2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

ACTIVEPUSHER: Active Learning and Planning with Residual Physics for Nonprehensile Manipulation

Zhuoyun Zhong, Seyedali Golestaneh, and Constantinos Chamzas

***Abstract*— Planning with learned dynamics models offers a promising approach toward versatile real-world manipulation, particularly in nonprehensile settings such as pushing or rolling, where accurate analytical models are difficult to obtain. However, collecting training data for learning-based methods can be costly and inefficient, as it often relies on randomly sampled interactions that are not necessarily the most informative. Furthermore, learned models tend to exhibit high uncertainty in underexplored regions of the skill space, undermining the reliability of long-horizon planning. To address these challenges, we propose ACTIVEPUSHER, a novel framework that combines residual-physics modeling with uncertainty-based active learning, to focus data acquisition on the most informative skill parameters. Additionally, ACTIVEPUSHER seamlessly integrates with model-based kinodynamic planners, leveraging uncertainty estimates to bias control sampling toward more reliable actions. We evaluate our approach in both simulation and real-world environments, and demonstrate that it consistently improves data efficiency and achieves higher planning success rates in comparison to baseline methods. The source code is available at https://github.com/elpis-lab/ActivePusher.**

I. INTRODUCTION

Model-based planning methods offer a powerful framework for generalizing robotic behavior and enabling long-horizon decision making [1]. Such methods often require a predictive model of the system's dynamics, especially in kinodynamic planning where dynamic constraints must be considered. The effectiveness of these approaches thus critically depends on the accuracy of the underlying forward dynamics model. Inaccuracies in this model can cause cascading errors during execution, particularly in contact-rich settings such as nonprehensile manipulation (e.g., pushing and rolling), where even minor deviations in predicted trajectories may accumulate and lead to irrecoverable task failure.

Accurately modeling the dynamics for these tasks is challenging. Analytical physics-based models often rely on simplified assumptions about friction, contact geometry, and mass distribution, making them brittle in practice [2]. As an alternative, data-driven approaches can learn dynamics directly from interaction data, either from scratch or by refining simplified analytical models through residual learning [3], [4]. However, these methods face two key limitations in real-world robotic settings:

- **Sample inefficiency:** Learning accurate models often requires large amounts of interaction data, which is costly and time-consuming to collect on physical systems.

This work was supported in part by NSF CRII Grant No. 2451108, an Amazon WPI RBE Gift, and WPI funds. All authors are affiliated with the Department of Robotics Engineering, Worcester Polytechnic Institute (WPI), Worcester, MA 01609, USA {zzhong3, sgolestaneh, cchamzas} @ wpi.edu

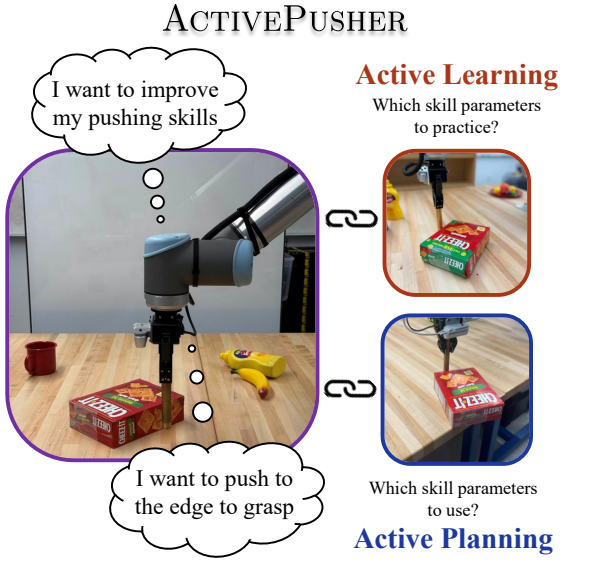


Fig. 1: Two key challenges addressed by ACTIVEPUSHER. During learning, the robot must choose the most informative skill parameters to efficiently improve its skills (Active Learning). When planning, the robot should select skill parameters with low model uncertainty to ensure reliable task completion (Active Planning).

- **Inaccuracy in underexplored regions:** Models may perform poorly in sparsely explored regions of the skill space, leading to unreliable predictions and large deviation from plans during execution.

In this paper, we propose ACTIVEPUSHER, a framework that tightly integrates residual physics, active learning, and active kinodynamic planning, to address both challenges in the context of nonprehensile pushing, illustrated in Fig. 1. The core idea is to quantify epistemic uncertainty in a learned neural network model with residual physics. This uncertainty estimate allows the system to actively target informative skill parameters for learning and reliable ones for planning.

To estimate the epistemic uncertainty of a neural network model, we leverage the Neural Tangent Kernel (NTK) [5]. During learning, rather than sampling pushing actions at random to practice, ACTIVEPUSHER actively queries the NTK for uncertainty estimation and identifies actions with the highest expected information gain using Batch Active learning via Information maTrices (BAIT) strategy [6]. This targeted exploration enables the model to improve rapidly with far fewer interactions. During planning, the uncertainty estimates are incorporated into a sampling-based kinodynamic planner [1], biasing action sampling toward high-confidence skill parameters to maximize task success.

By focusing on where the model is uncertain to learn, and where the model is certain to act, our approach tightly integrates learning and planning, enabling robust nonprehensile manipulation with few real-world interactions per task. Crucially, ACTIVEPUSHER operates without high-fidelity simulation, large offline datasets, or human demonstrations. Our main contributions are:

- **Active learning of skill models**. We introduce a principled framework for data-efficient nonprehensile skill learning by selecting skill parameters that maximize expected information gain, enabling data collection in the most informative manner.
- **Active uncertainty-aware planning**. We propose a novel planning strategy that integrates model uncertainty into an asymptotically optimal kinodynamic planner, guiding action sampling toward reliable actions and improving overall task success rate.
- **Empirical validation in simulation and the real world**. We demonstrate the effectiveness of our approach with multiple objects and nonprehensile manipulation tasks, showing significantly improved data efficiency and planning success over baselines.

