2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

UNO Push: Unified Nonprehensile Object Pushing via Non-Parametric Estimation and Model Predictive Control

Gaotian Wang, Kejia Ren, and Kaiyu Hang

***Abstract*— Nonprehensile manipulation through precise pushing is an essential skill that has been commonly challenged by perception and physical uncertainties, such as those associated with contacts, object geometries, and physical properties. For this, we propose a unified framework that jointly addresses system modeling, action generation, and control. While most existing approaches either heavily rely on *a priori* system information for analytic modeling, or leverage a large dataset to learn dynamic models, our framework approximates a system transition function via non-parametric learning only using a small number of exploratory actions (*ca.* 10). The approximated function is then integrated with model predictive control to provide precise pushing manipulation. Furthermore, we show that the approximated system transition functions can be robustly transferred across novel objects while being online updated to continuously improve the manipulation accuracy. Through extensive experiments on a real robot platform with a set of novel objects and comparing against a state-of-the-art baseline, we show that the proposed unified framework is a light-weight and highly effective approach to enable precise pushing manipulation all by itself. Our evaluation results illustrate that the system can robustly ensure millimeter-level precision and can straightforwardly work on any novel object.**

I. Introduction

Nonprehensile manipulation actions such as pushing, sliding, and toppling [1]–[3], can provide a rich set of physical possibilities for robots to interact with objects. More commonly than other motion primitives, pushing has been widely employed as a key component in manipulation systems to handle tasks where grasping is unnecessary or infeasible. In general, pushing-based manipulation is formulated either as a large-scale multi-objects rearrangement problem [4]–[7], or as a problem concerned with the precise motion control between a robot and a pushed object [8]. While both formulations are challenged by the intricate dynamics and various uncertainties in perception and physics, this work focuses on the precise control of pushing manipulation, as exemplified in Fig. 1 with the goal of minimizing the requirements on sensing and prior knowledge while optimizing the manipulation precision.

Analytic methods for precise pushing are traditionally quite heavy in terms of the involved system components, including contact analysis, object shape representations, modeling of system transitions, physical uncertainties, action generation, and control [8]–[13]. Nevertheless, such complex compositions often make the system integration prohibitively complex and render the solutions not generalizable nor scalable to different task setups, as limited by many modeling simplification assumptions. Alternatively, data-driven approaches [14]–[20] have shown unprecedented capabilities in handling complex manipulation tasks. Although data can enable such methods, it is also a limiting factor when the task setup changes or the perception design varies across systems, which would normally require the entire system to be trained again with a large amount of new data.

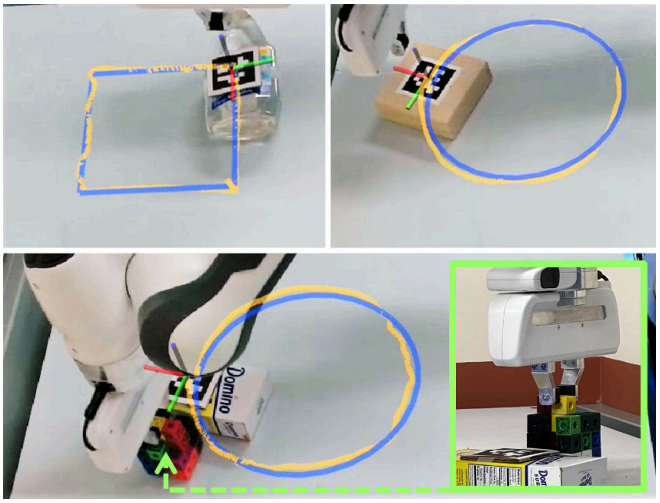

Fig. 1: A robot manipulator is tasked to manipulate an unknown object to trace reference paths (blue). Without analytically modeling the contacts or other physical components of the system, our UNO Push framework enables precise manipulation (yellow paths) by pushing the object with the gripper of the robot (top), or by an unknown object grasped by the gripper (bottom).

To address the aforementioned challenges, this work builds our system upon two insights. First, an approximated inaccurate system model can be used to close the control loop and ensure high precision. Second, this approximated model, which presents certain discrepancies against the real physical system, can still offer good performance when adaptively updated online to match the observed physical outcomes.

To this end, this work proposes a *unified* framework that addresses system modeling, action generation, and control of precise pushing all through non-parametric estimation, named as UNO Push for *Unified Nonprehensile Object Pushing*. The framework first builds the system transition model via *light-weight* non-parametric estimation to directly map from the robot actions to the object motions. Without requiring any *a priori* knowledge about the object or robot contact geometries or physics, nor any large dataset or object-specific offline training, our method can effectively approximate a transition model using a few exploratory actions (*ca.* 10).

The authors are with the Department of Computer Science, Rice University, Houston, TX 77005, USA. {gwang, kr43, kaiyu.hang}@rice.edu. This work is supported by NSF grant FRR-2133110 and Rice University Funds.

Then, via closed-loop Model Predictive Control built upon the approximated model, our framework generates real-time actions by observing the system state. Real-time feedback is used to adaptively update the approximated model online to continuously improve the control performance.

Through extensive experiments, we show that the *unified light-weight* UNO Push can directly work on novel objects using highly approximated models, which can be easily transferred and adapted to precisely manipulate other objects. By comparing against a state-of-the-art baseline approach, we show that our light-weight framework can ensure high manipulation precision without sophisticated modeling or object-specific pre-training. Through experiments of pushing by the robot gripper and by a grasped unmodeled object (Fig. 1), we illustrate the possibility of deploying our method on different robots without requiring any remodeling. We further show that even under unknown external perturbations, such as pushing through a cluttered area (Fig. 12), our UNO Push is able to effectively handle the uncertainties in the environment to ensure precise pushing.

