POINT CLOUD SEQUENCE ENCODING FOR MATERIAL-CONDITIONED GRAPH NETWORK SIMULATORS

Philipp Dahlinger[1]* **Balázs Gyenes**[1] **Niklas Freymuth**[1] **Tobias Würth**[2]

Tai Hoang[1] **Johannes Mitsch**[2] **Luise Kärger**[2] **Gerhard Neumann**[1]

[1]Autonomous Learning Robots, [2]Institute of Vehicle System Technology
Karlsruhe Institute of Technology, Karlsruhe

ABSTRACT

Graph Network Simulators (GNS) have emerged as powerful surrogates for complex physics-based simulation, offering inherent differentiability and orders-of-magnitude speedups over traditional solvers. However, GNS typically assume access to the underlying material parameters, such as stiffness or viscosity, and rely heavily on dense mesh-based data. These constraints severely limit their utility in realistic experimental settings. While recent meta-learning approaches address the parameter dependency by inferring properties from mesh trajectories, they still require mesh-based trajectories as contexts. In this work, we introduce Point cloud Encoding for Accurate Context Handling (*PEACH*), a novel framework that decouples parameter estimation from temporal dynamics. *PEACH* infers latent physical properties directly from sequences of point clouds using a spatial-temporal encoder. This parameter estimate then conditions a trajectory-level GNS, enabling accurate in-context simulation of unseen materials during inference. Experiments on simulated object deformation tasks demonstrate that *PEACH* achieves simulation accuracy matching purely mesh-based baselines, while relying on more realistic and accessible data.

1 INTRODUCTION

Simulating complex physical systems is fundamental to engineering disciplines ranging from structural mechanics (Landau et al., 2012) to fluid dynamics (Blazek, 2015). Traditional mesh-based methods, such as the Finite Element Method (FEM) (Brenner & Scott, 2008; Reddy, 2019), provide high precision, but quickly become computationally expensive. These methods also rely on precise knowledge of physical parameters, such as stiffness or density, which are frequently unknown in practice. Deep learning-based surrogates, particularly Graph Network Simulators (GNSs) (Battaglia et al., 2016; Pfaff et al., 2021; Brandstetter et al., 2022; Linkerhägner et al., 2023; Würth et al., 2025), have emerged as a promising alternative that offers orders-of-magnitude speedups and inherent differentiability. GNSs handle physical data by modeling arbitrary entities and their relations as a graph. Standard GNSs typically learn to evolve a given input mesh over time for known process conditions. However, they rely on expensive system identification methods (Åström & Eykhoff, 1971; Lennart, 1999; Isakov, 2006; Wu et al., 2015; Murthy et al., 2020; Antonova et al., 2023). for parameter estimation when process conditions, such as the material parameters, change.

While some methods use gradient-based inversion on pre-trained simulators to estimate parameters (Zhao et al., 2022; Wang & Wang, 2024), these require computationally expensive optimization loops at test time. Recently, Meta Neural Graph Operator (*MaNGO*) (Dahlinger et al., 2025) proposed to address these shortcomings via in-context learning. *MaNGO* adapts to unseen material properties during inference by encoding a context set of mesh-based trajectories into a high-dimensional latent representation. However, this reliance on dense, ground-truth meshes effectively

