# Dynamic-Resolution Model Learning for Object Pile Manipulation

Yixuan Wang[2*] Yunzhu Li[1,2*] Katherine Driggs-Campbell[2] Li Fei-Fei[1] Jiajun Wu[1]
[1]Stanford University [2]University of Illinois Urbana-Champaign
{yixuan22, yunzhuli, krdc}@illinois.edu, {feifeili, jiajunwu}@cs.stanford.edu

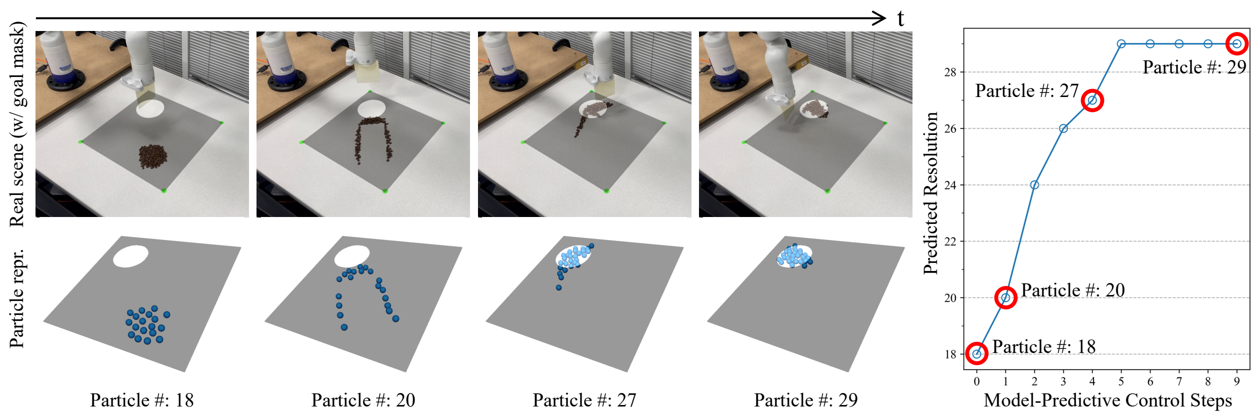


Figure 1. **Dynamic-Resolution Model Learning for Object Pile Manipulation in the Real World.** Depending on the progression of a task, representations at different granularity levels may be needed at each model-predictive control (MPC) step to make the most effective progress on the overall task. In this work, we construct dynamic-resolution particle representations of the environment and learn a *unified* dynamics model using graph neural networks (GNNs) that allows adaptive selection of the abstraction level. In this figure, we demonstrate a real-world task of gathering the object pile into a target region. Figures on the left show the task execution process and the corresponding particle representation. The plot on the right shows the predicted optimal resolution at each MPC step, where the red circles correspond to the frames on the left.

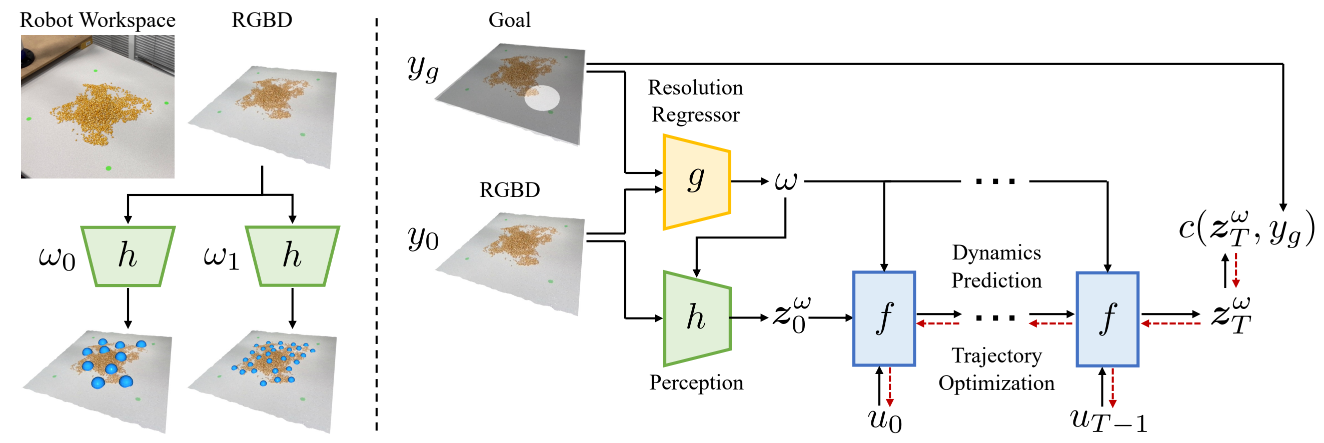
## Abstract

*Dynamics models learned from visual observations have shown to be effective in various robotic manipulation tasks. One of the key questions for learning such dynamics models is what scene representation to use. Prior works typically assume representation at a fixed dimension or resolution, which may be inefficient for simple tasks and ineffective for more complicated tasks. In this work, we investigate how to learn dynamic and adaptive representations at different levels of abstraction to achieve the optimal trade-off between efficiency and effectiveness. Specifically, we construct dynamic-resolution particle representations of the environment and learn a unified dynamics model using graph neural networks (GNNs) that allows continuous selection of the abstraction level. During test time, the agent can adaptively determine the optimal resolution at each model-predictive control (MPC) step. We evaluate our method in object pile manipulation, a task we commonly encounter in cooking, agriculture, manufacturing, and pharmaceutical applications. Through comprehensive evaluations both in the simulation and the real world, we show that our method achieves significantly better performance than state-of-the-art fixed-resolution baselines at the gathering, sorting, and redistribution of granular object piles made with various instances like coffee beans, almonds, corn, etc.*

## 1. Introduction

Predictive models are core to robotic systems for navigation [25], locomotion [28], and manipulation [22, 71]. In robotic manipulation, learned dynamics models have demonstrated impressive results. A learning-based dynamics model includes an encoder and a predictive model. Scene representation choices (e.g., latent vectors [19, 20, 33], object-centric [15, 69] or keypoint representations [37, 40, 66]) affect expressiveness and generalization capabilities, which makes it crucial for a given task.

Prior work uses a fixed representation for the entire task, but the optimal representation may differ depending on the object, task, or stage. An ideal representation balances efficiency and effectiveness [5, 61]. For instance, in object pile



(a) Representations at different resolutions

(b) Dynamics prediction and inverse planning

Figure 2. **Overview of the proposed framework.** (a) Our perception module $h$ processes the input RGBD image and generates particle representations at different levels of abstraction depending on the resolution $\omega$. (b) The resolution regressor $g$ takes the current observation $y_0$ and the goal $y_g$ as input. It then predicts the resolution $\omega$ we intend to represent the environment. The dynamics model $f$, conditioned on the dynamically-selected resolution $\omega$ and the input action $u_t$, predicts the temporal evolution of the scene representation $z_t^\omega$. During planning time, we calculate the task objective $c(z_T^\omega, y_g)$ and backpropagate the gradients to optimize the action sequence $\{u_t\}$.

manipulation, a more complex target configuration needs a finer model to capture all the details. While for the same targets, we might want representations at different abstraction levels for the most effective actions at different stages, as shown in Figure 1.

We focus on manipulating object piles, a crucial task in cooking, agriculture, manufacturing, and pharmaceutical scenarios. This task is highly challenging due to the environment's extremely high degrees of freedom [52], making it an ideal scenario to demonstrate how we can learn dynamics models at different levels of abstraction to achieve the optimal trade-off between efficiency and effectiveness.