II. Related Work

ACTIVEPUSHER draws ideas from several areas, such as residual learning, active learning, and kinodynamic planning. Here, we briefly review each of these areas in the context of nonprehensile manipulation, with a focus on pushing.

Residual Model Learning combines the strengths of analytical and data-driven approaches by training a neural network to predict corrections to an approximate physics model. In robotic manipulation, analytical models and simulations can offer useful priors but are often coarse approximations of real dynamics, sensitive to assumptions on physical parameters [7]. Conversely, fully data-driven methods [8] can model complex behaviors but require large amounts of real-world data. Residual learning reduces this burden by modeling only the error between physics and reality, improving data efficiency and real-world performance [3], [4]. Building on this paradigm, ACTIVEPUSHER further improves data efficiency by actively selecting informative data points for refinement.

Active Learning is a well-established topic in machine learning that improves data efficiency by actively selecting which data points to label with model uncertainty [6], [9], [10]. This paradigm naturally aligns with self-supervised robotic learning settings, where the data collection is expensive, and the robot should autonomously choose which interactions to collect. Several robotic learning methods have leveraged active learning to improve learning efficiency [11]. In the context of skill learning, recent approaches [12], [13] have applied active learning to accelerate skill learning with binary outcome using Gaussian Process (GP). Uncertainty estimates from such models have also been used to provide robustness in control synthesis. [14]. However, prior active skill learning methods focus primarily on binary success classification, with limited attention to regression-based skill models using neural networks. In contrast, we apply active learning in training predictive skill models with neural networks, and enable their integration into kinodynamic planners.

Nonprehensile Manipulation, particularly planar pushing problems, has been addressed by many approaches such as Model Predictive Control (MPC) and reinforcement learning (RL). RL methods [15], [16] can acquire complex behaviors, but require large amount of interaction data and often fail under distribution shift and sim-to-real transfer. MPC approaches that embed learned or physics models [7], [17], [18] provide robust execution but require high-frequency state observation and control for closed-loop correction. Meanwhile, both methods remain inherently myopic and prone to local minima. On the other hand, kinodynamic motion planning methods provide global reasoning and can be generalized to complex environments [19], [20]. However, previous works often rely on analytical models, assume regular-shaped objects, and handle execution errors through online replanning [20], which cannot always recover from failure. Our approach instead leverages a learned dynamics model and its uncertainty estimates to actively select reliable actions to build plans, enabling robust plans from the beginning.

III. Problem Statement

Kinodynamic Planning: Let $x \in \mathcal{X}$ denote the state and state-space, and $u \in \mathcal{U}$ denote the control and control-space of a robotic system [1]. We consider a dynamic system that follows time-invariant differential equations:

$$\dot{x}(t) = f^*(x(t), u(t)) \tag{1}$$

where f^* is the true (unknown) forward dynamics model of the system. Let $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$ denote the obstacle (invalid) state space, and $\mathcal{X}_{\text{free}} = \mathcal{X} \setminus \mathcal{X}_{\text{obs}}$ denote the free (valid) space. The start state is $x_{\text{start}} \in \mathcal{X}_{\text{free}}$, and the goal region is $X_{\text{goal}} \subseteq \mathcal{X}_{\text{free}}$.

The *kinodynamic planning* problem is to determine a control duration $\mathcal{T}$ and a control function $u : [0, \mathcal{T}] \to \mathcal{U}$ such that the resulting trajectory satisfies $x(0) = x_{\text{start}}$, $x(\mathcal{T}) \in X_{\text{goal}}$, and $x(t) \in \mathcal{X}_{\text{free}}$ for all $t \in [0, \mathcal{T}]$.

In this work, we focus on a planar pushing task, modeled as kinodynamic planning. We treat the state of the manipulated object as the system state and adopt an object-centric isotropy assumption, where the effect of a push is invariant to the object's current pose. In addition, we assume that a static equilibrium condition is achieved after each action. Formally, this reduces the system dynamics to:

$$\dot{x}(t) = f^*(u(t)) \tag{2}$$

Active Learning of Forward Dynamics Models: We define an interaction as the application of a control and observation of the associated states. Under the isotropic assumption, the interaction is simplified to $(u,\ \dot{x})$. A set of applied controls and observed state changes forms a dataset for model learning.

The *active learning of forward dynamics models* problem is to approximate the unknown dynamics model f^* with a learned model f, such that it predicts the outcomes of applied controls with high predictive accuracy, while minimizing the number of interactions required for training.

Active Kinodynamic Planning: Inspired by the concept of active learning with uncertainty quantification [6], [9], we define *active kinodynamic planning* as kinodynamic planning that explicitly incorporates uncertainty estimation from the learned dynamics model. The problem aims at finding a control trajectory $\bar{\mathbf{u}}$ such that the resulting trajectory $\bar{\mathbf{x}}$ satisfies the planning requirements with high confidence.

IV. Methodology

In this work, we discretize the continuous dynamic system into a discrete-time formulation. Each control action corresponds to a parameterized push skill that is executed over a fixed duration, and the system is observed only at the terminal state. This design choice allows us to work with skill abstractions and is consistent with practical sensing conditions, where object poses are often observable only after the push concludes (e.g., due to vision occlusion). The system dynamics becomes:

$$\Delta x_n = f^*(u_n) \tag{3}$$

We represent each object as a two-dimensional oriented bounding box (OBB) and the planar push control is parameterized by three variables, as shown in Fig. 2:

$$u = (s, o, d) \tag{4}$$

where $s \in \{1, \ldots, 4\}$ selects one of the box's sides, o is the lateral offset along that side from the center, and d is the total push distance. The end-effector follows a straight-line fixed-duration ($\tau = 2$ s) sinusoidal velocity profile:

$$v(t) = \frac{d}{\tau}\left[\sin\left(2\pi\,\frac{t}{\tau} - \frac{\pi}{2}\right) + 1\right] \tag{5}$$