As will be discussed in detail, the unified framework of data efficient model approximation, online model update, and closed-loop control, provides a very low barrier for UNO Push to be flexibly employed in real-world tasks without much pre-requisites. This is especially useful when the contact geometries and physics are unknown or when the task conditions do not allow for pre-training on the target object, making analytical modeling or offline training infeasible. The key contributions of our work are:

- A unified framework that addresses system modeling, action generation, and control of precise pushing all through non-parametric estimation;
- System motion models built through a small number of exploratory actions;
- Precise pushing manipulation with imprecisely approximated system models, which are continuously updated online using in-task experiences.

II. Related Work

Analytic Models: Precise control of planar pushing can be achieved through analytically modeling the control laws. Assuming necessary *a priori* knowledge are available, e.g., friction coefficient, object mass distribution, and object geometry, differentiable system transition models can be derived under some common simplification assumptions, such as point contact and quasi-static physics [8]. Thereafter, action generation can be enabled through various optimization formulations [12], and control loops are normally closed by nonlinear control schemes, such as model predictive control [13]. However, as existing analytic approaches aim at building models that are unnecessarily accurate, they are commonly not generalizable or scalable, especially against unknown and uncertain task setups.

Simulation-based Planning: For large-scale rearrangement manipulation problems, pushing actions are planned to sequentially reconfigure the system states [6]. In such problems, algorithms such as kinodynamic motion planning and trajectory optimization are more concerned with the discrete transitions between states, while the precise motions along the transitions are often ignored. Although also challenged by physical uncertainties [7], simulation-based planning methods are fundamentally different from precise control methods, such as the one addressed in this work, as they do not need to build the system models.

Data-Driven Approaches: Meta-learning of dynamic models [15], composite analytic and learned models [16], probabilistic approaches [17], [20], stochastic neural networks [18], and large public datasets [19], together with many other data-driven approaches, have shown unprecedented capabilities in handling complex tasks. Being the state-of-the-art work in precise pushing control, the study in [17], [20] have demonstrated that effective control performance can be achieved when system motion models are pre-trained on objects of uniform mass distributions, with additional assumptions that the shape of the object is known or well approximated. However, similar to data-driven approaches in other problems, data is a major limitation for model generalization, and a small change in task setup or perception design can often require a new dataset to be collected for transferring the model. In contrast, this work shows that an inaccurate model can be approximated with a small amount of data and a light-weight model, and can be integrated into a unified framework, while being updated online, to enable precise pushing control of objects of unknown geometries and physical properties.

III. Problem Formulation

In this work, we are interested in the problem of pushing-based nonprehensile manipulation, where a robot manipulator is tasked to continuously push a single object to trace some desired paths. We assume the motion of the object to be planar sliding without rolling or flipping, in a quasi-static manner. As such, the controlled object's motion can be modeled by a discrete-time dynamical system. We denote the configuration of the object at time t by $X_t \in SE(2)$, and define $\mathcal{U}$ to be the set consisting of all the allowed controls that the robot can execute to push the object. The object's configuration evolves according to the following deterministic system dynamics:

$$X_{t+1} = f(X_t, u_t) \tag{1}$$

where $f : SE(2) \times \mathcal{U} \mapsto SE(2)$ is the system transition function and $u_t \in \mathcal{U}$ is the control executed by the robot.

A. Manipulation Model Representation

Object's Configuration: We use a homogeneous representation for the object's configuration. That is, $X_t \in SE(2)$ is represented by a 3×3 transformation matrix. Given the object's configurations at adjacent time steps, X_t and X_{t+1}, the rigid body motion of the object at time t is defined by ${}^b g_t = X_t^{-1} \cdot X_{t+1} \in SE(2)$, where X_t^{-1} is the inverse of X_t. In this definition, the object's rigid body motion ${}^b g_t$ is always specified in the object's body frame. (*Note:* Throughout the

paper, a left superscript b in the notation indicates that the variable is defined in the object's body frame; otherwise, it is defined in the spatial frame.)

Furthermore, we define a distance function in the object's configuration space by $\Delta : SE(2) \times SE(2) \mapsto \mathbb{R}$. We calculate the distance between two arbitrary configurations of the object by the weighted summation of the difference in their orientations and the Euclidean distance between their positions.

Control: Typically, most control models for object pushing require precise locations of the robot-object contacts to be available. However, this is often impractical in the real-world tasks when handling objects of unknown shapes. To enable effective pushing actions for objects of arbitrary shapes, we represent the control by two angles defined in the object's body frame: ${}^b u_t = (\alpha_t, \beta_t)$. As illustrated in Fig. 2, we virtually attach a circle to the object's body frame, whose radius R is set larger than the size of the object. To execute a control, the robot first moves its end-effector to a point P on the circle, determined by the angle α_t. Then, the robot continuously moves the end-effector towards the object in the direction determined by an offset angle β_t until it has traveled a constant distance d after detecting the object's motion. Throughout the execution, the gripper orientation is maintained parallel to its motion direction. As such, the controls are specified relative to the object's body frame.

System Transition Models: Since the control is defined in the object's body frame, for the same object, its rigid body motion is independent of its configuration X_t. Therefore, we can represent the motion of the object by a transition function $\Gamma : \mathcal{U} \mapsto SE(2)$ invariant to the object's configuration, which maps a control to the object's rigid body motion. The system transition model in Eq. (1) can be re-written as:

$$X_{t+1} = X_t \cdot {}^b g_t = X_t \cdot \Gamma({}^b u_t) \tag{2}$$

In addition, we need an inverse model of the system transitions, $\Gamma^{-1} : SE(2) \mapsto \mathcal{U}$, which infers the control to be executed given a desired rigid body motion of the object.