Our aim is to learn a dynamics model that can adaptively express the world at different granularity levels based on the task objective and observation. To achieve this, we introduce a resolution regressor that predicts the optimal resolution using self-supervised learning with labels from Bayesian optimization [18]. Besides the resolution regressor, our model also includes perception, dynamics, and planning modules (Figure 2).

During task execution, we follow a model-predictive control (MPC) framework. At each MPC step, the resolution regressor predicts the resolution most effective for control optimization. The perception module then samples particles from the RGBD visual observation based on the predicted resolution. The derived particle-based scene representation, together with the robot action, will be the input to the dynamics model to predict the environment's evolution. The dynamics model can then be used for trajectory optimization to derive the action sequence. Specifically, the dynamics model is instantiated as a graph neural network consisting of node and edge encoders. Such compositional structures naturally generalize to particle sets of different sizes and densities—a unified graph-based dynamics model can support model-predictive control at various abstraction levels, selected continuously by the resolution regressor.

Our contributions are threefold: (1) a framework that dynamically determines the scene representation at different abstraction levels, (2) comprehensive evaluations showing the superiority of our dynamic scene representation over fixed resolution, and (3) a unified robotic manipulation system for various object pile manipulation tasks.

## 2. Method

In this section, we first present the problem formulation in Section A.1. We then discuss the structure of our dynamic-resolution dynamics models, how we learn a resolution regressor to automatically select the scene representation, and how we use the model for the downstream planning tasks in Section A.2, A.3 and A.4 respectively.

## 3. Experiments

In this section, we evaluate the proposed framework in various object pile manipulation tasks. In particular, we aim to answer the following three questions through the experiments. (1) Does a trade-off exist between efficiency and effectiveness as we navigate through representations at different abstraction levels? (2) Is a fixed-resolution dynamics model sufficient, or do we need to dynamically select the resolution at each MPC step? (3) Can our dynamic-resolution model accomplish three challenging object pile manipulation tasks: **Gather**, **Redistribute**, and **Sort**? Experiments details can be found in Section B

## References

[1] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6): 33–41, 1984. 12

[2] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003. 12

[3] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016. 12

[4] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. 12

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 1

[6] Matthew Michael Botvinick. Hierarchical reinforcement learning and decision making. *Current opinion in neurobiology*, 22(6):956–962, 2012. 12

[7] Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier, 1987. 12

[8] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013. 8

[9] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016. 12

[10] Peng Chang and Taşkın Padır. Model-based manipulation of linear flexible objects with visual curvature feedback. In *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1406–1412. IEEE, 2020. 12

[11] Andrea Cherubini, Valerio Ortenzi, Akansel Cosgun, Robert Lee, and Peter Corke. Model-free vision-based shaping of deformable plastic materials. *The International Journal of Robotics Research*, 39(14):1739–1759, 2020. 12

[12] Samuel Clarke, Travers Rhodes, Christopher G Atkeson, and Oliver Kroemer. Learning audio feedback for estimating amount and flow of granular material. *Proceedings of Machine Learning Research*, 87, 2018. 12

[13] Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992. 12

[14] Lokenath Debnath and Firdous Ahmad Shah. *Wavelet transforms and their applications*. Springer, 2002. 12

[15] Danny Driess, Zhiao Huang, Yunzhu Li, Russ Tedrake, and Marc Toussaint. Learning multi-object dynamics with compositional neural radiance fields. *arXiv preprint arXiv:2202.11855*, 2022. 1

[16] Nima Fazeli, Miquel Oller, Jiajun Wu, Zheng Wu, Joshua B Tenenbaum, and Alberto Rodriguez. See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion. *Science Robotics*, 4(26):eaav3123, 2019. 12

[17] Niklas Funk, Georgia Chalvatzaki, Boris Belousov, and Jan Peters. Learn2assemble with structured representations and search for robotic architectural construction. In *Conference on Robot Learning*, pages 1401–1411. PMLR, 2022. 12

[18] Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023. 2

[19] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019. 1

[20] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019. 1

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9): 1904–1916, 2015. 12

[22] François Robert Hogan and Alberto Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. *arXiv preprint arXiv:1611.08268*, 2016. 1, 12

[23] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. *Advances in neural information processing systems*, 31, 2018. 12

[24] Zixuan Huang, Xingyu Lin, and David Held. Mesh-based dynamics with occlusion reasoning for cloth manipulation. *arXiv preprint arXiv:2206.02881*, 2022. 12

[25] Boris Ivanovic, Amine Elhafsi, Guy Rosman, Adrien Gaidon, and Marco Pavone. Mats: An interpretable trajectory forecasting representation for planning and control. *arXiv preprint arXiv:2009.07517*, 2020. 1

[26] Ioannis G Kevrekidis, C William Gear, James M Hyman, Panagiotis G Kevrekidis, Olof Runborg, Constantinos Theodoropoulos, et al. Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis. *Commun. Math. Sci*, 1(4):715–762, 2003. 12

[27] Ioannis G Kevrekidis, C William Gear, and Gerhard Hummer. Equation-free: The computer-aided analysis of complex multiscale systems. *AIChE Journal*, 50(7): 1346–1355, 2004. 12

[28] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40:429–455, 2016. 1

[29] J Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013. 12

[30] J Nathan Kutz, Xing Fu, and Steven L Brunton. Multiresolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016. 12

[31] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018. 10, 12

[32] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation networks for model-based control under partial observation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1205–1211. IEEE, 2019. 7, 12

[33] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In *Conference on Robot Learning*, pages 112–123. PMLR, 2022. 1, 12

[34] 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. 12

[35] Qingkai Lu and Liangjun Zhang. Excavation learning for rigid objects in clutter. *IEEE Robotics and Automation Letters*, 6(4):7373–7380, 2021. 12

[36] 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. 10

[37] Lucas Manuelli, Yunzhu Li, Pete Florence, and Russ Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. *arXiv preprint arXiv:2009.05085*, 2020. 1

[38] David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010. 12

[39] Carolyn Matl, Yashraj Narang, Ruzena Bajcsy, Fabio Ramos, and Dieter Fox. Inferring the material properties of granular media for robotic tasks. In *2020 ieee international conference on robotics and automation (icra)*, pages 2770–2777. IEEE, 2020. 12

[40] Matthias Minderer, Chen Sun, Ruben Villegas, Forrester Cole, Kevin P Murphy, and Honglak Lee. Unsupervised learning of object structure and dynamics from videos. *Advances in Neural Information Processing Systems*, 32, 2019. 1

[41] Peter Mitrano, Dale McConachie, and Dmitry Berenson. Learning where to trust unreliable models in an unstructured world for deformable object manipulation. *Science Robotics*, 6(54):eabd8170, 2021. 12

[42] Carsten Moenning and Neil A Dodgson. Fast marching farthest point sampling. Technical report, University of Cambridge, Computer Laboratory, 2003. 7

[43] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li F Fei-Fei, Josh Tenenbaum, and Daniel L Yamins. Flexible neural representation for physics prediction. *Advances in neural information processing systems*, 31, 2018. 12

[44] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018. 12

[45] Monica N Nicolescu and Maja J Matarić. A hierarchical architecture for behavior-based robots. In *Proceedings of the first international joint conference on*

*Autonomous agents and multiagent systems: part 1*, pages 227–233, 2002. 12

[46] Tao Pang, HJ Suh, Lujie Yang, and Russ Tedrake. Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models. *arXiv preprint arXiv:2206.10787*, 2022. 12