*Correspondence to `philipp.dahlinger@kit.edu`

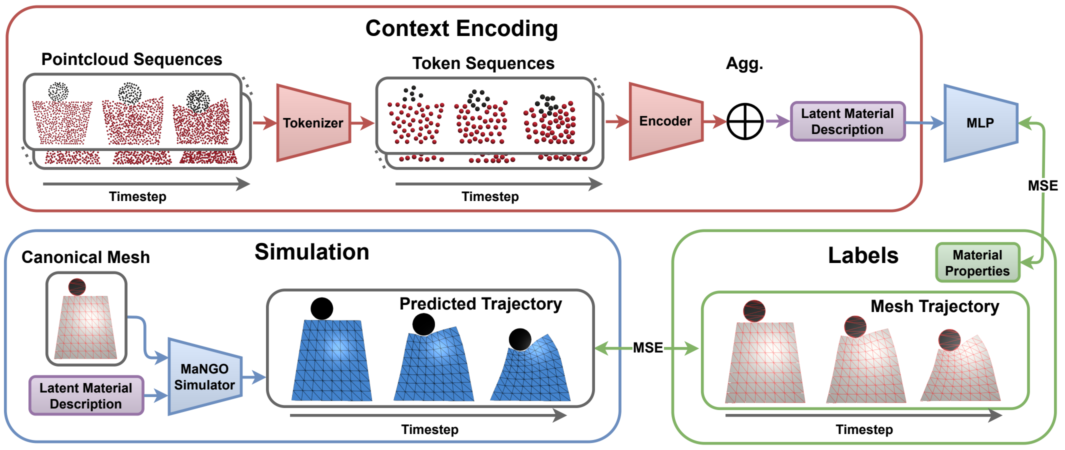

Figure 1: Overview of the *PEACH* framework. Point cloud sequences from several simulated or real trajectories are encoded into a single latent material description via tokenization and a subsequent Transformer-based sequence encoder. The aggregated latent vector conditions a *MaNGO* simulator, which predicts dynamics for an initial mesh. The method is trained to minimize the average simulation error, and uses an auxiliary loss on the latent material description to ensure physical consistency. During inference, only the context point cloud sequences and the initial geometry are needed to predict a simulation that follows the underlying, unknown material properties.

restricts the method to simulation-only settings. In contrast, real-world adaptation requires inferring underlying physics from sparse, noisy observations, rather than perfect mesh trajectories.

To bridge this gap, we introduce Point cloud Encoding for Accurate Context Handling (*PEACH*) 🍑. *PEACH* adopts a Conditional Neural Process (CNP) (Garnelo et al., 2018) approach to infer latent physical properties directly from multiple sequences of point clouds. The resulting latent material description is then used to condition a GNS for end-to-end training. To better align our training to context *sequences*, we utilize a trajectory-level architecture (Xu et al., 2024; Dahlinger et al., 2025) instead of iteratively rolling out system dynamics, as common in most GNSs. During training, we generate synthetic point clouds via ray-tracing on flexible material simulations. During inference, the model can be conditioned on arbitrary point cloud data, which can, e.g., be obtained via depth cameras, to predict dynamics on a canonical simulation mesh. This approach relaxes the strict context requirements of prior work, requiring only readily-available point-cloud sequences instead of mesh trajectories. Figure 1 provides a schematic overview.

We evaluate *PEACH* on complex object deformation environments, including a 2D `Deformable Block` that interacts with a collider, and a 3D `Sheet Deformation` scenario where a planar sheet is subjected to two forces perpendicular to the sheet (Dahlinger et al., 2025). We explore various encoder backbones for processing the point cloud contexts, including graph neural networks and the Point Spatio-Temporal Network (*PSTNet*) (Fan et al., 2021). Our experiments demonstrate that *PEACH* works best with a transformer backbone, achieving simulation accuracy on par with purely mesh-based baselines while operating on significantly sparser, more realistic data.

To summarize, we
(i) propose *PEACH*, an in-context learning framework for physical simulation that adapts to unseen materials using only point cloud sequences,
(ii) introduce an end-to-end architecture that encodes latent material parameters from such sequences, which then condition a trajectory-level GNS, and
(iii) demonstrate that *PEACH* matches the performance of simulators relying on mesh-based contexts on challenging 2D and 3D object deformation environments.

2 Point-cloud Encoding for Accurate Context Handling

Problem Setting. Each problem environment considers a family of physical systems that follow the same governing equations, but differ in their material properties. Each system instance is parameterized by a material vector, encoding properties such as stiffness or density. In this setting, each *task* corresponds to a fixed material configuration and consists of multiple trajectories with different initial conditions, such as different initial mesh geometries.

