*, pages 10828–10835. IEEE, 2022. doi: 10.1109/IROS47612.2022.9981101. URL https://doi.org/10.1109/IROS47612.2022.9981101.

[40] 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*, 44(13):2126–2155, 2025.

[41] Mark K Transtrum, Benjamin B Machta, and James P Sethna. Why are nonlinear fits to data so challenging? *Physical review letters*, 104(6):060201, 2010.

[42] Yi Zhang, Albert Van Bael, Antonio Andrade-Campos, and Sam Coppieters. Parameter identifiability analysis: Mitigating the non-uniqueness issue in the inverse identification of an anisotropic yield function. *International journal of solids and Structures*, 243:111543, 2022.

[43] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[44] Youn-Yeol Yu, Jeongwhan Choi, Woojin Cho, Kookjin Lee, Nayong Kim, Kiseok Chang, ChangSeung Woo, ILHO KIM, SeokWoo Lee, Joon Young Yang, et al. Learning flexible body collision dynamics with hierarchical contact mesh transformer. In *The Twelfth International Conference on Learning Representations*, 2024.

[45] Niklas Freymuth, Tobias Würth, Nicolas Schreiber, Balázs Gyenes, Andreas Boltres, Johannes Mitsch, Aleksandar Taranovic, Tai Hoang, Philipp Dahlinger, Philipp Becker, et al. Amber: Adaptive mesh generation by iterative mesh resolution prediction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[46] Naveen Raj Manoharan, Hassan Iqbal, and Krishna Kumar. Parameter-efficient conditioning for material generalization in graph-based simulators. *arXiv preprint arXiv:2511.05456*, 2025.

[47] Ankush Chakrabarty, Gordon Wichern, Vedang M Deshpande, Abraham P Vinod, Karl Berntorp, and Christopher R Laughman. Meta-learning for physically-constrained neural system identification. *Neurocomputing*, page 130945, 2025.

[48] Philipp Dahlinger, Niklas Freymuth, Tai Hoang, Tobias Würth, Michael Volpp, Luise Kärger, and Gerhard Neumann. Context-aware learned mesh-based simulation via trajectory-level meta-learning. *Transactions on Machine Learning Research*, 2025.

[49] Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. Foundationstereo: Zero-shot stereo matching. *arXiv*, 2025.

[50] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. doi: 10.1109/CVPR.2017.16.

[51] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.

[52] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 9245–9254. IEEE, 2019. doi: 10.1109/ICCV.2019.00934. URL https://doi.org/10.1109/ICCV.2019.00934.

[53] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. Pstnet: Point spatio-temporal convolution on point cloud sequences. In *International Conference on Learning Representations*, 2021.

[54] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJgbSn09Ym.

[55] Vitalis Vosylius and Edward Johns. Instant policy: In-context imitation learning via graph diffusion, 2024. URL https://arxiv.org/abs/2411.12633.

[56] Minkai Xu, Jiaqi Han, Aaron Lou, Jean Kossaifi, Arvind Ramanathan, Kamyar Azizzadenesheli, Jure Leskovec, Stefano Ermon, and Anima Anandkumar. Equivariant graph neural operator for modeling 3d dynamics. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=dccRCYmL5x.

[57] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

[58] Balázs Gyenes, Emiliyan Gospodinov, Jan Frieling, Enrico Krohmer, Xiaogang Jia, Nicolas Schreiber, Niklas Freymuth, and Gerhard Neumann. Fourier features let agents learn high precision policies with imitation learning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=y03xxeUkgN.

[59] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[60] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019.

[61] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[62] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[63] Jülich Supercomputing Centre. JUWELS Cluster and Booster: Exascale Pathfinder with Modular Supercomputing Architecture at Juelich Supercomputing Centre. *Journal of large-scale research facilities*, 7(A183), 2021. doi: 10.17815/jlsrf-7-183. URL http://dx.doi.org/10.17815/jlsrf-7-183.

[64] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[65] Chen Cai, Truong Son Hy, Rose Yu, and Yusu Wang. On the connection between mpnn and graph transformer. In *International Conference on Machine Learning*, pages 3408–3430. PMLR, 2023.

[66] François Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, and Stéphane Cotin. SOFA: A Multi-Model Framework for Interactive Physical Simulation. In Yohan Payan, editor, *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, volume 11 of *Studies in Mechanobiology, Tissue Engineering and Biomaterials*, pages 283–321. Springer, June 2012. doi: 10.1007/8415_2012_125. URL https://hal.inria.fr/hal-00681539.

[67] Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. Foundationstereo: Zero-shot stereo matching. *CVPR*, 2025.

[68] Nikolaus Kriegeskorte, Marieke Mur, and Peter A. Bandettini. Representational similarity analysis – connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2:4, 2008.

[69] Nathan Mantel. The detection of disease clustering and a generalized regression approach. *Cancer Research*, 27(2):209–220, 1967.

A Broader Impact

PEACH is a method for simulating deformable objects from point cloud observations, aimed at making physics-based simulation more accessible when material properties are unknown or hard to measure. Efficient real-to-sim transfer of this kind could reduce reliance on costly physical experiments in engineering and scientific workflows and support digital twin construction for monitoring and design. It can additionally lower the barrier to high-fidelity simulation in domains such as biomechanics, materials science, and soft-body engineering, where calibrating constitutive models is traditionally laborious.

At the same time, any improvement in simulation fidelity carries dual-use potential, for instance in the design of weapons or other harmful artifacts. We view *PEACH* as a generic enabling technology in this regard, with no direct path to harm beyond what is already implied by progress in learned simulation more broadly. Like any data-driven model, *PEACH* may also inherit biases from its training distribution. In particular, materials, geometries, and process conditions that are underrepresented during training are likely to be predicted less accurately, which could lead to silent failures if the method is used outside its validated regime. We therefore encourage practitioners to validate *PEACH* on their target domain before relying on its predictions in decision-critical settings.

B Methods and Hyperparameters

B.1 Point Cloud Encoder

In this section, we provide additional details on the point cloud encoders used in our comparisons.

PSTNet. We closely follow the method proposed in [53] and use the official implementation and architecture from `https://github.com/hehefan/Point-Spatio-Temporal-Convolution`. The spatial kernel radius is adapted to 0.1 to match our task normalization, while all other architectural components and hyperparameters are kept unchanged.

GNN Encoder. At each timestep, we perform FPS to select 52 center points per point cloud in the sequence. For each center point, we connect its 16 nearest neighbors and apply a PointNet layer [50] to each resulting patch, yielding an embedding of dimension 32. We concatenate a Fourier encoding of the corresponding timestep to this embedding.

The resulting spatio-temporal tokens are then assembled into a graph: within each timestep, every node is connected to its 4 nearest neighbors. In addition, each node is connected to its 2 nearest neighbors in the subsequent timestep and to its single nearest neighbor in the following timestep. This construction yields a spatio-temporal graph encoding the complete point cloud sequence.

The graph is processed using a message-passing graph neural network [43] without global features, consisting of 5 layers with a latent dimension of 128. Finally, an attention-based readout module is applied to obtain a single output token.

B.2 Test-Time Optimization

Test-Time Optimization represents the classical system identification approach, in which the unknown physical parameters are recovered by inverting a differentiable simulator on the observed data. We use the trained *Oracle* model as the simulator, as it receives the physical parameters ρ directly as input. We optimize the parameters in the same normalized space in which the *Oracle* model receives them, without constraining them to the training range, and initialize them with the mean parameters of the training set.

Objective. In each optimization step, the *Oracle* model predicts the mesh trajectories of all context trajectories given the current parameter estimate $\hat{\rho}$. We compare each predicted mesh with the observed point cloud of the same time step. Let $\mathcal{P}_t$ denote the observed point cloud with N points at time step t, and $\mathcal{F}_t$ the set of triangles of the predicted meshes, including both the deformable object and the collider. The point-to-mesh distance assigns each observed point to its closest triangle and

Table 1: Hyperparameters of Test-Time Optimization. The learning rate is multiplied by the decay factor after every decay interval.