[47] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021. 12

[48] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020. 12

[49] Haozhi Qi, Xiaolong Wang, Deepak Pathak, Yi Ma, and Jitendra Malik. Learning long-term visual dynamics with region proposal interaction networks. *arXiv preprint arXiv:2008.02265*, 2020. 12

[50] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pages 4470–4479. PMLR, 2018. 12

[51] 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. 12

[52] Connor Schenck, Jonathan Tompson, Sergey Levine, and Dieter Fox. Learning robotic manipulation of granular media. In *Conference on Robot Learning*, pages 239–248. PMLR, 2017. 2, 12

[53] Bokui Shen, Zhenyu Jiang, Christopher Choy, Leonidas J Guibas, Silvio Savarese, Anima Anandkumar, and Yuke Zhu. Acid: Action-conditional implicit visual dynamics for deformable object manipulation. *arXiv preprint arXiv:2203.06856*, 2022. 12

[54] 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. 12

[55] Tom Silver, Rohan Chitnis, Aidan Curtis, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Planning with learned object importance in large problem instances using graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11962–11971, 2021. 12

[56] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012. 8

[57] HJ Suh and Russ Tedrake. The surprising effectiveness of linear models for visual foresight in object pile manipulation. *arXiv preprint arXiv:2002.09093*, 2020. 12

[58] Hyung Ju Terry Suh, Tao Pang, and Russ Tedrake. Bundled gradients through contact via randomized smoothing. *IEEE Robotics and Automation Letters*, 7(2):4000–4007, 2022. 12

[59] Kuniyuki Takahashi, Wilson Ko, Avinash Ummadisingu, and Shin-ichi Maeda. Uncertainty-aware self-supervised target-mass grasping of granular foods. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2620–2626. IEEE, 2021. 12

[60] Russ Tedrake. Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for mit 6.832. *Working draft edition*, 3:4, 2009. 8

[61] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000. 1

[62] Hsiao-Yu Fish Tung, Zhou Xian, Mihir Prabhudesai, Shamit Lal, and Katerina Fragkiadaki. 3d-oes: Viewpoint-invariant object-factorized environment simulators. *arXiv preprint arXiv:2011.06464*, 2020. 12

[63] Neea Tuomainen, David Blanco-Mulero, and Ville Kyrki. Manipulation of granular materials by learning particle interactions. *IEEE Robotics and Automation Letters*, 7(2):5663–5670, 2022. 12

[64] Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2020. 12

[65] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017. 12

[66] Weiyao Wang, Andrew S Morgan, Aaron M Dollar, and Gregory D Hager. Dynamical scene representation and control with keypoint-conditioned neural ra-

diance field. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 1138–1143. IEEE, 2022. 1

[67] Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. *Advances in neural information processing systems*, 30, 2017. 12

[68] Yufei Ye, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani. Compositional video prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10353–10362, 2019. 12

[69] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019. 1, 12

[70] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 12

[71] Jiaji Zhou, Yifan Hou, and Matthew T Mason. Pushing revisited: Differential flatness, trajectory planning, and stabilization. *The International Journal of Robotics Research*, 38(12-13):1477–1489, 2019. 1, 12

[72] Guangxiang Zhu, Zhiao Huang, and Chongjie Zhang. Object-oriented dynamics predictor. *Advances in Neural Information Processing Systems*, 31, 2018. 12

[73] Yifan Zhu, Laith Abdulmajeid, and Kris Hauser. A data-driven approach for fast simulation of robot locomotion on granular media. In *2019 international conference on robotics and automation (ICRA)*, pages 7653–7659. IEEE, 2019. 12

# Appendix

## A. Method

### A.1. Problem Formulation

Our goal is to derive the resolution $\omega$ to represent the environment to achieve the best trade-off between efficiency and effectiveness for control optimization. We define the following trajectory optimization problem over a horizon $T$:

$$\begin{aligned} \min_{\{u_t\}} \quad & c(\boldsymbol{z}_T^\omega, y_g), \\ \text{s.t.} \quad & \omega = g(y_0, y_g), \\ & \boldsymbol{z}_0^\omega = h(y_0, \omega), \\ & \boldsymbol{z}_{t+1}^\omega = f(\boldsymbol{z}_t^\omega, u_t, \omega), \end{aligned} \tag{1}$$

where the resolution regressor $g(\cdot,\cdot)$ takes the current observation $y_0$ and the goal configuration $y_g$ as input and predicts the model resolution. $h(\cdot,\cdot)$, the perception module, takes in the current observation $y_0$ and the predicted resolution $\omega$, then derives the scene representation $\boldsymbol{z}_0^\omega$ for the current time step. The dynamics module $f(\cdot,\cdot,\cdot)$ takes the current scene representation $\boldsymbol{z}_t^\omega$, the input action $u_t$, and the resolution $\omega$ as inputs, and then predicts the representation's evolution at the next time step $\boldsymbol{z}_{t+1}^\omega$. The optimization aims to find the action sequence $\{u_t\}$ to minimize the task objective $c(\boldsymbol{z}_T^\omega, y_g)$.

In the following sections, we describe (1) the details of the perception module $h(\cdot,\cdot)$ and the dynamics module $f(\cdot,\cdot,\cdot)$ in Section A.2, (2) how we obtain the self-supervision for the resolution regressor $g(\cdot,\cdot)$ in Section A.3, and (3) how we solve Equation 1 in a closed planning loop in Section A.4.

### A.2. Dynamic-Resolution Model Learning

To instantiate the optimization problem defined in Equation 1, we use graphs of different sizes as the representation $\boldsymbol{z}_t^\omega = (\mathcal{O}_t, \mathcal{E}_t)$, where $\omega$ indicates the number of vertices in the graph. The vertices $\mathcal{O}_t = \{o_t^i\}_{i=1,\ldots,|\mathcal{O}_t|}$ denote the particle set and $o_t^i$ represents the 3D position of the $i^{\text{th}}$ particle. The edge set $\mathcal{E}_t = \{e_t^j\}_{j=1,\ldots,|\mathcal{E}_t|}$ denotes the relations between the particles, where $e_t^j = (u_t^j, v_t^j)$ denotes an edge pointing from particle of index $v_t^j$ to $u_t^j$.

To obtain the particle set $\mathcal{O}_t$ from the RGBD visual observation $y_t$, we first transform the RGBD image into a point cloud and then segment the point cloud to obtain the foreground according to color and depth information $\bar{y}_t \in \mathbb{R}^{N\times 3}$. We then deploy the farthest point sampling technique [42] to subsample the foreground but ensure sufficient coverage of $\bar{y}_t$. Specifically, given already sampled particles $o_t^{1,\ldots,i-1}$, we apply

$$o_t^i = \arg\max_{y^k \in \bar{y}_t} \min_{o_t^j \in o_t^{1,\ldots,i-1}} \|y^k - o_t^j\|_2^2 \tag{2}$$

to find the $i^{\text{th}}$ particle $o_t^i$. We iteratively apply this process until we reach $\omega$ particles. Different choices of $\omega$ indicate scene representations at different abstraction levels, as illustrated in Figure 2a. The edge set is constructed dynamically over time and connects particles within a predefined distance while limiting the maximum number of edges a node can have.