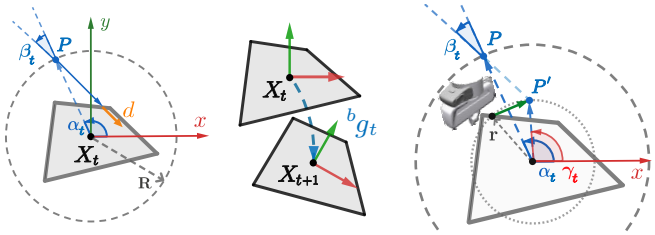

Fig. 2: *Left*: The representation of the control, through two angles α_t and β_t in the object's body frame; *Middle*: The rigid body motion of the object ${}^b g_t$; *Right*: The smoothened execution strategy of control. Instead of retreating the gripper back to the point P, the robot moves the gripper to a point P' closer to the object.

B. Precise Pushing Problem

Starting with the initial configuration of the object $X_0 \in SE(2)$, the robot is required to find and execute a sequence of controls, as defined in Sec. III-A, to push the object to M desired configurations sequentially. These configurations compose a reference path, represented by a sequence $\mathcal{Y} = \{Y_1, \cdots, Y_M\}$ where $Y_1, \cdots, Y_M \in SE(2)$.

Algorithm 1 Precise Pushing via UNO Push

Input: Object's initial configuration X_0, reference path $\mathcal{Y}$, a boolean argument s indicating whether to learn models from scratch, a distance threshold δ

1: $\Gamma, \Gamma^{-1} \leftarrow$ LEARNMODELS(X_0, s) ▷ Alg. 2
2: $t \leftarrow 0$
3: **while** $\Delta(X_t, Y_M) > \delta$ **do** ▷ Last Waypoint Y_M Not Reached
4: ${}^b u_t \leftarrow$ MPC$(X_t, \mathcal{Y})$ ▷ Alg. 4
5: $X_{t+1} \leftarrow$ SMOOTHENEDEXECUTE$(X_t, {}^b u_t)$ ▷ Alg. 5
6: $\Gamma, \Gamma^{-1} \leftarrow$ UPDATEMODELS$({}^b u_t, X_t, X_{t+1})$ ▷ Alg. 3
7: $t \leftarrow t + 1$
8: **end while**

In general, generating actions to accomplish the aforementioned manipulation task requires an accurate model of the system dynamics. However, analytical system models Γ and Γ^{-1} are not feasible as they require accurate object geometries and physical parameters such as the friction coefficient, which are not available without ideal and sophisticated sensing capability. Moreover, analytical Γ and Γ^{-1} are difficult to generalize on different objects, limiting their applications in the real world. To this end, we propose to approximate the system models by using manipulation experiences observed online, and integrate the approximated models into a Model Predictive Control (MPC) framework for generating effective actions. In this way, we unify the model approximation, action generation, and control into a light-weight yet efficient framework. The proposed framework, UNO Push, is presented in Alg. 1.

IV. Non-parametric Model Estimation

To approximate the system models Γ and Γ^{-1} defined in Sec. III-A without requiring a large amount of data or prior information about the system, we propose to represent Γ and Γ^{-1} by non-parametric models and estimate them through Gaussian Process Regression (GPR). As we try to keep our framework light-weight and limit the amount of data to be very small, the models Γ and Γ^{-1} can be estimated and updated online while the robot is manipulating the object. The predictions made by the approximated models will then be used to generate real-time actions.

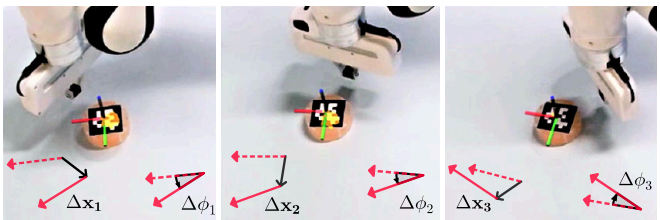

Fig. 3: Three example data points collected by pushing a cylinder through random controls. The red dashed and solid arrows represent the x-axis of the object's body frame before and after the push, respectively. The object's configuration has been changed through translation $\Delta \mathbf{x}$ and rotation $\Delta\phi$. In our experiments (Sec. VII), we applied the model learned on the cylinder object to directly manipulate other objects.

Algorithm 2 LearnModels($\cdot$)

Input: Object's initial configuration X_0, a boolean argument s indicating whether to learn models from scratch
Output: Learned models Γ and Γ^{-1}

if $s ==$ True **then** ▷ Learn Models from Scratch
  $\mathcal{D} \leftarrow \{\}$ ▷ Training Dataset
  for $i = 0, \cdots, N-1$ **do**
    $\alpha \leftarrow$ UNIFORM$(0, 2\pi)$ ▷ Uniform Sampling
    $\beta \leftarrow$ UNIFORM$(-0.2, 0.2)$
    ${}^b\hat{u}_i \leftarrow (\alpha, \beta)$ ▷ Random Control
    $X_{i+1} \leftarrow$ EXECUTE$({}^b\hat{u}_i)$ ▷ Observe Object's Configuration
    ${}^b\hat{g}_i \leftarrow X_i^{-1} \cdot X_{i+1}$ ▷ Object's Rigid Body Motion
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{({}^b\hat{u}_i, {}^b\hat{g}_i)\}$
  end for
  $\Gamma, \Gamma^{-1} \leftarrow$ GPR$(\mathcal{D})$ ▷ Gaussian Process Regression
else
  $\Gamma, \Gamma^{-1} \leftarrow$ COPYFROMOLD() ▷ Copy From Previous Tasks
end if
return Γ, Γ^{-1}

A. Model Learning

To learn the approximated models of Γ and Γ^{-1}, we build a training dataset $\mathcal{D} = \{({}^b\hat{u}_i, {}^b\hat{g}_i)\}_{i=0}^{N-1}$ of N data points through real-world manipulation. Each data point in $\mathcal{D}$ is a pair of control ${}^b\hat{u}_i \in \mathcal{U}$ the robot has executed and the observed rigid body motion of the object ${}^b\hat{g}_i \in SE(2)$. As detailed in Alg. 2 and illustrated in Fig. 3, we uniformly sample N controls within their allowed ranges and execute each sampled control on the robot to manipulate the object by pushing. In practice, the two angles of the control are sampled from $\alpha \in [0, 2\pi]$ radians and $\beta \in [-0.2, 0.2]$ radians. The range of β was made narrow to increase the probability of the robot making contact with the object. Meanwhile, the object's motion ${}^b\hat{g}_i$ resulting from the execution of ${}^b\hat{u}_i$ is observed by sensors. In the end, we add all these sampled controls and their corresponding observations into the training dataset $\mathcal{D}$ and regress the initial models of Γ and Γ^{-1} via GPR, to learn the underlying relationship between the controls and the object's motions. Both models Γ and Γ^{-1} are regressed with the same dataset $\mathcal{D}$, by swapping the domain and codomain of the data. Although trained with the same dataset, the estimated Γ and Γ^{-1} are not constrained to form a closed loop, that is, $\Gamma^{-1}(\Gamma(u)) \neq u$.