	`Deforming Block`	`Sheet Deformation`	`Bending Beam`	`Trampoline`
Learning rate	0.03	0.03	0.015	0.06
Optimization steps	50	200	200	300
Decay interval (steps)	25	100	100	100
Decay factor	0.5	0.25	0.25	0.25
Reverse term weight λ	1.0	0	0	0

computes the squared distance to the closest point on that triangle as

$$\mathcal{L}_{\text{p2m}}(t) = \frac{1}{N} \sum_{\mathbf{p} \in \mathcal{P}_t} \min_{f \in \mathcal{F}_t} d(\mathbf{p}, f)^2,$$

where $d(\mathbf{p}, f)$ denotes the Euclidean distance between the point $\mathbf{p}$ and the triangle f. This term only requires every observed point to lie close to the predicted surface, but does not penalize predicted surface regions that are not supported by any observation. We therefore optionally add a reverse mesh-to-point term, which computes the squared distance from the centroid $\mathbf{c}_f$ of each triangle to its nearest observed point, weighted by the triangle area a_f, as

$$\mathcal{L}_{\text{m2p}}(t) = \frac{\sum_{f \in \mathcal{F}_t} a_f \min_{\mathbf{p} \in \mathcal{P}_t} \|\mathbf{c}_f - \mathbf{p}\|^2}{\sum_{f \in \mathcal{F}_t} a_f}.$$

The total objective averages both terms over all time steps and all C context trajectories,

$$\mathcal{L}(\hat{\rho}) = \frac{1}{TC} \sum_{c=1}^{C} \sum_{t=1}^{T} \Big(\mathcal{L}_{\text{p2m}}^{(c)}(t) + \lambda \, \mathcal{L}_{\text{m2p}}^{(c)}(t) \Big),$$

with a weight λ for the reverse term. We minimize $\mathcal{L}$ with respect to $\hat{\rho}$ using Adam [64] and a step-wise learning rate decay. After the optimization, the *Oracle* model predicts the target trajectory using the optimized parameters. The reverse term is necessary for `Deforming Block`, where the optimization does not converge to meaningful parameters without it, and we disable it for all other environments. Table 1 lists the hyperparameters, which we tuned per environment. For the real-world recordings, we use the same hyperparameters as for the simulated `Trampoline` environment.

Context size. Backpropagating through full rollouts of multiple context trajectories is memory intensive, so the number of context trajectories is limited by GPU memory for some environments. Additionally, we found that more context trajectories do not consistently improve the result of Test-Time Optimization. We therefore report the best result across the context sizes we could evaluate, which favors this baseline compared to the fixed context size of 8 trajectories used for all other methods.

B.3 Mesh Encoder and Simulator

Mango Mesh Encoder. We closely follow the architecture of [16] and apply a one-dimensional convolutional neural network to each node along the temporal dimension. The mesh structure provides consistent point correspondences over time, enabling this temporal processing. Node features are then aggregated using a Deep Set architecture [65] to produce a permutation-invariant representation of the mesh. We use the official implementation, available at `https://github.com/ALRhub/mango`.

Mango Simulator. We closely follow the implementation from [16], using the official code available at `https://github.com/ALRhub/mango`. Rather than maintaining separate graphs per timestep, *MaNGO* represents the trajectory as a single static graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with temporally indexed node and edge feature tensors $\mathbf{m}_v \in \mathbb{R}^{T \times |\mathcal{V}| \times d_{\text{node}}}$ and $\mathbf{m}_e \in \mathbb{R}^{T \times |\mathcal{E}| \times d_{\text{edge}}}$. The initial node features and trajectory-level conditioning variables, such as latent physical parameters, are broadcast across all time steps and augmented with a learned time embedding per timestep. Edge features are constructed

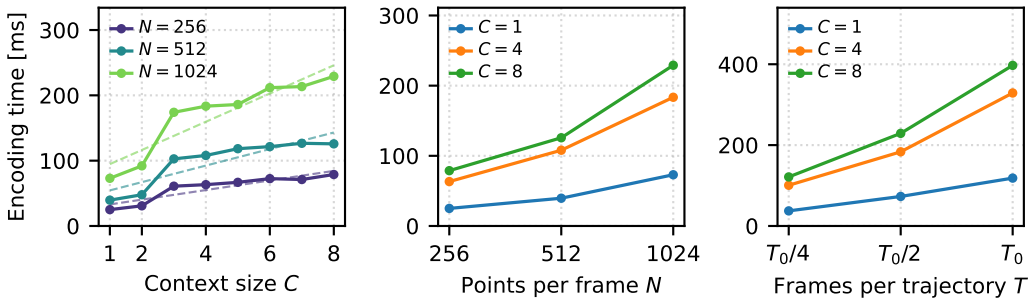


Figure 8: Encoding time of *PEACH* on the `Trampoline` environment. **Left.** Encoding time over the context size C for different numbers of points per frame N, with linear fits shown as dashed lines. **Center.** Encoding time over the number of points per frame. **Right.** Encoding time over the number of frames per trajectory, where T_0 denotes the full trajectory length.

from relative nodal positions. Both are then linearly projected into a latent dimension to obtain the initial feature tensors $\mathbf{m}_v^0$ and $\mathbf{m}_e^0$. Spatial reasoning within each time step is performed via a message passing Graph Neural Network (GNN) layer, which updates node and edge features based on local neighborhood interactions as

$$\mathbf{m}_e^{k+1} = f_{\mathcal{E}}^k(\mathbf{m}_e^k, \mathbf{m}_v^k, \mathbf{m}_w^k), \text{ with } e = (v, w), \qquad \tilde{\mathbf{m}}_v^{k+1} = f_{\mathcal{V}}^k(\mathbf{m}_v^k, \bigoplus_{e \in \mathcal{E}_v} \mathbf{m}_e^{k+1}).$$

Here, $\mathcal{E}_v \subseteq \mathcal{E}$ denotes the edges incident to node v, and $\bigoplus$ is a permutation-invariant aggregation. The functions $f_{\mathcal{E}}^k$ and $f_{\mathcal{V}}^k$ are learned MLPs. The resulting node features $\tilde{\mathbf{m}}_v^{k+1}$ are then processed independently over time by a 1D residual CNN to obtain $\mathbf{m}_v^{k+1}$. After K rounds of spatial message passing interleaved with temporal CNN processing, the model is trained by summing per-timestep MSE losses against ground truth mesh states over the full rollout.

B.4 Parameters and Runtime

PEACH has a total of 3.96M parameters, of which 3.01M belong to the *MaNGO* simulator and 0.96M to the point cloud encoder and auxiliary loss heads.

We report inference timings on an NVIDIA RTX 5090 for the `Trampoline` environment, using $N = 1024$ points per frame and every second time step of the trajectory unless stated otherwise. Inference consists of two stages. First, the encoder maps the context point cloud sequences to the aggregated latent material description. Second, the simulator predicts the full trajectory from the initial mesh and the latent description.

Encoding. Figure 8 shows how the encoding time depends on the context size C, the number of points per frame N, and the number of frames per trajectory T. The encoding time grows with the context size from 73 ms for a single context trajectory to 229 ms for eight. For all point counts, we observe a step between $C = 2$ and $C = 3$, which we attribute to implementation effects of the batched GPU computation, such as a change in kernel selection for larger batches, rather than to the method itself. Doubling the number of points or the number of frames increases the encoding time by a factor between 1.6 and 1.9. This is consistent with the linear dependence on the total number of space-time points $N \cdot T$ of farthest point sampling and patch grouping.

Simulation. The simulation rollout takes approximately 352 ms regardless of the context size, as it operates solely on the aggregated latent code. A complete prediction therefore takes between 0.4 and 0.6 s, depending on the context size.

Comparison. Figure 9 compares the inference time of *PEACH* with *Test-Time Optimization* and the FEM simulator used to generate the training data. *Test-Time Optimization* requires 82 s to adapt to a task and predict its trajectory, as each of its optimization steps involves a full forward and backward pass through the simulator for all context trajectories. *PEACH* replaces this iterative optimization with

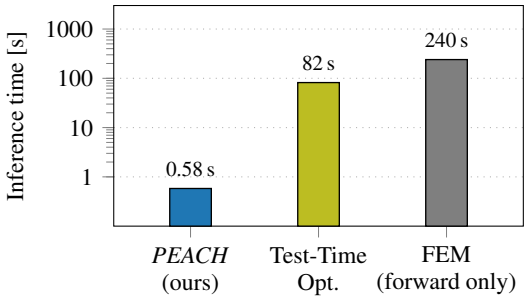


Figure 9: Inference time on the `Trampoline` environment on a logarithmic scale. *PEACH* and *Test-Time Optimization* include both the adaptation to the task and the prediction of the trajectory on an NVIDIA RTX 5090. The FEM time corresponds to a single forward rollout with known material parameters on a CPU.