We instantiate the dynamics model $f(\cdot,\cdot,\cdot)$ as graph neural networks (GNNs) that predict the evolution of the graph representation $\boldsymbol{z}_t^\omega$ under external actions $u_t$ and the selected resolution $\omega$. $f(\cdot,\cdot,\cdot)$ consists of node and edge encoders $f_{\mathcal{O}}^{\text{enc}}(\cdot,\cdot,\cdot)$, $f_{\mathcal{E}}^{\text{enc}}(\cdot,\cdot,\cdot)$ to obtain node and edge representations:

$$\begin{aligned} p_t^i &= f_{\mathcal{O}}^{\text{enc}}(o_t^i, u_t, \omega), \quad i = 1,\ldots,|\mathcal{O}_t|, \\ q_t^j &= f_{\mathcal{E}}^{\text{enc}}(o_t^{u_t^j}, o_t^{v_t^j}, \omega), \quad j = 1,\ldots,|\mathcal{E}_t|. \end{aligned} \tag{3}$$

We then have node and edge decoders $f_{\mathcal{O}}^{\text{dec}}(\cdot,\cdot)$, $f_{\mathcal{E}}^{\text{dec}}(\cdot,\cdot)$ to obtain the corresponding representations and predict the representation at the next time step:

$$\begin{aligned} r_t^j &= f_{\mathcal{E}}^{\text{dec}}(q_t^j, \omega), \quad j = 1,\ldots,|\mathcal{E}_t|, \\ \hat{o}_{t+1}^i &= f_{\mathcal{O}}^{\text{dec}}(p_t^i, \sum_{j\in\mathcal{N}_i} r_t^j), \quad i = 1,\ldots,|\mathcal{O}_t|, \end{aligned} \tag{4}$$

where $\mathcal{N}_i$ is the index set of the edges, in which particle $i$ is the receiver. In practice, we follow Li et al. [32] and use multi-step message passing over the graph to approximate the instantaneous propagation of forces.

To train the dynamics model, we iteratively predict future particle states over a time horizon of $T$ and then optimize the neural network's parameters by minimizing the mean squared error (MSE) between the predictions and the ground truth future states:

$$\mathcal{L} = \frac{1}{T\cdot|\mathcal{O}_t|}\sum_{t'=1}^{T}\sum_{i=1}^{|\mathcal{O}_t|}\|\hat{o}_{t+t'}^i - o_{t+t'}^i\|_2^2. \tag{5}$$

### A.3. Adaptive Resolution Selection

The previous sections discussed how to obtain the particle set and how we predict its evolution given a resolution $\omega$. In this section, we present how we learn the resolution regressor $g(\cdot,\cdot)$ in Equation 1 that can automatically determine the resolution in a self-supervised manner. Specifically, we intend to find the resolution $\omega$ that is the most effective for minimizing the task objective given the current observation $y_0$ and the goal $y_g$. We reformulate the optimization problem in Equation 1 by considering $\omega$ as a variable of the objective function as the following:

$$\begin{aligned} c^*(y_0, y_g, \omega) = \min_{\{u_t\}} \quad & c(\boldsymbol{z}_T^\omega, y_g), \\ \text{s.t.} \quad & \boldsymbol{z}_0^\omega = h(y_0, \omega), \\ & \boldsymbol{z}_{t+1}^\omega = f(\boldsymbol{z}_t^\omega, u_t, \omega). \end{aligned} \tag{6}$$

For a given $\omega$, we solve the above optimization problem via a combination of sampling and gradient descent using shooting methods [60] under a given time budget—the higher resolution representation will go through fewer optimization iterations. For simplicity, we denote the objective in Equation 6 as $c^*(\omega)$ in the following part of this section.

Given the formulation, we are then interested in finding the parameter $\omega$ that can minimize the following objective:

$$\begin{aligned} \min_{\omega} \quad & c^+(\omega) = c^*(\omega) + R(\omega), \\ \text{s.t.} \quad & \omega \in (\omega_{\min}, \omega_{\max}), \end{aligned} \tag{7}$$

where $R(\omega)$ is a regularizer penalizing the choice of an excessively large $\omega$ to encourage efficiency. Regularizer details can be found in supplementary materials. We use Bayesian optimization [56] to find the optimal $\omega$ by iteratively sampling $\omega$ and approximating $c^+(\omega)$ using the Gaussian process. At each sampling stage, we sample one or more data points $\omega_i$ according to the expected improvement of the objective function and evaluate their value $c^+(\omega_i)$. Then, at the approximation stage, we assume the distribution of $c^+(\omega)$ follows the Gaussian distribution $\mathcal{N}(\mu(\omega), \sigma^2)$; thus, the joint distribution of the evaluated points $\mathbf{\Omega}_{\text{train}} = [\omega_1, \dots, \omega_n]$ and the testing points $\mathbf{\Omega}_{\text{test}} = [\omega'_1, \dots, \omega'_m]$ can be expressed as the following:

$$\begin{aligned} \mathbf{C}_{\text{train}} &= [c^+(\omega_1), \dots, c^+(\omega_n)], \\ \mathbf{M}_{\text{train}} &= [\mu(\omega_1), \dots, \mu(\omega_n)], \\ \mathbf{C}_{\text{test}} &= [c^+(\omega'_1), \dots, c^+(\omega'_m)], \\ \mathbf{M}_{\text{test}} &= [\mu(\omega'_1), \dots, \mu(\omega'_m)], \\ \begin{bmatrix} \mathbf{C}_{\text{train}} \\ \mathbf{C}_{\text{test}} \end{bmatrix} &\sim \mathcal{N}\left( \begin{bmatrix} \mathbf{M}_{\text{train}} \\ \mathbf{M}_{\text{test}} \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{bmatrix} \right), \end{aligned} \tag{8}$$

where $\mathbf{K}$ is a kernel function matrix derived via $\mathbf{K} = K(\mathbf{\Omega}_{\text{train}}, \mathbf{\Omega}_{\text{train}})$. $K(\cdot, \cdot)$ is the kernel function used to compute the covariance. Similarly, $\mathbf{K}_* = K(\mathbf{\Omega}_{\text{train}}, \mathbf{\Omega}_{\text{test}})$ and $\mathbf{K}_{**} = K(\mathbf{\Omega}_{\text{test}}, \mathbf{\Omega}_{\text{test}})$.

Equation 8 shows the joint probability of $\mathbf{C}_{\text{train}}$ and $\mathbf{C}_{\text{test}}$ conditioned on $\mathbf{\Omega}_{\text{train}}$ and $\mathbf{\Omega}_{\text{test}}$. Through marginalization, we could fit $c^+(\omega)$ using the following conditional distribution:

$$\begin{aligned} & \mathbf{C}_{\text{test}} | \mathbf{C}_{\text{train}}, \mathbf{\Omega}_{\text{train}}, \mathbf{\Omega}_{\text{test}} \sim \\ & \quad \mathcal{N}(\mathbf{K}_*^\top \mathbf{K} \mathbf{C}_{\text{train}}, \mathbf{K}_{**} - \mathbf{K}_*^\top \mathbf{K}^{-1} \mathbf{K}_*). \end{aligned} \tag{9}$$

We can then use the mean value of the Gaussian distribution in Equation 9 as the metric to minimize $c^+(\omega)$. Therefore, the solution to Equation 7 is approximated as the following:

$$\begin{aligned} \omega^* = \arg\min_{\omega} \quad & \mathbf{K}_*^\top \mathbf{K} \mathbf{C}_{\text{train}} \\ \text{s.t.} \quad & \omega \in (\omega_{\min}, \omega_{\max}). \end{aligned} \tag{10}$$

To train the resolution regressor, we randomly generate a dataset containing the observation and goal pairs $(y_0, y_g)$. For each pair, we follow the above optimization process to generate the optimal resolution label $\omega^*$. We then train the resolution regressor $\omega = g(y_0, y_g)$ to predict the resolution based on the observation and the goal via supervised learning. Training the $\omega$ regressor is a self-supervised learning process, as the labels are automatically generated via an optimization process without any human labeling.

