Particle-Grid Neural Dynamics for Learning Deformable Object Models from Depth Images

Kaifeng Zhang[1], Baoyu Li[2], Kris Hauser[2], Yunzhu Li[1]
[2]Columbia University, [2]University of Illinois Urbana-Champaign

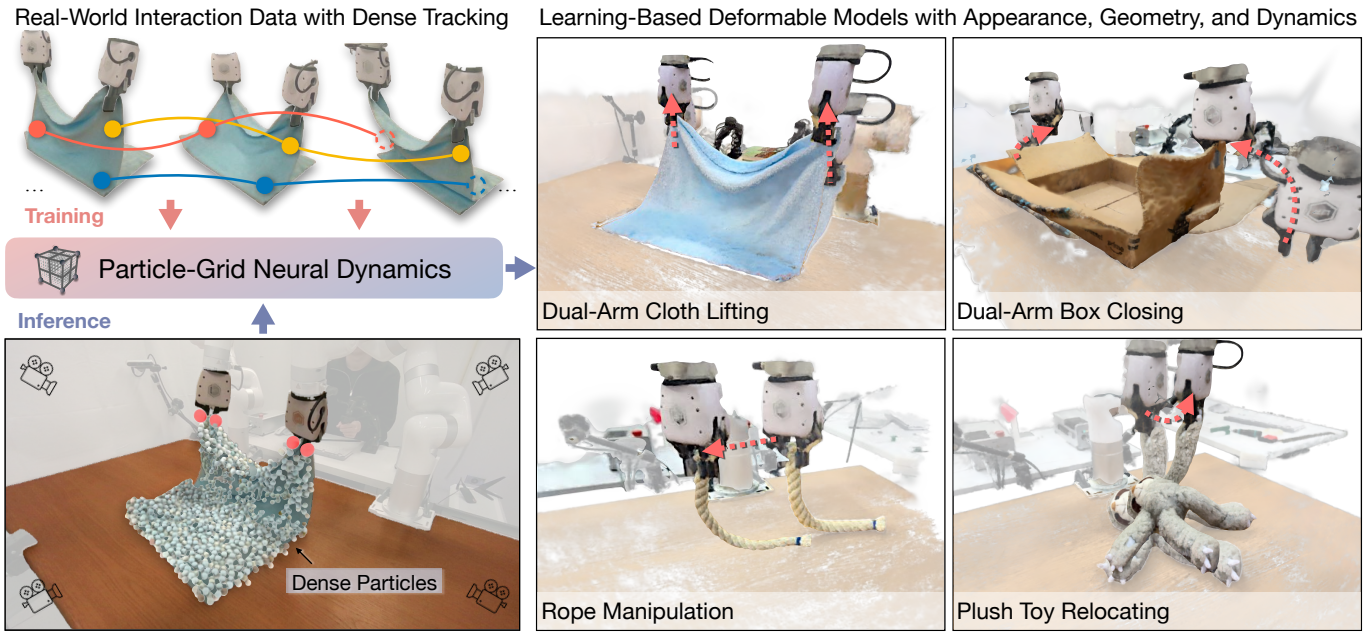


Fig. 1: Modeling deformable objects from RGB-D images presents a significant challenge due to occlusions and complex physical interactions. Our **Particle-Grid Neural Dynamics** framework learns the behavior of deformable objects directly from real-world observations. To train the model, we introduce a novel dense 3D tracking method that leverages foundational vision models for video tracking. The trained model predicts the motion of dense particles for future object states and robot-object interactions. We demonstrate the ability of Particle-Grid Neural Dynamics to model complex interactions across a diverse set of objects, including ropes, cloth, plush toy, box, and bread.

***Abstract*—Modeling the dynamics of deformable objects is challenging due to their diverse physical properties and the difficulty of estimating states from limited visual information. We address these challenges with a neural dynamics framework that combines object particles and spatial grids in a hybrid representation. Our particle-grid model captures global shape and motion information while predicting dense particle movements, enabling the modeling of objects with varied shapes and materials. Particles represent object shapes, while the spatial grid discretizes the 3D space to ensure spatial continuity and enhance learning efficiency. Coupled with Gaussian Splattings for visual rendering, our framework achieves a fully learning-based digital twin of deformable objects and generates 3D action-conditioned videos. Through experiments, we demonstrate that our model learns the dynamics of diverse objects—such as ropes, cloths, stuffed animals, and paper bags—from sparse-view RGB-D recordings of robot-object interactions, while also generalizing at the category level to unseen instances. Our approach outperforms state-of-the-art learning-based and physics-based simulators, particularly in scenarios with limited camera views. Furthermore, we showcase the utility of our learned models in model-based planning, enabling goal-conditioned object manipulation across a range of tasks.**

I. Introduction

Learning predictive models is crucial for a wide range of robotic tasks. In deformable object manipulation, an accurate predictive object dynamics model enables model-based planning, policy evaluation, and real-to-sim asset generation. However, developing dynamics models for deformable objects that are both accurate and generalizable remains a significant challenge. For example, physics-based simulators [12, 30] often struggle to generalize to the real world due to the inherent sim-to-real gap and the difficulties of system identification and state estimation. Meanwhile, video-based predictive models [9, 52] are computationally expensive, lack 3D spatial understanding, and are highly sensitive to viewpoint and appearance changes.

Recent work has focused on learning particle-based dynamics from RGB-D data, using GNNs to model particle sets as spatial graphs [45, 54]. While promising, these methods struggle with partial observations due to graph construction sensitivity and require carefully tuned message-passing steps to balance global context and smoothness. As a result, these approaches are often limited to simple simulated environments or real-world objects with trivially defined geometries, e.g., nearest neighbors.

To overcome these limitations, we propose particle-grid neural dynamics, a hybrid model combining object particles with fixed spatial grids. The model takes particle kinematic states as input and predicts velocity fields at grid points, using a global

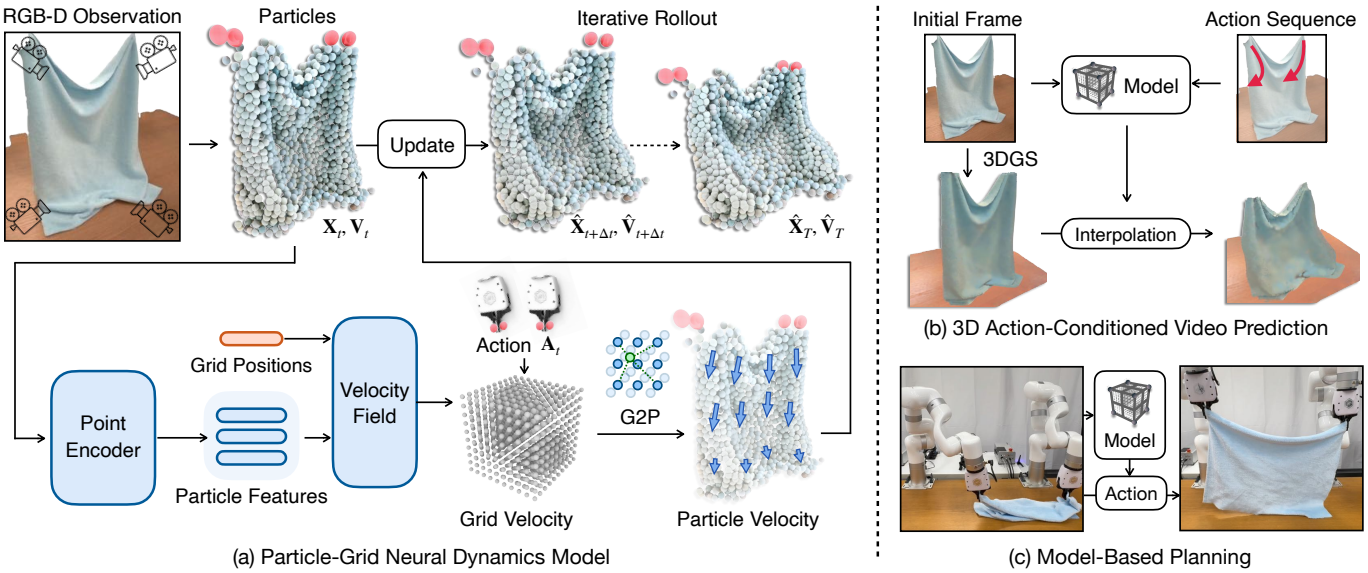


Fig. 2: **Overview of proposed framework: Particle-Grid Neural Dynamics. (a)** A diagram of our dynamics model. Given particles fused from multi-view depth images as input, our model predicts dense per-particle motion by first using a point encoder to extract particle features and predict the velocity field, which is then transformed into a grid representation to estimate the velocity distribution in 3D space. The model updates particle positions with the predicted velocity to perform iterative rollouts. **(b)** Our framework enables 3D action-conditioned video prediction by reconstructing objects with 3D Gaussian Splatting and interpolating the 6DoF transformation of Gaussian kernels using the predicted particle motions. **(c)** The model can be integrated into model-based planning frameworks to generate plausible motions for manipulating deformable objects.

point cloud encoder to extract comprehensive features. This design enhances robustness to partial observations, supports denser particle inputs, and regularizes predictions via the grid structure for spatial continuity and efficiency. By combining Lagrangian (particles) and Eulerian (grids) representations, our model draws inspiration from physics-based simulators [41, 12], while leveraging neural networks to generalize across materials and operate under partial observability.

Our model is trained entirely from RGB-D videos of robot-object interactions. We introduce a 3D particle fusion and tracking pipeline that uses foundational vision models [18, 36, 15] to estimate segmentation masks and pixel tracks, which are fused into persistent 3D trajectories for training.

Experiments show our method accurately simulates a wide range of deformable objects, including ropes, cloth, plush toys, boxes, paper bags, and bread, and outperforms state-of-the-art baselines. It also integrates seamlessly with appearance reconstruction techniques like 3D Gaussian Splatting (3DGS) [16], enabling photorealistic renderings with improved accuracy. We further demonstrate strong performance under sparse-view settings and validate its utility for real-world manipulation via integration with Model Predictive Control (MPC).

