Active Learning of Model Preconditions for Model-Deviation Aware Planning in Deformable Object Manipulation

Alex LaGrassa[1], Moonyoung Lee[1], and Oliver Kroemer[1]

***Abstract*— This paper presents an algorithm for actively selecting trajectories in order to train an estimator that predicts model deviation of an inaccurate dynamics model. The trained estimator is then used during test time to constrain planning to regions of state-action space where the dynamics model is predicted to be accurate with high probability. As collecting state-action space data can be costly, we propose an active learning algorithm to minimize data collection needed to reliably train a model deviation estimator. Preliminary results show improvements in both test performance, and in the qualitative selection of trajectories selected by the active learning approaches. We further show experiments that analyze the effect of using different acquisition functions on trajectories chosen during training, and on the test performance.**

I. Introduction

Although it is possible to leverage the long-horizon reasoning of model-based planning with deformable objects, accurate general-purpose modeling of deformable objects is difficult and computationally expensive [1, 9]. Furthermore, the resulting models are often inaccurate and require large amounts of data to train. This makes it hard to plan and robustly execute robot motions that involve deformable objects in settings where data is expensive.

In this paper, we address the problem of model-based planning with inaccurate deformable object models by actively collecting data to predict model error of a special-purpose deformable object model, which can then be used in planning to obtain more robust plans. Previous work has shown that a model deviation estimator (MDE) can predict model deviation, which can then be used as a constraint in planning as a *model precondition*: a region of state-action space where a dynamics model is sufficiently accurate for planning [8, 5]. MDEs can identify and reason about patterns in which state-action pairs where the model tends to deviate from real-world dynamics, but require ground-truth data which can be expensive.

Active learning can enable sample-efficient learning of expensive-to-compute functions [12]. In this work, we use the MDE to perform active learning to minimize data collection in the real world. Collecting and defining a good MDE dataset is challenging because the dataset needs to contain enough trajectories to find plans that are useful for tasks while being able to reject plans that would lead to undesirable results. Furthermore, when actively collecting data for a trajectory, model error in the earlier states can lead the robot to be unable to gather data from the later states due to the sequential nature of the problem. We show

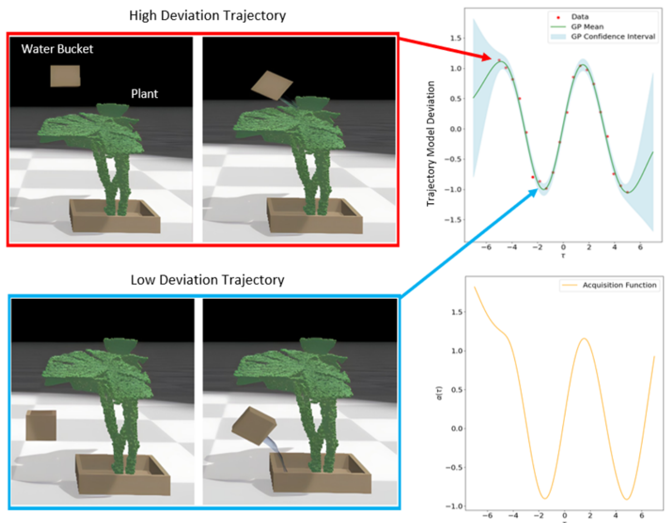


Fig. 1: Sketch of active learning approach over trajectories. The algorithm selects a trajectory that minimizes $\alpha(\tau)$, which is a function of estimated model deviation on the trajectory. In this example, a generally low deviation trajectory (bottom) scores higher than the high deviation trajectory (top), so that trajectory would be executed to collect data to estimate model deviation.

the intuition for the desired behaviour of our active learning approach over trajectories in Figure 1.

In this paper, we present an approach for active learning of a model deviation estimator to improve the accuracy and robustness of plans for a plant watering task. We then examine results analyzing the effect of several different acquisition functions on trajectories chosen during training, and reliability of the plans executed at test time.