### A.4. Closed-Loop Planning on Adaptive Repr.

Now that we have obtained the resolution regressor $g$, the perception module $h$, and the dynamics module $f$. We can wire things together to solve Equation 1 and use the optimized action sequence in a closed loop within a model-predictive control (MPC) framework [8]. Specifically, for each MPC step, we follow Algorithm 1, which first determines the resolution to represent the environment, then uses a combination of sampling and gradient descent to derive the action sequence through trajectory optimization using the shooting method. We then execute the first action from the action sequence in the real world, obtain new observations, and apply Algorithm 1 again. Such a process allows us to take feedback from the environment and adaptively select the most appropriate resolution at each step as the task progresses. Figure 2b also shows an overview of the future prediction and inverse planning process. Details including task objective definition and MPC hyperparameter are included in supplementary materials.

**Algorithm 1** Trajectory optimization at each MPC step

```
Input: Current observation $y_0$, goal $y_g$, time horizon $T$,
    the resolution regressor $g$, the perception module $h$,
    the dynamics module $f$, and gradient descent iteration
$N$
Output: Actions $u_{0:T-1}$

Predict the resolution $\omega \leftarrow g(y_0, y_g)$
Obtain the current representation $\boldsymbol{z}_0^\omega \leftarrow h(y_0, \omega)$
Sample $M$ action sequences $\hat{u}_{0:T-1}^{1:M}$
for $m = 1, \dots, M$ do
    for $i = 1, \dots, N$ do
        for $t = 0, \dots, T-1$ do
            Predict the next step $\boldsymbol{z}_{t+1}^\omega \leftarrow f(\boldsymbol{z}_t^\omega, \hat{u}_t^m, \omega)$
        end for
        Calculate the task loss $c^m \leftarrow c(\boldsymbol{z}_T^\omega, y_g)$
        if $i < N$ then
            Update $\hat{u}_{0:T-1}^m$ using gradients $\nabla_{\hat{u}_{0:T-1}^m} c^m$
        end if
    end for
end for
$m^* \leftarrow \arg\min_m c^m$
Return $\hat{u}_{0:T-1}^{m^*}$
```

## A.5. Perception Module Details

Building graph $z_t^\omega = (\mathcal{O}_t, \mathcal{E}_t)$ from point cloud $\bar{y}_t \in \mathbb{R}^{N\times 3}$ contains two phases. First, we sample $\mathcal{O}_t^{\text{fps}}$ from the point cloud using farthest-point sampling. Specifically, given already sampled particles $o_t^{1,\dots,i-1}$, we apply

$$o_t^i = \arg\max_{y^k \in \bar{y}_t} \min_{o_t^j \in o_t^{1,\dots,i-1}} \|y^k - o_t^j\|_2^2 \tag{11}$$

to find the $i^{\text{th}}$ particle $o_t^i$. We iteratively apply this process until we reach $\omega$ particles. We found that sampled particles from the point cloud are likely to be at the edge of underneath objects. To bias sampled particles towards object centers, we define $\mathcal{O}_t$ as mass centers of $\mathcal{O}_t^{\text{fps}}$ neighbor points. For example, for $o_t^{i,\text{fps}} \in \mathcal{O}_t^{\text{fps}}$, we find the corresponding $o_t^i \in \mathcal{O}_t$ by applying

$$\begin{aligned} \bar{y}_t{}' &= \{y | y \in \bar{y}_t, ||y - o_t^{i,\text{fps}}||_2 \le r_{\text{center}}\} \\ o_t^i &= \frac{1}{|\bar{y}_t{}'|} \sum_{y \in \bar{y}_t{}'} y. \end{aligned} \tag{12}$$

$r_{\text{center}}$ is the hyperparameter to determine the neighbor points.

To find edges $\mathcal{E}_t$, we first approximate particle displacements $\Delta\mathcal{O}_t$ using the action $u_t$. We compute the sweeping region given the action $u_t$. If $o_t^i$ is within the sweeping region, $\Delta o_t^i$ is the vector from $o_t^i$ to the pusher end. We define $\hat{\mathcal{O}}_t = \Delta\mathcal{O}_t + \mathcal{O}_t$. For $\hat{o}_t{}^i$, $(\hat{o}_t{}^i, \hat{o}_t{}^j) \in \mathcal{E}_t$ if it satisfies the following criteria:

$$\begin{aligned} ||\hat{o}_t{}^i - \hat{o}_t{}^j||_2 &< r_{\text{edge}} \\ \hat{o}_t{}^j &\in \text{kNN}(\hat{o}_t{}^i). \end{aligned} \tag{13}$$

$r_{\text{edge}}$ is the hyperparameter to determine the distance needed for two nodes to interact. $\text{kNN}(\hat{o}_t{}^i)$ is the set of k-nearest-neighbor of the node $\hat{o}_t{}^i$ and $k$ is a hyperparameter.

## A.6. Distribution Distance

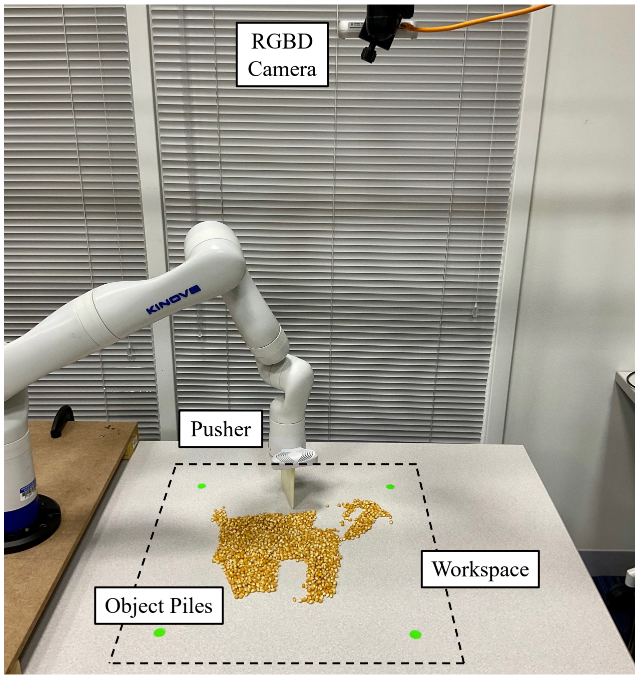
We compute distribution distance to evaluate the performance of different methods on gathering tasks quantitatively. The distribution distance is computed similarly to the task objective. We use the RGBD observation to segment out the foreground object and obtain foreground pixels $\mathcal{F}_t \in \mathbb{R}^{F\times 2}$. Similar to Equation 14, the distribution distance $d$ is defined using the following equation:

$$d = \sum_{f_i \in \mathcal{F}_t} \min_{q_j \in \mathcal{Q}_t} ||f_i - q_j||_2 + \sum_{q_j \in \mathcal{Q}_t} \min_{f_i \in \mathcal{F}_t} ||f_i - q_j||_2 \tag{14}$$

## A.7. High-Level Planner for Sort Task

In the **sort** task, we use a high-level planner to avoid colliding between two object piles. We use the A* search algorithm to find the high-level path. We represent every blob of object piles as a circle with a fixed radius in image space. Therefore, the state for one blob can be represented as its 2D blob center in image space. For the search algorithm, if there are $k$ blobs in the scene, the node is the concatenation of $k$ blob centers. One node is connected to nodes that can be reached by changing one blob center in a single step with no collision.