The system's state x is defined as the object's $SE(2)$ state T. The effect of the push $f^*(u)$ is therefore defined as the $SE(2)$ transformation between the object's initial and final poses. Unlike Euclidean states with additive updates, evolution on transformation is expressed through multiplication:

$$\begin{aligned} T_{n+1} &= T_n f^*(u_n) = T_n \Delta T_n, \\ f^*(u_n) &= \Delta T_n = {T_n}^{-1} T_{n+1} \end{aligned} \tag{6}$$

A. Residual Physics

In this section, we introduce the model $f(u)$, which will be trained to approximate the unknown true dynamics $f^*(u)$. To effectively predict in a low-data setting, we adopt the approach of learning residual physics, which integrates a physics-based model with a neural network [3], [4]. Rather than replacing the physics-based model, the neural network is tasked with learning the residual error, i.e. deviations from the idealized model output to the real observations. This preserves the physical plausibility while allowing the learned component to correct and improve overall accuracy.

For the analytical model, we follow the dynamic model proposed in [21] to predict object behavior given pusher motion. In this modeling, the object is treated as a rigid rectangle pushed under quasi-static assumption, with frictional forces obeying Coulomb's law. The model further requires

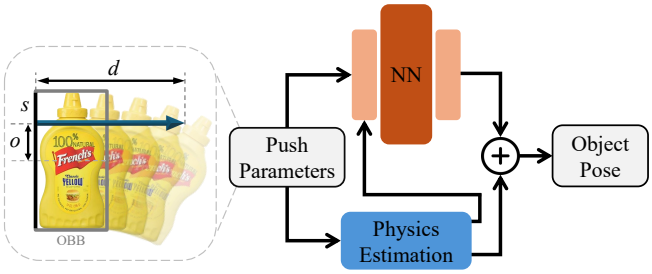


Fig. 2: Neural network (NN) with residual physics architecture. The network takes both the control parameters and the output of the physics model to predict residuals, which are then added to the physics-based output to produce the final prediction.

knowledge of the object's shape and the ratio of frictional moment to frictional force c_{ratio}. However, we do not assume having access to these exact parameters. Thus, the analytic prediction serves only as a coarse estimation on how the object will move. Furthermore, to keep the model compatible with our neural network components and enable efficient batch operations, additional simplifications are applied. We set a fixed $c_{ratio} = 0.0187$ as recommended in [21]. Also, we assume the contact point is fixed and the force is perpendicular to the point, maintaining perfect sticking contact throughout the push. Please refer to our source code for more details.

As illustrated in Fig. 2, our neural network takes both the skill parameters and the output of the physics equation as input. This design enables the network to reason about both the nominal dynamics and the data-driven corrections required to account for object-specific and contact-specific variations. The network finally outputs the residual, and the model combines it with the physics equation output to provide the final estimate. We train the combined model by minimizing the $SE(2)$ distance between prediction and observation, defined in Open Motion Planning Library (OMPL) [22]:

$$\mathcal{D}(T_i, T_j) = \mathcal{D}_{\mathbb{R}^2}(p_i, p_j) + w_o \cdot \mathcal{D}_{SO(2)}(r_i, r_j) \tag{7}$$

where $p \in \mathbb{R}^2$ and $r \in SO(2)$ are the positional and rotational component of the $SE(2)$ state T. $\mathcal{D}_{\mathbb{R}^2}(\cdot,\cdot)$ computes standard Euclidean norm between positions, while $\mathcal{D}_{SO(2)}(\cdot,\cdot)$ measures the geodesic angular difference between two $SO(2)$ angles. The scalar w_o weights the rotational component. In this work, we set $w_o = 0.2$ to bias toward positional accuracy.

B. Uncertainty Quantification

Traditionally, neural network-based dynamics models produce only point estimates of action outcomes, lacking a measure of their prediction uncertainty. By explicitly quantifying the epistemic uncertainty in the learned model, ACTIVEPUSHER enables both informative data acquisition and uncertainty-aware robust planning, as illustrated in Fig. 3. During learning, this uncertainty guides active data collection by prioritizing the most informative samples in under-explored regions, thereby improving data efficiency (Sec. IV-C). At execution time, the planner leverages this uncertainty to select reliable actions from well-explored regions of the action space, resulting in more robust planning (Sec. IV-D).

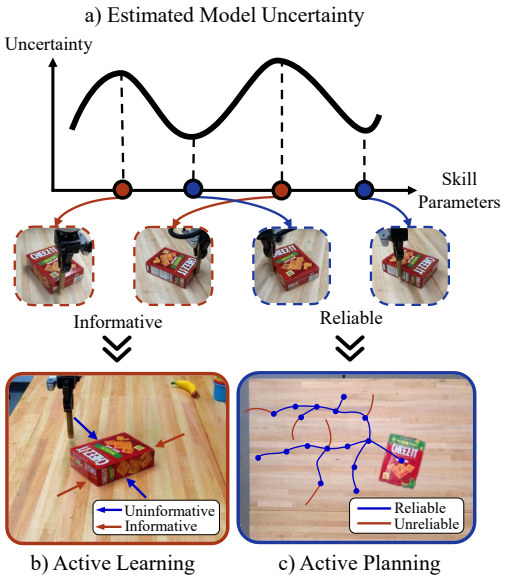


Fig. 3: a) ACTIVEPUSHER quantifies the model uncertainty of the learned model (Sec. IV-B). b) During the learning phase, ACTIVEPUSHER chooses the most *informative* push to apply to increase the learning efficiency (Sec. IV-C). c) During planning the most *reliable* pushes are chosen to maximize the task success rate (Sec. IV-D).

ACTIVEPUSHER estimates model uncertainty by leveraging the correspondence between neural networks and Gaussian Processes (GP) with the Neural Tangent Kernel (NTK) [5]. Under this view, the epistemic uncertainty of the learned dynamics model can be approximated through the predictive covariance of a GP with an NTK kernel.