II. Methods

A. Dynamics Model

1) State and Action Representation: In the dynamics model, object particles are denoted by $\mathbf{X}_t \in \mathbb{R}^{3\times n}$, where n is the number of particles, and t is the time. The velocity of the particles, $\mathbf{V}_t \in \mathbb{R}^{3\times n}$ is defined as the time derivative of $\mathbf{X}$ at time t. The action $\mathbf{A}_t$ represents the external effects caused to the object by the robot at time t.

2) Dynamics Function: We incorporate historical states from previous h timesteps as additional inputs, and approximate the function $\mathbf{f}$ using a neural network parameterized by θ:

$$\hat{\mathbf{X}}_{t+\Delta t} = \mathbf{X}_t + \Delta t \cdot \mathbf{f}_\theta(\mathbf{X}_{t-h\Delta t:t}, \mathbf{V}_{t-h\Delta t:t}, \mathbf{A}_t). \quad (1)$$

Our model utilizes a hybrid particle-grid representation to inject inductive bias related to spatial continuity and local information integration. The particle-grid dynamics function is comprised of the following components:

$$\mathbf{f}_\theta = \mathbf{h}^{\text{G2P}} \cdot \mathbf{g}^{\text{grid}} \cdot \mathbf{f}_\psi^{\text{field}} \cdot \mathbf{f}_\phi^{\text{feature}}, \quad (2)$$

where $\mathbf{f}_\phi^{\text{feature}}$ is the neural network-based point encoder for extracting feature from the input particles, $\mathbf{f}_\psi^{\text{field}}$ is the neural network-based function for parameterizing a neural velocity field based on the extracted features, and $\mathbf{g}^{\text{grid}}$ is a grid velocity editing (GVE) control method to encode collision surfaces and robot gripper movements, and $\mathbf{h}^{\text{G2P}}$ is the grid-to-particle integration function for calculating particle velocities from grid-based velocity field.

B. Data Collection and Training

We collect training data through teleoperation and automatic annotation using foundation models. Specifically, we record multi-view RGB-D videos of random robot-object interactions. For each camera view, we apply Segment-Anything [18, 36] to extract persistent object masks and use CoTracker [15] to generate 2D trajectories. Using depth information, we perform inverse projection to map these 2D velocities into 3D, resulting in multi-view fused point clouds with persistent particle tracking. To train the dynamics function $\mathbf{f}_\theta$, the loss function is defined as the mean squared error (MSE) between the predicted and actual particle positions.

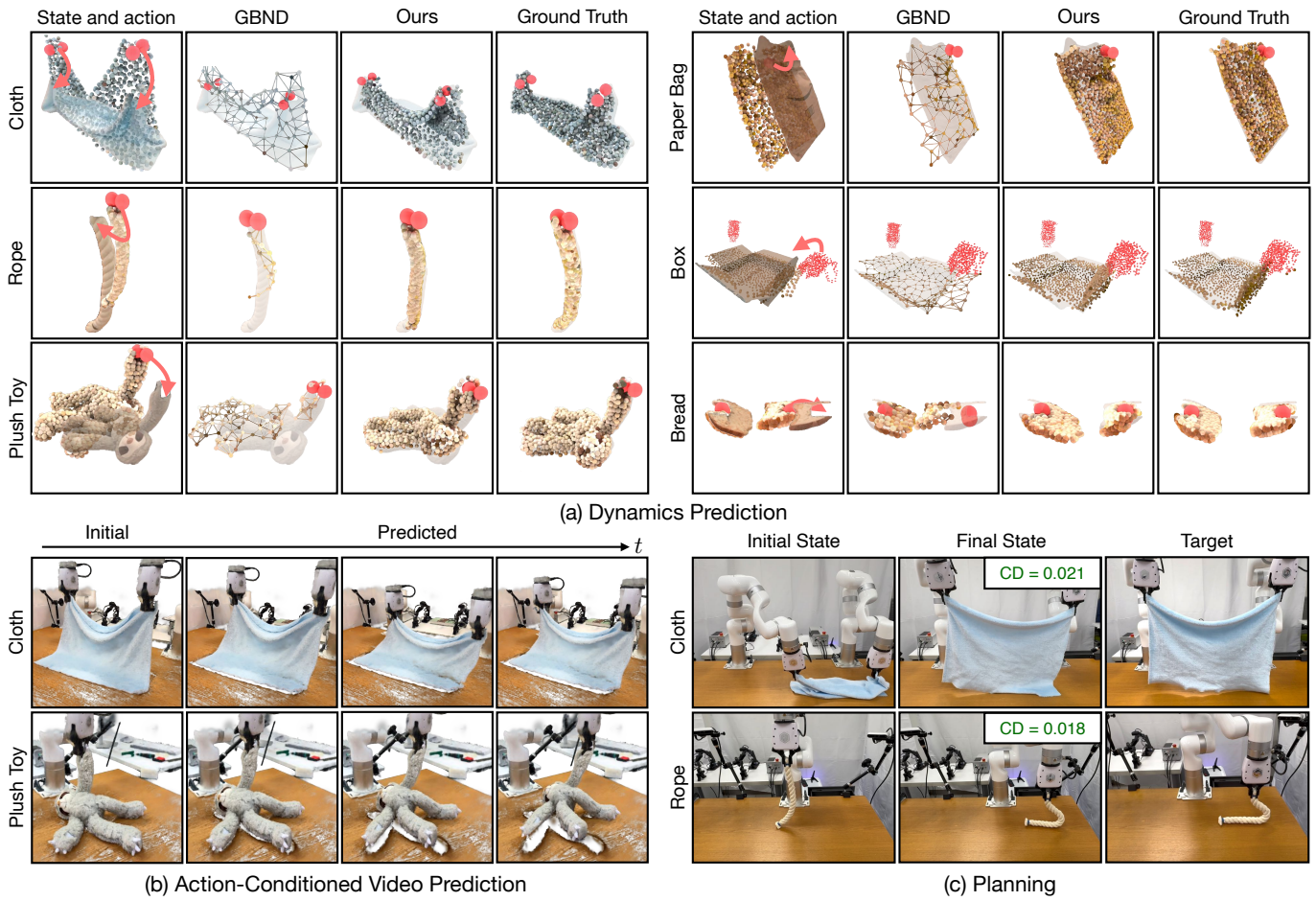


Fig. 3: **Qualitative Comparisons on Dynamics Prediction.** Given initial states and actions, we show the prediction results of the GBND baseline compared to our particle-grid neural dynamics model. The red spheres indicate the position and orientation of robot grippers. We overlay the predictions with ground truth final state images to highlight the prediction errors. Our model's predictions are more aligned with the ground truth, offering higher-density particle predictions and fewer artifacts compared to the baseline.

C. Action-Conditioned Video Prediction and Planning

Our predictions can be integrated with 3D Gaussian Splatting to achieve a realistic action-conditioned video prediction. To transform Gaussians between frames, we first apply the dynamics model to the point cloud set $\mathbf{X}$, yielding the next-frame prediction $\hat{\mathbf{X}}$. The 6-DoF motions of the Gaussian kernels are interpolated using Linear Blend Skinning (LBS) [42].

Additionally, our model can be integrated with MPC for model-based planning. Given RGB-D captures, we obtain object particles through segmentation, inverse projection into 3D, and downsampling. We use the Chamfer Distance between the predicted state and the target state as the cost function, and apply the MPPI [47] trajectory optimization algorithm.

III. Experiments

We conduct data collection and experiments using a bimanual xArm setup. The objects used in the experiments include rope, cloth, plush toys, paper bags, boxes, and bread.

A. Dynamics Learning and Prediction

We visualize the particle prediction results of our method alongside the baseline method, Graph-Based Neural Dynamics (GBND) [53, 54], in Fig. 3. Our method's predictions align more closely with the ground truth images and also exhibit higher resolution. In contrast, GBND predicts inadequate particle motions for objects like cloth, plush toy, box, bread, and generates artifacts for objects like rope.

B. Action-Conditioned Video Prediction and Planning

In Fig. 3b, we show the action-conditioned video prediction results by reconstructing the object using 3D Gaussian Splatting and deforming the Gaussians with dynamics predictions. The scenes are reconstructed using video scans of a static workspace. The results demonstrate that our model can generate 3D action-conditioned video predictions with high visual fidelity.

C. Planning

In planning experiments, we evaluate the model's ability to integrate with MPC to generate actions for manipulating deformable objects. Fig. 3c shows that our model produces results that are visually close to the target. For example, in the cloth lifting task, our method successfully lifts the cloth from a folded initial state, using both arms simultaneously. In rope manipulation, our method accurately bends the rope by pressing downward.

References

[1] Jad Abou-Chakra, Krishan Rana, Feras Dayoub, and Niko Suenderhauf. Physically embodied gaussian splatting: A realtime correctable world model for robotics. In *8th Annual Conference on Robot Learning*, 2024. 7

[2] Bo Ai, Stephen Tian, Haochen Shi, Yixuan Wang, Cheston Tan, Yunzhu Li, and Jiajun Wu. Robopack: Learning tactile-informed dynamics models for dense packing. *Robotics: Science and Systems (RSS)*, 2024. 7

[3] Kelsey R Allen, Tatiana Lopez Guevara, Yulia Rubanova, Kim Stachenfeld, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Graph network simulators can learn discontinuous, rigid contact dynamics. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1157–1167. PMLR, 14–18 Dec 2023. 7

[4] Dominik Bauer, Zhenjia Xu, and Shuran Song. Doughnet: A visual predictive model for topological manipulation of deformable objects. *European Conference on Computer Vision (ECCV)*, 2024. 7

[5] Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. Discrete elastic rods. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781450301121. doi: 10.1145/1399504.1360662. 7

[6] Yizhou Chen, Yiting Zhang, Zachary Brei, Tiancheng Zhang, Yuzhen Chen, Julie Wu, and Ram Vasudevan. Differentiable discrete elastic rods for real-time modeling of deformable linear objects, 2024. 7

[7] Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Diffpd: Differentiable projective dynamics. *ACM Trans. Graph.*, 41(2), nov 2021. ISSN 0730-0301. doi: 10.1145/3490168. URL https://doi.org/10.1145/3490168. 7

[8] Bardienus P Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Mike Zheng Shou, Shuran Song, and Jeffrey Ichnowski. Md-splatting: Learning metric deformation from 4d gaussians in highly deformable scenes. *arXiv preprint arXiv:2312.00583*, 2023. 7