To accelerate motion planning, we divide the image space into a sparse grid. The blob center will only be on the grid. In addition, to encourage a path with fewer steps, we add a constant cost for each path so that a path with fewer steps is preferred.



(a) Robot setup

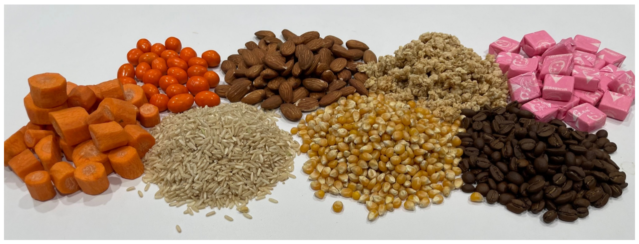
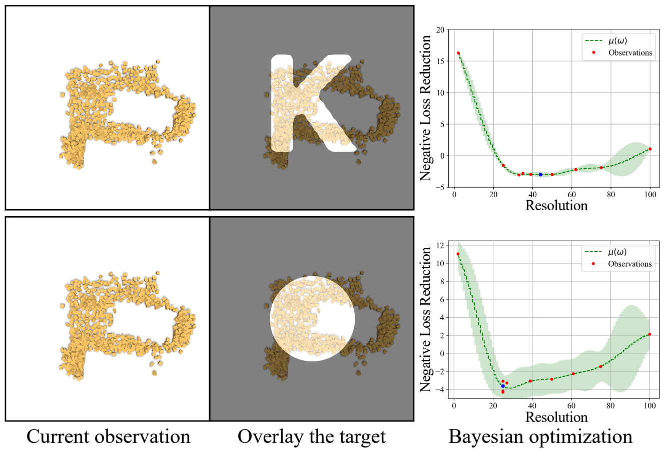
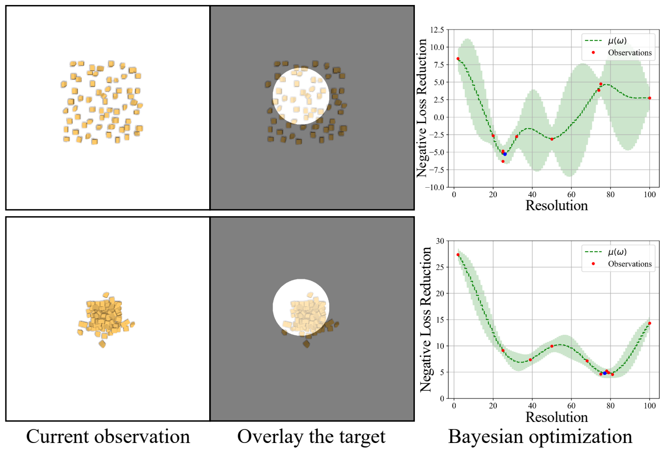

(b) Object piles considered in this work

Figure 3. **Robot setup and the testing object piles.** (a) The dashed black square shows the robot's workspace. The robotic manipulator, equipped with a pusher at the end effector, pushes the object piles within the workspace. A calibrated RGBD camera mounted at the top provides visual observations of the environment. (b) We show the object piles considered in this work, including M&M, almond, granola, candy, carrot, rice, corn, and coffee beans.



(a) Same initial but different goal configurations



(b) Same goal but different initial configurations

Figure 4. **Optimal resolution differs depending on the initial and goal configurations.** (a) We show two examples with the same initial but different goal configurations. We apply Bayesian optimization to solve the problem discussed in Section A.3 to find the optimal resolution for both cases. The example with a more complicated target shape requires a higher-resolution representation to be the most effective at making task progress. (b) When the goal is to gather the pieces in the center of the workspace, a coarse representation is sufficient for examples with spread-out pieces. The task progresses as long as the agent pushes any outlying pieces toward the goal region. In contrast, a higher-resolution representation is needed to reveal the subtle difference between the initial and goal configurations when they are close.

## B. Experiment

### B.1. Setup

We conduct experiments in both the simulation environment and the real world. The simulation environment is built using NVIDIA FleX [31, 36], a position-based simulator capable of simulating the interactions between a large number of object pieces. In the real world, we conducted experiments using the setup shown in Figure 3a. We use RealSense D455 as the top-down camera to capture the RGBD visual observations of the workspace. We attach a flat pusher to the robotic manipulator's end effector to manipulate the object piles.

### B.2. Tasks

We evaluate our methods on three object pile manipulation tasks that are common in daily life.

- **Gather**: The robot needs to push the object pile into a target blob with different locations and radii.
- **Redistribute**: The robot is tasked to manipulate the object piles into many complex target shapes, such as letters.
- **Sort**: The robot has to move two different object piles to target locations without mixing each other.

We use a unified dynamics model for all three tasks, which involve objects pieces of different granularities, appearances, and physical properties (Figure 3b).

### B.3. Trade-Off Between Efficiency and Effectiveness

The trade-off between efficiency and effectiveness can vary depending on the tasks, the current, and the goal configurations. As we have discussed in Section A.3, given the resolution $\omega$, we set a fixed time budget to solve the optimization problem defined in Equation 6. Intuitively, if the resolution is too low, the representation will not contain sufficiently detailed information about the environment to accomplish the task, the optimization of which is efficient but not effective enough to finish the task. On the contrary, if we choose an excessively high resolution, the representation will carry redundant information not necessary for the task and can be inefficient in optimization. We thus conduct experiments evaluating whether the trade-off exists (i.e., whether the optimal resolution $\omega$ calculated from Equation 10 is different for different initial and goal configurations).

We use Bayesian optimization and follow the algorithm described in Section A.3 to find the optimal trade-off on **Gather** and **Redistribute** tasks in the simulation. As shown in Figure 4a, higher-resolution dynamics models do not necessarily lead to better performance due to their optimization inefficiency. Compared between goal configurations, even if the current observation is the same, a more complicated goal typically requires a higher resolution representation to make the most effective task progression. More specifically, when the target region is a plain circle, the coarse representation captures the rough shape of the object pile, sufficient for the task objective, allowing more efficient optimization than the higher-resolution counterparts. However, when the target region has a more complicated shape, low-resolution representation fails to inform downstream MPC of detailed object pile shapes. Therefore, high-resolution representation is necessary for effective trajectory optimization.

The desired representation does not only depend on goal configurations. Even if the goal configurations are the same, different initial configurations can also lead to different optimal resolutions, as illustrated in Figure 4b. When the initial configuration is more spread out, the most effective way of decreasing the loss is by pushing the outlying pieces to the goal region. Our farthest sampling strategy, even with just a few particles, could capture outlying pieces and helps the agent to make good progress. Therefore, when pieces are sufficiently spread out, higher particle resolution does not necessarily contain more useful information for the task but makes the optimization process inefficient. On the other hand, when the initial configuration concentrates on the goal region, to effectively decrease the task objective, MPC needs more detailed information about the object pile's geometry to pinpoint the mismatching area. For example, the agent needs to know more precise contours of the goal region and the outlying part of object piles to decide how to improve the planning results further. Low-resolution representations will be less effective in revealing the difference between the current observation and the goal, thus less helpful in guiding the agent to make action decisions.