For each task, we generate training sequences using a FEM solver. Here, each trajectory consists of a mesh-based simulation as well as a corresponding sequence of point clouds obtained by sampling or ray-tracing the mesh states. At inference time, we need to infer the unknown material parameters and do not have access to mesh states. Instead, the model must rely on a variable-sized *context set* comprised of sequences of real-world or simulated point cloud observations to estimate material parameters. It then uses this estimate to predict trajectory-level dynamics for an initial mesh.

Latent Material Inference and Conditioned Simulation. Given a context set, each sequence is independently encoded into a latent vector using a point cloud sequence encoder. These vectors are aggregated into a single task-level *latent material description*. We aggregate using a learned softmax aggregation, which allows the model to adapt from a variable number of context trajectories. The resulting latent material representation conditions a trajectory-level GNS, which we implement using a *MaNGO* simulator (Dahlinger et al., 2025). The simulator receives an initial mesh and predicts future mesh dynamics for the sequence in a single forward pass.

During training, the predicted trajectories are compared to ground-truth mesh simulations using a Mean Squared Error (MSE) loss. Gradients are propagated end-to-end through both the simulator and the point cloud encoder. To improve the alignment between latent representations and underlying material properties, the latent material description is additionally compared to material property labels using a lightweight decoder MLP. The material labels are only available during training, and the decoder MLP is discarded at inference time.

Spatio-Temporal Point Cloud Encoder. To encode an entire point cloud sequence into a single latent vector, we represent each point by both its spatial coordinates and a normalized time value. Our point cloud encoder can then treat the sequence as a point cloud over 4D space time. To reduce the number of input points, we adapt the encoder architecture from Gyenes et al. (2024) and apply Farthest Point Sampling in 4D space-time to select patch centers. For each center, a local spatio-temporal neighborhood is constructed using nearest-neighbor queries. Each neighborhood is processed using a PointNet-style architecture to produce a token embedding, to which a Fourier-encoded representation of the patch center's 4D location is added. The resulting set of tokens is processed by a Transformer-based backbone and pooled into a single embedding, which serves as the latent representation of the corresponding point cloud trajectory.

3 Experiments

Environments. We evaluate *PEACH* on its ability to provide latent material descriptions from raw point cloud sequences to facilitate downstream simulation of a canonical mesh. We consider two simulated environments, namely `Deformable Block` and `Sheet Deformation` (Dahlinger et al., 2025)[1]. `Deformable Block` consists of 2D trapezoids with Poisson's ratios uniformly sampled from $[-0.9, 0.49]$ that are deformed by a circular collider. The dataset for this environment contains $600/100/100$ train/val/test tasks, each containing 16 trajectories of 52 simulation steps. In `Sheet Deformation`, a 2D sheet is subjected to external forces perpendicular to the sheet, resulting in 3D deformations. The sheet's Young's Modulus is sampled log-uniformly from $[10, 500]$. Here, the dataset includes $460/50/50$ train/val/test tasks with 16 trajectories of 50 steps each.

For both environments, we generate point clouds for each simulation step via raycasting from a static, front-facing camera followed by Farthest Point Sampling (FPS) (Eldar et al., 1997) to retain 512 points per time step. For `Deformable Block`, points belonging to the collider are explicitly marked. The lack of temporal point correspondences in these sequences significantly increases task difficulty compared to mesh-based inputs. More information about the environments is given in Appendix B.

Baselines and Setup. We compare against *MaNGO* (Dahlinger et al., 2025), which operates on mesh trajectories. *MaNGO* acts as an upper bound that assesses the impact of losing explicit mesh

[1]Respectively called `Deformable Plate` (*easy*) and `Planar Bending` in Dahlinger et al. (2025).