[9] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017. 1

[10] Marcos García, César Mendoza, Luis Pastor, and Angel Rodríguez. Optimized linear fem for modeling deformable objects. *Comput. Animat. Virtual Worlds*, 17(3–4): 393–402, July 2006. ISSN 1546-4261. 7

[11] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Kumar Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. Visuospatial foresight for multi-step, multi-task fabric manipulation. *arXiv preprint arXiv:2003.09044*, 2020. 7

[12] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018. 1, 2, 7, 9, 11

[13] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation, 2020. 7

[14] Isabella Huang, Yashraj Narang, Ruzena Bajcsy, Fabio Ramos, Tucker Hermans, and Dieter Fox. Defgraspnets: Grasp planning on 3d fields with graph neural nets, 2023. 7

[15] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proc. arXiv:2410.11831*, 2024. 2, 9, 15

[16] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. 2, 7, 14, 15

[17] Thomas Kipf, Elise Van der Pol, and Max Welling. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019. 7

[18] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 2, 9, 15

[19] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *ICLR*, 2019. 7

[20] Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 7

[21] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, 2020. 7

[22] Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, pages 256–266. PMLR, 2022. 7

[23] Fei Liu, Entong Su, Jingpei Lu, Mingen Li, and Michael C. Yip. Robotic manipulation of deformable rope-like objects using differentiable compliant position-based dynamics. *IEEE Robotics and Automation Letters*, 8(7):3964–3971, 2023. doi: 10.1109/LRA.2023.3264766. 7

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

7

[25] Ziang Liu, Genggeng Zhou, Jeff He, Tobia Marcucci, Li Fei-Fei, Jiajun Wu, and Yunzhu Li. Model-based control with sparse neural dynamics. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 7

[26] Alberta Longhini, Marco Moletta, Alfredo Reichlin, Michael C Welle, David Held, Zackory Erickson, and Danica Kragic. Edo-net: Learning elastic properties of deformable objects from graph dynamics. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3875–3881. IEEE, 2023. 7

[27] Alberta Longhini, Marcel Büsching, Bardienus Pieter Duisterhof, Jens Lundell, Jeffrey Ichnowski, Mårten Björkman, and Danica Kragic. Cloth-splatting: 3d state estimation from RGB supervision for deformable objects. In *8th Annual Conference on Robot Learning*, 2024. 7

[28] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 7, 14

[29] Pingchuan Ma, Peter Yichen Chen, Bolei Deng, Joshua B Tenenbaum, Tao Du, Chuang Gan, and Wojciech Matusik. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *International Conference on Machine Learning*, pages 23279–23300. PMLR, 2023. 7, 11

[30] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 33 (4):1–12, 2014. 1, 7

[31] Jing-Chen Peng, Shaoxiong Yao, and Kris Hauser. 3d force and contact estimation for a soft-bubble visuotactile sensor using fem. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024. doi: 10.1109/ICRA57147.2024.10610233. 7

[32] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020. 7

[33] Kavya Puthuveetil, Sasha Wald, Atharva Pusalkar, Pratyusha Karnati, and Zackory Erickson. Robust body exposure (robe): A graph-based dynamics modeling approach to manipulating blankets over people. *IEEE Robotics and Automation Letters*, 2023. 7

[34] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 8

[35] Ri-Zhao Qiu, Ge Yang, Weijia Zeng, and Xiaolong Wang. Language-driven physics-based scene synthesis and editing via feature splatting. In *European Conference on Computer Vision (ECCV)*, 2024. 7

[36] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024. 2, 9

[37] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020. 7

[38] Keyi Shen, Jiangwei Yu, Huan Zhang, and Yunzhu Li. Bab-nd: Long-horizon motion planning with branch-and-bound and neural dynamics, 2024. 7

[39] Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks. *arXiv preprint arXiv:2205.02909*, 2022. 7

[40] Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. *arXiv preprint arXiv:2306.14447*, 2023. 7

[41] Deborah Sulsky, Shi-Jian Zhou, and Howard L Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer physics communications*, 87(1-2):236–252, 1995. 2, 7, 8

[42] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3):80–es, jul 2007. ISSN 0730-0301. doi: 10.1145/1276377.1276478. URL https://doi.org/10.1145/1276377.1276478. 3, 9, 14

[43] Changhao Wang, Yuyou Zhang, Xiang Zhang, Zheng Wu, Xinghao Zhu, Shiyu Jin, Te Tang, and Masayoshi Tomizuka. Offline-online learning of deformation model for cable manipulation with graph neural networks. *IEEE Robotics and Automation Letters*, 7(2):5544–5551, 2022. doi: 10.1109/LRA.2022.3158376. 7

[44] Yixuan Wang, Yunzhu Li, Katherine Driggs-Campbell, Li Fei-Fei, and Jiajun Wu. Dynamic-Resolution Model Learning for Object Pile Manipulation. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.047. 7

[45] William F. Whitney, Tatiana Lopez-Guevara, Tobias Pfaff, Yulia Rubanova, Thomas Kipf, Kimberly Stachenfeld, and Kelsey R. Allen. Learning 3d particle-based simulators from rgb-d videos, 2023. 1, 7

[46] William F. Whitney, Jacob Varley, Deepali Jain, Krzysztof Choromanski, Sumeet Singh, and Vikas Sindhwani. Modeling the real world with high-density visual particle dynamics, 2024. 7

[47] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017. 3, 10

[48] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024. 14

[49] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198*, 2023. 7

[50] Shangjie Xue, Shuo Cheng, Pujith Kachana, and Danfei Xu. Neural field dynamics model for granular object piles manipulation. In *Conference on Robot Learning*, pages 2821–2837. PMLR, 2023. 7

[51] Mengyuan Yan, Yilin Zhu, Ning Jin, and Jeannette Bohg. Self-supervised learning of state estimation for manipulating deformable linear objects. *IEEE Robotics and Automation Letters*, 5(2):2372–2379, 2020. doi: 10.1109/LRA.2020.2969931. 7

[52] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023. 1

[53] Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Adaptigraph: Material-adaptive graph-based neural dynamics for robotic manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024. 3, 7, 15

[54] Mingtong Zhang, Kaifeng Zhang, and Yunzhu Li. Dynamic 3d gaussian tracking for graph-based neural dynamics modeling. *arXiv preprint arXiv:2410.18912*, 2024. 1, 3, 7, 9, 11, 14

[55] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T. Freeman. PhysDreamer: Physics-based interaction with 3d objects via video generation. *arxiv*, 2024. 7

[56] Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and simulation of elastic objects with spring-mass 3d gaussians. *arXiv preprint arXiv:2403.09434*, 2024. 7

[57] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning, 2024. 7

APPENDIX

Contents

APPENDIX A RELATED WORK

A.1 *Physics-Based Deformable Modeling.*

The simulation of deformable objects is essential for advancing both the modeling and robotic manipulation of soft, flexible materials. Researchers have introduced various analytical physics-based approaches, including, but not limited to, the mass-spring system [24, 20], the Finite Element Method (FEM) [10, 31, 7], the Discrete Elastic Rods (DER) [5, 6], the Position-Based Dynamics (PBD) [30, 21, 23], and the Material Point Method (MPM) [41, 12, 13, 29]. However, real-world analytical deformable modeling remains challenging due to the difficulty in property identification and state estimation. While our method uses a hybrid particle-grid representation similar to MPM, we leverage neural networks as message integrators and reduce dependence on full-state information as input. This enhances robustness to partial observations and enables the handling of a wide variety of deformable objects without requiring predefined material-specific constitutive laws. Recently, the emergence of 3D Gaussian Splatting (3DGS) [16, 28, 8] has enabled the fusion of Gaussian Splatting reconstructions with physics-based models to simulate the dynamics of deformable objects [49, 35, 56, 55, 1]. In our particle-grid neural dynamics model, the particle motion predictions can also be integrated with Gaussian Splatting reconstructions, facilitating the simultaneous modeling and rendering of deformable objects, purely learned from real data.

A.2 *Learning-Based Deformable Modeling.*

Learning-based dynamics models, which use deep neural networks to model the future evolution of dynamic systems, have demonstrated effectiveness across various robotic tasks [11, 51, 17, 57, 50, 4]. Among these learning-based methods, Graph-Based Neural Dynamics (GBND) have shown great promise, as they explicitly model spatial relational biases within complex physical systems [19, 32, 37, 3, 45]. Previous research has investigated the use of GBND across a range of material types, such as rigid bodies [14, 25, 2], plasticine [39, 40], fabrics [22, 26, 33, 27], ropes [43], and granular piles [44, 38]. Beyond simulation and single-material scenarios, GBND has also demonstrated flexibility and generalization in modeling diverse materials using a unified framework [53, 54]. However, these approaches often operate on spatially sparse graph vertices, rely on expert knowledge to determine graph connectivity, and do not consider partial observations. For learning dense particle dynamics, Whitney et al. [46] propose transformer-based backbones for higher computational efficiency. However, their work mainly focuses on modeling rigid objects for grasping and pushing tasks. In contrast, our work emphasizes deformable object modeling using a hybrid particle-grid neural dynamics framework, achieving dense particle prediction while remaining robust to incomplete observations and flexible for diverse types of deformable objects with distinct physical properties.

APPENDIX B METHODS

Our Particle-Grid Neural Dynamics framework models the dynamics of objects represented by a set of particles. The core of this framework is a dynamics function that predicts the future motion of each particle based on its current and historical states, as well as the current action of the robot's end effector. A detailed description of the model is provided in Section B.1 and B.2. We introduce the data collection pipeline and the model's training method in Section B.3. Additionally, we explore the integration of Particle-Grid Neural Dynamics with 3D Gaussian Splatting for 3D video rendering, as discussed in Section B.4. The application of this model within a Model Predictive Control (MPC) framework is covered in Section B.5. An overview of our method is also provided in Fig. 2.

B.1 *Particle-Grid Neural Dynamics*

B.11 State Representation: We intend to learn a particle-based dynamics model, which represents the target object as a collection of particles $\mathbf{X}_t \in \mathbb{R}^{3\times n}$, where n is the number of particles, and t is the time. The velocity of the particles, $\mathbf{V}_t \in \mathbb{R}^{3\times n}$ is defined as the time derivative of $\mathbf{X}$ at time t.