Given a fully-connected neural network $f_\theta(u)$ with infinite width, parameterized by weights θ, its NTK stays constant during training and is defined as

$$k_{\text{NTK}}(u, u') = \langle \nabla_\theta f_\theta(u), \nabla_\theta f_\theta(u') \rangle \tag{8}$$

where $\nabla_\theta f_\theta(u)$ denotes the gradient of the network output with respect to the parameters θ, u and u' are two different inputs, and $\langle \cdot, \cdot \rangle$ is the inner product of two gradient vectors.

After training with gradient descent to convergence, the predictive distribution of an infinite-width neural network $f_\theta(u)$ is equivalent to that of Gaussian process regression with the NTK as its kernel [5]. Therefore, its predictive distribution, capturing model epistemic uncertainty, can be approximated by a GP with the NTK:

$$\mathcal{GP}(0, k_{\text{NTK}}(u, u')). \tag{9}$$

NTK encodes a notion of similarity between two inputs from the perspective of the neural network. The gradient $\nabla_\theta f_\theta(u)$ reflects how sensitive the network's prediction $f_\theta(u)$ is to infinitesimal perturbations in parameter θ. Intuitively, two inputs u and u' are considered similar if their gradients are aligned. In this case, model parameter updates that improve the prediction at u will also tend to improve the prediction at u'. This similarity is directly relevant to epistemic uncertainty, as the model is less uncertain in regions where multiple inputs share aligned gradients. This notion of similarity is precisely captured by the NTK, where gradient similarity is computed.

For finite-width networks, the kernel will change during the training process. But in practice, NTK after training convergence (empirical NTK) still provides accurate model uncertainty estimates [5], [23].

Assume we train neural network model $f(u)$ with training data $\mathcal{S}_{train}$ and their corresponding observed labels $\mathcal{L}_{train}$. Here, a set of data $\mathcal{S}$ is a collection of skill action u. As mentioned, we can model f as a GP with k_{NTK}. In addition, we consider a data pool $\mathcal{S}_{\text{pool}}$, which is the set of candidate actions whose labels have not yet been observed. Conditioned on the observed $\mathcal{S}_{train}$ and $\mathcal{L}_{train}$, the resulting posterior predictive covariance enables estimation of model uncertainty on unobserved data points within $\mathcal{S}_{\text{pool}}$. Formally, given a pre-defined inherent data noise σ_d, the posterior predictive covariance of the GP over an unlabeled data pool $\mathcal{S}_{\text{pool}}$ is:

$$\begin{aligned} \text{Cov}(\mathcal{S}_{\text{pool}}) &= K_{\text{NTK}}(\mathcal{S}_{\text{pool}}, \mathcal{S}_{\text{pool}}) \\ &\quad - K_{\text{NTK}}(\mathcal{S}_{\text{pool}}, \mathcal{S}_{\text{train}}) K_t^{-1} K_{\text{NTK}}(\mathcal{S}_{\text{train}}, \mathcal{S}_{\text{pool}}), \\ \text{where } K_t &= K_{\text{NTK}}(\mathcal{S}_{\text{train}}, \mathcal{S}_{\text{train}}) + \sigma_d^2 I. \end{aligned} \tag{10}$$

where $K_{\text{NTK}}(\cdot, \cdot)$ is the Gram matrix of pairwise NTK values between two vector inputs, capturing similarity of two sets of the data. We set $\sigma_d = 0.005$ in our experiments. Since σ_d is fixed, the posterior covariance of a GP (Eq. 10) primarily reflects epistemic uncertainty arising from limited training data. By isolating the diagonal terms of the posterior covariance matrix, we obtain per-sample model uncertainty estimates for $\mathcal{S}_{pool}$:

$$\text{Var}(\mathcal{S}_{\text{pool}}) = \text{Diag}(\text{Cov}(\mathcal{S}_{\text{pool}})) \tag{11}$$

C. Active Learning

Algorithm 1: `Active Learning`

Input: Kernel k, training round N, batch size B, initial training set $\mathcal{S}_{train}$, pool set $\mathcal{S}_{pool}$

for $i \leftarrow 1$ ***to*** N **do**
- $\mathcal{S}_{sel} \leftarrow \texttt{Acquire}(k, \mathcal{S}_{train}, \mathcal{S}_{pool}, B)$;
- Acquire labels of $\mathcal{S}_{sel}$;
- $\mathcal{S}_{train} \leftarrow \mathcal{S}_{train} \cup \mathcal{S}_{sel}$;
- $\mathcal{S}_{pool} \leftarrow \mathcal{S}_{pool} \setminus \mathcal{S}_{sel}$;
- Train model on $\mathcal{S}_{train}$ with corresponding labels and update kernel k;

Rather than passively training on a randomly collected dataset, active learning tries to iteratively query the most informative batch of samples to improve model performance with fewer labels [6], [9]. The general active learning process in a kernel setting is defined in Alg. 1. In each of the N training rounds, we perform uncertainty estimation over all unlabeled data in $\mathcal{S}_{\text{pool}}$ and select $\mathcal{S}_{\text{sel}}$ set with B informative samples. After querying their labels and moving them into the training set $\mathcal{S}_{\text{train}}$, we retrain the model with the expanded $\mathcal{S}_{\text{train}}$ and proceed to the next round.

Given predictive uncertainty estimation, one can apply different data acquisition strategies. In this work, we adopt the BAIT algorithm [6] with Fisher information to actively select the most informative action batch to collect.

