Dynamic-Resolution Model Learning for Object Pile Manipulation

Yixuan Wang[1], Yunzhu Li[1,2], Katherine Driggs-Campbell[1], Li Fei-Fei[2], and Jiajun Wu[2]

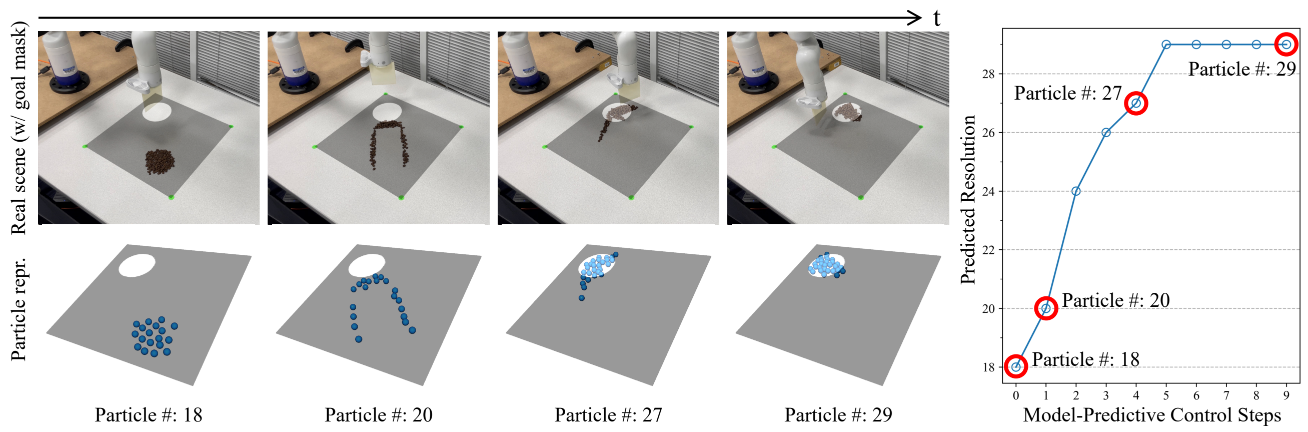

Fig. 1. **Dynamic-Resolution Model Learning for Object Pile Manipulation in the Real World.** Depending on the progression of a task, representations at different granularity levels may be needed at each model-predictive control (MPC) step to make the most effective progress on the overall task. In this 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.**

I. Introduction

Predictive models are core to robotic systems for navigation [8], locomotion [9], and manipulation [7, 21]. 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 [6, 5, 11], object-centric [20, 3] or keypoint representations [13, 12, 19]) 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 [18, 1]. For instance, in object pile 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 [15], 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

[1]University of Illinois at Urbana-Champaign. [2]Stanford University

a resolution regressor that predicts the optimal resolution using self-supervised learning with labels from Bayesian optimization [4]. 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.

We test our model in different object pile manipulation tasks, such as gathering, redistributing, and sorting piles of various objects, including corn kernels, coffee beans, almonds, and candy pieces. Our model can adaptively determine the resolution of the scene representation based on the current observation and task goal, enabling the successful completion of these tasks.

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.

II. Method

In this section, we first present the overall problem formulation. 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 in a closed loop for the downstream planning tasks.

A. Problem Formulation

Our goal is to derive the resolution ω 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 ω, 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 ω 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 II-B, (2) how we obtain the self-supervision for the resolution regressor $g(\cdot,\cdot)$ in Section II-C, and (3) how we solve Equation 1 in a closed planning loop in Section II-D.

B. 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 ω indicates the number of vertices in the graph. The vertices $\mathcal{O}_t = \{o_t^i\}_{i=1,\dots,|\mathcal{O}_t|}$ denote the particle set and o_t^i represents the 3D position of the i^{th} particle. The edge set $\mathcal{E}_t = \{e_t^j\}_{j=1,\dots,|\mathcal{E}_t|}$ 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 deploy the farthest point sampling technique [14] to subsample $\mathcal{O}_t$ from the foreground but ensure sufficient coverage. Different choices of ω 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 ω. In practice, we follow Li et al. [10] and use multi-step message passing over the graph to approximate the instantaneous propagation of forces. The dynamics model is trained using the mean squared error (MSE).

C. Adaptive Resolution Selection

The previous sections discussed how to obtain the particle set and how we predict its evolution given a resolution ω. 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 ω 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 ω as a variable of the objective function as the following:

$$\begin{aligned} c^*(y_0, y_g, \omega) = \min_{\{u_t\}} \quad & c(\boldsymbol{z}_T^{\omega}, y_g), \\ \text{s.t.} \quad & \boldsymbol{z}_0^{\omega} = h(y_0, \omega), \\ & \boldsymbol{z}_{t+1}^{\omega} = f(\boldsymbol{z}_t^{\omega}, u_t, \omega). \end{aligned} \tag{2}$$

For a given ω, we solve the above optimization problem via a combination of sampling and gradient descent using shooting methods [17] under a given time budget—the higher resolution