B.12 Action Representation: The action $\mathbf{A}_t$ represents the external effects caused to the object by the robot at time t. We define $\mathbf{A}_t$ as

$$\mathbf{A}_t = (\mathbf{y}, \mathbf{T}_t, \dot{\mathbf{T}}_t, \mathbf{o}_t), \tag{3}$$

namely, the combination of the end-effector type $\mathbf{y}$, the 6-DoF pose of the end effector $\mathbf{T}_t$, its time derivative $\dot{\mathbf{T}}_t$, and the end-effector state, specifically the open distance of the gripper $\mathbf{o}_t$, a 1D variable.

B.13 Dynamics Function: We consider the change of state caused by the state itself (e.g., objects falling due to gravity) and the robot's actions (e.g., robots grasping an object thus making it move) in the object's dynamic functions:

$$\hat{\mathbf{V}}_{t+\Delta t} = \mathbf{f}(\mathbf{X}_t, \mathbf{V}_t, \mathbf{A}_t), \tag{4}$$

where $\mathbf{f}$ is the dynamics function predicting the state evolution. Since the particle representation typically cannot capture the full state information (e.g., internal stress or contact mode with other objects), a common practice is to incorporate historical states as additional inputs for making predictions:

$$\hat{\mathbf{V}}_{t+\Delta t} = \mathbf{f}(\mathbf{X}_{t-h\Delta t:t}, \mathbf{V}_{t-h\Delta t:t}, \mathbf{A}_t), \tag{5}$$

where h is the history window size. In our neural dynamics model, we approximate the function $\mathbf{f}$ using a neural network parameterized by θ and derive the next particle positions by applying forward Euler time integration:

$$\hat{\mathbf{X}}_{t+\Delta t} = \mathbf{X}_t + \Delta t \cdot \mathbf{f}_\theta(\mathbf{X}_{t-h\Delta t:t}, \mathbf{V}_{t-h\Delta t:t}, \mathbf{A}_t). \tag{6}$$

B.14 The Particle-Grid Model: Learning the neural dynamics parameters θ with an end-to-end neural network usually leads to unstable predictions due to accumulative error. Motivated by the use of hybrid Lagrangian-Eulerian representations in MPM [41], our model also utilizes a hybrid particle-grid representation to inject inductive bias related to spatial continuity and local information integration. Specifically, we define a uniformly distributed grid in the space as

$$\mathbf{G}_{l_x,l_y,l_z,\delta} = \{(k_x\delta, k_y\delta, k_z\delta) | k_i \in [l_i], \forall i \in \{x, y, z\}\}. \tag{7}$$

The parameters l_x, l_y, l_z, δ control the spatial limits and resolution of the grid. Empirically, we set δ to $1\,\mathrm{cm}$ or $2\,\mathrm{cm}$ to balance computational cost and resolution. To enforce translational invariance during dynamics prediction, we always translate the positions of the particles and the robot end effector to the volume defined by $\mathbf{G}_{l_x,l_y,l_z,\delta}$.

The particle-grid dynamics function is defined by the following components:

$$\mathbf{f}_\theta = \mathbf{h}^{\text{G2P}} \cdot \mathbf{g}^{\text{grid}} \cdot \mathbf{f}_\psi^{\text{field}} \cdot \mathbf{f}_\phi^{\text{feature}}, \tag{8}$$

where $\mathbf{f}_\phi^{\text{feature}}$ is the neural network-based point encoder for extracting feature from the input particles (Sec. B.21), $\mathbf{f}_\psi^{\text{field}}$ is the neural network-based function for parameterizing a neural velocity field based on the extracted features (Sec. B.22), and $\mathbf{g}^{\text{grid}}$ is a grid velocity editing (GVE) control method to encode collision surfaces and robot gripper movements (Sec. B.24), and $\mathbf{h}^{\text{G2P}}$ is the grid-to-particle integration function for calculating particle velocities from grid-based velocity field (Sec. B.23).

B.2 Model Components

B.21 Point Encoder: The point encoder encodes particle positions and velocities to per-particle latent features $\mathbf{Z}_t \in \mathbb{R}^{d\times n}$, where d is the feature dimension:

$$\mathbf{Z}_t = \mathbf{f}_\phi^{\text{feature}}(\mathbf{X}_{t-h\Delta t:t}, \mathbf{V}_{t-h\Delta t:t}). \tag{9}$$

We use PointNet [34] as the encoder. The encoder captures global information from the set of all particles, including the object's shape and the historical motion of the particles, which are used to implicitly infer the object's physical properties and dynamic state. This is essential for handling incomplete observations, where the feature extraction network must extract occlusion-robust features for subsequent velocity decoding.

B.22 Neural Velocity Field: In this step, we use a neural implicit function $\mathbf{f}_\psi^{\text{field}}$ to predict a spatial velocity grid, at time t, from the extracted point features. For $g \in \mathbf{G}_{l_x,l_y,l_z,\delta}$, the function $\mathbf{f}_\psi^{\text{field}}$ is instantiated as an MLP that takes the grid locations $\mathbf{x}_g$ and the corresponding locality-aware feature $\mathbf{z}_{g,t}$ as inputs, then predict per-grid velocity vector $\mathbf{v}_{g,t}$ by

$$\hat{\mathbf{v}}_{g,t} = \mathbf{f}_\psi^{\text{field}}(\gamma(\mathbf{x}_g), \mathbf{z}_{g,t}), \tag{10}$$

where γ is the sinusoidal positional encoding, and the locality-aware feature $\mathbf{z}_{g,t}$ is defined as the average pooling of particle features within the neighborhood of grid location:

$$\mathbf{z}_{g,t} = \frac{\sum_{p\in\mathcal{N}_r(\mathbf{X}_t,\mathbf{x}_g)} \mathbf{z}_{p,t}}{|\mathcal{N}_r(\mathbf{X}_t, \mathbf{x}_g)|}, \tag{11}$$

where $\mathcal{N}_r(\mathbf{X}_t, \mathbf{x}_g)$ is the set of indices of particles within $\mathbf{X}_t$ whose positions are within radius r of the grid location $\mathbf{x}_g$. By incorporating the radius hyperparameter r, we can control the number of particle features a grid point attends to, thus encouraging the network to predict velocities that are dependent on local geometry. Empirically, we set $r = 0.2\,\mathrm{m}$.

B.23 G2P: After calculating the grid's velocities, we transfer from the Eulerian grid to Lagrangian particles via spline interpolation. Following MPM, we utilize a continuous B-spline kernel to transfer grid velocities $\hat{\mathbf{v}}_{g,t}$ to particle velocities $\hat{\mathbf{v}}_{p,t}$:

$$\hat{\mathbf{v}}_{p,t} = \sum_{g\in\mathbf{G}} \hat{\mathbf{v}}_{g,t} w_{pg,t}, \tag{12}$$

where the $w_{pg,t}$ is the value of the B-spline kernel defined on the grid position $\mathbf{x}_g$ and evaluated at the particle location $\mathbf{x}_{p,t}$. It assigns larger weights to closer grid-particle pairs, achieving smooth spatial interpolation. The predictions $\hat{\mathbf{V}} \in \mathbb{R}^{3\times n}$ serves as the final output of the dynamics function $\mathbf{f}_\theta$ and is used to perform time integration in Eq. 6.

B.24 Controlling Deformation: We present two methods for controlling deformations by interactions with external objects: Grid Velocity Editing (GVE) and Robot Particles (RP). GVE is inspired from MPM approaches and we use it for grasped interactions and object-ground interaction. Simply put, the operator $\mathbf{g}^{\text{grid}}$ changes the velocities on the grid to match physical constraints. For ground contact, we project velocity back from the contact surface and incorporate friction terms. To define the motion of a rigidly grasped point, we calculate

Method	Metric	Cloth	Rope	Plush	Box	Bag	Bread
MPM [12]	MSE ↓	$0.176_{\pm 0.107}$	$0.138_{\pm 0.072}$	$0.163_{\pm 0.148}$	–	$0.226_{\pm 0.026}$	$0.034_{\pm 0.014}$
GBND [54]		$0.077_{\pm 0.033}$	$0.062_{\pm 0.025}$	$0.078_{\pm 0.028}$	$0.045_{\pm 0.008}$	$0.030_{\pm 0.011}$	$0.031_{\pm 0.014}$
Particle		$0.059_{\pm 0.039}$	$0.061_{\pm 0.051}$	$0.060_{\pm 0.027}$	$0.025_{\pm 0.009}$	$0.021_{\pm 0.016}$	$0.038_{\pm 0.018}$
Ours		$\mathbf{0.045_{\pm 0.023}}$	$\mathbf{0.039_{\pm 0.032}}$	$\mathbf{0.043_{\pm 0.018}}$	$\mathbf{0.022_{\pm 0.007}}$	$\mathbf{0.016_{\pm 0.005}}$	$\mathbf{0.020_{\pm 0.011}}$
MPM [12]	CD ↓	$0.156_{\pm 0.091}$	$0.115_{\pm 0.054}$	$0.153_{\pm 0.207}$	–	$0.183_{\pm 0.032}$	$0.031_{\pm 0.010}$
GBND [54]		$0.083_{\pm 0.034}$	$0.073_{\pm 0.027}$	$0.064_{\pm 0.016}$	$0.062_{\pm 0.014}$	$0.042_{\pm 0.007}$	$0.031_{\pm 0.013}$
Particle		$0.051_{\pm 0.034}$	$0.059_{\pm 0.058}$	$0.043_{\pm 0.019}$	$0.032_{\pm 0.011}$	$0.025_{\pm 0.016}$	$0.044_{\pm 0.031}$
Ours		$\mathbf{0.043_{\pm 0.022}}$	$\mathbf{0.038_{\pm 0.036}}$	$\mathbf{0.033_{\pm 0.013}}$	$\mathbf{0.015_{\pm 0.003}}$	$\mathbf{0.021_{\pm 0.005}}$	$\mathbf{0.018_{\pm 0.012}}$
MPM [12]	EMD ↓	$0.093_{\pm 0.078}$	$0.081_{\pm 0.052}$	$0.092_{\pm 0.131}$	–	$0.110_{\pm 0.027}$	$0.021_{\pm 0.009}$
GBND [54]		$0.035_{\pm 0.017}$	$0.036_{\pm 0.016}$	$0.032_{\pm 0.010}$	$0.032_{\pm 0.008}$	$0.016_{\pm 0.005}$	$0.016_{\pm 0.008}$
Particle		$0.029_{\pm 0.024}$	$0.036_{\pm 0.037}$	$0.025_{\pm 0.013}$	$0.018_{\pm 0.006}$	$0.012_{\pm 0.010}$	$0.023_{\pm 0.016}$
Ours		$\mathbf{0.022_{\pm 0.013}}$	$\mathbf{0.021_{\pm 0.021}}$	$\mathbf{0.017_{\pm 0.007}}$	$\mathbf{0.016_{\pm 0.005}}$	$\mathbf{0.009_{\pm 0.003}}$	$\mathbf{0.010_{\pm 0.008}}$

