DART: Learning-Enhanced Model Predictive Control for Dual-Arm Non-Prehensile Manipulation

Anonymous

***Abstract*— *What appears effortless to a human waiter remains a major challenge for robots.* Manipulating objects non-prehensilely on a tray is inherently difficult, and the complexity is amplified in dual-arm settings. Such tasks are highly relevant to service robotics in domains such as hotels and hospitality, where robots must transport and reposition diverse objects with precision. We present DART, a novel dual-arm framework that integrates nonlinear Model Predictive Control (MPC) with an optimization-based impedance controller to achieve accurate object motion relative to a dynamically controlled tray. The framework systematically evaluates three complementary strategies for modeling tray–object dynamics as the state transition function within our MPC formulation: (i) a physics-based analytical model, (ii) an online regression-based identification model that adapts in real-time, and (iii) a reinforcement learning–based dynamics model that generalizes across object properties. Our pipeline is validated in simulation with objects of varying mass, geometry, and friction coefficients. Extensive evaluations highlight the trade-offs among the three modeling strategies in terms of settling time, steady-state error, control effort, and generalization across objects. To the best of our knowledge, DART constitutes the first framework for non-prehensile Dual-Arm manipulation of objects on a tray. Project Link: https://dart-icra.github.io/dart/**

I. INTRODUCTION

Service robots performing human like tasks face a fundamental limitation: most rely exclusively on grasping to manipulate objects [1], struggling with items that are large, fragile, or awkwardly positioned i.e. necessitating non-prehensile manipulation, i.e., manipulating objects without grasping [2], [3]. This field encompasses diverse strategies including pushing [4], throwing [5], and rolling [6], exploiting friction, inertia, and contact forces without rigid grasping to offer promising alternatives for challenging scenarios. Among these techniques [3], [5]–[9], tray-based object manipulation moves objects through controlled surface tilting, presenting a natural, human-inspired approach with broad applications in restaurants, hospitals, and laboratories. However, extending this capability to dual-arm systems introduces various challenges: the shared tray dynamically couples both arms, requiring precise coordination while simultaneously controlling object motion through uncertain dynamics. Unlike single-arm manipulation, this problem demands understanding and predicting how tray tilting affects object trajectories while maintaining stable, compliant control of the two manipulators.

When two robotic arms coordinate to manipulate a tray for controlling object motion on its surface, several critical constraints emerge [10]. **First**, the tray creates dynamic coupling between the arms, meaning any motion by one arm directly affects the tray orientation and, consequently, the object's trajectory and the other arm's required response. **Second**, precise object control requires accurate modeling of complex tray–object dynamics involving friction, inertia, and contact mechanics that vary significantly with object properties such as mass, geometry, and surface characteristics. **Third**, the system must maintain compliance to handle interaction forces, while executing coordinated tilting motions, as rigid control can amplify disturbances and destabilize the entire manipulation process. These coupled dynamics create a challenging control problem where traditional approaches struggle with adaptability.

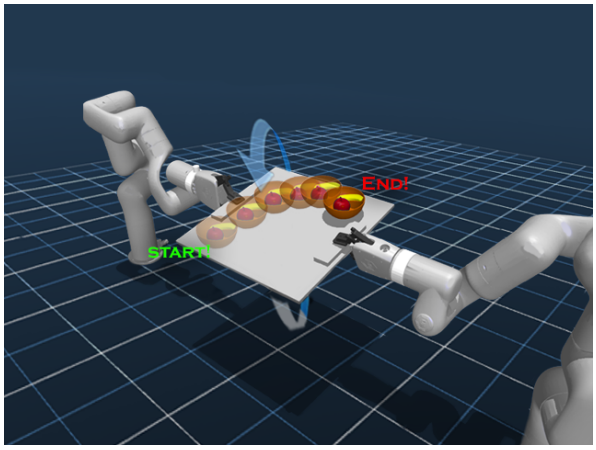

Fig. 1: **Non-Prehensile Manipulation** using a Dual xArm7 setup. Our framework transports a bowl of fruits to the goal location on the tray through a sequence of tilts given by MPC executed by the dual-arm system.

Existing research has largely treated non-prehensile manipulation and dual-arm coordination as separate problems. Non-prehensile manipulation work has primarily focused on single-arm systems [4], [11], [12], while dual-arm coordination research typically assumes grasped objects with well-defined contact models [13]–[16]. The intersection of dual-arm coordination and non-prehensile manipulation remains largely unexplored, despite its practical importance and technical complexity. Accurately modeling object-tray dynamics presents a fundamental challenge: analytical models may oversimplify complex contact mechanics, while purely data-driven approaches may lack interpretability and fail to generalize across diverse objects and conditions.

To address these limitations, we propose DART (Dual-Arm Robot Tray manipulation), a unified framework that integrates nonlinear Model Predictive Control (MPC) with optimization-based impedance control. Our MPC computes optimal tray tilt trajectories to guide objects to desired

Manuscript 3775 submitted to 2026 IEEE International Conference on Robotics and Automation (ICRA). Received September 16, 2025.

goals while accounting for system constraints and object dynamics. Simultaneously, our impedance controller ensures stable, compliant execution by both arms, managing interaction forces and maintaining coordination throughout the manipulation process. To summarize, our contributions are:

1) We introduce the to the best of our knowledge a first framework for dual-arm non-prehensile tray manipulation, formulating a control problem that bridges bimanual coordination with contact-rich object manipulation.
2) We develop three complementary dynamics modeling strategies: physics-based, online regression-based, and reinforcement learning-based. A key innovation of our framework is the integration of RL-learned state transition dynamics as a constraint within the MPC, enabling the controller to generalize across varying object geometries, masses, and surface interactions without requiring elaborate fine-tuning for each new object.
3) We demonstrate through extensive experiments across objects with varying mass, geometry, and surface properties that our framework performs well. We analyze and compare these approaches across metrics including settling time, steady-state error, and control effort.

II. RELATED WORKS

A. Non-Prehensile Manipulation

Non-prehensile manipulation is crucial for tasks involving hard-to-grasp objects or cluttered workspaces [2], [3]. This field encompasses diverse techniques such as throwing [5], pushing [17], rolling [6], and sliding [8], [18]. Several studies have explored the non-prehensile manipulation of objects on horizontal surfaces [9], [11], [12].

Prior research on planar object transportation has followed two distinct approaches. One focuses on maintaining object stability during transport, explicitly preventing sliding or slippage [11], [12]. In contrast, other works deliberately exploit sliding as the manipulation mechanism [8]. Many previous studies assume the presence of external barriers to constrain object motion and rely on precise knowledge of surface friction coefficients within their model-based controllers [18]. While such setups enable impressive controllability, they limit generalization to unconstrained environments with unknown friction properties. Recent advances have addressed these limitations through adaptive and learning-based approaches [8], but the vast majority of non-prehensile manipulation research remains constrained to single-arm systems.

B. Dual-Arm Manipulation

Dual-arm manipulation extends robotic capabilities beyond single-arm limitations, enabling coordinated object transport and bimanual assembly [10], [14]. Dual-arm systems are pivotal for manipulating heavier payloads that exceed single-arm capacity limits [19], [20], providing enhanced stability through distributed load-bearing, crucial for tray based transportation and manipulation where payload weight, tray stability, and precise orientation control must be simultaneously managed [10].

Traditional approaches utilize impedance-based control [21], [22] to handle dynamic coupling between manipulators, while recent advances have employed learning methods including imitation learning [23], vision-language models [24], and reinforcement learning [25]. These approaches have demonstrated impressive capabilities in coordinated manipulation tasks requiring precise force regulation and synchronization between arms [26], [27], which is essential for maintaining stable tray orientation during transport.