Intuitively, the Fisher information encodes how sensitively the model's predictions respond to changes in its parameters. Covering the Fisher space ensures that the selected batch spans the directions along which the model can still learn the most. BAIT aims to select B samples whose combined per-sample Fisher embeddings best approximate (in Frobenius norm) the global Fisher information, yielding a representative batch that jointly captures both model uncertainty and diversity. Specifically, it seeks to minimize the trace of the inverse Fisher information matrix of the selected batch (i.e., the model uncertainty after selecting a batch), pre-multiplied by the Fisher information of the entire unlabeled pool:

$$\mathcal{S}_{\text{sel}} = \underset{\mathcal{S}_{\text{sel}} \subseteq \mathcal{S}_{\text{pool}}}{\arg\min} \operatorname{tr}\left(\left(\sum_{u \in \mathcal{S}_{\text{sel}}} I(u;\theta)\right)^{-1}\left(\sum_{u \in \mathcal{S}_{\text{pool}}} I(u;\theta)\right)\right) \tag{12}$$

where $I(u;\theta)$ is the Fisher information matrix associated with model parameter θ at input u. In Gaussian regression, the Fisher information matrix of a set $\mathcal{S}$ reduces to the outer product of the output gradients:

$$\sum_{u \in \mathcal{S}} I(u;\theta) = \nabla_\theta f_\theta(\mathcal{S})^\top \nabla_\theta f_\theta(\mathcal{S}) \tag{13}$$

Let $k[\mathcal{S}]$ denote the kernel k conditioned on $\mathcal{S}$, as shown in [10], one can prove that, with NTK kernel:

$$\sum_{u \in \mathcal{S}_{pool}} k[\mathcal{S}_{sel}](u,u) = c \operatorname{tr}\left(G_{\mathcal{S}_{sel}}^{-1} G_{\mathcal{S}_{pool}}\right), \quad \text{where } G_{\mathcal{S}} := \nabla_\theta f_\theta(\mathcal{S})^\top \nabla_\theta f_\theta(\mathcal{S}) \tag{14}$$

Combined with Eq. 13, we show that optimizing Eq. 14 is equivalent to optimizing the Fisher objective Eq. 12 in BAIT.

Different from the original BAIT, we made the following changes. First, considering that our neural network is relatively small, we use the full gradient NTK instead of the last-layer gradient as in the original design. Second, rather than computing the Fisher information of merely the selected set $\mathcal{S}_{sel}$ and pool set $\mathcal{S}_{pool}$, we expand Eq. 14 by also considering the current training set $\mathcal{S}_{train}$. These changes better exploit the full representational capacity of the network and ensure that acquisition decisions account for both the unlabeled pool and the knowledge already contained in the training set. In summary, our acquisition strategy is as follows:

$$\texttt{Acquire}(k, \mathcal{S}_{train}, \mathcal{S}_{pool}, B) = \underset{\mathcal{S}_{sel} \subseteq \mathcal{S}_{pool}}{\arg\min} \sum_{u \in \mathcal{S}_{train} \cup \mathcal{S}_{pool}} k_{\text{NTK}}[\mathcal{S}_{train} \cup \mathcal{S}_{sel}](u,u) \tag{15}$$

Optimizing such a Fisher objective is intractable given the many potential different combinations for $\mathcal{S}_{sel}$. To address this, the same greedy forward–backward selection algorithms, proposed in [6], are adopted to greedily select $\mathcal{S}_{sel}$.

D. Active Planning

We formulate nonprehensile pushing as a kinodynamic planning problem in the object's $SE(2)$ state space. In this formulation, each parameterized push action u becomes a discrete control that drives the object's state through forward simulation of the learned dynamics. We use an asymptotically near optimal kinodynamic planner, specifically SST [24], to explore the object's state space directly. SST incrementally expands a tree of dynamically feasible trajectories by forward-propagating sampled controls and pruning redundant nodes to maintain sparsity. Optionally, given a optimization objective, it progressively improves trajectory cost while steering the search toward the goal region.

In the absence of model error, control sequences found by SST succeed by design. In practice, however, accumulated prediction errors in the learned dynamics can lead to execution failures. To improve robustness, we integrate epistemic uncertainty estimates into the control sampling step.

Algorithm 2: `Active Sampling`

Input: Kernel k, training set $\mathcal{S}_{\text{train}}$, batch size b

$\mathcal{S}_U \leftarrow$ `random_sampling`(b);
Var $\leftarrow$ `query_uncertainty`$(k, \mathcal{S}_{\text{train}}, \mathcal{S}_U)$ (*Eq. 11*);
$u \leftarrow \operatorname{argmin}_{u \in \mathcal{S}_U} \text{Var}[u]$;
return u

Specifically, as summarized in Alg. 2, at each planning step, a batch of candidate pushing controls of size b is first sampled. We then evaluate the epistemic uncertainty of the learned model for these controls (Eq. 11). Instead of selecting randomly, we choose the control with the lowest predicted uncertainty. This strategy biases control sampling in planning toward more reliable actions, which is based on the intuition that the model is more accurate in well-explored regions of the skill space.

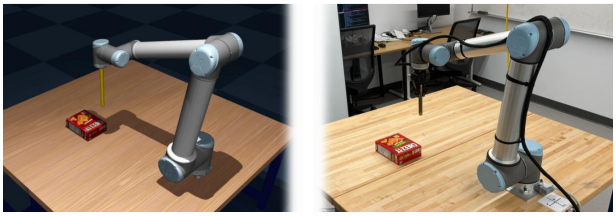
Because the biased active sampler still samples from all the control space (i.e., every action has non-zero probability of being sampled), the planner retains the probabilistic completeness and asymptotic optimality guarantees of SST. In effect, uncertainty estimates guide the planner to prioritize reliable regions of the skill space while maintaining theoretical coverage of the entire domain.

V. Experiments

In this section, we provide details of our models, training procedure and the experiments conducted in both simulation and real world. Our experiment setup includes a 6-DOF UR10 robot and multiple objects from the YCB dataset [25], with different geometric shapes and physical characteristics, to test the robustness of our approach across a range of physical characteristics. All the learning and planning experiments are run on a workstation with an NVIDIA RTX 4070 Ti Super GPU and 32GB of RAM.