TABLE I: **Quantitative Results on Dynamics Prediction.** We compare our method with the Mateiral Point Method (MPM) [12], Graph-Based Neural Dynamics (GBND) [54], and a particle-based dynamics model without the grid representation. We report the mean and standard deviation of the prediction error over a 3-second future horizon. The best results are highlighted in bold and blue.

the set of grid points $\mathbf{G}_{\text{grasp},t}$ within a distance a of the grasp center point $\mathbf{x}_{\text{grasp},t}$. For each point $g \in \mathbf{G}_{\text{grasp},t}$, we modify the velocities as follows:

$$\mathbf{v}_{g,t} = \omega_t \times (\mathbf{x}_g - \mathbf{x}_{\text{grasp},t}) + \dot{\mathbf{x}}_{\text{grasp},t} \tag{13}$$

where ω_t is the angular velocity of the gripper at time t.

The Robot Particles method allows us to model nonprehensile actions for the Box example in which the object is pushed. Here, we represent the robot gripper with additional particles that carry gripper action information, and fuse this into the object point cloud. Specifically, at each step, we augment the point cloud by

$$\mathbf{X}_t^{\text{aug}} = \mathbf{X}_t \cup \mathbf{X}_{\text{robot},t}, \tag{14}$$

$$\mathbf{V}_t^{\text{aug}} = \mathbf{V}_t \cup \mathbf{V}_{\text{robot},t}, \tag{15}$$

and model the particle-grid dynamics function on the augmented point cloud. This injects action information into particle features but does not explicitly force particles to move at a prescribed velocity, thus supporting nonprehensile manipulation. Our implementation samples points from the gripper shape and calculates their velocities based on the end-effector transformation from proprioception.

B.3 Data Collection and Training

We collect training data through teleoperation and automatic annotation using foundation models. Specifically, we record multi-view RGB-D videos of random robot-object interactions. For each camera view, we apply Segment-Anything [18, 36] to extract persistent object masks across the video. The segmented objects are then cropped and tracked over time using CoTracker [15], providing 2D trajectories. Using depth information, we perform inverse projection to map these 2D velocities into 3D, resulting in multi-view fused point clouds with persistent particle tracking.

With the collected tracking data, we define particle sets and their trajectories over a look-forward time window as $\mathbf{X}_{t-h\Delta t:t+K\Delta t} \in \mathbb{R}^{3\times n\times T}$, alongside corresponding robot actions $\mathbf{A}_{t-h\Delta t:t+K\Delta t}$, where K is the horizon length hyperparameter. Empirically, we set $h = 2$ and $K = 5$. Model training begins from a given point cloud at time t, followed by iterative dynamics model rollouts for K steps. Since the dynamics function $\mathbf{f}_\theta$ is fully differentiable, we optimize the network parameters ϕ and ψ using gradient descent. The loss function is defined as the mean squared error (MSE) between the predicted and actual particle positions:

$$\mathcal{L} = \sum_{i=1}^{K} \|\hat{\mathbf{X}}_{t+i\Delta t} - \mathbf{X}_{t+i\Delta t}\|_2^2, \tag{16}$$

where $\hat{\mathbf{X}}_{t+i\Delta t}$ is the predicted particle positions at step i.

B.4 Rendering and Action-Conditioned Video Prediction

Our predictions can be integrated with 3D Gaussian Splatting (3DGS) to achieve a realistic rendering of the results. The 3DGS reconstruction of the object is defined as

$$\mathcal{G} = \{\mathbf{X}_{\text{GS}}, \mathbf{C}, \mathbf{R}_{\text{GS}}, \mathbf{S}, \mathbf{O}\}, \tag{17}$$

where $\mathbf{X}_{\text{GS}}$, $\mathbf{C}$, $\mathbf{R}_{\text{GS}}$, $\mathbf{S}$, and $\mathbf{O}$ represent the Gaussian kernels' center location, color, rotation, scale, and opacity, respectively. To transform Gaussians between frames, we first apply the dynamics model to the point cloud set $\mathbf{X}$, yielding the next-frame prediction $\hat{\mathbf{X}}$. The points $\hat{\mathbf{X}}$ can either be sampled from $\mathbf{X}_{\text{GS}}$ or obtained from additional point cloud observations within the same coordinate frame with $\mathbf{X}_{\text{GS}}$. The 6-DoF motions of the Gaussian kernels are interpolated using Linear Blend Skinning (LBS) [42], which updates $\mathbf{X}_{\text{GS}}$ and $\mathbf{R}_{\text{GS}}$ by treating $\mathbf{X}$ as control points and interpolating their predicted motion to generate new Gaussian centers and rotations. We assume that the color, scale, and opacity of the Gaussian splatting remain constant.

B.5 Planning

Our model can be integrated with Model Predictive Control (MPC) for model-based planning. Given multi-view RGB-D captures, we obtain object particles through segmentation, inverse projection into 3D space, and downsampling. The downsampled particles serve as inputs to the dynamics model for future prediction. With a specified cost function, the MPC

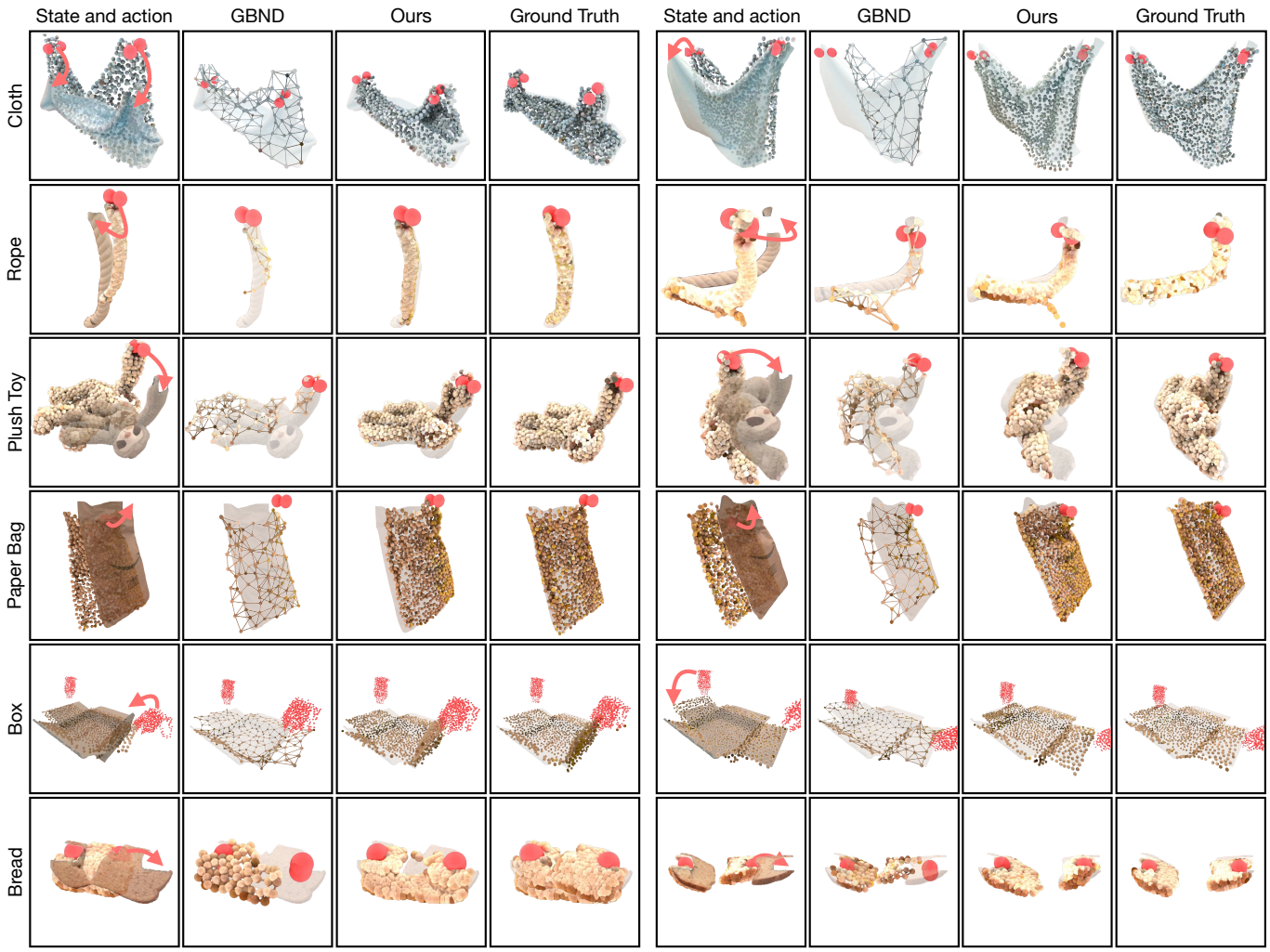


Fig. 4: **Qualitative Comparisons on Dynamics Prediction.** Given initial states and actions, we show the prediction results of the GBND baseline compared to our particle-grid neural dynamics model. The red spheres indicate the position and orientation of robot grippers. We overlay the predictions with ground truth final state images to highlight the prediction errors. Our model's predictions are more aligned with the ground truth, offering higher-density particle predictions and fewer artifacts compared to the baseline.

framework rolls out the dynamics model using sampled actions and optimizes the total cost. In our experiments, we use the Chamfer Distance between the predicted state $\hat{\mathbf{X}}$ and the target state $\mathbf{X}_{\text{target}}$ as the cost function:

$$\mathbf{J}(\hat{\mathbf{X}}_{1:N}, \mathbf{A}_{1:N}) = \sum_{t=1}^{N} \text{CD}(\hat{\mathbf{X}}_t, \mathbf{X}_{\text{target}}). \tag{18}$$

We apply the Model-Predictive Path Integral (MPPI) [47] trajectory optimization algorithm to minimize the cost and to synthesize the robot's actions. During deployment, we perform online, iterative planning to achieve closed-loop control.