II. Problem formulation

The goal of this work is to select a dataset $\mathcal{D}$ of (s, a, s') tuples in order to train an MDE that uses $\mathcal{D}$. We assume we are given a distribution of planning problems that can be sampled from in the form of start states and goal states, a pre-existing dynamics model $\hat{f}(s, a)$ that can compute the next state s', and a planner that uses $\hat{f}(s, a)$ with the corresponding MDE to find plans to the goal. Plans are trajectories of high-level parameterized actions $a_{0:T-1}$ and predicted states $s_{0:T}$ such that s_T is a goal state. Each trajectory is denoted by τ. The agent may use the planner during training time and sample from the same *distribution* of planning problems that will be seen at test time, but not the same problems. We assume that the state needs to be fully observable to extract meaningful distances between s' and $\hat{f}(s)$, and correspondingly a distance function of the form $d(s_i, s_j)$ that returns a scalar value representing the distance between two states.

[1]Robotics Institute, Carnegie Mellon University

III. Related work

We build on work for predicting and avoiding model error [7, 8], which has also been done using probabilistic models [10, 4] as a form of model preconditions, as well as work in skill precondition learning. Some work has used GPs to define skill preconditions because of their data-efficiency and meaningful uncertainty quantification [11, 12, 13]. The most closely related work to active model precondition learning is active skill precondition learning [12], which also uses GPs to determine which action parameters to sample for successfully executed skills given a context variable that represents the state. In contrast, this work is for a model precondition, which is more specific than a skill precondition though related in many cases. Furthermore, [12] does not account for the sequential nature of trajectories in active learning: the most informative point may not be reached with the current controller and model if the model is inaccurate. [3] addresses the sequential dependence of selecting points that require following a controller to reach by separately optimizing information gain to select a point and steering towards it. The results in [2] show that optimizing over trajectories instead of single points leads to more efficient exploration and better trajectories, which we do as well to learn MDEs instead of the dynamics.

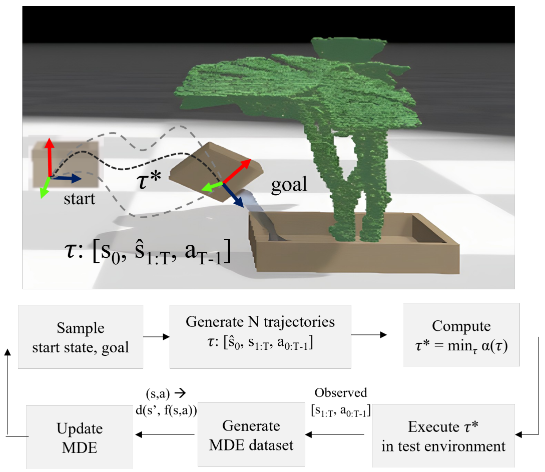
IV. Approach

We first illustrate the intuition for our approach. We then describe our proposed algorithm that uses knowledge of where and how much the model is predicted to deviate to determine how to prioritize data collection.

Intuitively, our approach efficiently collects data to define a model precondition. Data is sampled efficiently by prioritizing coverage of diverse dynamics so that the planner can consistently compute plans within the model precondition. We select trajectories that are relevant for planning by using trajectories that reach a goal while maintaining a balance between success and exploration through a utility function that prioritizes low error and an exploration bonus based on the predicted variance of $\hat{d}(s, a)$.

We estimate the model precondition using a model deviation estimator(MDE), which predicts a distribution for $d(s', \hat{f}(s, a))$ given (s, a) that we denote as $\hat{d}(s, a)$. The distribution is a Gaussian distribution with mean $\mu(s, a)$ and $\sigma(s, a)$. The model precondition of a model $\hat{f}$ is defined as: $\text{pre}(\hat{f}) = \{(s, a) | P(\hat{d}(s, a) > d_{\max}) < \delta\}$ for some small δ. The constraint is defined as $\mu(s, a) + \beta\sigma(s, a) < d_{\max}$, where β is set based on δ to tune the risk tolerance.