However, existing dual-arm research exclusively assumes rigidly grasped objects with controlled dynamics. This fundamental assumption fails in tray-based scenarios where objects slide freely, creating entirely different contact dynamics that existing frameworks do not address. Single-arm non-prehensile approaches cannot leverage the enhanced force distribution and coordination that dual-arm systems provide.

To bridge this gap, our DART framework introduces the first integration of non-prehensile tray manipulation with dual-arm coordination, combining complementary dynamics modeling strategies within a unified nonlinear MPC and optimization-based impedance control architecture for dual-arm non-prehensile object manipulation. The framework's generality is a key strength, enabling adaptation to diverse object geometries, masses, and surface properties without requiring extensive recalibration or object-specific modeling. DART enables precise manipulation of objects through controlled tray tilting, a capability essential for service robotics applications in domains such as hospitality, healthcare, and domestic environments.

III. SYSTEM DESCRIPTION

We consider a system of two manipulator arms rigidly grasping a tray as shown in Fig. 2, and an object of mass m is placed on it. The world frame $\{\mathcal{W}\}$ is fixed to the environment, the tray frame $\{\mathcal{B}\}$ aligned with $\{\mathcal{W}\}$ at zero tilt, and the object frame $\{\mathcal{O}\}$ is attached to the object's center of mass (CoM). The object-tray configuration and dual arm system are described as follows.

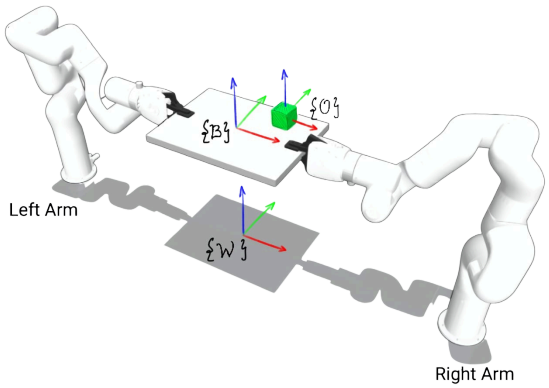

Fig. 2: **Task Setup:** Two robotic arms are placed with the bases fixed to the ground. The tray is rigidly grasped and an object is placed on the tray. The world frame is represented by $\{\mathcal{W}\}$, body frame by $\{\mathcal{B}\}$ and object frame by $\{\mathcal{O}\}$.

Manuscript 3775 submitted to 2026 IEEE International Conference on Robotics and Automation (ICRA). Received September 16, 2025.

A. Object–Tray Model

The pose of the object in $\{\mathcal{W}\}$ is defined as $\mathbf{x} = [\mathbf{p}^\top,\ \boldsymbol{\theta}^\top]^\top$, where $\mathbf{p} \in \mathbb{R}^3$ is the position and $\boldsymbol{\theta} \in \mathbb{R}^3$ is the orientation (roll-pitch-yaw). The linear and angular velocities are $\mathbf{v} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$. The twist in $\{\mathcal{W}\}$ is defined as $\boldsymbol{\nu} = [\mathbf{v}^\top, \boldsymbol{\omega}^\top]^\top$. The object state in $\{\mathcal{W}\}$ is defined as $\mathbf{X} = [\mathbf{x}^\top, \boldsymbol{\nu}^\top]^\top$. The control input $\mathbf{u} = [\mathrm{u}_\alpha,\ \mathrm{u}_\beta]^\top \in \mathbb{R}^2$ represents the commanded roll (u_α) and pitch (u_β) angles of the tray in the world frame $\{\mathcal{W}\}$. These tilt angles are constrained by the tray actuation limits $|\mathrm{u}_\alpha| \leq \mathrm{u}_\alpha^{\max}$, $|\mathrm{u}_\beta| \leq \mathrm{u}_\beta^{\max}$ (refer to Table I). The desired goal state of the object in $\{\mathcal{W}\}$ is denoted by $\mathbf{X}^{\mathrm{ref}}$. The object pose and twist in $\{\mathcal{O}\}$ are denoted by $\mathbf{x}_\mathrm{o}$ and $\boldsymbol{\nu}_\mathrm{o}$ respectively. Friction coefficient between the object and tray is represented by μ.

B. Dual-Robot Model

We consider a system of two torque-controlled manipulators with 7 degrees of freedom (DoF). The joint positions, velocities, and acceleration vectors are denoted by $\mathbf{q}_{\mathrm{L/R}}$, $\dot{\mathbf{q}}_{\mathrm{L/R}}$, $\ddot{\mathbf{q}}_{\mathrm{L/R}}$, where the subscript L/R refers to the *left* or *right* arm, respectively. The joint-space mass matrix is $\boldsymbol{M}(\mathbf{q})$, and the combined centripetal, Coriolis, and gravitational effects are given by $\boldsymbol{h}(\mathbf{q}, \dot{\mathbf{q}})$. The manipulator's Jacobian and its derivative are given by $\boldsymbol{J}(\mathbf{q})$ and $\dot{\boldsymbol{J}}(\mathbf{q})$ respectively. The joint torques applied to each arm are $\boldsymbol{\tau}_{\mathrm{L/R}}$.

The two end-effectors rigidly grasp the tray at fixed, predefined grasp poses $\{G_L, G_R\} \in SE(3)$, enabling coordinated tray manipulation. The poses of the end-effector are expressed as $\mathbf{y}_{\mathrm{L/R}} = \begin{bmatrix}\mathbf{y}_{\mathrm{pos,L/R}}, & \mathbf{y}_{\mathrm{ori,L/R}}\end{bmatrix}^\top$, where $\mathbf{y}_{\mathrm{pos,L/R}} \in \mathbb{R}^3$ is the Cartesian position of the end-effector and $\mathbf{y}_{\mathrm{ori,L/R}} \in \mathbb{R}^3$ is its orientation (e.g., roll–pitch–yaw) in $\{\mathcal{W}\}$. The desired end–effector poses $\mathbf{y}_{\mathrm{L/R}}^{\mathrm{ref}}$ are obtained from the desired tray pose as their relative placement w.r.t. the tray CoM stays constant.

IV. METHOD

We adopt a two-layer control framework for non-prehensile manipulation of the object as described below:

1) **High-level controller:** A nonlinear MPC (refer Sec. IV-A) for manipulating the object from its current state $\mathbf{X}$ to the desired state $\mathbf{X}^{\mathrm{ref}}$, computes $\mathbf{u}$ over the prediction horizon N. The dynamics rollout constraint in this MPC is modeled in three different ways as described in Sec. IV-B.
2) **Low-level controller:** A torque controller, adapted from an optimization-based dual-arm impedance control framework (refer to Sec. IV-C) [26], [28], computes $\boldsymbol{\tau}_{\mathrm{L/R}}$ for each manipulator in order to realize the commanded tray tilts $\mathbf{u}$, as given by the high-level controller.

A. Model Predictive Control

The high-level MPC controller computes optimal control commands $\mathbf{u}_\mathrm{k}$ and object states $\mathbf{X}_\mathrm{k}$ for a horizon of length N. It solves the nonlinear optimization in Eq. 1 for each discrete time step $\mathrm{k} \in [0, \mathrm{N})$ which is sampled at $\mathrm{T_s}$. The tracking error at step k is defined as $\mathbf{e}_\mathrm{k} = (\mathbf{X}_\mathrm{k} - \mathbf{X}^{\mathrm{ref}})$.