In real world, object pose is estimated with FoundationPose [26] using an Intel RealSense Depth Camera D435 mounted on the end effector. Additionally, for real-world data collection, we implement a simple reset algorithm that automatically pushes the object back toward the center of the robot's high-manipulability workspace whenever it drifts outside that region. This mechanism enables autonomous, continuous learning without any human intervention. To

execute the push parameters with the robot, we use a global redundancy resolution method [27] as inverse kinematics (IK) solutions, which guarantees valid and smooth joint trajectories within a certain workspace region.



(a) MuJoCo Simulation (b) Real World

Fig. 4: Experiment Setup

We perform our simulation experiments with the MuJoCo Simulator [15], which allows parallel execution of multiple environments, enabling efficient large-scale testing. The simulated scene in the simulation environment replicates our real-world setup, as shown in Fig. 4.

We evaluate the proposed method in two settings: (i) Skill Learning, where we measure active learning performance, and (ii) long-horizon Kinodynamic Planning, where we assess the effectiveness of both active learning and planning.

A. Skill Learning

As discussed in Sec. IV-C, active learning can run continuously to collect training data and update models. However, for the purpose of repetitive evaluation and fair comparison across methods, we adopt a standard pool-based evaluation setup [6], [9]. In simulation, we first construct a candidate pool of 9,000 push actions for each object, while in the real world, each object has a pool of 1,000 actions. During training, at each acquisition stage, a batch of 10 samples $\mathcal{S}_{sel}$ is selected from the pool to expand the training set and update the model. This process is repeated for 10 stages until 100 data are collected. We collect independent test sets of 1,000 samples for each object in either simulation or real world.

Four objects in simulation (Banana, Mug, Cracker Box and Mustard Bottle) and two objects in the real world (Cracker Box and Mustard Bottle) are used, and we evaluate the following five methods:

- *Pure Physics:* The analytical dynamics model adapted from [21].
- *MLP Random:* A fully connected multi-layer perceptron (MLP) consisting of five hidden layers with sizes [32, 64, 64, 32, 32] trained with random push actions.
- *Residual Random:* The hybrid model as described in Sec. IV-A. The physics is the same as *Pure Physics*. Meanwhile, the neural network architecture and data collection method are the same as *MLP Random*.
- *MLP BAIT:* The same MLP architecture as *MLP Random*, but trained via our NTK-driven active learning pipeline to collect informative data.
- *Residual BAIT:* The hybrid residual model as in *Residual Random*, but trained with active learning pipeline.

All models are trained using a learning batch size of 20 for 1000 epochs until convergence. Each training is repeated

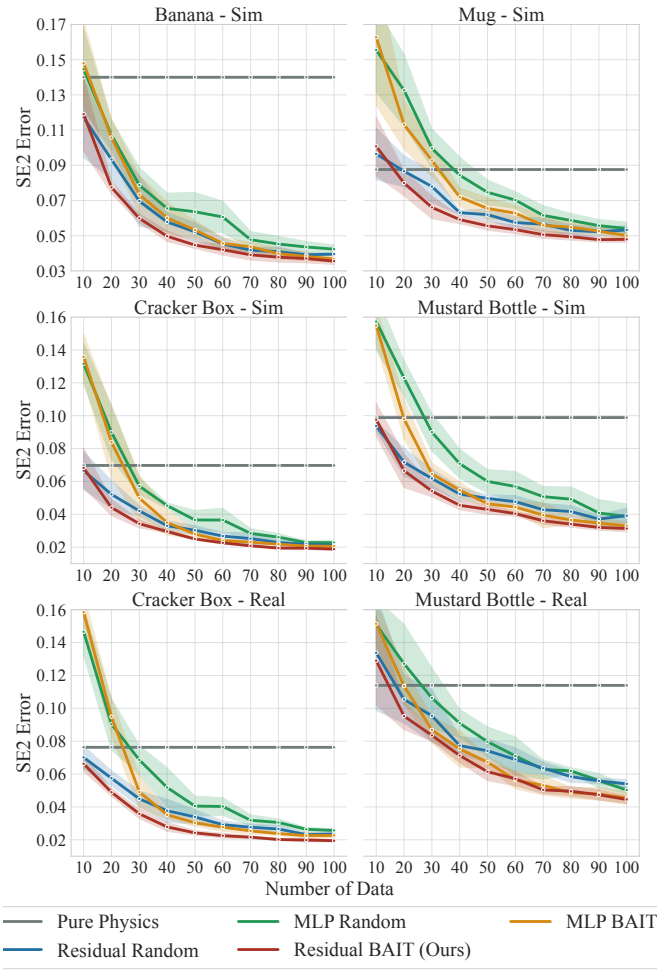
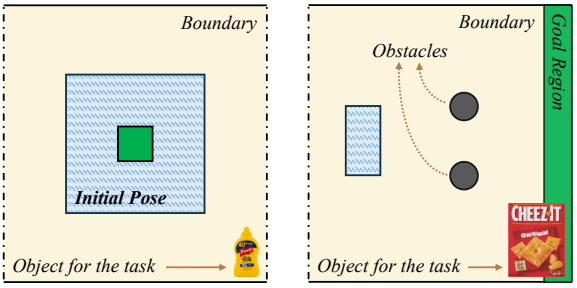


Fig. 5: Skill Learning results show SE2 prediction error for 2 real objects, and 4 simulated objects from [25]: Banana, Mug, Cracker Box and Mustard Bottle. The active learning methods outperform random data collection, and models with residual physics perform better in low-data regime.

5 times and the summarized results for prediction $SE(2)$ error (Eq. 7) on the test set are shown in Fig. 5. The models with residual physics show a clear improvement when data is limited. Additionally, the active learning approach consistently outperforms random sampling across all objects, either by reaching the same level of accuracy with fewer training samples or by achieving higher accuracy given the same amount of data. By jointly leveraging active learning and analytical physics, our method *Residual BAIT* can achieve comparable accuracy to the final performance of baseline *MLP Random* while requiring only 55% of the training data on average. In practice, active learning can be run continuously, with data collection terminated once validation performance converges, or according to a task-specific criterion [28].