Table 2: List of the used hyperparameters

Parameter	Value
Num Points in Pointcloud	512
Num Points in Pointcloud (Trampoline)	1024
FPS downsampling ratio (Tokenizer)	0.1
Patch size (Tokenizer)	16
Time scaling (Tokenizer)	16
Num. Fourier frequencies	8
Transformer Attention heads	4
Latent material representation dimension	128
Simulator Node feature dimension	128
Simulator Message passing blocks	15
Simulator Aggregation function	Mean
Simulator Activation function	Leaky ReLU
Optimizer	AdamW [62]
Learning rate	5.0×10^{-5}
Weight decay	1.0×10^{-4}
Min Context Size (Training)	1
Max Context Size (Training)	8
Latent representation aggregation	Learned Softmax
Auxiliary Loss Scale (Phys. Parameters)	0.02
Auxiliary Loss Scale (SDF)	0.5
Noise Scale MGN (Def. Block)	0.0007
Noise Scale MGN (Bend. Beam)	0.0007
Noise Scale MGN (Trampoline)	0.0005
Noise Scale MGN (Sheet Def.)	0.0005

a single forward pass and is thus about 140 times faster. A single FEM rollout requires 4 minutes on an AMD EPYC 74F3 24-Core Processor, even with known material parameters. Identifying unknown parameters with the FEM simulator would require many such rollouts and thus take considerably longer.

B.5 Hyperparameters

We provide a list of the used hyperparameters in Table 2.

C Simulation Datasets

This section provides detailed information about the datasets used in our experiments. The key characteristics of each dataset are summarized in Table 3. `Deforming Block` and `Sheet Deformation`

consist of 16 simulation trials per task with various initial conditions, `Trampoline` and `Bending Beam` consist of 10 simulation trials per task.

Table 3: Dataset descriptions

Name	Train/Val/Test Splits	Number of Steps	Number of Nodes
Deforming Block	600/100/100	52	81
Sheet Deformation	460/50/50	50	225
Bending Beam	380/60/60	100	180
Trampoline	600/100/100	25	1700

C.1 Trampoline.

This dataset models a thin square deformable sheet clamped by a rigid frame and deformed by a spherical impactor of varying size and density falling under gravity. Rigid body motion prediction of the ball is enforced by predicting per-node velocities and averaging them across all ball nodes. The sheet has initial stress-free edge length $l_{\text{init}} = 245$ mm. The sheet is discretized by 9604 triangular membrane elements and 4901 nodes, with membrane thickness t varied as described below. To emulate the rigid frame in the physical setup, the sheet is pre-stretched to an edge length of $l = 260$ mm by applying Dirichlet boundary conditions along its outer boundary.

Data generation follows a two-stage sampling procedure. In the first stage, a simulation configuration is defined by sampling the sheet material parameters together with the properties of a rigid spherical impacter. The sheet material is modeled as a nearly incompressible viscoelastic solid with rubber-like behavior. The instantaneous Young's modulus is sampled uniformly from $E \in [1, 5]$ MPa, the Poisson's ratio is fixed to $\nu = 0.499$, and the membrane thickness is sampled from $t \in [0.15, 0.4]$ mm. Rate dependence is modeled using a single-term Prony series with shear relaxation ratio $g \in [0.01, 0.3]$ and relaxation time $\tau \in [10^{-2}, 10^{1}]$ s. Volumetric relaxation is neglected ($k = 0$). The impacter diameter is chosen from 15 to 55 mm in steps of 5 mm, with each diameter represented by a pre-meshed geometry of approximately 1000 elements. Its mass is sampled from $[0.009, 0.269]$ kg, subject to density bounds between 268 kg/m^3 (3D-printed PLA with 10% infill) and 7850 kg/m^3 (steel). Gravitational loading is applied as an equivalent concentrated force at the impacter's center of mass.

In the second stage, 10 trajectories are generated for each fixed stage-one configuration by varying only the initial conditions of the spherical impacter. Specifically, the in-plane drop position is sampled from a Gaussian distribution centered at the sheet origin and truncated to $[-75, 75]$ mm in each in-plane direction, while the drop height is sampled within $[120, 280]$ mm. Contact between the impacter and the sheet is modeled using hard normal contact to prevent interpenetration. Tangential contact is governed by a Coulomb friction law with coefficient $\mu = 0.7$. All simulations are performed in ABAQUS/EXPLICIT using explicit time integration.

C.2 Bending Beam.

This scene consists of 2D beams clamped on the left side that bend and deform over time due to time-varying external forces applied on the right side. The beam is modeled as a rectangular domain embedded in two dimensional space. The left boundary of the beam is clamped with zero displacement, while external forces are applied to the right boundary. The load consists of one horizontal and one vertical force component. The horizontal force magnitude is sampled uniformly in $[-1.0{\cdot}10^{-7}, 1.0{\cdot}10^{-7}]$, while the vertical force magnitude is sampled uniformly in $[-3.0{\cdot}10^{-5}, 3.0{\cdot}10^{-5}]$. The forces are varied smoothly over time using cubic spline interpolation, starting from zero force, reaching the sampled target magnitude, and returning to zero before the end of the simulation. The beam height is sampled normally with mean 0.3 and standard deviation 0.01, and the beam length is sampled normally with mean 10.0 and standard deviation 1.0. The maximum characteristic mesh length, used for mesh generation, is sampled normally with mean 0.2 and standard deviation 0.02. The material behavior is modeled using an isotropic Kelvin–Voigt viscoelastic material model, combining an elastic stress response with a viscous stress contribution. Young's modulus E is sampled log-uniformly in $[0.5, 5.0]$, Poisson's ratio ν is sampled uniformly

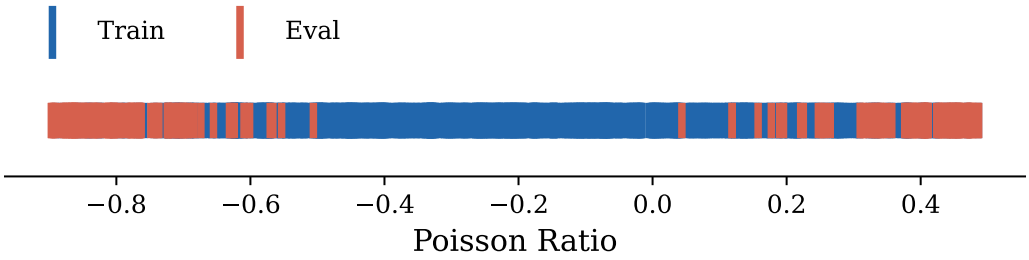


Figure 10: Out-of-distribution evaluation split for the `Deforming Block OOD` environment. Each tick marks one task, colored by split assignment. Eval tasks are drawn from the tails of the Poisson ratio distribution, ensuring that evaluated material properties lie outside the training range.

in $[0.0, 0.45]$, and the viscosity parameter τ is sampled log-uniformly in $[0.05, 0.5]$. Geometric non-linearities due to large deformations are considered. The beam is discretized using linear triangular elements. The simulations are implemented in `scikit-fem` and solved using Newton iterations with a residual tolerance of 10^{-8}. We create a total of 500 simulations, each consisting of $40{,}000$ time steps. For the GNS, we subsample to 100 time steps per trajectory.

C.3 Deforming Block.

This scene consists of 2D trapezoids deformed by a circular collider. This environment was originally implemented by Linkerhägner et al. [6] and uses Simulation Open Framework Architecture (SOFA) [66]. This dataset considers the deformation of a two dimensional rectangular block in two dimensional space. There are different trapezoidal initial geometries of the block. The nodes located at the bottom edge of the block are clamped with zero displacement. The load is applied by means of a contact between a rigid circular collider and the top surface of the block. Across simulation trials, the collider radius is sampled uniformly between $10\,\%$ and $40\,\%$ of the block size and moves vertically downward at a constant velocity. The contact is modeled using a hard contact definition, which is rigid in the normal direction and frictionless in the tangential direction. The material behavior of the block is modeled using isotropic linear elasticity with a fixed Young's modulus E, while the Poisson's ratio ν is sampled uniformly per task from $[-0.9,\ 0.49]$. The block is discretized using 81 nodes and 128 triangular elements.