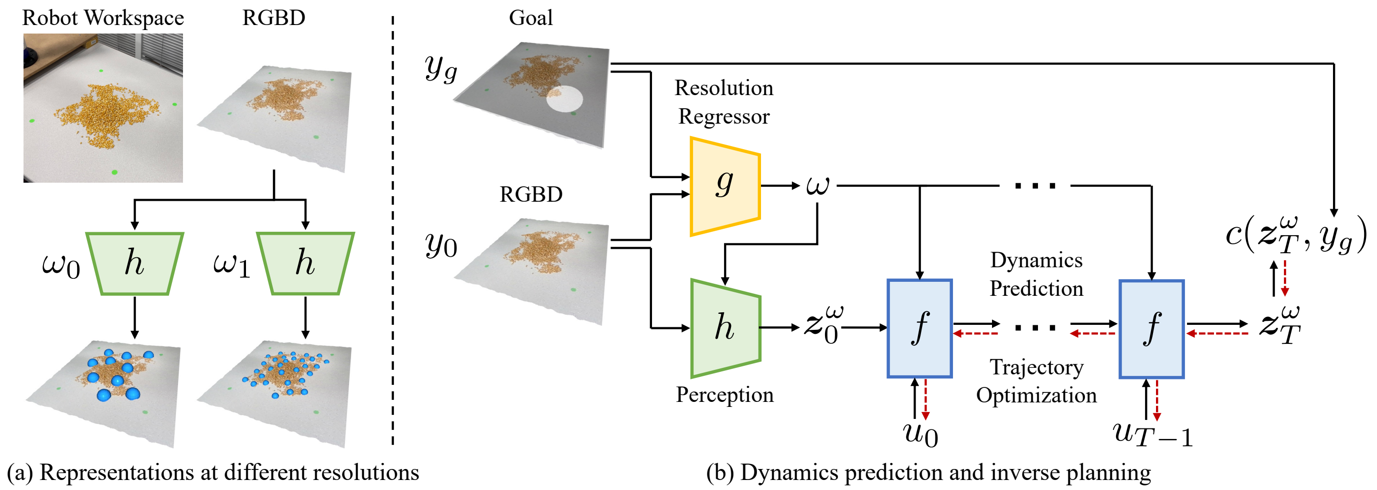

Fig. 2. **Overview of the proposed framework.** (a) Our perception module h processes the input RGBD image and generates particle representations at different levels of abstraction depending on the resolution ω. (b) The resolution regressor g takes the current observation y_0 and the goal y_g as input.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

representation will go through fewer optimization iterations. For simplicity, we denote the objective in Equation 2 as $c^*(\omega)$ in the following part of this section.

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

$$\begin{aligned} \min_{\omega} \quad & c^+(\omega) = c^*(\omega) + R(\omega), \\ \text{s.t.} \quad & \omega \in (\omega_{\min}, \omega_{\max}), \end{aligned} \tag{3}$$

where $R(\omega)$ is a regularizer penalizing the choice of an excessively large ω to encourage efficiency. We use Bayesian optimization [16] to find the optimal ω by iteratively sampling ω and approximating $c^+(\omega)$ using the Gaussian process.

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 ω^*. 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 ω regressor is a self-supervised learning process, as the labels are automatically generated via an optimization process without any human labeling.

D. 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 [2]. Figure 2b also shows an overview of the future prediction and inverse planning process.

III. 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 step?

A. Tasks

- **Gather**: Push the object pile into a target blob.
- **Redistribute**: Push the object piles into complex shapes.
- **Sort**: Sort two object piles to target locations without mixing each other.

B. Trade-Off Between Efficiency and Effectiveness

The trade-off between efficiency and effectiveness can vary depending on the current, and the goal configurations. As we have discussed in Section II-C, given the resolution ω, we set a fixed time budget to solve Equation 2. Intuitively, if the resolution is too low, the representation will not contain sufficiently detailed information to accomplish the task, the optimization of which is efficient but not effective enough to finish the task. On the contrary, the representation will carry redundant information for the task and can be inefficient in optimization for excessively high resolution.

We use Bayesian optimization and follow the algorithm described in Section II-C to find the optimal trade-off on **Gather** and **Redistribute** tasks in the simulation. We found higher-resolution dynamics models do not necessarily lead to better performance due to their optimization inefficiency.

C. Is a Single Resolution Model Sufficient?

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 ω, 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.

References

[1] 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

[2] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013. 3

[3] 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

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

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

[6] 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

[7] 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

[8] 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

[9] 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

[10] 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. 2

[11] 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] 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

[13] 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

[14] Carsten Moenning and Neil A Dodgson. Fast marching farthest point sampling. Technical report, University of Cambridge, Computer Laboratory, 2003. 2

[15] 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. 1

[16] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012. 3

[17] 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. 2

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

[19] Weiyao Wang, Andrew S Morgan, Aaron M Dollar, and Gregory D Hager. Dynamical scene representation and control with keypoint-conditioned neural radiance field. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 1138–1143. IEEE, 2022. 1

[20] 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

[21] 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