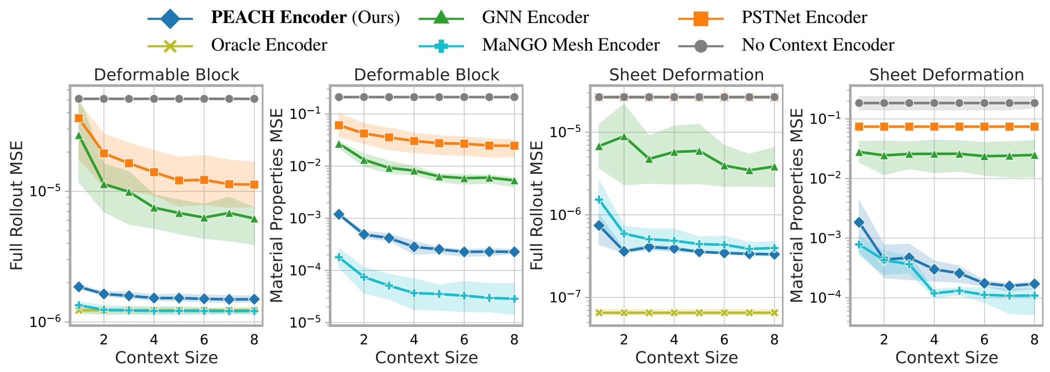

Figure 2: Full simulation rollout and material property MSE across different encoder architectures as a function of the number of context sequences for (**left**) `Deformable Block` and (**right**) `Sheet Deformation`. *PEACH* consistently performs similar to the mesh-based *MaNGO*, improving over other point cloud encoders and achieving near-Oracle performance on `Deformable Block`.

topology. We additionally evaluate several point cloud sequence architectures as *PEACH*'s encoder. *PSTNet* (Fan et al., 2021) uses spatio-temporal convolutions to capture local geometric evolutions across 4D space-time. Additional encoders include a *Spatio-temporal GNN* that constructs dynamic graphs per frame using nearest neighbors inside the same and adjacent time frames. Our proposed architecture as described in Section 2 uses a *Transformer Encoder* that processes the point tokens and aggregates them into a single trajectory-level embedding.

We train each method end-to-end, evaluating simulation fidelity and material prediction accuracy using MSE against the full trajectory and ground truth vector, respectively. For the trajectory evaluation, we also include an upper bound *Oracle Encoder*, which receives ground-truth material properties. We similarly include a *No Context Encoder* with an empty context set to establish a lower bound representing regression toward mean material behavior. All models are trained using batches of 1 to 8 randomly sampled trajectories to improve adaptation capability. We repeat each experiment across 3 random seeds using the AdamW optimizer (Loshchilov & Hutter, 2019) with a learning rate of 5.0×10^{-5}. Appendix A provides an overview of used methods and lists detailed hyperparameters.

4 Results

Figure 2 shows that *PEACH* consistently outperforms all point cloud baselines across both environments and scales best with increased context set sizes. These results indicate that the Transformer-based encoder better aggregates spatio-temporal point cloud information, particularly when temporal correspondences are absent.

Further, *PEACH*'s material predictions and simulation accuracy are comparable to those of *MaNGO*, demonstrating that the learned material embeddings from point clouds are competitive to the ones learned from meshes. Appendix C provides exemplary trajectory visualizations, which show that *PEACH*'s simulations are consistent and visually mostly indistinguishable from those of *MaNGO*. These results confirm that *PEACH* provides accurate material identification and physically consistent trajectory prediction from point cloud sequences, bridging the gap to mesh-based methods.

5 Conclusion

We introduced Point cloud Encoding for Accurate Context Handling (*PEACH*), an in-context learning framework that adapts graph network simulators to unseen material properties using only sequences of point cloud observations. By decoupling material inference from mesh-based dynamics prediction, *PEACH* extends prior mesh-dependent approaches to more realistic observation settings while retaining the efficiency and accuracy of trajectory-level graph network simulators. Our experiments on simulated deformation environments demonstrate that *PEACH* achieves performance on par with mesh-based context methods, despite relying on significantly sparser input data.