The out-of-distribution evaluation split for this environment is visualized in Figure 10, where eval tasks are drawn from the tails of the Poisson ratio range $[-0.9,\ 0.49]$.

C.4 Sheet Deformation.

This dataset considers the deformation of a thin square 2D deformable sheet subjected to out-of-plane forces, resulting in 3D deformations. The sheet has initial stress-free edge length $l = 140$ mm. The thickness of the sheet is intrinsically defined as $t = 2.0$ mm. Nodes located at the outer edge of the sheet are clamped with zero displacement. The load is applied by two discrete external forces acting perpendicular to the thickness direction of the sheet. The force magnitude is fixed, while the normal force direction (upward or downward) and point of application are sampled uniformly. To enable a continuous range of force application locations, the forces are applied uniformly to all nodes within a small radius around the sampled application location. The forces are ramped up linearly from zero to their target magnitude over the simulated time. The material behavior of the sheet is modeled using isotropic linear elasticity with Young's modulus E sampled log-uniformly in $[10, 500]$ MPa and Poisson's ratio ν=0.3. The sheet is discretized using 225 nodes and 392 triangular elements. Geometric non-linearities due to large deformations are considered. The simulation for the described environment is implemented in the finite element software `Abaqus/Standard` and solved using implicit time integration.

D Real World Experiments

We create a real-world version of **Trampoline** dataset for use in our sim to real experiments. To mimic the simulated scene, solid balls of various materials and densities (see Table 4) are dropped onto rubber sheets of various thicknesses (see Table 5). The balls are dropped using a Franka Emika Panda robot, where the drop position is sampled from a volume of $4\,\mathrm{cm} \times 4\,\mathrm{cm} \times 5\,\mathrm{cm}$. The scene is recorded using a StereoLabs Zed Mini stereo camera at a resolution of 720×1280 frame rate of $30\,\mathrm{fps}$.

Diameter (mm)	**Mass** (g)	**Material**
40	269	Steel Solid
40	69	Steel Hollow
50	110	Steel Hollow
60	162	Steel Hollow
40	21	PLA 50% Infill

Table 4: Sphere specifications used in the real-world experiments.

Color	**Thickness** (mm)	**Average Density** $(\mathrm{kg\,m^{-3}})$
Black	0.381	992.7
Green	0.254	992.7
Yellow	0.1524	992.7

Table 5: Sheet specifications used in the real-world experiments.

Depth estimates for each recorded frame are recomputed using FoundationStereo [67] (smaller architecture based on Vit-small). Images are resized to 480x853 and are converted into point clouds by unprojecting the depth using the cameras' intrinsic parameters. The camera extrinsics are calibrated by mounting Aruco markers on the frame around the rubber sheet and optimizing for minimal reprojection error (OpenCV's `solvePnPRansac`). Point clouds are then transformed into a canonical world frame such that the bottom left corner of the sheet is at the origin and the sheet is on the xy plane.

We discard time steps before the ball begins to fall and limit trajectories to 20 time steps (approx. $0.667\,\mathrm{s}$). Points in the background of the scene are cropped out, and voxel downsampling is applied with a voxel edge length of $5\,\mathrm{mm}$. To remove spurious artifacts, any points that have fewer than 20 neighbors within a radius of $15\,\mathrm{mm}$ are filtered out. Finally, FPS [51] is applied to reduce the point clouds to 2048 points.

Sphere points are identified via color segmentation, isolating the distinctly colored ball from the scene. The color labels are refined through an iterative algorithm that updates each point's label based on its own color and the colors of its neighbors, smoothing spurious misclassifications at object boundaries. Given the known sphere diameter, a spherical mask is fitted to the identified sphere points by optimizing the center position, which is then used to remove robot arm points that fall outside the mask. Finally, FPS is applied to reduce the point clouds to 1024 points.

D.1 Robustness to Corrupted Observations

Real-world point clouds can contain missing regions, for example due to occlusions or reflective surfaces, as well as spurious points from reconstruction errors. To evaluate how robust *PEACH* is to such artifacts, we corrupt the context observations of the real-world `Trampoline` scene at test time.

Setup. We use the *PEACH* model from the main experiments, as well as a *PEACH* model trained with the identical setup but without data augmentation. For each corruption level, we remove 1%, 9%, or 25% of the points of each context point cloud, which we refer to as low, medium, and high corruption. Instead of removing points uniformly at random, we remove them in small spatially connected patches, which resembles missing regions in real sensor data. We additionally insert the

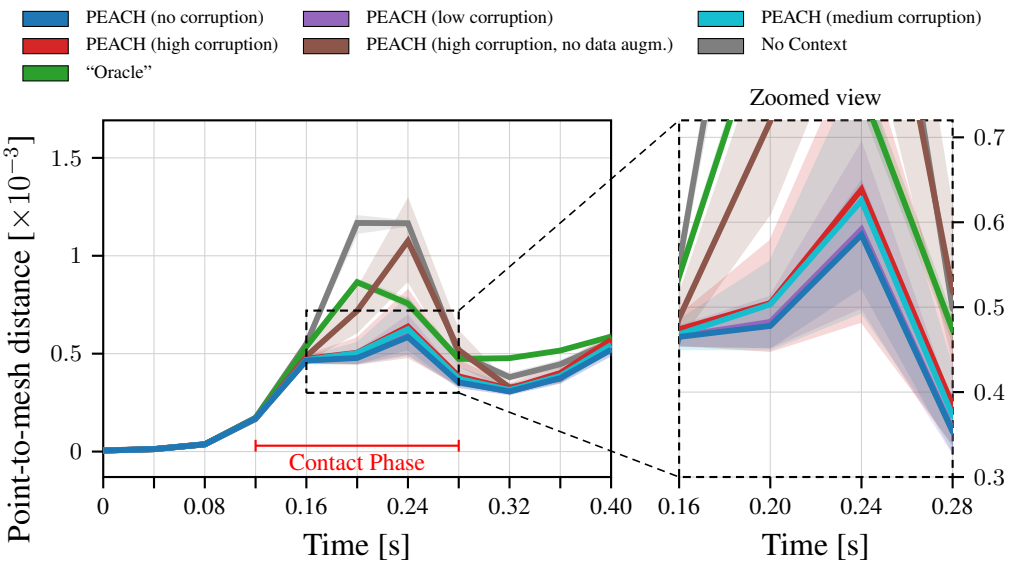


Figure 11: Mean point-to-mesh distance on the real-world `Trampoline` scene with corrupted context observations. Low, medium, and high corruption remove 1%, 9%, and 25% of the context points in small patches and add the same number of spurious points, while the target observations used for evaluation remain uncorrupted. The right panel shows a zoomed view of the contact phase, marked by the dashed rectangle in the left panel.

same number of spurious points at random locations in the scene to simulate reconstruction artifacts. Only the context trajectories are corrupted. The encoder infers the latent material description from the corrupted context, and the simulator predicts the target trajectory from it. We evaluate this prediction against the uncorrupted observations of the target trajectory using the same point-to-mesh distance as in Figure 5. Note that these corruptions are similar in kind to the region dropout and point artifacts used in our data augmentation, but they are applied to real-world instead of simulated observations and at levels that we did not tune for this evaluation.

Results. Figure 11 shows the results. *PEACH* handles all three corruption levels similarly well, and its errors remain close to the uncorrupted case throughout the trajectory. Even under high corruption, *PEACH* is clearly more accurate than the *Oracle* during the contact phase, although the *Oracle* does not rely on the context observations and is therefore unaffected by the corruption. In contrast, the *PEACH* model trained without data augmentation deteriorates considerably under high corruption, and its error at the peak of the contact phase approaches that of the *No Context* baseline. This shows that data augmentation is essential for the robustness of *PEACH* to corrupted real-world observations.

E Additional Quantitative Results.

We provide full rollout MSE as a function of context size for all four environments in Figure 12. *PEACH* consistently improves with more context trajectories and achieves performance competitive with the Oracle baselines, while outperforming all learned point cloud encoders across environments.