B. Kinodynamic Planning

In this experiment, we demonstrate the performance of learned models integrated with a kinodynamic planner for two downstream tasks. The planning was conducted in the object's state space, defined as $\mathcal{X} = SE(2)$, with a control space $\mathcal{U} \subset \mathbb{R}^3$ corresponding to the pushing parameters described in Sec. IV. The valid state space $\mathcal{X}_{\text{free}}$ is constrained to the table surface and are free of collisions with obstacles.



(a) Push to Region (b) Push to Edge

Fig. 6: 2D schematic of the planning tasks

The two task environments are shown in Fig. 6. In the simpler "Push to Region" task, a mustard bottle is initialized randomly and needs to be pushed to a $0.1\text{m} \times 0.1\text{m}$ region at the table center. The more challenging "Push to Edge" requires pushing a non-graspable cracker box toward the table edge to enable a feasible pick-up, with two cylinder obstacles (radius 0.04m) present. The goal is defined geometrically: the object's center of mass remains on the table while at least one corner extends beyond 0.03m from the boundary.

We use SST [24] in OMPL [22] with a path-length optimization objective to plan for both tasks. In simulation, we solve 100 randomly generated problems for each task and repeat the planning 5 times. For the real world experiment, we solve 20 randomly generated problems once. For all the problems, we compare the default uniform sampling method (Regular Planning) with our proposed active sampling method (Active Planning). To also assess the impact of model prediction accuracy on planning outcomes, we evaluate baseline model *MLP Random* and proposed model *Residual BAIT*, described in the previous section Sec. V-A, with different data sizes.

All generated plans are executed in an open-loop manner. Performance is evaluated by task success rate and path-tracking error. A task is considered successful if the goal is reached while maintaining validity, and the path-tracking error is defined as the average $SE(2)$ deviation along the trajectory. The results are summarized in Fig. 7.

The results of the experiment show that the planning performance is closely tied to the model prediction accuracy as expected. More accurate models generally produce plans with higher task success rates and lower execution errors. In addition, incorporating active action sampling further improves performance. By steering the planner toward actions in which the model is more confident, active planning tends to select actions with potentially lower execution error. As a result, active planning consistently outperforms regular planning in both success rate and tracking accuracy.

We note that active planning produces slightly longer paths, as the planner favors sampling actions in lower-uncertainty regions. In our experiments, this results in an increase of approximately 9–13% in average path length, while improving success rate and execution accuracy. The additional computational overhead is negligible in practice. Estimating the model uncertainty for 10,000 candidate actions takes only 8.1 millisecond on average in a batched implementation with neural-network gradient features.

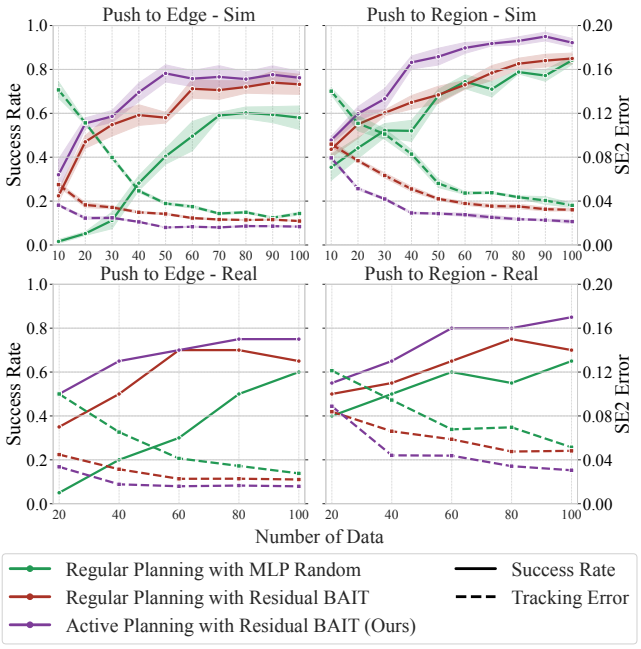


Fig. 7: Kinodynamic Planning results show that more accurate dynamics models lead to higher task success rates. Additionally, active planning tends to select actions with lower execution error and further improves task success.

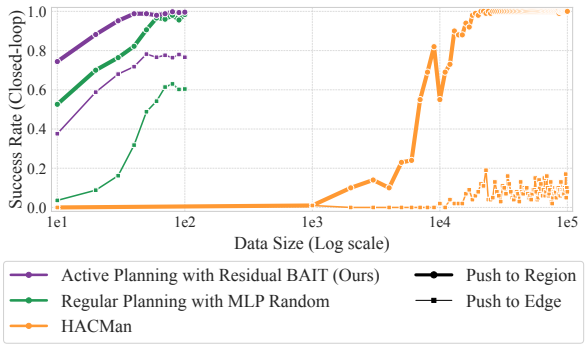


Fig. 8: Closed-loop Execution results show our method outperforms HACMan, especially in Push to Edge task, while requiring far less data and transferring effectively to complex environments with novel obstacles.

C. Closed-loop Execution

All previous experiments are conducted in an open-loop setting. In practice, replanning given current state can improve robustness. To examine this, we repeat the same tasks in a closed-loop manner. We define a *recoverable failure* as an execution that does not reach the goal but remains in the valid state space. The system can replan within 1 second and continue execution. In contrast, *irrecoverable failures* occur when the object collides with obstacles, falls off the table, or exceeds the planning horizon of 10 control steps. We additionally compare against HACMan [16], a state-of-the-art RL method that executes closed-loop, with discrete actions learned entirely from interactions. Results are in Fig. 8.