Based on the intuition that motion models of different objects have similarities in their patterns, our framework has the option to transfer the models learned from previous manipulation tasks as the manipulation models for a novel object. In such cases, when manipulating a novel object, our framework can skip the real-world data collection step (lines 2-11 of Alg. 2) and directly use the models transferred from previous tasks of manipulating a different object, to speed up the current manipulation task. Even though the models transferred from manipulating a different object are not accurate enough at the beginning, they can be updated online to improve the manipulation performance.

B. Online Model Update

When the robot uses the learned models of Γ and Γ^{-1} in Sec. IV-A to push the object, it at the same time keeps exploring the system transition models. Hence, we propose to adaptively update the model online using the newly executed actions as detailed in Alg. 3. Specifically, whenever a control ${}^bu_t$ is executed to manipulate the object, ${}^bu_t$ associated with the observed object's motion ${}^bg_t$ will be added to the training dataset $\mathcal{D}$ to update the models Γ and Γ^{-1}.

Algorithm 3 UpdateModels($\cdot$)

Input: Last executed control ${}^bu_t$, object's configuration before last execution X_t, object's configuration after last execution X_{t+1}
Output: Updated Models Γ and Γ^{-1}

$\mathcal{D} \leftarrow$ ACQUIREDATASET() ▷ Current Training Dataset
for $({}^b\hat{u}_i, {}^b\hat{g}_i) \in \mathcal{D}, i = 1, \cdots, |\mathcal{D}|$ **do**
  if $\|{}^bu_t - {}^b\hat{u}_i\| < \epsilon$ **then**
    $\mathcal{D}$.REMOVE$(({}^b\hat{u}_i, {}^b\hat{g}_i))$
  end if
end for
${}^bg_t \leftarrow X_t^{-1} \cdot X_{t+1}$ ▷ Object's Rigid Body Motion
$\mathcal{D} \leftarrow \mathcal{D} \cup \{({}^bu_t, {}^bg_t)\}$
$\Gamma, \Gamma^{-1} \leftarrow$ GPR$(\mathcal{D})$ ▷ Gaussian Process Regression
return Γ, Γ^{-1}

Moreover, if the newly executed control ${}^bu_t$ is very similar to any control ${}^b\hat{u}_i$ which already exists in the dataset $\mathcal{D}$ (i.e., their difference $\|{}^bu_t - {}^b\hat{u}_i\|$ is less than a threshold ϵ), the outdated control ${}^b\hat{u}_i$ will be removed from the dataset $\mathcal{D}$. This mechanism helps keep the size of the dataset small to facilitate efficient model approximation. More importantly, it enforces updating the models with the most recent data points, which are more relevant to the current task setup and the physical state of the robot-object system.

Importantly, online model update is useful especially when transferring the approximated models between manipulations of different objects. When the models are transferred by previous experiences of manipulating an old object, updating the models with online data can ensure fast adaptation of the models to the manipulation of the new object.

V. MPC-BASED ACTION GENERATION

With the system models Γ and Γ^{-1} approximated and adaptively updated online in Sec. IV, a model-based control scheme can be applied to generate real-time controls taking into account the predictions made by Γ and Γ^{-1}. To this end, as illustrated in Alg. 4, we integrate the approximated models Γ and Γ^{-1} in a Model Predictive Control (MPC) framework to close the control loop, for generating effective robot actions to precisely manipulate the object.

As illustrated in Fig. 4, with the object currently at configuration X_t, the approximated models Γ and Γ^{-1} are used to simulate Q controlled trajectories of the object's configuration, up to a prediction horizon L. Each trajectory, $\mathcal{T}^q = \{\hat{X}_k^q\}_{k=0}^L$, $q = 1, \cdots, Q$, is simulated by iteratively propagating the system through Eq. (3)-(6) with $\hat{X}_0^q = X_t$.

At each iteration of trajectory simulation, assuming the object configuration is $\hat{X}_k^q$ as predicted at the current step, the nearest waypoint in the reference trajectory $Y_{j^*} \in \mathcal{Y}$ is found by Eq. (3). The next waypoint Y_{j^*+1} is used as a reference goal to calculate the object's desired motion ${}^b\hat{g}_k^q$ by Eq. (4). We perturb this desired motion by applying a random transformation $g_\xi \in SE(2)$ to it, which is generated by randomly sampling a small rotation and translation. This perturbed motion is passed to the inverse model Γ^{-1} for

Algorithm 4 Model Predictive Control (MPC)