Appendix C Experiments

Our experiments are designed to address the following questions:

- How well does the particle-grid model learn the dynamics of various types of deformable objects?
- Does the model perform effectively under limited visual observation (e.g., sparse views)?
- Can we train a unified model for multiple instances within an object category, and how well does it generalize to unseen instances?
- Can the model improve the performance of 3D action-conditioned video prediction and model-based planning?

We evaluate our method on a diverse set of challenging deformable objects, including cloth, rope, plush toys, bags, boxes, and bread. Our results demonstrate that it outperforms previous state-of-the-art approaches in dynamics prediction accuracy while remaining robust to incomplete camera views. Additionally, we validate the model's capability in category-level training and its effectiveness in downstream applications, such as video prediction and planning.

C.1 Experiment Setup

We conduct data collection and experiments using a bimanual xArm setup, with each robot arm having seven degrees of freedom. The objects used in the experiments include rope, cloth, plush toys, paper bags, boxes, and bread. An illustration

Method	Metric	Cloth	Rope	Plush	Box	Bag	Bread
MPM [12]	$\mathcal{J}$-Score / IoU ↑	$40.5_{\pm 20.3}$	$12.5_{\pm 9.5}$	$37.5_{\pm 14.8}$	–	$34.3_{\pm 5.5}$	$42.9_{\pm 10.3}$
GBND [54]		$56.7_{\pm 13.9}$	$19.2_{\pm 15.6}$	$42.7_{\pm 10.5}$	**$83.0_{\pm 5.5}$**	**$78.0_{\pm 82.8}$**	$49.8_{\pm 13.9}$
Particle		$58.5_{\pm 18.7}$	$24.4_{\pm 17.7}$	$49.0_{\pm 12.5}$	$82.2_{\pm 6.1}$	$74.5_{\pm 11.7}$	$39.3_{\pm 15.4}$
Ours		**$63.2_{\pm 16.4}$**	**$29.5_{\pm 16.5}$**	**$59.7_{\pm 11.1}$**	$82.4_{\pm 5.0}$	$76.8_{\pm 8.8}$	**$55.8_{\pm 10.1}$**
MPM [12]	$\mathcal{F}$-Score ↑	$28.0_{\pm 11.3}$	$37.0_{\pm 14.2}$	$40.4_{\pm 12.2}$	–	$13.1_{\pm 5.9}$	$54.3_{\pm 12.1}$
GBND [54]		$32.2_{\pm 18.2}$	$41.9_{\pm 18.9}$	$35.8_{\pm 11.2}$	**$74.0_{\pm 8.8}$**	$54.6_{\pm 13.4}$	$60.1_{\pm 16.1}$
Particle		$41.0_{\pm 19.6}$	$45.4_{\pm 19.5}$	$43.0_{\pm 14.0}$	$73.1_{\pm 8.8}$	$52.7_{\pm 19.5}$	$46.8_{\pm 17.7}$
Ours		**$42.6_{\pm 20.4}$**	**$52.6_{\pm 17.6}$**	**$53.8_{\pm 12.9}$**	$68.8_{\pm 12.9}$	**$60.3_{\pm 14.7}$**	**$64.6_{\pm 12.7}$**
MPM [12]	LPIPS ↓	$0.141_{\pm 0.060}$	$0.052_{\pm 0.018}$	$0.103_{\pm 0.085}$	–	$0.145_{\pm 0.020}$	$0.059_{\pm 0.020}$
GBND [54]		$0.120_{\pm 0.055}$	$0.042_{\pm 0.018}$	$0.081_{\pm 0.023}$	$0.106_{\pm 0.039}$	$0.097_{\pm 0.019}$	$0.052_{\pm 0.015}$
Particle		$0.109_{\pm 0.048}$	$0.044_{\pm 0.032}$	$0.072_{\pm 0.024}$	$0.082_{\pm 0.031}$	$0.069_{\pm 0.019}$	$0.054_{\pm 0.020}$
Ours		**$0.099_{\pm 0.044}$**	**$0.041_{\pm 0.033}$**	**$0.057_{\pm 0.018}$**	**$0.079_{\pm 0.034}$**	**$0.065_{\pm 0.010}$**	**$0.042_{\pm 0.014}$**

TABLE II: **Quantitative Results on 3D Action-Conditioned Video Prediction.** We compared our method on 3D action-conditioned video prediction quality with MPM [12], GBND [54], and particle-based baselines. The $\mathcal{J}$-Score/IoU and the $\mathcal{F}$-Score measures mask similarities and the LPIPS score measures appearance-wise similarities between predicted frames and ground truth video recordings. We report the mean and standard deviation of the prediction error over a 3-second horizon. The best results are highlighted in bold and blue.

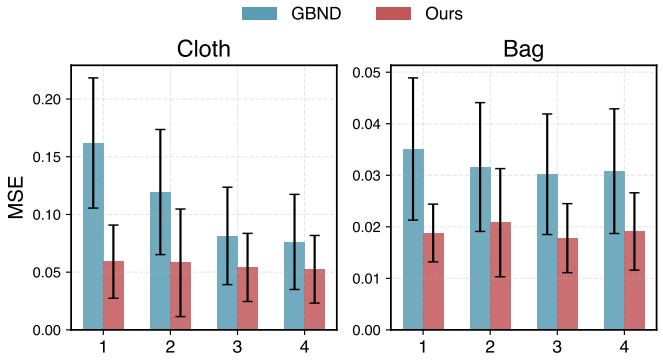


Fig. 5: **Quantitative Comparisons on Prediction under Partial Views.** We compare our method with the GBND baseline in the cloth and paper bag categories while varying the number of input camera views. We report the mean and standard deviation of the dynamics prediction error. Our method consistently achieves lower error than the baseline, and its error increase rate as the number of camera views decreases is also lower.

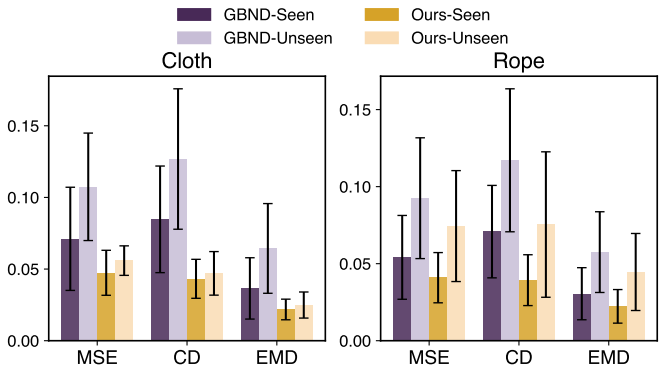


Fig. 6: **Quantitative Comparisons on Generalization.** Our method is compared with GBND on seen and unseen instances of the rope and cloth categories. We present the mean and standard deviation of dynamics prediction error. Our method's prediction error is lower on both seen and unseen instances compared to the baseline.

of our workspace setup and objects is provided in Fig. 11.

C.10a Rope: A single robot arm grasps one end of a rope, while the other end remains free on the table surface. The robot manipulates the rope in 3D space, generating various deformation patterns such as bending and dragging.

C.10b Cloth: Two robot arms grasp a rectangular piece of cloth and manipulate it in 3D space. The lower half of the cloth remains in contact with the table, resulting in significant deformations under lifting, moving, and folding actions.

C.10c Plush: A single robot arm grasps one limb of a plush toy while the rest of the toy remains in contact with the table. The robot manipulates the plush in 3D space, creating deformation patterns such as limb movements and flipping.

C.10d Paper Bag: One robot arm grasps and stabilizes one side of an envelope-shaped mailer bag, while the other manipulates it in 3D space. The robot performs various actions, including opening, closing, and rotating the bag.

C.10e Box: Two robot arms are used to open and close shipping boxes. The grippers remain closed, and the manipulation is performed in a nonprehensile manner, utilizing the surfaces of the grippers to push against the movable parts of the box.

C.10f Bread: Two robot arms are used to tear pieces of bread. The grippers remain closed, holding the bread in the air. One robot arm stays still while the other pulls, creating stretching effects and eventual breakage.

The baseline models in our comparisons are as follows:

- MPM-based deformable object simulation [12, 29]: This baseline assumes a hyperelastic material with an unknown uniform Young's modulus and friction coefficient with the tabletop. Parameter identification is performed via gradient descent.
- Graph-Based Neural Dynamics (GBND) [54]: This model represents objects using subsampled sparse vertices, along with the robot end-effectors, and employs a Graph Neural Network (GNN) to predict particle motions.
- Particle-based dynamics model (ours w/o grid): In this baseline, we ablate the grid representation in our model and directly query the velocity field at particle positions to predict per-particle velocities.

For additional information on the experiment setups and baseline implementations, please refer to Appendix **??**.