We now describe our algorithm for actively learning MDEs. The algorithm works as follows and is illustrated in Figure 2. Starting in the upper left corner, the agent samples a planning problem, generates a set of diverse candidate trajectories, and then finds a trajectory that minimizes a utility function $\alpha(\tau)$. It then executes that trajectory in the real world and saves the observed (s, a, s') tuples. One a batch of M trajectories has been executed, the robot updates the MDE using the new data.


start
goal
τ^*
τ: $[s_0, \hat{s}_{1:T}, a_{T-1}]$
Sample start state, goal
Generate N trajectories τ: $[\hat{s}_0, s_{1:T}, a_{0:T-1}]$
Compute $\tau^* = \min_\tau \alpha(\tau)$
Execute τ^* in test environment
Observed $[s_{1:T}, a_{0:T-1}]$
Generate MDE dataset
$(s,a) \rightarrow d(s', f(s,a))$
Update MDE
Fig. 2: Overview of our method. At each iteration, a planning problem is sampled, and then a trajectory that minimizes the acquisition function is executed. After every M trajectories, the agent updates the MDE.

At test time, the robot uses the final model deviation estimator trained on data $\mathcal{D}$. It finds plans predicted to be robust by rejecting transitions that are predicted to have a deviation higher than a threshold $\delta_{\max}$ that the system can correct in closed loop-control.

A. Learning MDEs

We use a standard Gaussian Process (GP) regression model with the addition of a heteroscedastic noise model. Data collected for the MDE is in the form of (s, a, s') tuples from executing action a in the target environment (expected to be the real world) from state s, and then observing s'. The labels for the MDE dataset are the distances between s' and the predicted next state $\hat{f}$: $d(\hat{f}(s, a), s')$. For numerical reasons, we normalize the labels to be zero-mean and unit-variance by subtracting the mean of $d(\hat{f}(s, a), s')$, μ_z and dividing by the standard deviation of the data, σ_z. To maintain a zero-mean prior in the space of $\hat{d}$, we fix the prior mean of the GP to $-\frac{\mu_z}{\sigma_z}$. The kernel is a Matern kernel and all hyperparameters are optimized for the z-score normalized data. Input features remain unscaled. Whereas a homoscedastic GP was sufficient in [4], we needed to use a heteroscedastic noise model for this more complex problem.

B. Active learning

In active learning, the robot samples a set of N candidate trajectories using RRT, and then selects the one that minimizes $\alpha(\tau)$. The value for $\alpha(\tau)$ depends on the values for each individual (s, a) pair along the trajectory, which is of the form $s_{1:T}, a_{1:T-1}$. The acquisition function we use for each individual t is denoted as $\alpha_t(s_t, a_t)$. We define $\alpha_t(s, a)$ in a form inspired from Lower Confidence Bound: $\mu(x) - c\sigma(x)$ where c (which can be negative) is a scaling factor that controls the degree to which exploration is encouraged. We propose a form of our acquisition function as follows:

$$\min_\tau \max_{t<len(\tau)} \gamma^t \mu(\hat{d}(s, a)) + c\sigma(\hat{d}(s, a))) \qquad (1)$$

We fix $d_{\max}$ for all of training but allow a broader training model precondition by increasing β as the robot accumulates more data.