Input: Observed object's configuration X_t, reference path $\mathcal{Y}$
Output: Generated control for execution ${}^b u_t$

for $q = 1, \cdots, Q$ **do**
  $\mathcal{T}^q \leftarrow \{\hat{X}_0^q = X_t\}$ ▷ Simulated Trajectory
  for $k = 0, \cdots, L-1$ **do**
    $j^* \leftarrow \arg\min_j \Delta(\hat{X}_k^q, Y_j)$ ▷ Nearest Waypoint in $\mathcal{Y}$
    ${}^b\hat{g}_k^q \leftarrow \hat{X}_k^{q\,-1} \cdot Y_{j^*+1}$ ▷ Desired Motion of Object
    $g_\xi \leftarrow$ RandTransformation() ▷ Perturbation
    ${}^b\hat{u}_k^q = \Gamma^{-1}({}^b\hat{g}_k^q \cdot g_\xi)$ ▷ Predicted Control
    $\hat{X}_{k+1}^q \leftarrow \hat{X}_k^q \cdot \Gamma({}^b\hat{u}_k^q)$ ▷ Predicted Configuration
    $\mathcal{T}^q \leftarrow \mathcal{T}^q \cup \{\hat{X}_{k+1}^q\}$
  end for
end for
$q^* \leftarrow \arg\min_q \sum_{k=1}^{L} \left(\min_j \Delta(Y_j, \hat{X}_k^q)\right)$ ▷ Optimal
${}^b u_t \leftarrow {}^b\hat{u}_0^{q^*}$ ▷ First Predicted Control in $\mathcal{T}^{q^*}$
return ${}^b u_t$

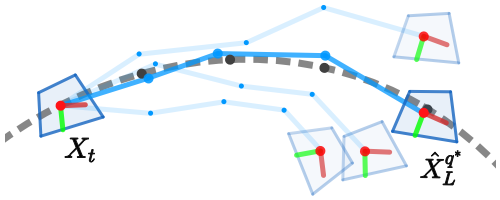

Fig. 4: Trajectory simulation and optimization by MPC. By iteratively propagating the object's configuration with the estimated models Γ and Γ^{-1} and random perturbations, a bunch of trajectories (blue) are simulated to a horizon L. The optimal one (thick blue), which is closest to the reference trajectory (dashed), is selected to extract the control input for execution.

predicting a desired control ${}^b\hat{u}_k^q$ by Eq. (5). Assuming this predicted control ${}^b\hat{u}_k^q$ is executed next, we can forward propagate the system by Γ to predict the outcome configuration $\hat{X}_{t+1}^q$ as in Eq. (6), to be used in the next iteration. It is worth noting that the Q trajectories are simulated differently due to the randomness in the perturbation g_ξ.

$$j^* = \underset{j \in \{1,\cdots,M\}}{\arg\min}\ \Delta(\hat{X}_k^q, Y_j) \tag{3}$$

$${}^b\hat{g}_k^q = \hat{X}_k^{q\,-1} \cdot Y_{j^*+1} \tag{4}$$

$${}^b\hat{u}_k^q = \Gamma^{-1}({}^b\hat{g}_k^q \cdot g_\xi) \tag{5}$$

$$\hat{X}_{k+1}^q = \hat{X}_k^q \cdot \Gamma({}^b\hat{u}_k^q) \tag{6}$$

Over the Q differently simulated trajectories, the optimal trajectory $\mathcal{T}^{q^*}$ with the lowest cost will be found by Eq. (7), and the first predicted control in $\mathcal{T}^{q^*}$ will be extracted for execution by the robot. As the entire procedure of trajectory simulation and optimization (visualized in Fig. 4) is performed at each time step by MPC, our unified framework is able to enable the robot to precisely manipulate the object through generated pushing actions in a closed loop.

$$q^* = \underset{q \in \{1,\cdots,Q\}}{\arg\min} \sum_{k=1}^{L} \left(\min_{j \in \{1,\cdots,M\}} \Delta(Y_j, \hat{X}_k^q)\right) \tag{7}$$

The generated control approaches optimal control as the number of simulated trajectories Q increases. When $Q = 0$, our control scheme will be reduced to executing a control directly predicted by the inverse model Γ^{-1}, thus becoming greedy and non-optimal.

Algorithm 5 SmoothenedExecute($\cdot$)

Input: Object's current configuration X_t, control ${}^b u_t$
Output: Object's outcome configuration X_{t+1}

1: ${}^b u_{t-1} \leftarrow$ GetLastControl() ▷ From the Previous Time Step
2: **if** $\|{}^b u_{t-1} - {}^b u_t\| < \sigma$ **then**
3: $p_t \leftarrow$ GetObjectPosition(X_t) ▷ Object's Position
4: $p_{EE} \leftarrow$ GetHandPosition() ▷ End-Effector's Position
5: $r \leftarrow \|p_t - p_{EE}\|$
6: $(\alpha_t, \beta_t) \leftarrow {}^b u_t$
7: $\gamma_t \leftarrow \alpha_t + \beta_t - \arcsin\frac{R \sin\beta_t}{r}$ ▷ R : The Virtual Circle Radius
8: $P' \leftarrow (r\cos\gamma_t, r\sin\gamma_t)$ ▷ In Object's Body Frame
9: MoveHandTo(P')
10: $X_{t+1} \leftarrow$ Push($\overline{PP'}, d$) ▷ $\overline{PP'}$: Direction; d: Distance
11: **else**
12: $X_{t+1} \leftarrow$ Execute(${}^b u_t$) ▷ Not Smoothened
13: **end if**
14: **return** X_{t+1}

VI. Smoothening the Execution of Controls

By our representation of controls defined in Sec. III-A and shown in Fig. 2, whenever executing a control generated by MPC, the robot needs to first retreat its gripper back to a point P on the virtual circle before approaching the object to push it. This makes the gripper move back and forth, causing the object not continuously pushed by the robot. To this end, we implemented an optimized execution in Alg. 5 by smoothly connecting adjacent control executions to facilitate continuous manipulation of the object.

We first calculate the Euclidean distance between the current control ${}^b u_t = (\alpha_t, \beta_t)$ and the control executed at the previous time step ${}^b u_{t-1} = (\alpha_{t-1}, \beta_{t-1})$. When the distance is less than a threshold σ, instead of reaching the point P, the new execution strategy will let the gripper first approach a point P' closer to the object than P and then move the gripper in the desired direction to apply the push. The point P' is chosen such that 1) its distance to the object should equal the current distance between the gripper and the object (i.e., r at line 3 in Alg. 5); 2) the direction of the line segment $\overline{PP'}$ matches the desired pushing direction of ${}^b u_t$. To guarantee $\overline{PP'}$ aligns with the desired pushing direction, we solve an auxiliary angle γ_t at line 5 in Alg. 5 by the Law of Sines in a triangle.