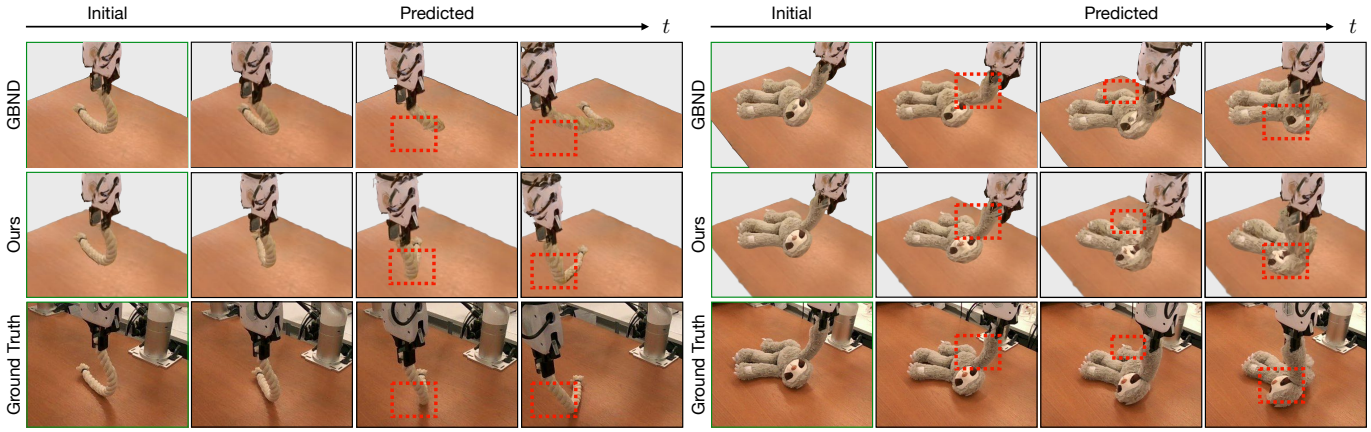


Fig. 7: **Qualitative Comparisons on 3D Action-Conditioned Video Prediction.** We show our method and the GBND baseline's prediction on two examples of rope and plush toy, compared with the ground truth video. The predictions are based on the 3DGS reconstructions on the first frame (leftmost image) and the robot action sequence. Differences are highlighted with red dashed boxes. Our method aligns better with the ground truth while the baseline method predicts visually nonrealistic deformations.

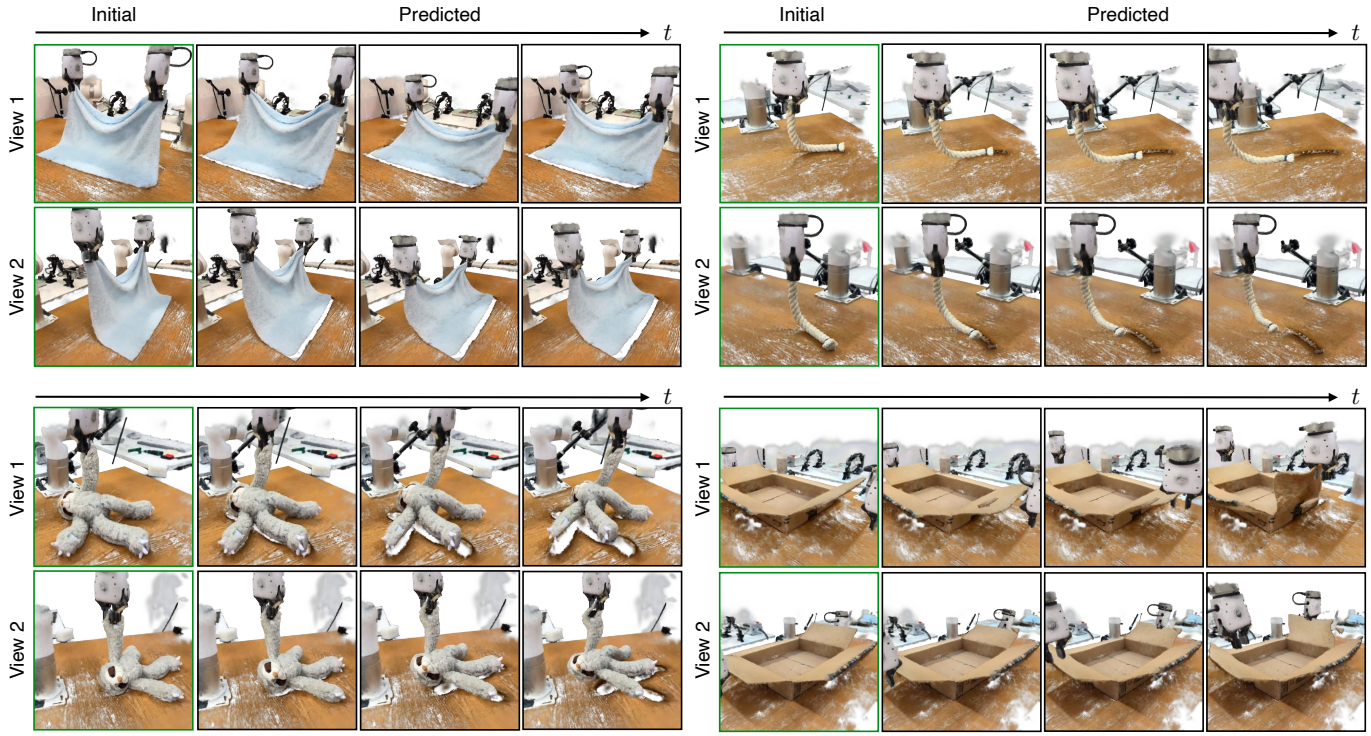


Fig. 8: **Qualitative Visualizations of Simulation from Scanned Scenes.** Our method gives higher-quality video prediction results with high-resolution Gaussians reconstructed from phone scans. Given the initial reconstruction (green frame), we apply our particle-grid dynamics model to simulate the segmented object, and visualize from different views.

C.2 Dynamics Learning and Prediction

We evaluate accuracy using a held-out set of robot-object interactions. The interaction videos are divided into 3-second clips, and the dynamics rollout accuracy is assessed using 3D point cloud metrics that compute the distance between predicted particle positions and ground truth future points. The metrics include Mean Squared Error (MSE), Chamfer Distance (CD), and Earth Mover's Distance (EMD). All metrics are calculated with m or m^2 as the unit of measurement. For the box category, we use the Robot Particles control method representation, and since it is not directly compatible with MPM. Therefore, we omit MPM from the box comparison.

Quantitative results are shown in Table I, where our method outperforms all baselines in terms of dynamics rollout accuracy. We visualize the particle prediction results of our method alongside the baselines in Fig. 4. Our method's predictions align more closely with the ground truth images and also exhibit higher resolution. In contrast, GBND predicts inadequate particle motions for objects like cloth, plush toy, box, bread, and it generates artifacts for objects like rope.

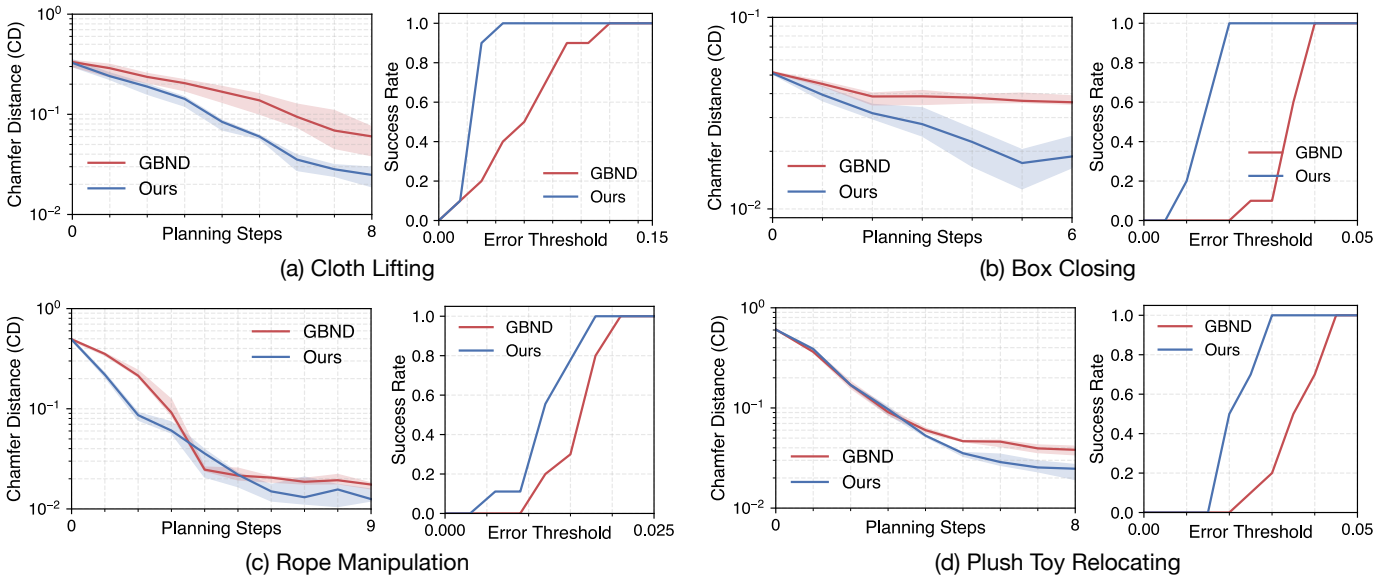


Fig. 9: **Quantitative Comparisons on Planning.** For four manipulation tasks—cloth lifting, box closing, rope manipulation, and plush toy relocating—we present the error curve and the final success rate curve with respect to the error threshold for task success. The error is always measured using the Chamfer Distance between the current and target point clouds. Our method outperforms the GBND baseline in both error reduction rate and success rate.