## B.4. Is a Single Resolution Dynamics Model Sufficient?

Although there is a trade-off between representation resolution and task progression, can we benefit from this trade-off in trajectory optimization? We compared our dynamic-resolution dynamics model with fixed-resolution dynamics models on **Gather** and **Redistribute** tasks. Figure 1 shows how our model changes its resolution prediction as MPC proceeds in the real world. Trained on the generated dataset of optimal $\omega$, our regressor learned that fixing a resolution throughout the MPC process is not optimal. Instead, our regressor learns to adapt the resolution according to the current observation feedback. In addition, for the example shown in Figure 1, we can see that the resolution increases as object piles approach the goal. This matches our expectation as explained in Section B.3.

We quantitatively evaluate different fixed-resolution baselines and our adaptive representation learning algorithms in simulation. We record the final step distribution distance between object piles and the goal. Specifically, given a distance threshold $\tau_p$, the number of tasks with a distance lower than $\tau_p$ is $N_p$, and the total number of tasks is $N$. The task score is then defined as $N_p/N$ (i.e., y-axis in Figure 5). Our adaptive resolution model almost always achieves the highest task score, regardless of the threshold used.

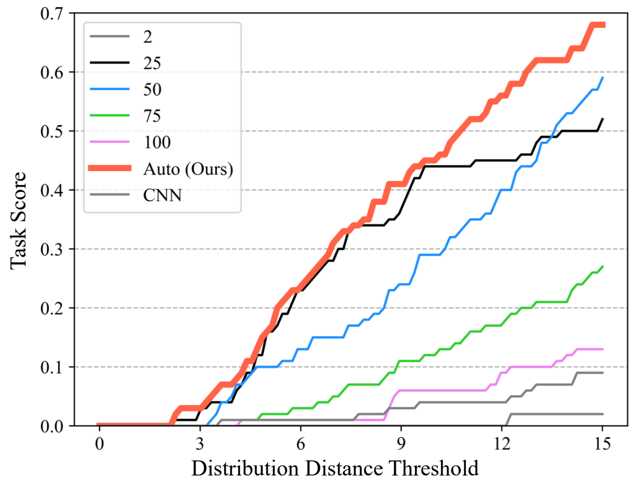


Figure 5. **Model-predictive control (MPC) results.** We evaluated the MPC performance on different representation choices. We use the task score as the evaluation metric. Task execution trial results in a distribution distance lower than the threshold is considered a success. The task score is the number of successful trials divided by the total number of task trials for both the **Gather** and **Redistribute** tasks. Our method automatically and adaptively selects the scene representation, which achieves the best overall performance compared with the scores of fixed-resolution baselines and a method that uses convolutional neural networks (CNN) as the dynamics model class.

Figure 6a shows a qualitative comparison between the fixed-resolution baselines and our dynamic-resolution selection method on the **Gather** task in the real world. All methods start from a near-identical configuration. We can see from the qualitative results that our method manipulates the object pile to a configuration closest to the goal region, whereas the best-performing fixed-resolution baseline still has some outlying pieces far from the goal region. In addition, we could see from the quantitative evaluation curve in Figure 6b that our model is always the best throughout the whole MPC process. Representations with an excessively high resolution are unlikely to converge to a decent solution within the time budget, as demonstrated by resolutions 75 and 100. Conversely, if the representation is too low resolution, it will converge to a loss much higher than our model. A resolution of 25 reached a comparable final loss to our method. However, because the same resolution was ineffective for initial timesteps, its loss does not reduce as rapidly as our adaptive approach. Because our model could adapt to different resolutions in different scenes, making it more effective at control optimization.

That is why our model could reach the goal region faster than all other fixed-resolution models and consistently performs better at all timestamps, highlighting the benefits of adaptive resolution selection.

### B.5. Can a Unified Model Achieve All Tasks?

We further demonstrate that our method could work on all three tasks and diverse object piles. For the **Gather** task, we test our method on different objects with different initial and goal configurations. From left to right in Figure 6c, our agent gathers different object piles made with almond, granola, or M&M$^{TM}$. Different appearances and physical properties challenge our method's generalization capability. For example, while almonds and granola are almost quasi-static during the manipulation, M&M$^{TM}$ will roll around and have high uncertainties in its dynamics. In addition, unlike almonds and M&M$^{TM}$, granola pieces are non-uniform. Our method has a good performance for all these objects and configurations.

For the **Redistribute** task, we redistribute carrots and almonds into target letters 'J', 'T', and 'U' with spread-out initial configurations. The final results match the desired letter shape. Please check our supplementary materials for video illustrations of the manipulation process.

For the **Sort** task, we use a high-level motion planner to find the intermediate waypoints in the image space. Then we use a similar method as **Gather** task to push the object pile into the target location. For the three examples shown in Figure 6e, we require object piles to go to their own target locations while not mixing with each other. Objects with different scales and shapes are present here. For example, coffee beans have smaller granularity and round shapes, while candies are relatively large and square. Here we demonstrate success trials of manipulating the object piles to accomplish the **Sort** task for different objects and goal configurations. Please check our video for the manipulation process.

## C. Related Work

### C.1. Scene Representation at Different Levels

To build multi-scale models of the dynamical systems, prior works have adopted wavelet-based methods and windowed Fourier Transforms to perform multi-resolution analysis [13, 14, 29]. Kevrekidis et al. [26, 27] investigated equation-free, multi-scale modeling methods via computer-aided analysis. Kutz et al. [30] also combined multi-resolution analysis with dynamic mode decomposition for the decomposition of multi-scale dynamical data. Our method is different in that we directly learn from vision data for the modeling and planning of real-world manipulation systems.

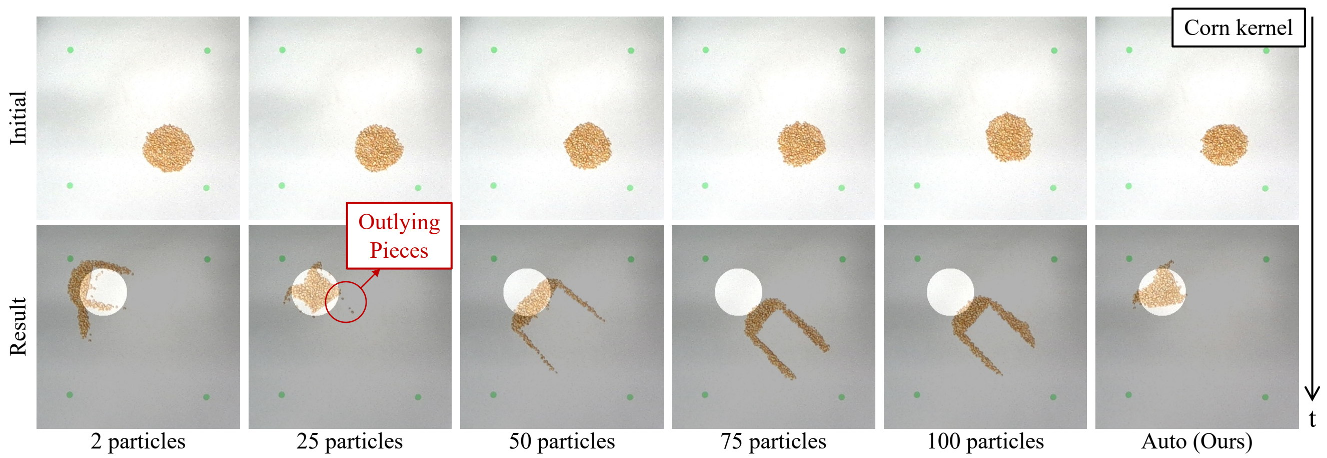
In computer vision, Marr [38] laid the foundation by proposing a multi-level representational framework back in 1982. Since then, people have investigated pyramid methods in image processing [1, 7] using Gaussian, Laplacian, and Steerable filters. Combined with deep neural networks, the multi-resolution visual representation also showed stunning performance in various visual recognition tasks [21, 70]. In the field of robotics, reinforcement learning researchers have also studied task- or behavior-level abstractions and come up with various hierarchical reinforcement learning algorithms [2, 6, 16, 44, 45, 47, 65]. Our method instead focuses on spatial abstractions from vision, where we learned structured representations based on particles to model the object interactions within the environment at different levels.