VII. Experiments

We evaluated our proposed framework UNO Push on a Franka Emika Panda robot, with the manipulated object tracked by cameras via AprilTags [21]. Through real-world experiments, we would like to investigate the performance of our framework from two aspects: 1) without any explicit modeling of the system or physical properties like object shapes and inertial parameters, what precision can be achieved by the proposed UNO Push; and 2) how the robustness of the approximated system models is affected by different settings of the framework and external perturbations. Moreover, in comparison with a state-of-the-art baseline which relies on system transition models trained

directly on the target objects [20], our framework achieved comparable manipulation precision in manipulating objects with unknown shapes and mass distributions.

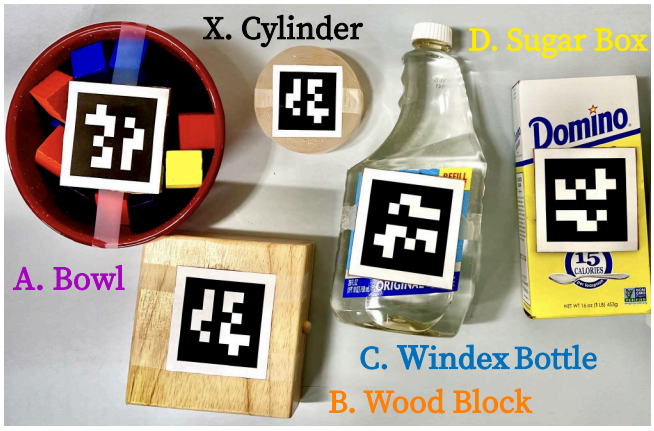

Fig. 5: The objects used in the experiments. A. Bowl (YCB #0024) B. Wood block (YCB #0071) C. Windex bottle (YCB #0022) D. Sugar box (YCB #0004) X. A Cylinder object for sampling.

We selected several objects from the YCB dataset [22] for experiments, as shown in Fig. 5. In all our experiments, we consistently ran our proposed framework (Alg. 1) at a frequency of 20Hz. The pushing distance d was set to 4mm. In all the experiments except for those in Sec. VII-A, no matter which target object was manipulated for testing, the non-parametric system models were learned only by the cylinder (object X in Fig. 5) with 10 sampled actions. For quantitative evaluation, we used a single circle with a radius of 0.15m as our reference path across all the experiments, as shown in Fig. 1. To evaluate the precision of object manipulation, we used Mean Absolute Error (MAE) as our metric, which was computed by averaging the absolute distance between the object's actual path and the reference path over the entire manipulation.

A. Robustness of Non-parametric Models

We tested UNO push with four different experimental settings described in Fig. 6. In different settings, the non-parametric system models were learned on object X or on the target object itself being manipulated (objects A-D in Fig. 5), and the online model update was opted to be enabled or not. We fixed the hyperparameters of MPC to $L = 20$ and $Q = 50$. Evaluated with different numbers of data points for model learning (i.e., $N = 5, 10, 20, 50$), the manipulation performance under different settings is reported in Fig. 7. The reported MAE was averaged over four trials, each on a different tested object (objects A-D).

First, without online model update enabled (Setting #1), more data collected with the cylinder X can cause worse manipulation performance on a novel object. Trained with more data collected on the cylinder, the estimated models were more likely to be overfitted to the cylinder and thus made more inaccurate predictions about the motions of the novel object. Second, by comparing #1 against #2 (or #3 against #4), no matter which object or how much data was

Setting	Description
#1	model learned on the **cylinder X** and transferred to the target object *without* online update
#2	model learned on the **cylinder X** and transferred to the target object *with* online update
#3	model learned on the **target object** *without* online update
#4	model learned on the **target object** *with* online update

Fig. 6: Four different settings for model estimation in the experiments of Sec. VII-A.

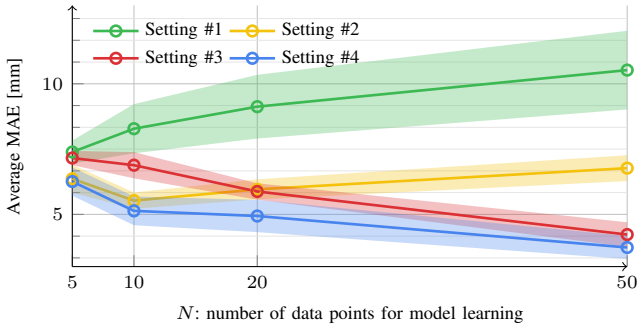

Fig. 7: Performance evaluation under different settings defined in Fig. 6, with different number of data points for model learning and transfer. The shaded regions indicate the standard deviations.

used to pre-train the model, online model update always facilitated better manipulation.This is because online model update enables more extensive exploration of the system models specific to the object being manipulated, resulting in more accurate model estimation. Last, when the models were learned on the target object being manipulated rather than on the cylinder (#3 or #4), more data points enabled better model estimation due to more valid explorations of the underlying system dynamics.

In general, the results have shown that a small amount of data is sufficient for our framework to achieve high-precision manipulation. With only 10 actions explored to train the initial models of Γ and Γ^{-1} (under Setting #2), an averaged MAE of less than 6mm was achieved on novel objects.

B. MPC Performance Evaluation

In this experiment, we evaluated the precision achieved with our framework by varying the prediction horizon L and the number of simulated trajectories Q in MPC. These two parameters were selected from $L = 5, 10, 20$ and $Q = 0, 10, 20, 50, 200$. For each different combination of L and Q, we conducted four trials on four objects (Object A-D shown in Fig. 5). We averaged the MAE over the four trials to summarize the results in Fig. 8. The system models were estimated under Setting #2 defined in Fig. 6. That is, we saved Γ and Γ^{-1} learned by manipulating the cylinder X through 10 actions, and transferred them as the system models for all other objects while updating them online.