On the *Push to Region* task, ACTIVEPUSHER can reach 100% success rate with replanning. HACMan can also reach 100% success but requires orders of magnitude more data. On the more challenging *Push to Edge* task, HACMan struggles with distribution shift: obstacles are randomized during training but differ in placement during evaluation.

In contrast, our method requires far less data and transfers effectively to novel obstacles and goal constraints.

Additionally, sim-to-real transfer remains a challenge for RL-based approaches. As reported in [16], HACMan achieves only about 70% success in simple planar push tasks without obstacles. In contrast, our method can be trained directly and efficiently with limited real-world data, achieving robust performance without additional adaptation even with obstacles.

Also, closed-loop replanning alone does not eliminate all failures. In the *Push to Edge* task, an inappropriate initial plan can drive the object into irrecoverable states. This emphasizes the importance of generating robust initial plans.

VI. Conclusion and Future Work

In this paper, we present ActivePusher, a framework that integrates residual physics, active learning, and active kinodynamic planning. By modeling epistemic uncertainty using NTK, our method actively gathers informative data with BAIT strategy, and biases planning toward reliable actions. Our experiments demonstrate that ActivePusher achieves higher prediction accuracy and planning success with fewer interactions compared to baselines. This integration of learning and planning offers a promising path toward data-efficient and reliable nonprehensile manipulation.

Our current formulation relies on several simplifying assumptions to enable analytical modeling and a relatively small neural network. The major simplification is to approximate rigid objects as oriented bounding boxes, and assume sticking contact during pushing. These assumptions allow efficient learning and uncertainty estimation but limit the applicability of the framework to planar quasi-static nonprehensile manipulation. Importantly, the core contribution of this work is uncertainty estimation for both active learning and planning, which is largely independent of the specific problem modeling, and can be integrated with more sophisticated modeling approaches. Future work will broaden the scope beyond simple planar pushing toward richer contact dynamics, diverse geometries, and $SE(3)$ nonprehensile skills. Another direction is to jointly model epistemic and aleatoric uncertainty to better capture action-dependent noise.

References

[1] A. Orthey, C. Chamzas, and L. E. Kavraki, "Sampling-based motion planning: A comparative review," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, no. 1, pp. 285–310, July 2024.

[2] M. T. Mason, "Toward robotic manipulation," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. Volume 1, 2018, pp. 1–28, 2018.

[3] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, 2020.

[4] A. Ajay, J. Wu, N. Fazeli, M. Bauza, L. P. Kaelbling, J. B. Tenenbaum, and A. Rodriguez, "Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3066–3073.

[5] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[6] J. Ash, S. Goel, A. Krishnamurthy, and S. Kakade, "Gone fishing: Neural active learning with fisher embeddings," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8927–8939, 2021.

[7] F. R. Hogan and A. Rodriguez, "Reactive planar non-prehensile manipulation with hybrid model predictive control," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 755–773, 2020.

[8] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.

[9] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70. PMLR, 2017.

[10] D. Holzmüller, V. Zaverkin, J. Kästner, and I. Steinwart, "A framework and benchmark for deep batch active learning for regression," *J. Mach. Learn. Res.*, vol. 24, no. 1, Jan. 2023.

[11] A. T. Taylor, T. A. Berrueta, and T. D. Murphey, "Active learning in robotics: A review of control principles," *Mechatronics*, vol. 77, p. 102576, 2021.

[12] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Learning compositional models of robot skills for task and motion planning," *The International Journal of Robotics Research*, vol. 40, no. 6-7, pp. 866–894, 2021.

[13] A. LaGrassa, M. Lee, and O. Kroemer, "Task-oriented active learning of model preconditions for inaccurate dynamics models," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 16 445–16 445.

[14] R. Reed, L. Laurenti, and M. Lahijanian, "Promises of deep kernel learning for control synthesis," *IEEE Control Systems Letters*, 2023.

[15] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, *et al.*, "Mujoco playground," *arXiv preprint arXiv:2502.08844*, 2025.

[16] W. Zhou, B. Jiang, F. Yang, C. Paxton, and D. Held, "Hacman: Learning hybrid actor-critic maps for 6d non-prehensile manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 241–265.

[17] M. Bauza, F. R. Hogan, and A. Rodriguez, "A data-efficient approach to precise and controlled pushing," in *Conference on Robot Learning*. PMLR, 2018, pp. 336–345.

[18] G. Wang, K. Ren, and K. Hang, "Uno push: Unified nonprehensile object pushing via non-parametric estimation and model predictive control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 9893–9900.

[19] K. Ren, G. Wang, A. S. Morgan, L. E. Kavraki, and K. Hang, "Object-centric kinodynamic planning for nonprehensile robot rearrangement manipulation," *IEEE Transactions on Robotics*, 2025.

[20] K. Ren, P. Chanrungmaneekul, L. E. Kavraki, and K. Hang, "Kinodynamic rapidly-exploring random forest for rearrangement-based nonprehensile manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8127–8133.

[21] K. Lynch, H. Maekawa, and K. Tanie, "Manipulation and active sensing by pushing using tactile feedback," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 1992, pp. 416–421.

[22] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, .

[23] M. A. Mohamadi, W. Bae, and D. J. Sutherland, "Making look-ahead active learning strategies feasible with neural tangent kernels," *Advances in Neural Information Processing Systems*, vol. 35, pp. 12 542–12 553, 2022.

[24] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[25] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[26] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "Foundationpose: Unified 6d pose estimation and tracking of novel objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 868–17 879.

[27] Z. Zhong, Z. Li, and C. Chamzas, "Expansion-grr: Efficient generation of smooth global redundancy resolution roadmaps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 8854–8860.

[28] N. Kumar, T. Silver, W. McClinton, L. Zhao, S. Proulx, T. Lozano-Pérez, L. P. Kaelbling, and J. Barry, "Practice makes perfect: Planning to learn skill parameter policies," *Planning*, vol. 1, p. 2.