While this work takes an important step toward real-world adaptive simulation, all evaluations are conducted using simulated point clouds. Thus, real-world sensing effects such as noise, occlusions, and domain shift are not yet considered.

ACKNOWLEDGMENTS

This work is part of the DFG AI Resarch Unit 5339 regarding the combination of physics-based simulation with AI-based methodologies for the fast maturation of manufacturing processes. The financial support by German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) is gratefully acknowledged. 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. This work has been supported by the German Federal Ministry of Research, Technology, and Space (BMFTR) under the Robotics Institute Germany (RIG).

REFERENCES

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

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

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 (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

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

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

Susanne C Brenner and L Ridgway Scott. *The mathematical theory of finite element methods*, volume 3. Springer, 2008.

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

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)*, pp. 77–85, 2017. doi: 10.1109/CVPR.2017.16.

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.

Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. The farthest point strategy for progressive image sampling. *IEEE transactions on image processing*, 6(9):1305–1315, 1997.

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.

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 (ed.), *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, volume 11 of *Studies in Mechanobiology, Tissue Engineering and Biomaterials*, pp. 283–321. Springer, June 2012. doi: 10.1007/8415_2012_125. URL https://hal.inria.fr/hal-00681539.

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.

Balázs Gyenes, Nikolai Franke, Philipp Becker, and Gerhard Neumann. Pointpatchrl - masked reconstruction improves reinforcement learning on point clouds. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pp. 5236–5253. PMLR, 2024. URL `https://proceedings.mlr.press/v270/gyenes25a.html`.

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

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

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

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.

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`.

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.

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`.

Junuthula Narasimha Reddy. *Introduction to the finite element method*. McGraw-Hill Education, 2019.

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*, pp. 8459–8468. PMLR, 2020.

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`.

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.

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.

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`.

Qingqing Zhao, David B. Lindell, and Gordon Wetzstein. Learning to solve pde-constrained inverse problems with graph networks. 2022.

A Methods and Hyperparameters

A.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 Fan et al. (2021) 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 furthest point sampling 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 (Charles et al., 2017) 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 (Sanchez-Gonzalez et al., 2020) 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.

***PEACH* Transformer Encoder.** For our method, we employ a spatio-temporal furthest point sampling to select the center points. All timesteps are first normalized to the unit interval and then scaled by a factor of 16, producing a temporal scaling comparable to the spatial normalization. For each center point, we extract its 16 nearest neighbors and apply a PointNet layer to the resulting patch. Temporal and spatial Fourier encodings are then added to these embeddings to incorporate both spatial and temporal information.

We implement an Attention Pooling module that combines a Transformer encoder with a cross-attention-based pooling mechanism. Input tokens are first processed through a Transformer encoder using multi-head self-attention, allowing them to exchange information and build contextual representations. The resulting token embeddings are then passed to an Attention Pooling layer, where a single learnable query token attends to the encoded sequence via cross-attention, summarizing the variable-length input into a fixed set of output tokens.

A.2 Mesh Encoder and Simulator

Mango Mesh Encoder. We closely follow the architecture of Dahlinger et al. (2025) 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 (Cai et al., 2023) 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 Dahlinger et al. (2025), using the official code available at `https://github.com/ALRhub/mango`.

A.3 Hyperparameters

We provide a list of the used hyperparameters in Table 1.

Table 1: List of the used hyperparameters

Parameter	Value
Furthest Point 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 (Loshchilov & Hutter, 2019)
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

B Datasets

This section provides detailed information about the datasets used in our experiments. The key characteristics of each dataset are summarized in Table 2. Each task consists of 16 simulation trials with various initial conditions.

Table 2: 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

B.1 Deforming Block.

This environment was originally implemented by Linkerhägner et al. (2023) and uses Simulation Open Framework Architecture (SOFA) (Faure et al., 2012). 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.

B.2 Sheet Deformation.