F Statistical Analysis of the Latent Material Description

We complement the qualitative latent space visualizations in Appendix G.1 with two quantitative analyses of the latent material description $\mathbf{r}$. Appendix F.1 investigates how $\mathbf{r}$ behaves within a task as the number of context trajectories grows, and Appendix F.2 investigates whether $\mathbf{r}$ preserves the structure of the physical parameters across tasks.

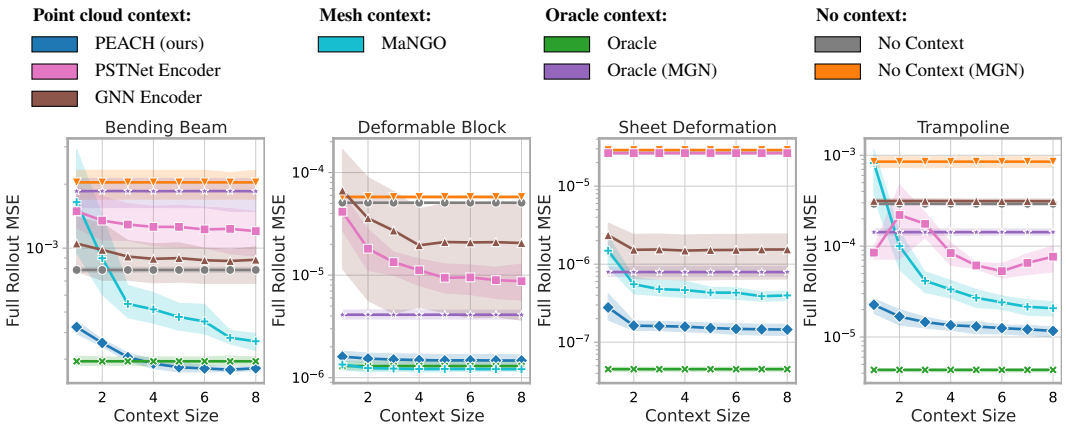


Figure 12: Full simulation rollout MSE over the context size across different baselines. PEACH consistently performs similarly to the mesh-based MaNGO, improving over other point cloud-based baselines and achieving performance competitive with the Oracle models.

F.1 Convergence with Context Size

Ideally, the latent material description converges to a fixed descriptor of the task, and additional context trajectories reduce its variance at a rate comparable to averaging independent observations.

Setup. Each test task consists of N trajectories that share the same physical properties, with $N = 16$ for `Deforming Block` and `Sheet Deformation` and $N = 10$ for `Bending Beam` and `Trampoline`. For every context size $C \in \{1, \dots, 8\}$, we sample up to 250 context sets S_C of C trajectories without replacement, or use all possible sets if fewer exist, and encode each set into its aggregated latent representation $\mathbf{r}(S_C)$. As the latent space has no natural scale, we standardize each of its dimensions by its mean and standard deviation over all latent vectors of an environment. We then measure the variance of the standardized representations $\tilde{\mathbf{r}}$ across the sampled context sets of a task as

$$\mathrm{Var}(C) = \frac{1}{M_C - 1} \sum_{m=1}^{M_C} \left\| \tilde{\mathbf{r}}(S_C^{(m)}) - \bar{\mathbf{r}}_C \right\|^2,$$

where M_C is the number of sampled context sets and $\bar{\mathbf{r}}_C$ their mean representation.

Variance model. For a plain average of C independent samples, the variance decreases as C^{-1}. Since our context sets are drawn without replacement from a finite pool of N trajectories, the variance across context sets is additionally reduced by the finite population correction $(N - C)/(N - 1)$, which becomes substantial for large C relative to N. We therefore fit the model

$$\mathrm{Var}(C) = \sigma^2 \, C^{-\beta_r} \, \frac{N - C}{N - 1}$$

to each task individually via least squares in log space. Under this model, $\beta_r = 1$ corresponds exactly to averaging independent trajectory embeddings. Values $\beta_r < 1$ indicate a slower variance reduction, for example due to redundant context trajectories, while $\beta_r > 1$ indicates that the aggregation reduces the variance faster than a plain average. Note that the learned softmax aggregation of *PEACH* is not a plain average, so the model serves as a reference rather than an exact description.

Results. Figure 13 shows the normalized variance $\mathrm{Var}(C)/\mathrm{Var}(1)$ for all test tasks, and Table 6 reports the fitted exponents. In all environments, the variance of $\mathbf{r}$ decreases consistently with the context size, and the curves of individual tasks closely follow the median, indicating that the latent description converges for all tasks rather than only on average. For `Deforming Block`, `Sheet Deformation`, and `Bending Beam`, the median exponents lie close to $\beta_r = 1$, i.e., the aggregated representation behaves like an average over independent observations of the task. `Bending Beam` shows the largest spread across tasks, which we attribute to its three varying material parameters that are not equally observable in every trajectory. For `Trampoline`, the variance decreases more slowly

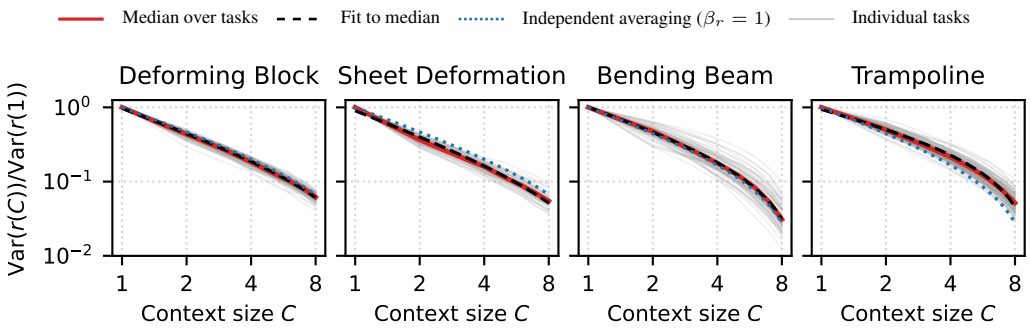


Figure 13: Variance of the latent material description $\mathbf{r}$ across context sets of size C, normalized per task by its variance at $C = 1$. The dashed line shows the fitted power law with finite population correction on the median curve, and the dotted line shows the expected decrease for averaging independent samples.

Table 6: Fitted variance exponent β_r of the latent material description. We report the median over all test tasks, with the interquartile range in brackets. $\beta_r = 1$ corresponds to averaging independent samples.

	`Deforming Block`	`Sheet Deformation`	`Bending Beam`	`Trampoline`
β_r	$1.03\ [0.99, 1.06]$	$1.08\ [1.04, 1.13]$	$0.96\ [0.77, 1.07]$	$0.73\ [0.64, 0.83]$

with $\beta_r = 0.73$. Here, the trajectories of a task only differ in the drop position and height of the ball, so additional context trajectories provide partially redundant information about the material. Overall, the consistent variance reduction matches the steady improvement of the simulation accuracy with increasing context size reported in Appendix E.

F.2 Correspondence to Physical Parameters

A useful latent material description should organize tasks according to their physical properties, i.e., tasks with similar parameters should be encoded close to each other. We test this with a representational similarity analysis [68] across all test tasks of an environment.

Setup. For each test task, we estimate its latent description as the mean of the standardized representations $\tilde{\mathbf{r}}(S_C)$ over all sampled context sets of the largest context size $C = 8$, which yields the most converged estimate. We normalize each ground-truth physical parameter by its standard deviation across the test tasks of an environment, so that all parameters contribute equally. We then compute two matrices of pairwise Euclidean distances between tasks, one in the space of physical parameters ρ and one in the latent space $\mathbf{r}$, and measure their correspondence via the Spearman correlation between their upper triangles. Since every task contributes to many pairs, the pairs are not independent, and we assess significance with a Mantel test [69] that permutes the task labels of one matrix 9999 times. As a complementary local measure, we report the neighbor agreement, i.e., the fraction of each task's 3 nearest neighbors in ρ that are also among its 3 nearest neighbors in $\mathbf{r}$, averaged over all tasks.

Additionally, we estimate the effective dimensionality of the latent descriptions across tasks via the participation ratio

$$\mathrm{PR} = \frac{\left(\sum_i \lambda_i\right)^2}{\sum_i \lambda_i^2},$$

where λ_i are the eigenvalues of the covariance matrix of the task-level latent descriptions. The participation ratio ranges from 1, if all variance lies along a single direction, to the full latent dimension, if the variance is spread evenly. We compute it on the mean-centered but otherwise unstandardized representations, since standardizing each dimension would equalize the variance of uninformative directions and thereby inflate the estimate.