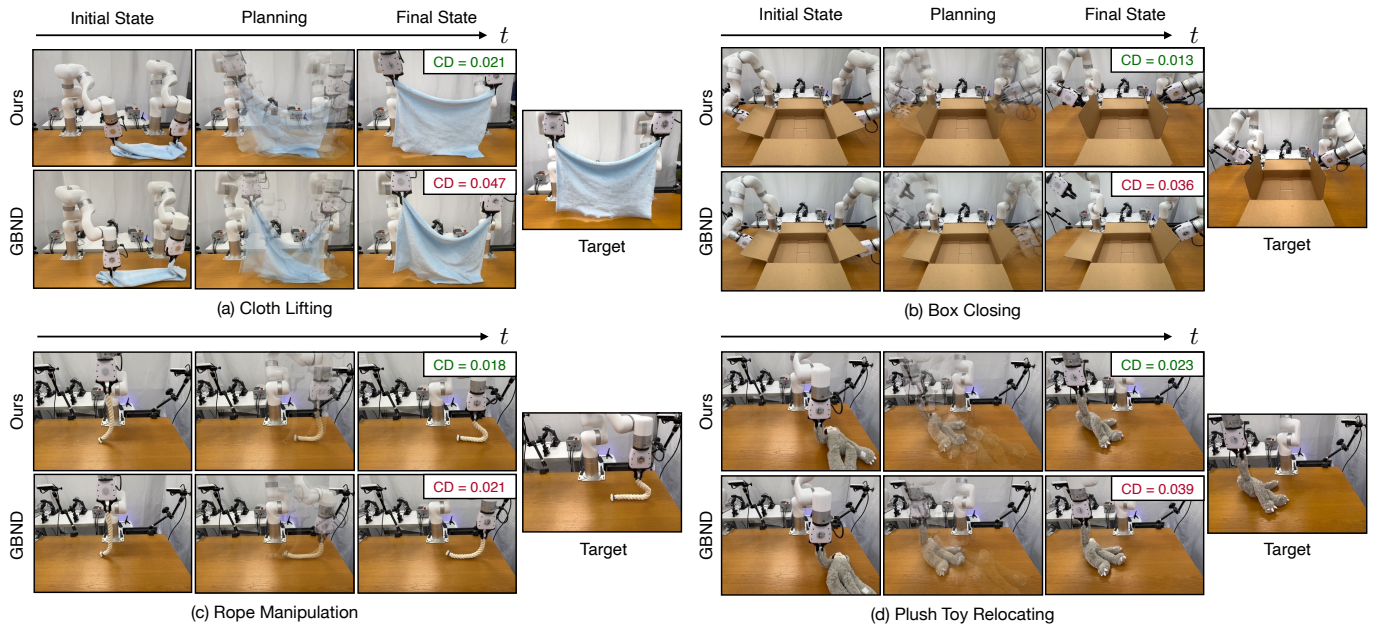


Fig. 10: **Qualitative Comparisons on Planning.** For each of the four tasks, we visualize a representative planning sequence for both our method and the GBND baseline. Given similar initial states and the same number of planning steps, our method achieves a lower final error, as measured by Chamfer Distance (CD), and produces results that are visually more similar to the target.

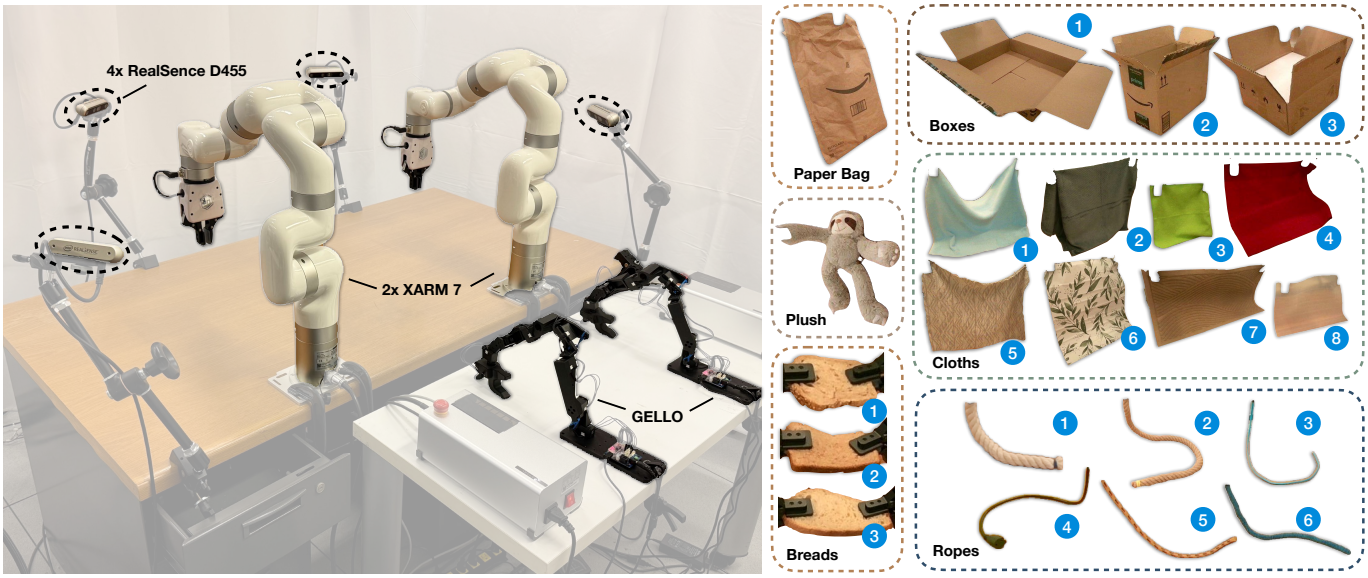
C.3 Sparse-View Dynamics Prediction

We evaluate the performance of our method in sparse-view scenarios by training the model on partial observation data. Specifically, during training, we use point cloud from a randomly sampled number of camera views as model input. During evaluation, we test the model's performance with 1 to 4 camera views.

The results shown in Fig. 5 demonstrate that our model outperforms the GBND baseline in dynamics prediction accuracy, regardless of the number of input views. Additionally, the performance drop when decreasing the number of views is also less significant than the baseline. For the cloth category, the baseline performance drops significantly when decreasing from 4 camera views to 1 camera view, while our model maintains a low prediction error.

C.4 Category-Level Model

Next, we assess the model's ability to generalize across multiple instances within the same category by training on a combined dataset of various object instances. For ropes, the model is trained on 4 distinct ropes and evaluated on 2



(a) Robot workspace (b) Diverse objects

Fig. 11: **Experiment Setup. (a)** Our robot workspace includes four calibrated RGB-D cameras positioned at each corner of the table, along with a GELLO [48] system for teleoperating the dual xArm 7 robotic arms equipped with parallel grippers. **(b)** Our experiments involved six types of materials: (i) a paper bag, (ii) a stuffed animal, (iii) three varieties of bread, (iv) three boxes of different shapes, (v) eight cloth pieces varying in fabric type and size, and (vi) six ropes differing in length, thickness, and stiffness.

unseen ropes. For cloths, it is trained on 6 cloth instances and tested on 2 unseen cloths. The 6 rope instances are cotton rope, jute rope, utility rope, cable, paracord, and yarn, with the cotton rope and utility rope included in the test set and unseen during training. For cloths, the 8 instances include a flannel blanket, cotton towel, microfiber cloth, cotton bed sheet, curtain, wallpaper, mat, and foam sheet, with the flannel blanket and foam sheet included in the test set. These instances are selected to have diverse physical properties and shapes, allowing us to thoroughly evaluate the model's generalization.

The results in Fig. 6 show that our model achieves lower prediction errors than the GBND baseline across both categories and for both seen and unseen instances. Notably, for the cloth category, the baseline method exhibits a significantly performance drop on unseen instances, whereas our method keeps a relatively low error, demonstrating better generalization to novel objects at test time.

C.5 Action-Conditioned Video Prediction

For action-conditioned video prediction, we use the predicted point cloud trajectories to interpolate Gaussian kernel transformations using LBS [42]. We reconstruct Gaussians from 4 input views using Gaussian Splatting [16]. The Gaussians are trained with a segmentation mask loss, following previous works [54, 28]. Videos are rendered with a fixed input camera pose. The video prediction quality is assessed using mask-based metrics, including $\mathcal{J}$-Score (IoU), $\mathcal{F}$-Score (contour matching accuracy), and the image-based metric LPIPS.

The resulting metrics are shown in Table II. Our approach achieves the best overall performance. For categories with relatively large objects, for instance boxes and paper bags, we observe that spiky Gaussian reconstructions often negatively impact mask prediction performance, especially when objects undergo significant deformation. The higher mask alignment scores in GBND and Particle baselines are largely due to inadequate particle motion predictions.

In Fig. 7, we show the action-conditioned video prediction results by reconstructing the object using Gaussian Splatting from 4 views and deforming the Gaussians with predictions from our dynamics model. Our method achieves higher-quality rendering and better alignment with the ground truth.

In Fig. 8, we further demonstrate that our method can be used for simulation based on high-quality phone-scanned GS reconstructions. The scenes are reconstructed using video scans of a static workspace. Coupled with our learned particle-grid neural dynamics, we can generate 3D action-conditioned video predictions with even greater visual fidelity.

C.6 Planning

In planning experiments, we evaluate the model's ability to integrate with MPC to generate actions for manipulating objects. We test on 4 tasks with distinct object types: cloth lifting, box closing, rope manipulation, and plush toy relocating. For each task, we conduct 10 repetitive experiments. Performance is assessed using error curves and task success rates.

The quantitative results are shown in Fig. 9. Across all four planning tasks, our method achieves a lower terminal error and a higher error reduction rate compared to the GBND baseline. In Fig. 10, we visualize the initial states, intermediate steps, and final states, comparing them to the target. In all four tasks, our method produces results that are visually closer to the target. For example, in the box closing task, our method successfully lifts both sides of the box, whereas the baseline struggles to predict the correct actions and often loses contact with the box. In rope manipulation, our method accurately bends the rope

by pressing downward, while the baseline fails to achieve this due to lower prediction resolution.

APPENDIX D DISCUSSIONS

D.1 Limitations

While we have demonstrated that Particle-Grid Neural Dynamics can model diverse types of deformable objects, the current framework has several limitations: (i) The current formulation assumes a fixed number of particles during iterative rollout, making it inapplicable to scenarios involving the appearance or disappearance of particles. This limitation could be addressed by correcting the particle sets with new observations or modeling the per-frame visibility of particles. (ii) The model implicitly infers an object's physical properties from its point cloud and short-term motion history. While this is sufficient for modeling a single object instance or multiple instances with distinct shapes and physical properties, a more systematic approach to modeling physical properties is needed for interpretable identification and adaptation at test time. This could involve learning a parameter-conditioned neural dynamics model [53]. (iii) Training the model and applying it to video prediction depend on accurate predictions from computer vision models such as Segment-Anything [18], CoTracker [15], and Gaussian Splatting [16]. Failures in these perception and reconstruction models could negatively impact our method's performance.

D.2 Conclusion

In this paper, we introduce Particle-Grid Neural Dynamics, a novel framework for learning neural dynamics models of deformable objects directly from sparse-view RGB-D recordings of robot-object interactions. By leveraging a hybrid particle-grid representation to capture object states and robot actions in 3D space, our method outperforms previous graph-based neural dynamics models in terms of prediction accuracy and modeling density. This advancement enables the modeling of a wide range of challenging deformable objects. Additionally, integration with 3D Gaussian Splatting facilitates 3D action-conditioned video prediction, simultaneously capturing both object geometry and appearance changes, thereby creating a learning-based digital twin of real-world objects. We further demonstrate that our model can be applied to various deformable object manipulation tasks, achieving improvements in both task execution efficiency and success rate.