This dataset considers the deformation of a thin square two dimensional deformable sheet with edge length $l = 140$ mm embedded in three dimensional space. 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

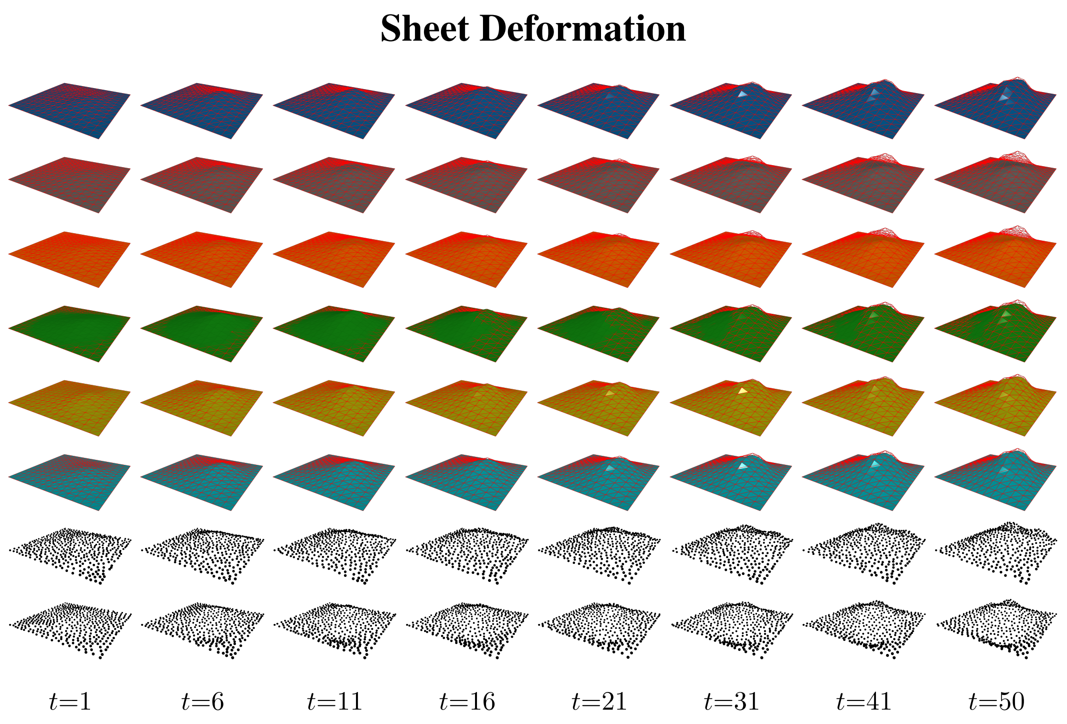

Figure 3: Predicted simulation of a `Sheet Deformation` test task by PEACH Encoder (blue), No Context Encoder (grey), PSTNet Encoder (orange), GNN Encoder (green), Oracle Encoder (yellow), and MaNGO Mesh Encoder (turquoise). All visualizations show the colored **predicted mesh**, a **collider**, and a **wireframe** (red) of the ground-truth simulation. The last two rows show exemplary point cloud sequences from the context set.

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.

C VISUALIZATIONS

We provide additional qualitative results comparing all evaluated methods on both datasets. Figure 3 shows qualitative trajectory predictions for the `Sheet Deformation` dataset, while Figure 4 presents corresponding results for the `Deformable Block` dataset. These visualizations illustrate qualitative differences between the methods and complement the quantitative results reported in the main paper. They also present two exemplary point cloud sequences from the same task used by the models to estimate material properties.