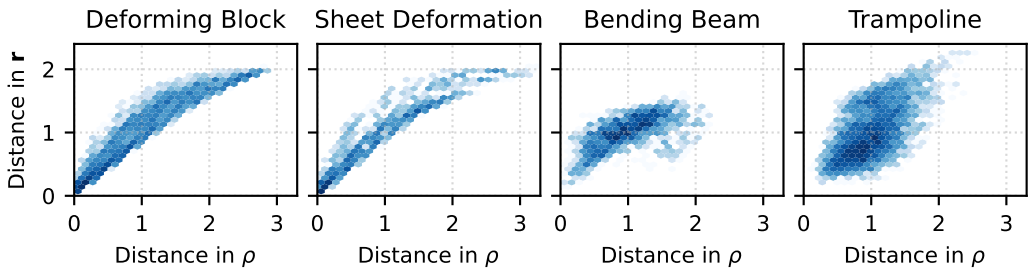


Figure 14: Pairwise distances between test tasks in the space of ground-truth physical parameters ρ and in the latent space $\mathbf{r}$, both normalized by their mean. Each hexagon counts task pairs on a logarithmic color scale. Tasks with similar physical parameters are consistently encoded close to each other, while the larger spread in `Bending Beam` and `Trampoline` stems from parameters with little influence on the observed dynamics.

Table 7: Correspondence between the latent material description $\mathbf{r}$ and the ground-truth physical parameters ρ across test tasks. Spearman correlations are computed between pairwise task distances and tested with a Mantel test using 9999 permutations, all with $p < 10^{-3}$. Neighbor agreement is the fraction of each task's 3 nearest neighbors in ρ that are also among its 3 nearest neighbors in $\mathbf{r}$, with the chance level in brackets.

	`Deforming Block`	`Sheet Deformation`	`Bending Beam`	`Trampoline`
Test tasks	100	50	60	100
Spearman correlation	0.97	0.96	0.64	0.60
Neighbor agreement	0.69 [0.03]	0.85 [0.06]	0.54 [0.05]	0.31 [0.03]
Varying parameters	1	1	3	5
Participation ratio	1.56	1.49	4.15	2.78

Results. Figure 14 shows the pairwise task distances in both spaces, and Table 7 summarizes the statistics. For the single-parameter environments `Deforming Block` and `Sheet Deformation`, the distances correspond almost perfectly with Spearman correlations of 0.97 and 0.96. The relationship is monotone but concave, as distances in $\mathbf{r}$ saturate for very different materials, which the rank-based correlation does not penalize. For `Bending Beam` and `Trampoline`, the correlations of 0.64 and 0.60 are lower but still highly significant, and the neighbor agreement is roughly ten times above chance level in all environments. We attribute the larger spread in these environments to parameters with little influence on the observed dynamics. Since all parameters contribute equally to the distance in ρ, tasks that differ mainly in such a parameter appear far apart in ρ but behave almost identically, and are thus correctly encoded close to each other in $\mathbf{r}$.

The participation ratio supports this interpretation. For the single-parameter environments, it lies close to 1, i.e., the latent descriptions essentially vary along a single direction. For `Trampoline`, the participation ratio of 2.78 is smaller than the five varying parameters. As shown in Figure 15, the behavior of this scene is mainly governed by the loading ratio, which combines the ball mass, the Young's modulus, and the sheet thickness into a single quantity, and by the relaxation time, while the shear relaxation ratio has little visible effect. For `Bending Beam`, the participation ratio of 4.15 exceeds the three varying material parameters. This is consistent with our interpretation that the latent description does not merely encode the material parameters, but an effective behavior that also compensates for approximation errors of the learned simulator, which need not align with the directions of the individual parameters.

G Visualizations

We provide additional qualitative results comparing all evaluated methods on all four datasets.

G.1 Latent Space Visualizations

Figure 15 shows a PCA projection of the task-level latent codes $\mathbf{r}$ produced by the *PEACH* encoder across all four environments. Each point corresponds to a single test task, colored by its material properties. The smooth, structured organization of the latent space, particularly the near-perfect one-dimensional manifold recovered for the single-parameter environments `Sheet Deformation` and `Deforming Block`, demonstrates that the encoder successfully disentangles material properties from trajectory-level variation, without any explicit supervision on the latent space geometry.

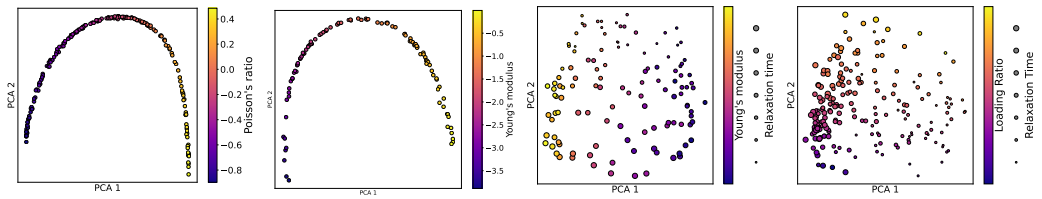


Figure 15: PCA visualization of the latent space on the Deforming block, Sheet deformation, Bending beam and Trampoline environments (left to right). Each point represents an encoded task, with different coloring and scaling references based on the task's material properties.

G.2 Qualitative Trajectory Predictions

We provide qualitative trajectory predictions for all evaluated datasets. Figure 16 shows results for the `Deforming Block` dataset, Figure 17 for `Sheet Deformation`, Figure 18 for the `Trampoline` dataset, and Figure 19 for `Bending Beam`.