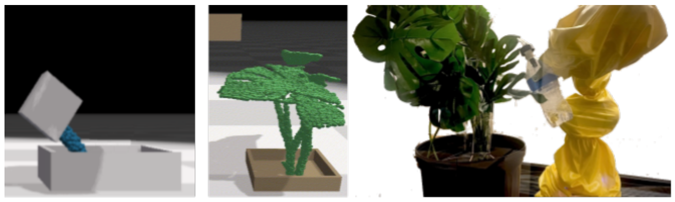
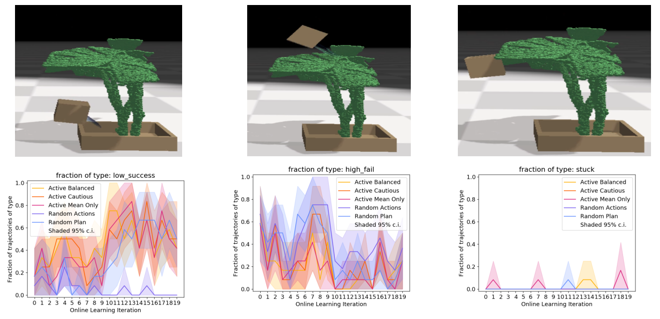

a) Training environment b) Simulator test environment c) Real-world test environment

Fig. 3: Experimental setup. a) The simulated environment where data is collected to train an initial model. There are only two containers for pouring, and simulation is inexpensive. b) a simulated environment for evaluation with a plant that we use for more extensive evaluation c) the real-world environment



(a) successfully pour *above* the plant (b) unsuccessfully pour *above* (c) stuck in plant

Fig. 4: Ratio of types of trajectories observed during training (examples of each shown above)

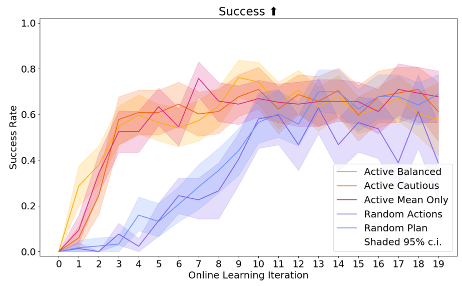


Fig. 5: Learning curves of the success rate of finding and executing plants to the goal at test time. The shaded region is the 95% confidence interval.

V. Experiments

We conduct experiments in a simulated water pouring environment [6] using a dynamics model trained in less expensive simulated environment without a plant, shown along with its real-world equivalent in Figure 3.

The initial poses of the target box is fixed but the initial pose of the controlled box is randomized across a wide range, and the plant pose is varied slightly. The task is to pour a desired amount of water into the target box without spilling more than two percent. The state space for planning is the poses and volumes of both boxes. We run each experiment with 4 random seeds for 20 iterations containing 10 trajectories in each iteration. 9 candidate trajectories are generated for each of the active methods. During training, we use $d_{\max} = 0.07$ with $\beta = -2$ for iterations below 10, and $\beta = 1$ otherwise as a constraint during planning.

The baselines we compare against are *Random Actions*, which samples a trajectory of length 6 using the action sampler the planner uses, and *Random Plan*, which selects a random trajectory that reaches the goal. The purpose of RP is to test the effect of active learning using the MDE over only using data relevant to planning. To compare different ways of using the MDE during planning, we compare three different values of c. The higher c is, the higher the exploration bonus. *Active Cautious* (AC) uses $c = -0.1$, *Active Balanced* (AB) uses $c = 0.1$, and *Active Mean Only* tests the impact of only using the mean by setting $c = 0$.

Trajectory Diversity and Dataset Analysis: We first analyzed the types of trajectories collected during online learning with various acquisition functions. We observe the most significant difference in the fraction of pours that successfully poured into the target container without spilling where active learning methods tend to achieve more successful pours below the plant leaves with less data(Figure 4a). The ratios of pours above the plant are comparable across all methods including *Active Cautious*, indicating that the negative incentive to explore is not affecting trajectory types significantly. Very few samples get stuck for all methods. We hypothesize that the low effect of the exploration bonus is due to the low prevalence of low-mean, high-variance points when sampling randomly.