The cost function penalizes the state tracking error $\mathbf{e}_\mathrm{k}$, the control effort $\mathbf{u}_\mathrm{k}$, the control rate change $\Delta\mathbf{u}_\mathrm{k}$, and the terminal state error $\mathbf{e}_\mathrm{N}$ to ensure smooth tray motion:

$$\min_{\mathcal{X},\,\mathcal{U}} \sum_{\mathrm{k}=0}^{\mathrm{N}-1} \left(\mathbf{e}_\mathrm{k}^\top \mathbf{Q}\, \mathbf{e}_\mathrm{k} + \begin{bmatrix}\mathbf{u}_\mathrm{k} \\ \Delta\mathbf{u}_\mathrm{k}\end{bmatrix}^\top \mathbf{Q}_{\mathbf{R}} \begin{bmatrix}\mathbf{u}_\mathrm{k} \\ \Delta\mathbf{u}_\mathrm{k}\end{bmatrix} \right) + \mathbf{e}_\mathrm{N}^\top \mathbf{Q}_\mathrm{N} \mathbf{e}_\mathrm{N} \tag{1a}$$

$$\text{s.t. } \mathbf{X}_{\mathrm{k}+1} = \Phi\big(\mathbf{X}_\mathrm{k}, \mathbf{u}_\mathrm{k}, \Delta t\big), \tag{1b}$$

$$\boldsymbol{\nu}_{\min} \leq \boldsymbol{\nu}_\mathrm{k} \leq \boldsymbol{\nu}_{\max} \tag{1c}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_\mathrm{k} \leq \mathbf{u}_{\max} \tag{1d}$$

$$\Delta\mathbf{u}_{\min} \leq \Delta\mathbf{u}_\mathrm{k} \leq \Delta\mathbf{u}_{\max} \tag{1e}$$

Here, $\mathbf{Q}$ and $\mathbf{Q}_\mathrm{N}$ denote the weight matrices for the state tracking error along the horizon and at the terminal step, respectively. In our implementation, $\mathbf{Q}$ is structured to separately weight position and velocity errors ($\mathbf{Q} = \mathrm{diag}(\mathbf{Q}_\mathrm{p}, \mathbf{Q}_\mathrm{v})$) (refer to Table I). The matrix $\mathbf{Q}_\mathbf{R}$ penalizes both the control input $\mathbf{u}_\mathrm{k}$ and its rate of change $\Delta\mathbf{u}_\mathrm{k}$, where

$$\Delta\mathbf{u}_\mathrm{k} = \begin{cases} \mathbf{u_0} - \mathbf{u}_{\mathrm{prev}}, & \mathrm{k} = 0, \\ \mathbf{u}_\mathrm{k} - \mathbf{u}_{\mathrm{k}-1}, & 0 < \mathrm{k} \leq \mathrm{N} \end{cases} \tag{2}$$

Here $\mathbf{u}_{\mathrm{prev}}$ is defined as the command executed at the previous simulation timestep. Initially, $\mathbf{u}_{\mathrm{prev}}$ is defined as $[0, 0]^\top$. The decision variables are stacked as $\mathcal{X} = [\mathbf{X}_0^\top,\ \mathbf{X}_1^\top,\ \ldots,\ \mathbf{X}_\mathrm{N}^\top]^\top$ and $\mathcal{U} = [\mathbf{u}_0^\top,\ \mathbf{u}_1^\top,\ \ldots,\ \mathbf{u}_\mathrm{N}^\top]^\top$, which together describe the object state and control trajectories over the horizon. The object dynamics are enforced as a discrete-time state-transition mapping $\Phi(\cdot)$ (Eq. 1b). We obtain this function by numerical integration of the object dynamic models illustrated in Sec. IV-B.

B. Object–Tray Dynamics

Within our MPC framework (Sec. IV-A), the state–transition function (Eq. 1b) is instantiated with three alternative dynamics models. We denote the complete MPC pipeline under each instantiation as :

- **Physics–Based MPC (PMPC):** Uses analytical object dynamics with simple contact models and therefore requires modeling the dynamics of the object.
- **Regression–Based MPC (RMPC):** Augments a nominal dynamics model with an online linear regressor to capture unmodeled dynamics and contact effects, reducing reliance on exact dynamics.
- **Reinforcement–Learning–Based MPC (LMPC):** Starts from a nominal dynamics model and learns object dynamics parameters via reinforcement learning (RL) to directly capture unmodeled behaviors.

PMPC is purely physics–based, whereas RMPC and LMPC relax the need for exact dynamics and contact models through online adaptation or learning. The three variants are detailed below.

*1) **Physics-Based Dynamics Model**:* The object dynamics in the frame $\{\mathcal{O}\}$ as adapted from [29] is expressed as

$$\mathbf{M}_\mathrm{o}(\mathbf{x}_\mathrm{o})\, \dot{\boldsymbol{\nu}}_\mathrm{o} + \mathbf{C}_\mathrm{o}(\mathbf{x}_\mathrm{o}, \boldsymbol{\nu}_\mathrm{o})\, \boldsymbol{\nu}_\mathrm{o} + \mathbf{g}_\mathrm{o}(\mathbf{x}) = \mathbf{F}_\mathrm{o}(\mathbf{x}_\mathrm{o}, \boldsymbol{\nu}_\mathrm{o}, \mathbf{u}) \tag{3}$$

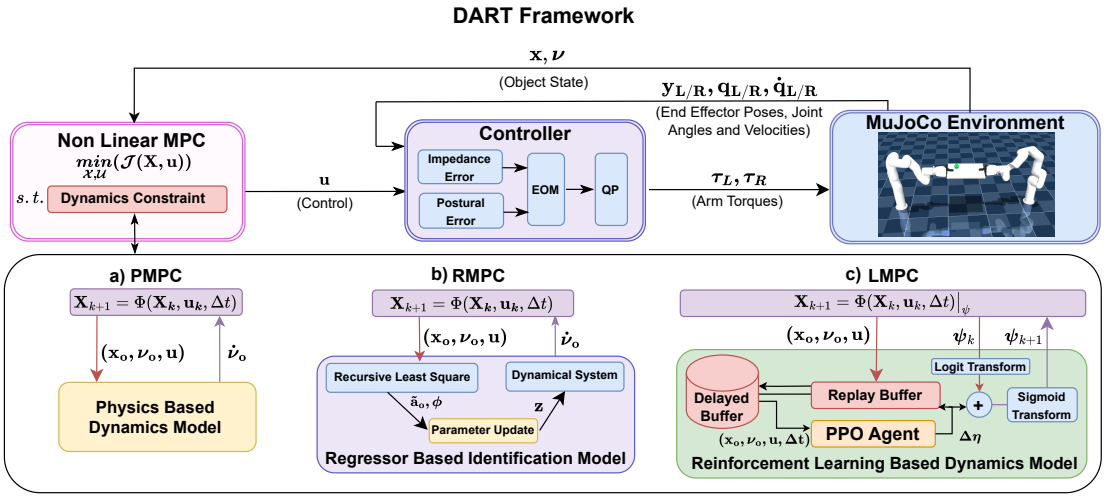

Fig. 3: **DART Framework:** Our proposed framework takes the current object state $\mathbf{X}$ and the desired target state $\mathbf{X}^{\text{ref}}$. We choose $\boldsymbol{\nu}^{\text{ref}}$ as $\mathbf{0}_{6\times1}$ as inputs. These are fed into a nonlinear MPC, which computes the optimal tray-tilt commands ($\mathbf{u}$). These commands are then passed to an optimization-based impedance controller, which computes the torques required to realize the tilts. Feedback from the simulator updates the object state for closing the loop for the next MPC step. The object-tray dynamics is modeled as a state transition constraint for the MPC. We propose three models for this state transition constraint, namely **(a) PMPC:** an analytical physics-based dynamics model, **(b) RMPC:** a regressor-based model which learns unmodeled dynamics and **(c) LMPC:** a PPO agent used to estimate the object dynamics.