### C.2. Compositional Model Learning for Robotics

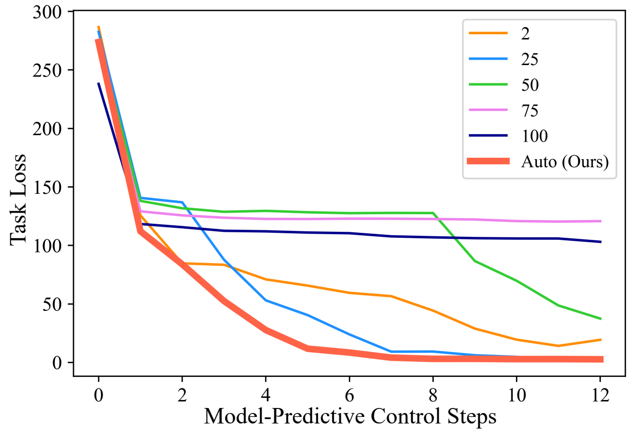
Physics-based models have demonstrated their effectiveness in many robotic manipulation tasks (e.g., [22, 46, 58, 71]). However, they typically rely on complete information about the environment, limiting their use in scenarios where full-state estimation is hard or impossible to acquire (e.g., precise shape and pose estimation of each one of the object pieces in Figure 1). Learning-based approaches provide a way of building dynamics models directly from visual observations. Prior methods have investigated various scene representations for dynamics modeling and manipulation of objects with complicated physical properties, including clothes [24, 34], ropes [10, 41], fluids [33], softbodies [53], and plasticine [54]. Among the methods, graph-structured neural networks (GNNs) have shown great promise by introducing explicit relational inductive biases [4]. Prior works have shown GNNs' effectiveness in modeling compositional dynamical systems involving the interaction between multiple objects [3, 9, 17, 32, 50, 55], systems represented using particles or meshes [31, 43, 48, 51, 64], or for compositional video prediction [23, 49, 62, 67, 68, 69, 72]. However, these works typically assume scene representation at a fixed resolution, whereas our method learns a unified graph dynamics model that can generalize to scene representations at different levels of abstraction.

### C.3. Object Pile Manipulation

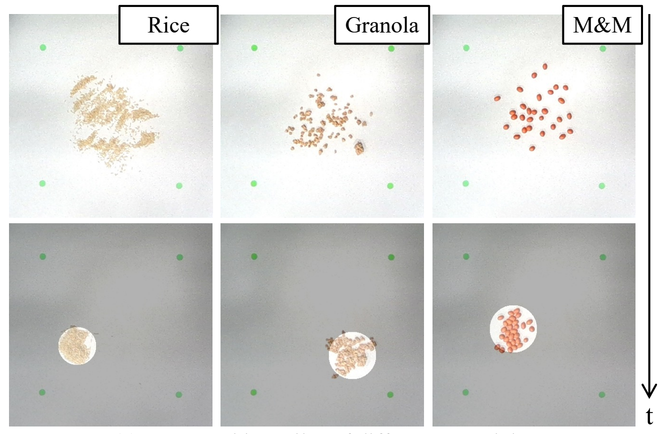
Robotic manipulation of object piles and granular pieces has been one of the core capabilities if we want to deploy robot systems for complicated tasks like cooking and manufacturing. Suh and Tedrake [57] proposed to learn visual dynamics based on linear models for redistributing the object pieces. Along the lines of learning the dynamics of granular pieces, Tuomainen et al. [63] and Schenck et al. [52] also proposed the use of GNNs or convolutional neural dynamics models for scooping and dumpling granular pieces. Other works introduced success predictors for excavation tasks [35], a self-supervised mass predictor for grasping granular foods [59], visual serving for shaping deformable plastic materials [11], or data-driven methods to calibrate the physics-based simulators for both manipulation and locomotion tasks [39, 73]. Audio feedback has also shown to be effective at estimating the amount and flow of granular materials [12]. Our work instead focuses on three tasks us-



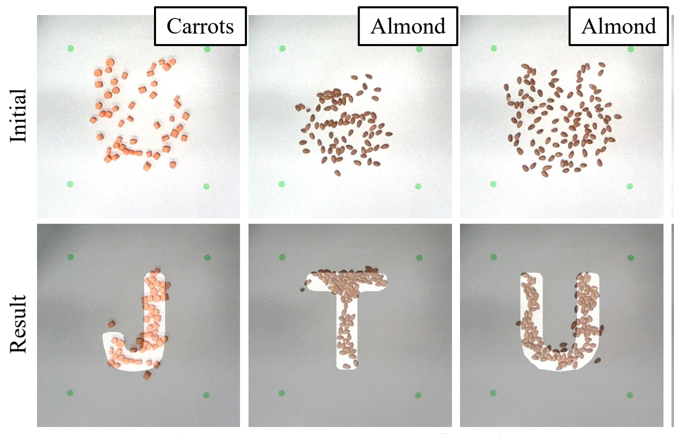
(a) Qualitative comparison between fixed-resolution baselines and our automatic resolution selection method



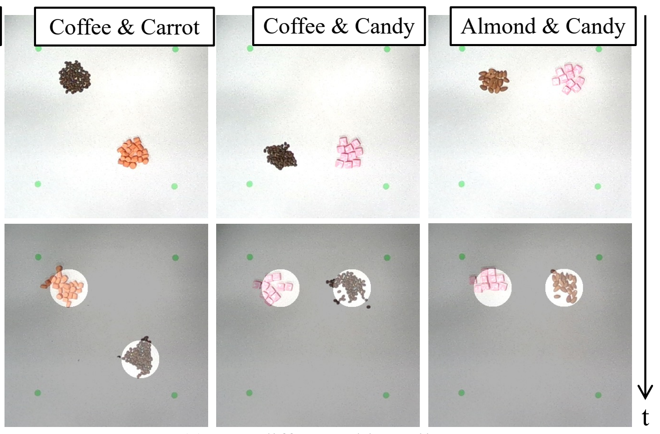
(b) Task loss reduction on the gathering trial shown in (a)



(c) **Gather** object piles of different materials



(d) **Redistribute** to target configurations



(e) **Sort** different object piles

Figure 6. **Qualitative results in the real world.** (a) Our method outperforms in gathering corn pieces into the target region qualitatively. (b) Quantitative comparisons for the qualitative results in (a). Starting from similar initial configurations, our automatic resolution selection method performs the best throughout the MPC steps. (c) Our method is evaluated on the **Gather** task with various objects varying in scales and physical properties. (d) Our method can **redistribute** object pieces into complicated target configurations, such as letter shapes. (e) Our method can be combined with a high-level planner for more complex tasks, such as sorting different object piles into target regions.

ing a unified dynamic-resolution graph dynamics to balance efficiency and effectiveness for real-world deployment.

*Acknowledgments:* This work is in part supported by Stanford Institute for Human-Centered Artificial Intelligence (HAI), Toyota Research Institute (TRI), NSF RI #2211258, ONR MURI N00014-22-1-2740, and Amazon.