From the results, we can see that the manipulation precision was improved when more simulated trajectories were generated. This is because, with more perturbed trajectories to optimize over, MPC is able to search more extensively

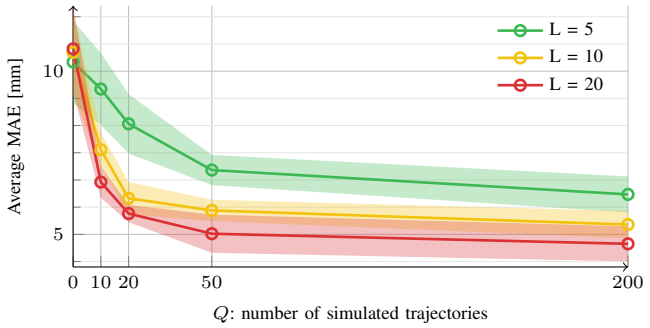

Fig. 8: Performance evaluation w.r.t. different L and Q in MPC, where the shaded regions indicate the standard deviations.

to find a more optimal control that has a higher probability of pushing the object in its desired direction. Moreover, with the same number of simulated trajectories, a larger prediction horizon L improves the manipulation performance. The reason is that a large L enables the system models to predict the long-horizon outcome of controls, providing more reliable evidence for generating effective controls. Particularly, when $Q = 0$, MPC was directly executing the predicted control from Γ^{-1}, which in most cases was not effective enough due to the inaccuracy of model approximation. This in turn verifies the importance of optimization in MPC. In general, with inaccurately approximated system models, MPC could achieve millimeter-level precision of manipulation.

C. Comparison with Baseline

In a comparative evaluation against a baseline [20], as reported in Fig. 9, our framework achieved comparable precision to the analytical method and the data-driven method in the baseline [20] for both circle and square paths, while requiring only 10 initial data points without any sophisticated modeling, significantly less than the baseline's $5,000$ data points under this specific test. As illustrated in Fig. 10, the outcome paths of the executions across various objects were consistently good, regardless of the object manipulated or the shape of the reference path.

It's worth noting that other results from the baseline have achieved a very good accuracy of 14mm with just 10 data points. We used the results with $5,000$ data points reported in [20] as our comparison baseline in Fig. 9, since that was the best result in [20] with the most comprehensive evaluation for both circle and square paths.

Moreover, while the baseline [20] was trained offline for a square object of uniform mass, our approach does not require any *a priori* knowledge about the contact geometries or physics, nor does it require any object-specific training. This makes it a more generalizable and low-barrier solution for nonprehensile manipulation tasks in everyday tasks.

D. Qualitative Evaluation

To further test the robustness of UNO Push, three additional evaluation tasks were performed. First, we applied our framework to trace letter-shaped paths of "R", "I", "C", and "E", with different objects, as shown in Fig. 11. The results

Path	Method	Error (mm)
Circle	[20] (analytical), $v = 20$ mm/s	2.89
	[20] (data-driven), $v = 20$ mm/s	6.53
	UNO Push (ours) $v = 50$ mm/s	5.87
Square	[20] (analytical), $v = 50$ mm/s	4.95
	[20] (data-driven), $v = 50$ mm/s	6.60
	UNO Push (ours), $v = 50$ mm/s	5.42

Fig. 9: Performance evaluation by comparison with a baseline, where v denotes the motion velocity of the robot gripper.

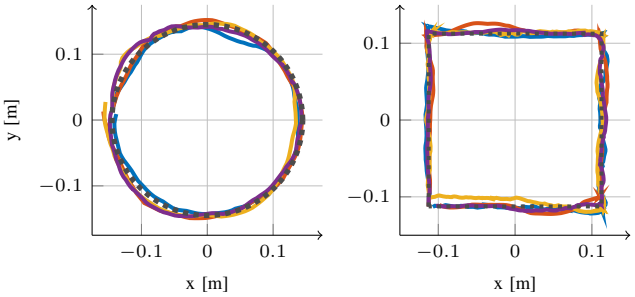

Fig. 10: The object's actual paths of UNO Push in experiments repeatd from the baseline [20], with the goal of tracing a circle (left) and a square (right) reference path. The lines are color-coded in the same way as the objects A-D shown in Fig. 5.

show the robustness of our framework to manipulate through complex paths even with sharp turns (from 90° to 180°).

As demonstrated in Fig. 12, our framework could also push objects through unknown perturbations caused by interactions with the object clutter, as assisted by the sufficient frequency of online model updates and MPC (20Hz). Finally, as shown in Fig. 1, even with an unmodeled object grasped by the robot gripper, our method was still able to push the target object through object-object contact. This again demonstrates that our framework is able to work with different physical uncertainties and unmodeled contacts.

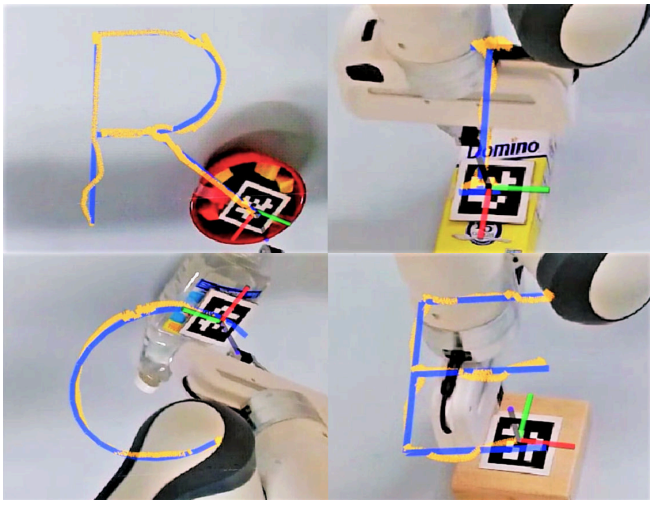

Fig. 11: Example object pushing manipulation to trace letter-shaped paths with four different objects. The blue lines are the reference paths, and the yellow lines are the actual paths of the object.