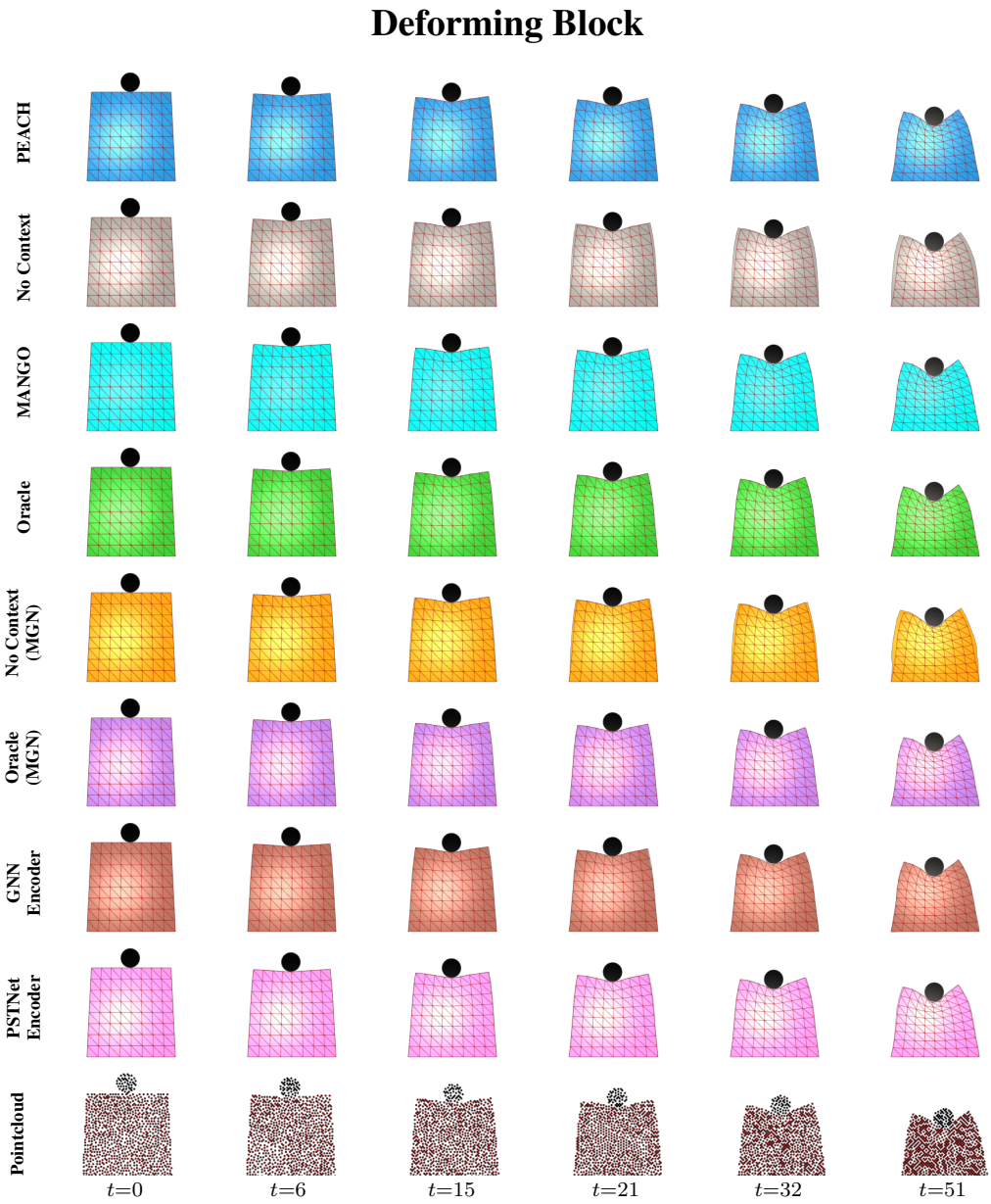


Figure 16: Predicted simulation of a `Deforming Block` test task by PEACH (blue), No Context (gray), MANGO (cyan), Oracle (green), No Context (MGN) (orange), Oracle (MGN) (purple), GNN Encoder (brown), and PSTNet Encoder (pink). All visualizations show the colored **predicted mesh**, a **collider**, and a **wireframe** (red) of the ground-truth simulation. The last row shows an exemplary point cloud sequence from the context set.

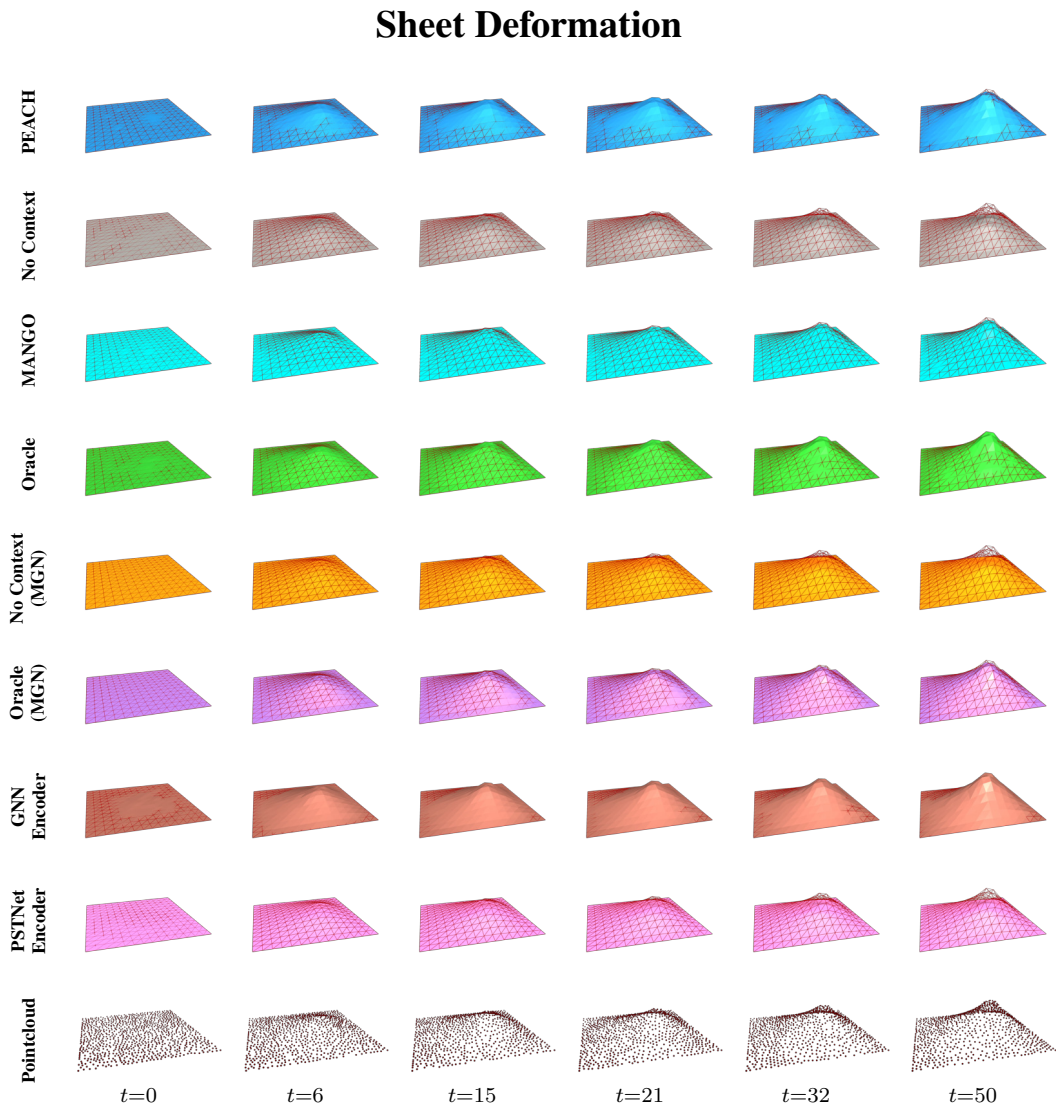


Figure 17: Predicted simulation of a `Sheet Deformation` test task by PEACH (blue), No Context (gray), MANGO (cyan), Oracle (green), No Context (MGN) (orange), Oracle (MGN) (purple), GNN Encoder (brown), and PSTNet Encoder (pink). All visualizations show the colored **predicted mesh** and a **wireframe** (red) of the ground-truth simulation. The last row shows an exemplary point cloud sequence from the context set.