End-to-end Performance Metrics: We evaluated the end performance of the proposed approach by monitoring the success rate in reaching the goal at test time by using the MDE learned on datasets acquired from each algorithm to form a model precondition defined using $d_{\max} = 0.07$ and β=2. For each iteration, the planner with MDEs learned using each method are evaluated for 30 randomly sampled planning problems. Results are shown in Figure 5.

We observe a large data efficiency improvement when using an active learning algorithm over the baselines. Iterations 1 and 2 may inconclusively indicate a small benefit with very low data with higher c. The performance for all algorithms that sample trajectories to the goal converges to finding plans by iteration 13. Although the success rate in finding plans is significantly lower for *Random Actions*than for *Random Plan*, we find a smaller gap in success reaching the goal, which indicates potentially more accurate MDEs when not constraining data to be from plans.

VI. Conclusions

Overall, our approach provides a promising solution for actively learning model deviation estimators to improve the accuracy and robustness of robot motion planning, particularly in the context of deformable object manipulation. Preliminary results show that our algorithm for active learning can use the MDE to identify informative samples efficiently and effectively, but too small of an effect in how the variance estimate is used to conclude anything about how it affects exploration. Our most immediate future goals are to evaluate the impact on acquisition functions on the accuracy of model preconditions. To more thoroughly evaluate the accuracy of the learned model preconditions, we will evaluate how well-calibrated the MDE predictions are on a set of test trajectories. We have also been validating that the algorithm works on the more challenging real-robot environment shown in Figure 3.

References

[1] Christopher J Bates, Ilker Yildirim, Joshua B Tenenbaum, and Peter Battaglia. “Modeling human intuitions about liquid flow with particle-based simulation”. In: *PLoS computational biology* 15.7 (2019), e1007210.

[2] Mona Buisson-Fenet, Friedrich Solowjow, and Sebastian Trimpe. “Actively learning gaussian process dynamics”. In: *Learning for dynamics and control*. PMLR. 2020, pp. 5–15.

[3] Alexandre Capone, Gerrit Noske, Jonas Umlauft, Thomas Beckers, Armin Lederer, and Sandra Hirche. “Localized active learning of Gaussian process state space models”. In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 490–499.

[4] Alex LaGrassa, Steven Lee, and Oliver Kroemer. “Learning Skills to Patch Plans Based on Inaccurate Models”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021*. IEEE, 2020, pp. 9441–9448.

[5] Alex Licari LaGrassa and Oliver Kroemer. “Learning Model Preconditions for Planning with Multiple Models”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 491–500.

[6] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. “SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 432–448.

[7] Dale McConachie, Thomas Power, Peter Mitrano, and Dmitry Berenson. “Learning When to Trust a Dynamics Model for Planning in Reduced State Spaces”. In: *IEEE Robotics and Automation Letters* (2020).

[8] Peter Mitrano, Dale M^{c}Conachie, and Dmitry Berenson. “Learning Where to Trust Unreliable Models in an Unstructured World for Deformable Object Manipulation”. In: *Science Robotics* (2021).

[9] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. “Learning Mesh-Based Simulation with Graph Networks”. In: *International Conference on Learning Representations*. 2020.

[10] Thomas Power and Dmitry Berenson. “Keep It Simple: Data-Efficient Learning for Controlling Complex Systems With Simple Models”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 1184–1191.

[11] Rin Takano, Hiroyuki Oyama, and Yuki Taya. “Robot Skill Learning with Identification of Preconditions and Postconditions via Level Set Estimation”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 10943–10950. DOI: 10.1109/IROS47612.2022.9981933.

[12] Zi Wang, Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. “Active Model Learning and Diverse Action Sampling for Task and Motion Planning”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Oct. 2018, pp. 4107–4114. DOI: 10.1109/IROS.2018.8594027.

[13] Zi Wang, Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. “Learning compositional models of robot skills for task and motion planning”. In: *The International Journal of Robotics Research* 40.6-7 (2021), pp. 866–894.