VIII. Conclusion

In this paper, we proposed a unified framework, named UNO Push, for pushing-based nonprehensile object manipu-

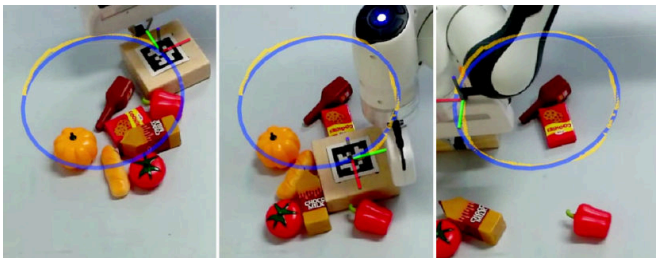

Fig. 12: Example object pushing manipulation through a cluttered area to demonstrate the performance under unmodeled external perturbations.

lation. It unifies system model estimation, action generation, and control through light-weight non-parametric learning and closed-loop MPC. With extensive experiments on a real 7-DoF robot, we showed that our framework can achieve millimeter-level manipulation precision, without requiring heavy data collection, sophisticated system modeling, or offline training on the target object.

Our work offers broad possibilities for applications of pushing-related manipulation tasks. As UNO Push provides an efficient approach for fundamental pushing tasks like trajectory tracking, it could serve as a low-level controller or motion primitive for various complex nonprehensile manipulation tasks. These tasks, such as non-prehensile rearrangement and multi-modal manipulation planning in household environments, often require precise manipulation of different objects without accurate object models and object-specific training.

Despite promising results, our method has limitations. First, our approach is derived under the assumption of planar pushing and quasi-static scenarios, so it may be difficult to deal with more dynamic motions of the object like rolling and flipping, or more dynamic actions like hitting and throwing objects on a 2D plane. Second, the target object is assumed to be a rigid body, and the proposed approach might not directly perform effectively on deformable or soft objects. Third, the reliance on visual markers for accurate object tracking poses challenges in environments where marker placement is impractical or where occlusions may occur.

In future work, we plan to extend the framework to tasks that involve more complex physical interactions, such as pushing a group of multiple objects together under formation constraints, as well as pushing objects that move with motions beyond quasi-static patterns, e.g., rolling. We are also interested in exploring the possibilities of applying the UNO Push to more complex nonprehensile rearrangement tasks, such as multi-object sorting. These tasks would require multi-modal manipulation skills, such as grasping and pushing for task and motion planning.

REFERENCES

[1] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," *The International Journal of Robotics Research*, vol. 15, no. 6, pp. 533–556, 1996.

[2] K. Hang, A. S. Morgan, and A. M. Dollar, "Pre-grasp sliding manipulation of thin objects using soft, compliant, or underactuated hands," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 662–669, 2019.

[3] K. Lynch, "Toppling manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4, 1999, pp. 2551–2557 vol.4.

[4] H. Song, J. A. Haustein, W. Yuan, K. Hang, M. Y. Wang, D. Kragic, and J. A. Stork, "Multi-object rearrangement with monte carlo tree search: A case study on planar nonprehensile sorting," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 9433–9440.

[5] E. Huang, Z. Jia, and M. T. Mason, "Large-scale multi-object rearrangement," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 211–218.

[6] J. A. Haustein, J. King, S. S. Srinivasa, and T. Asfour, "Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable states," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3075–3082.

[7] W. C. Agboh and M. R. Dogar, "Real-time online re-planning for grasping under clutter and uncertainty," in *IEEE International Conference on Humanoid Robots (HUMANOIDS)*. IEEE, 2018, pp. 1–8.

[8] J. Zhou, Y. Hou, and M. T. Mason, "Pushing revisited: Differential flatness, trajectory planning, and stabilization," *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1477–1489, 2019.

[9] X. Cheng, E. Huang, Y. Hou, and M. T. Mason, "Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6520–6526.

[10] M. T. Mason, "Progress in nonprehensile manipulation," *The International Journal of Robotics Research*, vol. 18, no. 11, pp. 1129–1141, 1999.

[11] M. C. Koval, N. S. Pollard, and S. S. Srinivasa, "Pre-and post-contact policy decomposition for planar contact manipulation under uncertainty," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 244–264, 2016.

[12] 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.

[13] F. Bertoncelli, F. Ruggiero, and L. Sabattini, "Linear time-varying mpc for nonprehensile object manipulation with a nonholonomic mobile robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 11 032–11 038.

[14] W. Yuan, K. Hang, D. Kragic, M. Y. Wang, and J. A. Stork, "End-to-end nonprehensile rearrangement with deep reinforcement learning and simulation-to-reality transfer," *Robotics and Autonomous Systems*, vol. 119, pp. 119–134, 2019.

[15] M. Bauza, F. Alet, Y.-C. Lin, T. Lozano-Pérez, L. P. Kaelbling, P. Isola, and A. Rodriguez, "Omnipush: accurate, diverse, real-world dataset of pushing dynamics with rgb-d video," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4265–4272.

[16] A. Kloss, S. Schaal, and J. Bohg, "Combining learned and analytical models for predicting action effects from sensory data," *The International Journal of Robotics Research*, vol. 41, no. 8, pp. 778–797, 2022.

[17] M. Bauza and A. Rodriguez, "A probabilistic data-driven model for planar pushing," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3008–3015.

[18] 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 International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3066–3073.

[19] K. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, "More than a million ways to be pushed," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[20] 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.

[21] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3400–3407.

[22] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-cmu-berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.