where $\mathbf{M}_o \in \mathbb{R}^{6\times6}$, $\mathbf{C}_o \in \mathbb{R}^{6\times6}$, and $\mathbf{g}_o \in \mathbb{R}^6$ denote, respectively, the inertia matrix, the centrifugal and Coriolis matrix, and the gravity vector of the object. Here $\mathbf{F}_o(\mathbf{x}_o, \boldsymbol{\nu}_o, \mathbf{u}) = \mathbf{J}_c(\mathbf{x}_o)^\top \boldsymbol{f}(\mathbf{x}_o, \boldsymbol{\nu}_o, \mathbf{u}) \in \mathbb{R}^6$ denotes the generalized contact wrench produced due to the tray's contact interactions. $\mathbf{J}_c(\mathbf{x}_o)$ represents the contact Jacobian and $\boldsymbol{f}(.)$ encodes the stacked contact forces. Frictional effects are modeled using the Stribeck friction model [30].

The tray orientation (tilt) is controlled through the dual-arm manipulators. The tilt $\mathbf{u}$ modifies the components of the gravity vector acting along the tray's tangent plane. Thus, $\mathbf{g}_o(\mathbf{x}_o)$ depends on the tray tilt and we can say $\mathbf{g}_o(\mathbf{x}_o) \sim \mathbf{G}_o(\mathbf{x}_o, \mathbf{u})$. After updating and rearranging, we compute the acceleration of the object as follows:

$$\dot{\boldsymbol{\nu}}_o = \mathbf{M}_o^{-1}(\mathbf{x}_o)\Big[\mathbf{F}_o(\mathbf{x}_o, \boldsymbol{\nu}_o, \mathbf{u}) - \mathbf{C}_o(\mathbf{x}_o, \boldsymbol{\nu}_o)\,\boldsymbol{\nu}_o - \mathbf{G}_o(\mathbf{x}_o, \mathbf{u})\Big] \tag{4}$$

which can be compactly written in continuous time as

$$\dot{\boldsymbol{\nu}}_o = \mathcal{F}\big(\mathbf{x}_o,\, \boldsymbol{\nu}_o,\, \mathbf{u}\big) \tag{5}$$

Numerical Integration of Eq. 5 yields the state-transition function $\Phi(\cdot)$ used as the rollout constraint in the MPC:

$$\mathbf{X}_{k+1} = \Phi\big(\mathbf{X}_k,\, \mathbf{u}_k\big) \tag{6}$$

Φ is computed by transforming the pose from $\{\mathcal{W}\}$ to $\{\mathcal{O}\}$, evaluating object dynamics in $\{\mathcal{O}\}$, integrating $\boldsymbol{\nu}$ in $\{\mathcal{O}\}$ and then updating the pose in $\{\mathcal{W}\}$ using the appropriate map.

*2) **Regression-Based Identification Model**:* We employ a regressor-based model, formulated as a Recursive Least Squares (RLS) [31] estimator, to learn the unmodeled dynamics.

For each Cartesian axis $\mathrm{j} \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$ in $\{\mathcal{O}\}$, we construct 3×1 feature vectors ${}^{\mathrm{j}}\boldsymbol{\phi}_t = [\mathrm{v_j}, \tanh(\mathrm{v_j}/\epsilon_t), 1]^\top$ and ${}^{\mathrm{j}}\boldsymbol{\phi}_r = [\omega_j, \tanh(\omega_j/\epsilon_r), 1]^\top$ for the translational and rotational components of the unmodeled dynamics respectively. Here, ϵ_t and ϵ_r are regularization terms.

These vectors $({}^{\mathrm{j}}\boldsymbol{\phi}_t,\ {}^{\mathrm{j}}\boldsymbol{\phi}_r)$ capture Coriolis and damping effects, Coulomb friction and other bias, which correlate with velocity, tanh expressions and a numerical constant respectively.

The adaptive parameter vectors, ${}^{\mathrm{j}}\mathbf{z}_t, {}^{\mathrm{j}}\mathbf{z}_r \in \mathbb{R}^3$, and the feature vectors ${}^{\mathrm{j}}\boldsymbol{\phi}_t$ and ${}^{\mathrm{j}}\boldsymbol{\phi}_r$ are used to model the dynamics as follows:

$$\hat{\mathbf{a}}_o = \bar{\mathbf{a}}_o + \Delta\hat{\mathbf{a}}_{o,k} \tag{7}$$

$$\bar{\mathbf{a}}_o = \mathbf{M}_o^{-1}(\mathbf{x}_o)\Big[-\mathbf{C}_o(\mathbf{x}_o, \boldsymbol{\nu}_o)\,\boldsymbol{\nu}_o - \mathbf{G}_o(\mathbf{x}_o, \mathbf{u})\Big] \tag{8}$$

where $\Delta\hat{\mathbf{a}}_{o,k} =$

$$\big[{}^{\mathrm{x}}\boldsymbol{\phi}_t^\top {}^{\mathrm{x}}\mathbf{z}_t, {}^{\mathrm{y}}\boldsymbol{\phi}_t^\top {}^{\mathrm{y}}\mathbf{z}_t, {}^{\mathrm{z}}\boldsymbol{\phi}_t^\top {}^{\mathrm{z}}\mathbf{z}_t, {}^{\mathrm{x}}\boldsymbol{\phi}_r^\top {}^{\mathrm{x}}\mathbf{z}_r, {}^{\mathrm{y}}\boldsymbol{\phi}_r^\top {}^{\mathrm{y}}\mathbf{z}_r, {}^{\mathrm{z}}\boldsymbol{\phi}_r^\top {}^{\mathrm{z}}\mathbf{z}_r\big]^\top \tag{9}$$

Online Parameter Identification: At each step k, we compute the acceleration discrepancy defined as $\tilde{\mathbf{a}}_o = \big[\tilde{\mathbf{a}_{o,t}}^\top, \tilde{\mathbf{a}_{o,r}}^\top\big]^\top \in \mathbb{R}^6$, $\tilde{\mathbf{a}}_{o,k} = \mathbf{a}_{o,k} - \bar{\mathbf{a}}_{o,k}$ where $\mathbf{a}_{o,k} = (\boldsymbol{\nu}_{o,k} - \boldsymbol{\nu}_{o,k-1})/\mathrm{T_s}$.

We then update the adaptive parameter vectors ${}^{\mathrm{j}}\mathbf{z}_t$ and ${}^{\mathrm{j}}\mathbf{z}_r$ using the regressors ${}^{\mathrm{j}}\boldsymbol{\phi}_{t,k} = {}^{\mathrm{j}}\boldsymbol{\phi}_t(\boldsymbol{\nu}_{o,k-1})$ and ${}^{\mathrm{j}}\boldsymbol{\phi}_{r,k} = {}^{\mathrm{j}}\boldsymbol{\phi}_r(\boldsymbol{\nu}_{o,k-1})$.