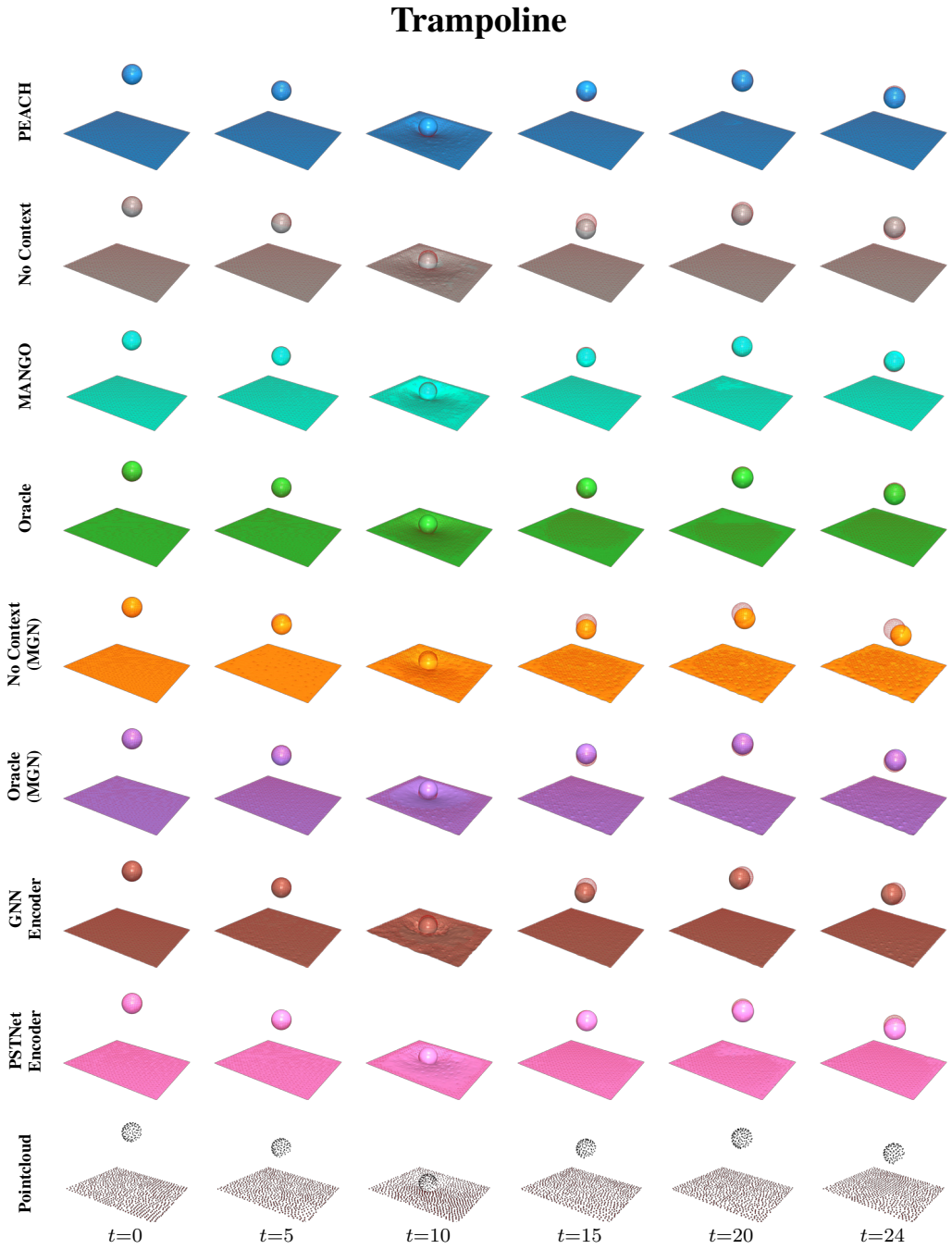


Figure 18: Predicted simulation of a `Trampoline` test task by PEACH (blue), No Context (gray), MANGO (cyan), Oracle (green), No Context (MGN) (orange), Oracle (MGN) (purple), GNN Encoder (brown), and PSTNet Encoder (pink). All visualizations show the colored **predicted mesh** and a **wireframe** (red) of the ground-truth simulation. The last row shows an exemplary point cloud sequence from the context set.

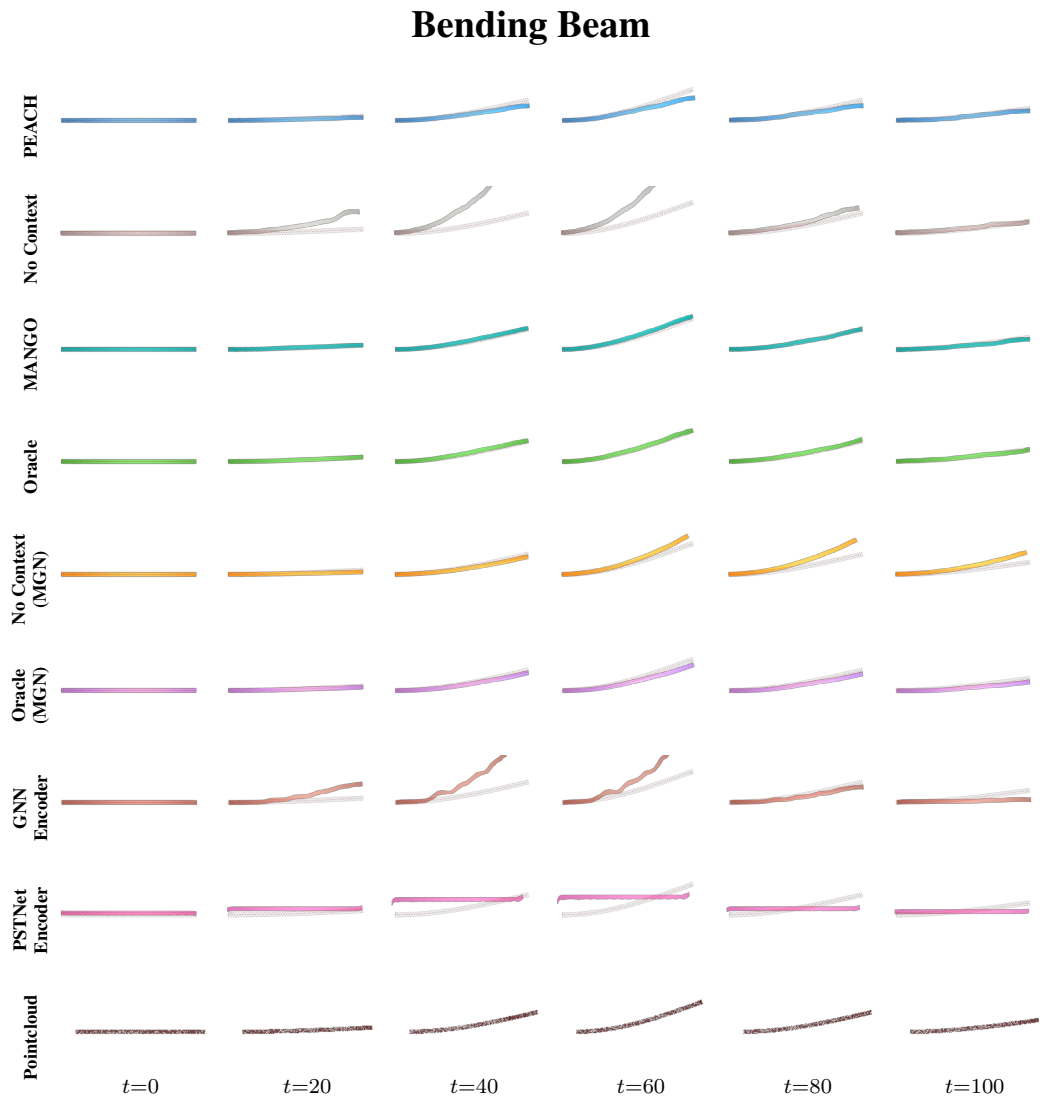


Figure 19: Predicted simulation of a `Bending Beam` test task by PEACH (blue), No Context (gray), MANGO (cyan), Oracle (green), No Context (MGN) (orange), Oracle (MGN) (purple), GNN Encoder (brown), and PSTNet Encoder (pink). All visualizations show the colored **predicted mesh** and a **wireframe** (red) of the ground-truth simulation. The last row shows an exemplary point cloud sequence from the context set.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim that *PEACH* (1) infers material properties from raw point cloud observations without mesh registration, (2) outperforms point cloud-based baselines and is competitive with mesh-based and oracle methods, and (3) generalizes to real-world data without retraining. All three claims are directly supported by the quantitative simulation results in Section 4, the ablation study, and the real-world trampoline experiment.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the dedicated Limitations and Future Work paragraph at the end of Section 5, covering aspects such as the assumption of known object geometry at test time, the restriction to a single deformable object per scene, and the potential for extending the approach to additional sensor modalities.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: We do not include theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The appendix provides detailed descriptions of all four simulation environments (Appendix C), full training hyperparameters and architectural details (Appendix B), and code is included in the supplementary material. The appendix additionally details the real-world experiment setup (Appendix D).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is provided in the supplementary material. Both simulation datasets and real-world measurements will be released upon acceptance. Appendix C describes the data generation procedure in sufficient detail.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and evaluation details are specified in Appendix B, covering the optimizer, learning rate, batch size, and model architecture. Data splits and dataset-specific details are described in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All quantitative results reported include error bars indicating the bootstrapped $95\,\%$ confidence intervals across 4 random seeds, allowing assessment of the statistical reliability of the reported results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources are reported in the experiments section, including the GPU hardware used for training, the HPC cluster infrastructure, and wall-clock runtimes for both training and inference (Inference runtimes are in the Appendix).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper conforms with the NeurIPS Code of Ethics. The work focuses on physical simulation and material property estimation, involves no human subjects, personal data, or sensitive applications, and raises no foreseeable ethical concerns.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This work addresses physical simulation and material property estimation from point cloud observations. We discuss its broader impact in Appendix A.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released artifacts consist of physics simulation code and datasets of deformable object trajectories. These do not present a meaningful risk of misuse and therefore no specific safeguards beyond standard open-source licensing are required.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All third-party code and models used in this work, including baseline implementations, are properly credited with citations to the original papers. License terms are respected throughout.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The newly introduced simulation datasets are documented in Appendix C, describing the environment setup, mesh discretization, material parameter ranges, and data generation procedure. Documentation will be provided alongside the datasets upon release.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve human subjects or study participants, and therefore no IRB approval or risk disclosure is required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: No LLMs are used as part of the core methodology. Any use of LLM-based tools was limited to writing assistance and does not impact the scientific content, methodology, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.