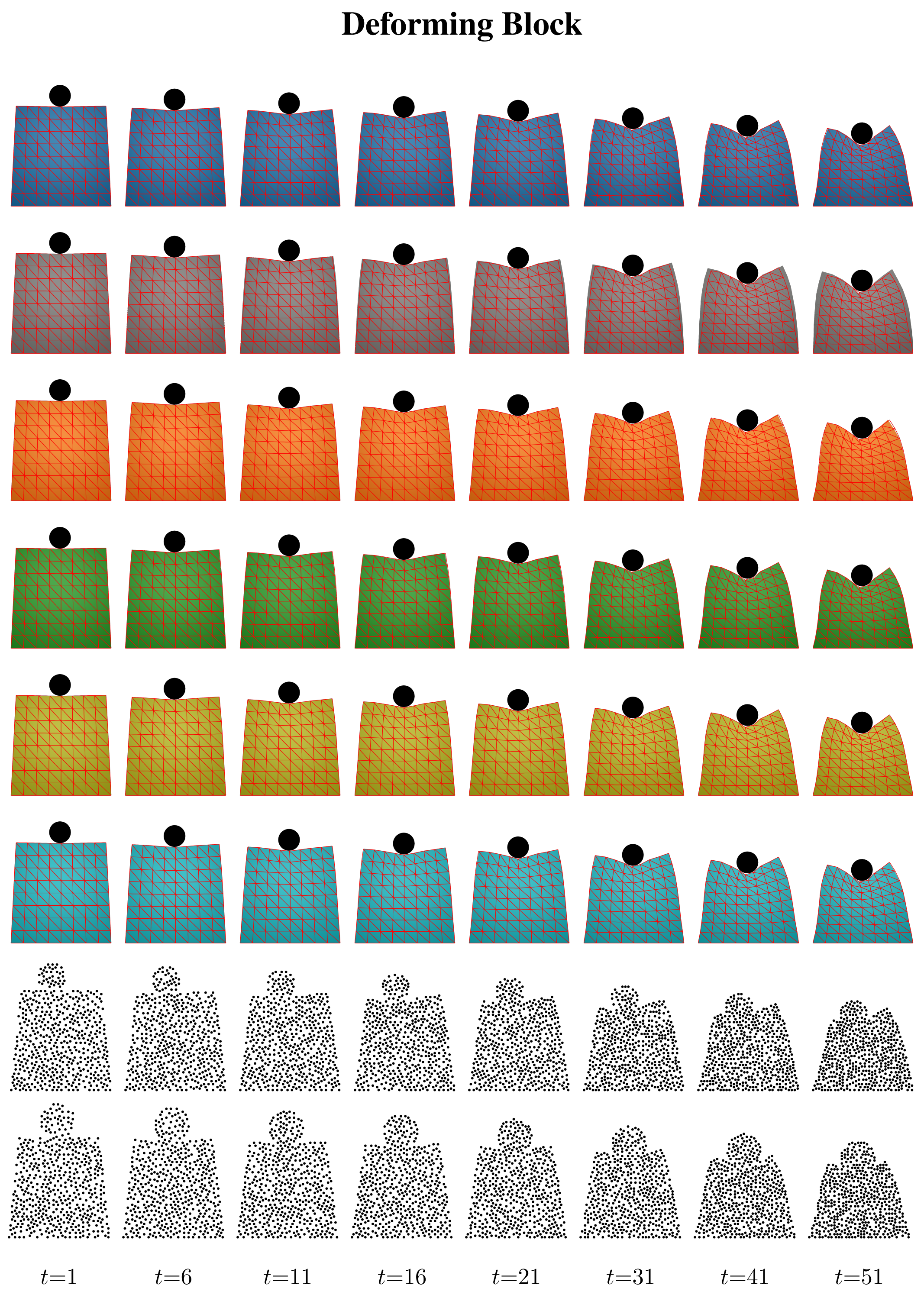

Figure 4: Predicted simulation of a `Deforming Block` test task by PEACH Encoder (blue), No Context Encoder (grey), PSTNet Encoder (orange), GNN Encoder (green), Oracle Encoder (yellow), and MaNGO Mesh Encoder (turquoise). All visualizations show the colored **predicted mesh**, a **collider**, and a **wireframe** (red) of the ground-truth simulation. The last two rows show exemplary point cloud sequences from the context set.