For all $\mathrm{j} \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$ and forgetting factor $\lambda \in (0, 1]$, the RLS update is:

$$\begin{aligned}
{}^{\mathrm{j}}\mathrm{K}_{t,k} &= \frac{{}^{\mathrm{j}}\mathrm{P}_{t,k-1}{}^{\mathrm{j}}\phi_{t,k}}{\lambda + {}^{\mathrm{j}}\phi_{t,k}^\top {}^{\mathrm{j}}\mathrm{P}_{t,k-1}{}^{\mathrm{j}}\phi_{t,k}} \\
{}^{\mathrm{j}}\mathrm{z}_{t,k} &= {}^{\mathrm{j}}\mathrm{z}_{t,k-1} + {}^{\mathrm{j}}\mathrm{K}_{t,k}\big({}^{\mathrm{j}}\tilde{\mathbf{a}}_{o,t,k} - {}^{\mathrm{j}}\phi_{t,k}^\top {}^{\mathrm{j}}\mathrm{z}_{t,k-1}\big) \\
{}^{\mathrm{j}}\mathrm{P}_{t,k} &= \frac{1}{\lambda}\big({}^{\mathrm{j}}\mathrm{P}_{t,k-1} - {}^{\mathrm{j}}\mathrm{K}_{t,k}{}^{\mathrm{j}}\phi_{t,k}^\top {}^{\mathrm{j}}\mathrm{P}_{t,k-1}\big)
\end{aligned} \tag{10}$$

Here ${}^{\mathrm{j}}\mathrm{P}_{t,k}$ denotes the error covariance matrix and ${}^{\mathrm{j}}\mathrm{K}_{t,k}$ is the gain vector for axis j. An analogous update is applied to

${}^{\mathrm{j}}\mathbf{z}_{\mathrm{r,k}}$ for all $\mathrm{j} \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$. We use $\dot{\boldsymbol{\nu}}_{\mathrm{o}} = \hat{\mathbf{a}}_{\mathrm{o,k}}$ and numerically integrate it to obtain the state transition constraint similar to Eq. 6.

*3) **Reinforcement Learning-Based Dynamics Model**:* We use proximal policy optimization (PPO) [32], an RL method to learn the parameter vector $\boldsymbol{\psi}$ which collects the state independent components of $\mathbf{M}_{\mathrm{o}},\ \mathbf{C}_{\mathrm{o}},\ \mathbf{G}_{\mathrm{o}}$ and $\mathbf{F}_{\mathrm{o}}$.

The system dynamics with the learned parameter vector is represented as

$$\dot{\boldsymbol{\nu}}_{\mathrm{o}} = \widehat{\mathbf{M}}_{\mathrm{o}}(\boldsymbol{\psi}, \mathbf{x_o})^{-1}\Big[\widehat{\mathbf{F}}_{\mathrm{o}}(\boldsymbol{\psi}, \mathbf{x_o}, \boldsymbol{\nu}_{\mathrm{o}}) - \widehat{\mathbf{C}}_{\mathrm{o}}(\boldsymbol{\psi}, \mathbf{x}_{\mathrm{o}}, \boldsymbol{\nu}_{\mathrm{o}})\,\boldsymbol{\nu}_{\mathrm{o}} - \widehat{\mathbf{G}}_{\mathrm{o}}(\boldsymbol{\psi}, \mathbf{x_o}, \mathbf{u})\Big] \tag{11}$$

PPO is employed to compute the structured updates of the parameter vector $\boldsymbol{\psi}$, enabling the dynamics model to account for unmodeled disturbances, parametric uncertainty, and model mismatch.

To ensure that $\widehat{\boldsymbol{M}}_{\mathrm{o}}$ remains positive definite and $\widehat{\boldsymbol{C}}_{\mathrm{o}}$ and $\widehat{\boldsymbol{G}}_{\mathrm{o}}$ are physically plausible, we constrain $\boldsymbol{\psi}$ by means of a tanh squashing function. Further, to prevent saturation, we adopt a Logit–Sigmoid reparameterization to ensure $\boldsymbol{\psi}$ remains bounded, while the learning process is unconstrained. Each parameter is normalized and mapped into logit space.

$$\boldsymbol{r} = \boldsymbol{\psi}\ /\psi_{\max} \qquad \boldsymbol{\eta} = \ln\left(\boldsymbol{r}\ /\ (1-\boldsymbol{r})\right) \tag{12}$$

The policy provides $\Delta\boldsymbol{\eta}$ and we map $\boldsymbol{\eta}' = \boldsymbol{\eta} + \Delta\boldsymbol{\eta}$ back through the sigmoid

$$\boldsymbol{r}' = \sigma(\boldsymbol{\eta}') \qquad \boldsymbol{\psi}' = \psi_{\max}\boldsymbol{r}' \tag{13}$$

This ensures adaptation occurs in an unconstrained space while the realized parameters remain bounded and stable.

Experience from state transitions and MPC commands, along with sampled actions, is stored in a dense replay buffer for policy updates. A secondary sparse buffer collects samples at a lower rate, promoting generalization across variations in geometry, mass, and friction, enabling PPO to adapt robustly to diverse dynamical conditions.

Finally, the policy receives a reward signal $\mathbf{R}$ which encourages low position error and low velocity of the object simultaneously. It also collects the L1 norm of $\Delta\mathbf{u}$ as a penalty.

$$\mathbf{R} = e^{\frac{-\|\mathbf{p}-\mathbf{p}^{\mathrm{ref}}\|_2^2}{2\sigma_{\mathrm{p}}^2}}\left(w_{\mathrm{p}}\ + w_{\mathrm{v}}\ e^{\frac{-\|\mathbf{v}-\mathbf{v}^{\mathrm{ref}}\|_2^2}{2\sigma_{\mathrm{v}}^2}}\right) - \|\Delta\mathbf{u}\|_1 \tag{14}$$

where w_{p} and w_{v} are scalar weights that are tuned to the desired scaling between the position and velocity minimization rewards.

C. Controller

At each step, the high–level MPC (Sec. IV-A) outputs the commanded tray tilts $\mathbf{u}$ (roll–pitch in $\{\mathcal{W}\}$). The tray CoM remains fixed while its orientation follows $\mathbf{u}$.

We adapt the QP–based dual–arm impedance controller of [26], [28] to realize these poses. At each time step, the optimal joint accelerations are computed by solving

$$(\ddot{\mathbf{q}}_L^*, \ddot{\mathbf{q}}_R^*) = \arg\min_{(\ddot{\mathbf{q}}_L, \ddot{\mathbf{q}}_R)} w_{\mathrm{imp}}\big(\|\boldsymbol{e}_{\mathrm{imp}_L}\|_2^2 + \|\boldsymbol{e}_{\mathrm{imp}_R}\|_2^2\big) + w_{\mathrm{pos}}\big(\|\boldsymbol{e}_{\mathrm{pos}_L}\|_2^2 + \|\boldsymbol{e}_{\mathrm{pos}_R}\|_2^2\big) \tag{15}$$

subject to joint position, velocity, and torque limits.

The impedance–task error $\boldsymbol{e}_{\mathrm{imp}}$ is

$$\boldsymbol{e}_{\mathrm{imp}} = \ddot{\mathbf{y}} - \boldsymbol{\Lambda}^{-1}\big[\boldsymbol{D}\big(\dot{\mathbf{y}}^{\mathrm{ref}} - \dot{\mathbf{y}}\big) + \boldsymbol{K}\big(\mathbf{y}^{\mathrm{ref}} - \mathbf{y}\big)\big] \tag{16}$$

$$\boldsymbol{\Lambda} = \left(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^{\top}\right)^{-1}, \quad \ddot{\mathbf{y}} = \boldsymbol{J}\ddot{\mathbf{q}} + \dot{\boldsymbol{J}}\dot{\mathbf{q}} \tag{17}$$

The postural–task error biases each arm to its grasp configuration:

$$\boldsymbol{e}_{\mathrm{pos}} = \ddot{\mathbf{q}} - 2\sqrt{\boldsymbol{K}_{\mathrm{null}}}\,(\dot{\mathbf{q}}^{\mathrm{ref}} - \dot{\mathbf{q}}) + \boldsymbol{K}_{\mathrm{null}}(\mathbf{q}^{\mathrm{ref}} - \mathbf{q}), \tag{18}$$

with $\boldsymbol{K}, \boldsymbol{K}_{\mathrm{null}}$ and $\boldsymbol{D}$ as stiffness and damping matrices (Table I). The damping matrix $\boldsymbol{D}$ is computed using $\sqrt{\boldsymbol{\Lambda}}\sqrt{\mathbf{K}} + \sqrt{\mathbf{K}}\sqrt{\boldsymbol{\Lambda}}$.

Finally, the optimal joint accelerations yield the torques

$$\boldsymbol{\tau} = \boldsymbol{M}(\mathbf{q})\,\ddot{\mathbf{q}}^* + \boldsymbol{h}(\mathbf{q}, \dot{\mathbf{q}}), \tag{19}$$

which are applied to the simulator. This converts the MPC generated tray tilts into dual–arm torques realizing the desired motion while respecting all constraints.

V. EXPERIMENTS

We evaluate our proposed framework as a simulated dual-arm non-prehensile tray manipulation task, where two UFactory xArm7 manipulators jointly tilt a tray to drive an object on the tray towards a desired target state. We emphasize robustness to variations in object mass, geometry, and tray–object friction. All experiments are conducted in the MuJoCo simulator with identical physical models, initial conditions, and prediction horizons to ensure unbiased comparisons across the methods.

A. Task Setup

In each trial, a tray with mass of 1 kg and dimensions of $40\,\mathrm{cm} \times 30\,\mathrm{cm} \times 1\,\mathrm{cm}$, is manipulated by the control input $\mathbf{u}$ executed as in Sec. IV-C.

We consider three object geometries: {cube, cylinder, sphere}, representing area, line, and point contacts, respectively. Each object is evaluated at two different masses $\{1, 2\}$ kg and three different friction coefficients $\{0.05, 0.10, 0.20\}$, resulting in $3 \times 3 \times 2 = 18$ unique object configurations. Fixing the initial object position to be at the tray's center, we uniformly sample target positions at a radial distance of $[0.08\ , 0.12]$ m from the center and average over them to get the final metrics for each configuration. The xArm7 bases are mounted with a fixed separation of 1.4 m, providing sufficient dexterous workspace for coordinated tray manipulation.

Symbol	Description	Value
T_s	Simulation/MPC time step	0.002s
$\mathrm{n_x}$	State dimension	8
$\mathrm{n_u}$	Control dimension	2
N	Prediction horizon	20
$\mathbf{u}_\alpha^{\max}, \mathbf{u}_\beta^{\max}$	Max. roll/pitch command	0.6 rad
$\mathbf{Q_p}$	Pose error weight	250 $\mathbf{I_6}$
$\mathbf{Q_v}$	Velocity error weight	2 $\mathbf{I_6}$
$\mathbf{Q_R}$	Control effort weight	0.2 $\mathbf{I_4}$
$\boldsymbol{K}$	Cartesian stiffness (N/m)	$\mathrm{diag}(5000\ \mathbf{I}_3, 50\ \mathbf{I}_3)$
$\boldsymbol{K}_{\mathrm{null}}$	Joint stiffness (posture)	7 $\boldsymbol{I}_3$ N/m

TABLE I: Controller parameters used in all experiments.

B. Controller Parameters

The weight matrices used by the MPC in the high-level task, as well as those used by the low-level QP based impedance controller are listed in Table I. These are the default values used in all experiments in Sec. V unless mentioned otherwise.

The weight matrices $\mathbf{Q_p}$, $\mathbf{Q_v}$, $\mathbf{Q_N}$, and $\mathbf{Q_R}$ are tuned iteratively. Increasing $\mathbf{Q_p}$ places greater emphasis on accurate final positioning of the object, while higher $\mathbf{Q_v}$ helps dampen overshoot. $\mathbf{Q_N}$ enforces convergence to the desired state, and $\mathbf{Q_R}$ penalizes excessive torque changes, promoting smooth and feasible tray motions without saturating the torques.This tuning balances positioning accuracy with control effort.

C. Training Details for LMPC

The reinforcement learning policy was trained in simulation using MuJoCo, running three environments in parallel and collecting 500 episodes of 20,000 steps each—sufficient to observe convergence given an initial, arbitrary, non-zero parameter vector. We used PPO with a learning rate of 3×10^{-4}, value function coefficient $c_v = 0.5$, and entropy coefficient $\beta = 0.01$. Training covered a total of $500 \times 20{,}000$ steps (≈ 2 hours of wall-clock time) and was evaluated every 2048 iterations. Further, we have adopted domain randomization to randomly select any one of the 18 physical object configurations to improve the generalization capabilities of the RL agent.

D. Implementation Details

Both RMPC and LMPC are implemented in Python using the PyTorch framework. The nonlinear MPC and the optimization-based dual arm controller are solved using the Interior Point Optimization (IPOPT) solver in CasADi [33]. All simulations and training experiments were executed on a Ryzen 9 8945HS CPU and NVIDIA GeFORCE RTX 4090 GPU.

To ensure real–time performance during training and evaluation, the codebase was parallelized using Python's `multiprocessing` library.

Specifically, the CasADi solver for the MPC and the dual–arm controller and the MuJoCo simulation environment were each executed in separate processes, which communicate via a shared memory buffer in CPU memory. Synchronization is achieved using multiple mutex locks, with the MuJoCo solver's updates as a reference clock.

E. Evaluation Metrics

The data collected for evaluation is sampled at a frequency of 500Hz for a duration of 20 seconds. At each timestep $i \in [1, \mathrm{n}]$, the position error is defined as $\mathbf{e}_i = \mathbf{p}^{\mathrm{ref}} - \mathbf{p}_i$, with n the total number of steps. We assess the three MPC methods (PMPC, RMPC, LMPC) using three widely used performance measures [34], [35].

- **Steady State Error:** Measures the mean error over the last 4s (20 % of the samples) as all three methods reach steady state before this. It is computed as:

$$\hat{\mathrm{e}}_{\mathrm{ss}} = \frac{1}{0.2\mathrm{n}} \sum_{i=0.8\mathrm{n}+1}^{\mathrm{n}} \|\mathrm{e}_i\|^2 \tag{20}$$

- **Settling Time:** Measures the time taken to reach the steady state with a tolerance of 1%. It is defined as the smallest time index i_s such that

$$\|\mathrm{e}_i\| \leq \varepsilon, \quad \forall\, i \geq i_s \tag{21}$$

  where $\varepsilon = 1.01\|\hat{\mathrm{e}}_{\mathrm{ss}}\|$.
- **Control Effort:** Evaluates the efficiency and smoothness of the applied torques. We compute energy and rate costs from the joint torques $\boldsymbol{\tau}_{\mathrm{k}}$ of both xArm7s as

$$\mathrm{J}_{\mathrm{effort}} = \mathrm{w_E} \sum_{i=1}^{n} \|\boldsymbol{\tau}_i\|^2 \Delta t + \mathrm{w_R} \sum_{i=2}^{\mathrm{n}} \|\frac{\boldsymbol{\tau}_i - \boldsymbol{\tau}_{i\text{-}1}}{\Delta t}\|^2, \tag{22}$$

  where $\mathrm{w_E}$, $\mathrm{w_R}$ are the associated weights.

VI. RESULTS

We evaluate the performance of DART across a range of object configurations using the metrics defined in Sec. V-E.

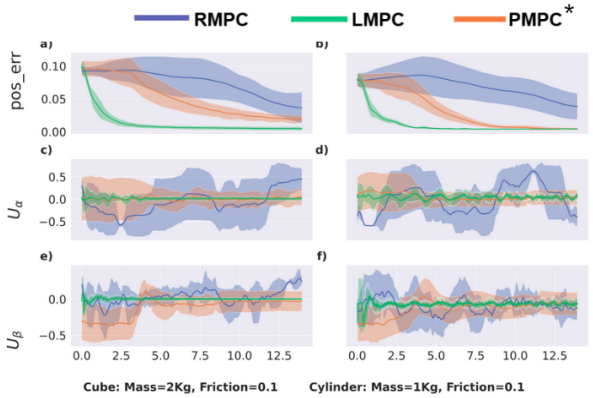

Fig. 4: Performance of two objects, Cube on the left and Cylinder on the right visualized as a standard deviation and average plot denoting the position error and tilt commands given by the LMPC RMPC and PMPC*. The color legend corresponding to the three methods shown on top

A. Quantitative Results

Table II summarizes the performance of all three controllers across all the object configurations described in Sec. V-A. We observe that LMPC has better settling time and and steady state error. However, it's control effort is relatively high. RMPC requires low control effort for low mass objects but it shows higher settling time. PMPC offers

Object	Mass(Kg)	Friction	Settling Time(s)				Steady State Error(m)				Control Effort			
			PMPC	RMPC	LMPC	PMPC*	PMPC	RMPC	LMPC	PMPC*	PMPC	RMPC	LMPC	PMPC*
Cube	1	0.05	6.70	17.14	**3.89**	6.20	0.060	0.017	**0.0066**	0.016	2174.35	**943.46**	2007.70	1180.78
		0.1	7.34	16.92	**3.47**	7.01	0.020	0.018	**0.0075**	0.015	2131.62	**949.57**	1885.27	1125.92
		0.2	7.43	17.02	**2.57**	6.44	0.020	0.026	**0.016**	0.016	2209.89	**1164.74**	1884.86	1206.55
	2	0.05	7.16	21.11	**4.85**	6.71	0.050	0.017	**0.0059**	0.014	2328.56	3873.42	**2085.31**	*1651.14*
		0.1	6.52	21.70	**4.91**	6.62	0.039	0.013	**0.0055**	0.014	**2192.75**	3906.57	2258.05	*1652.74*
		0.2	6.89	21.30	**3.86**	6.58	0.039	0.014	**0.0117**	0.014	2275.39	3941.35	**2182.49**	*1650.09*
Cylinder	1	0.05	6.24	16.53	**4.02**	5.83	0.060	0.025	**0.0030**	*0.0019*	1704.56	**992.04**	1780.91	1105.64
		0.1	5.92	16.60	**4.22**	5.16	0.031	0.021	**0.0026**	*0.0017*	1718.19	**893.93**	1782.94	1105.09
		0.2	4.97	18.08	**4.74**	5.47	**0.0019**	0.020	0.0044	*0.0016*	1172.72	**924.44**	2041.98	1172.72
	2	0.05	4.66	15.82	**3.22**	5.38	0.0315	0.013	**0.0044**	*0.0023*	**1569.18**	2925.26	2063.84	1569.18
		0.1	4.12	19.48	**2.93**	5.34	**0.0024**	0.014	0.0044	0.0025	**1582.98**	2855.69	1988.34	*1582.98*
		0.2	4.96	18.61	**4.12**	5.64	0.040	0.012	**0.0041**	*0.0024*	**1659.26**	2970.69	1926.88	1659.26
Sphere	1	0.05	5.39	15.58	**1.48**	4.27	**0.0052**	0.034	0.0166	*0.0051*	1817.05	**1102.22**	2007.70	1156.36
		0.1	4.27	13.74	**1.21**	3.84	**0.0047**	0.030	0.0207	0.0052	1651.87	**998.74**	1885.27	1139.02
		0.2	4.30	12.45	**1.39**	3.79	**0.0060**	0.030	0.0223	0.0060	1735.35	**1164.49**	1884.86	1245.72
	2	0.05	6.05	11.97	**1.34**	4.12	0.054	0.028	**0.0304**	*0.0044*	**2068.16**	3461.95	2085.31	*1607.79*
		0.1	4.65	10.83	**1.20**	4.55	**0.0054**	0.031	0.0301	*0.0048*	**2041.12**	3527.24	2258.05	*1673.52*
		0.2	5.76	12.80	**0.82**	4.25	**0.0137**	0.027	0.0262	*0.0058*	**2107.73**	3515.69	2182.49	*1677.63*

TABLE II: Performance comparison of different object geometries with varying mass and friction parameters, each configuration averaged over 20 radially sampled targets. The table evaluates the PMPC, RMPC, LMPC methods, initialized with the default MPC parameter(Table I), highlighting the best in bold. Further, PMPC* with the $*$ mark represents the performance of a higly fine-tuned Physics Based Dynamics MPC and the scenarios in which it out-performs the rest have been shown in italics.

better steady state error for rolling contacts and low control effort for high mass rolling contacts.

We further observe that RMPC and LMPC generalize well across objects, masses, and friction coefficients, which can be observed in their consistent settling times and steady state errors. PMPC cannot generalize and requires object-specific parameters to be provided. This is observed in Table II.

While LMPC outperforms the other two methods in terms of settling times, it requires training and computation V-C. RMPC adapts real time and does not require any training.

We also evaluate PMPC* which represents the PMPC method with fine tuned MPC control specific to each object configuration. We observe that PMPC* achieves the best performance in terms of steady state error. However, it can be noted that LMPC performs equally well in terms of steady state error.

Fig. 4 visualizes the above observations for 2 different object configurations, showing the trends in steady state error across RMPC, LMPC and PMPC*, and the respective commands produced by these methods. This portrays the trends observed across Table II. LMPC converges the fastest while PMPC* shows a smooth settling to the lowest steady state error for the Cube.

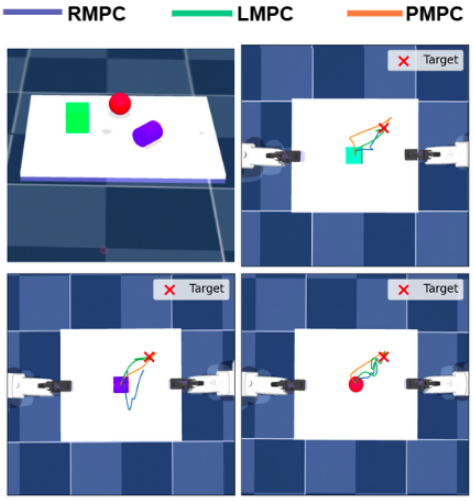

Fig. 5: **Visualizations of the trajectories:** This figure depicts the trajectories followed by a cube(green), sphere(red), and a cylinder(purple) with mass 1kg and friction 0.2 , manipulated by using 3 different MPC methods. The curves shows only the path of the object taken on the tray, longer path doesn't mean longer conververgence. The color legend corresponding to the three methods shown on top

B. Qualitative Results

We qualitatively evaluate our framework across diverse and challenging scenarios. Fig. 5 shows trajectories for varying object shapes, each presenting a different manipulation challenge due to different contact interactions. We chose a shorter goal to observe the trajectories given by different controllers to test the ability to react faster. The controller behaviours are clearly differentiated. PMPC takes long (Fig. 5) yet faster (Table II) trajectories which can be treated a trade-off of simplified contact dynamics model as Eq. 5. RMPC converges towards the target while adapting online; however, in some scenarios, it overshoots with a sub-optimal estimate of dynamics but manages to converge as it updates it parameters. As observed in the Fig. 5 cylinder shows some steady state error of 2-2.3 cm which is consistent with Table II. LMPC is always taking the shortest and fastest trajectories compared to the other two methods(Fig. 5) with good overall performance along with genaralization over different object configurations. These rollouts highlight how environment parameters strongly affect manipulation dynamics, underscoring the need for controllers that generalize across conditions rather than overfit to a single setup.

VII. CONCLUSION AND FUTURE WORK

This paper presents DART, a first framework for dual-arm non-prehensile tray manipulation. By integrating non-linear MPC with optimization-based impedance control, our approach enables object transportation with accuracy up to

2 mm on dynamically controlled trays, a capability critical for service robots. Our comparison of three dynamics modeling strategies reveal various strengths: physics-based models offer interpretability, regression-based models provide online adaptation, while reinforcement learning delivers superior generalization with faster convergence.

Through validation across objects with varying mass, geometry, and friction properties, we demonstrate that DART effectively navigates the complex, coupled dynamics between dual manipulators and sliding objects. Future work will extend this framework to handle surfaces with variable friction, uneven surfaces, complex objects, orientation control through active yawing, and non-planar object manipulation.

REFERENCES

[1] M. Tröbinger, C. Jähne, Z. Qu, J. Elsner, A. Reindl, S. Getz, T. Goll, B. Loinger, T. Loibl, C. Kugler *et al.*, "Introducing garmi-a service robotics platform to support the elderly at home: Design philosophy, system overview and first results," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5857–5864, 2021.

[2] W. Zhou, B. Jiang, F. Yang, C. Paxton, and D. Held, "Hacman: Learning hybrid actor-critic maps for 6d non-prehensile manipulation," *arXiv preprint arXiv:2305.03942*, 2023.

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

[4] A. Gutiérrez-Giles, F. Ruggiero, V. Lippiello, and B. Siciliano, "Closed-loop control of a nonprehensile manipulation system inspired by the pizza-peel mechanism," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 1580–1585.

[5] A. C. Satici, F. Ruggiero, V. Lippiello, and B. Siciliano, "A coordinate-free framework for robotic pizza tossing and catching," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 3932–3939.

[6] D. Serra, F. Ruggiero, A. Donaire, L. R. Buonocore, V. Lippiello, and B. Siciliano, "Control of nonprehensile planar rolling manipulation: A passivity-based approach," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 317–329, 2019.

[7] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, "More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 30–37.

[8] H. Raei, E. De Momi, and A. Ajoudani, "A reinforcement learning approach to non-prehensile manipulation through sliding," *IEEE Robotics and Automation Letters*, 2025.

[9] N. Jawale, B. Boots, B. Sundaralingam, and M. Bhardwaj, "Dynamic non-prehensile object transport via model-predictive reinforcement learning," 2024. [Online]. Available: https://arxiv.org/abs/2412.00086

[10] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation—a survey," *Robotics and Autonomous systems*, vol. 60, no. 10, pp. 1340–1353, 2012.

[11] A. Heins and A. P. Schoellig, "Keep it upright: Model predictive control for nonprehensile object transportation with obstacle avoidance on a mobile manipulator," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 7986–7993, 2023.

[12] M. Selvaggio, A. Garg, F. Ruggiero, G. Oriolo, and B. Siciliano, "Non-prehensile object transportation via model predictive non-sliding manipulation control," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 5, pp. 2231–2244, 2023.

[13] M. Grotz, M. Shridhar, T. Asfour, and D. Fox, "Peract2: Benchmarking and learning for robotic bimanual manipulation tasks," 2024. [Online]. Available: https://arxiv.org/abs/2407.00278

[14] F. Krebs and T. Asfour, "A bimanual manipulation taxonomy," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 031–11 038, 2022.

[15] G. Zhai, Y. Zheng, Z. Xu, X. Kong, Y. Liu, B. Busam, Y. Ren, N. Navab, and Z. Zhang, "Da ˆ{2} dataset: Toward dexterity-aware dual-arm grasping," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8941–8948, 2022.

[16] M. F. Karim, M. S. Hashmi, S. Bollimuntha, M. R. Tapeti, G. Singh, N. Govindan, and K. M. Krishna, "Dg16m: A large-scale dataset for dual-arm grasping with force-optimized grasps," *arXiv preprint arXiv:2503.08358*, 2025.

[17] B. Serhan, H. Pandya, A. Kucukyilmaz, and G. Neumann, "Push-to-see: Learning non-prehensile manipulation to enhance instance segmentation via deep q-learning," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1513–1519.

[18] J. Shi, J. Z. Woodruff, P. B. Umbanhowar, and K. M. Lynch, "Dynamic in-hand sliding manipulation," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 778–795, 2017.

[19] A. Edsinger and C. C. Kemp, "Two arms are better than one: A behavior based control system for assistive bimanual manipulation," in *Recent Progress in Robotics: Viable Robotic Service to Human: An Edition of the Selected Papers from the 13th International Conference on Advanced Robotics*. Springer, 2007, pp. 345–355.

[20] S. Pantanetti, F. Emiliani, D. Costa, G. Palmieri, and A. Bajrami, "From single to dual-arm collaborative robotic assembly: A case study at i-labs," in *2024 20th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, 2024, pp. 1–6.

[21] F. Caccavale, P. Chiacchio, A. Marino, and L. Villani, "Six-dof impedance control of dual-arm cooperative manipulators," *IEEE/ASME Transactions on Mechatronics*, vol. 13, no. 5, pp. 576–586, 2008.

[22] J. Lee, P. H. Chang, and R. S. Jamisola, "Relative impedance control for dual-arm robots performing asymmetric bimanual tasks," *IEEE transactions on industrial electronics*, vol. 61, no. 7, pp. 3786–3796, 2013.

[23] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[24] K. F. Gbagbe, M. A. Cabrera, A. Alabbas, O. Alyunes, A. Lykov, and D. Tsetserukou, "Bi-vla: Vision-language-action model-based system for bimanual robotic dexterous manipulations," *arXiv preprint arXiv:2405.06039*, 2024.

[25] Y. Chen, T. Wu, S. Wang, X. Feng, J. Jiang, Z. Lu, S. McAleer, H. Dong, S.-C. Zhu, and Y. Yang, "Towards human-level bimanual dexterous manipulation with reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5150–5163, 2022.

[26] M. F. Karim, S. Bollimuntha, M. S. Hashmi, A. Das, G. Singh, S. Sridhar, A. K. Singh, N. Govindan, and K. M. Krishna, "Davil: Adaptive dual-arm manipulation with reinforcement learning and variable impedance control," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 11 896–11 903.

[27] H. Hu and J. Cao, "Adaptive variable impedance control of dual-arm robots for slabstone installation," *ISA transactions*, vol. 128, pp. 397–408, 2022.

[28] J. van Steen, G. v. d. Brandt, N. v. d. Wouw, J. Kober, and A. Saccon, "Quadratic programming-based reference spreading control for dual-arm robotic manipulation with planned simultaneous impacts," *IEEE Transactions on Robotics*, vol. 40, pp. 3341–3355, 2024.

[29] R. Subburaman, M. Selvaggio, and F. Ruggiero, "A non-prehensile object transportation framework with adaptive tilting based on quadratic programming," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3581–3588, 2023.

[30] B. Armstrong-Helouvry, "Stick-slip arising from stribeck friction," in *Proceedings., IEEE International Conference on Robotics and Automation*, 1990, pp. 1377–1382 vol.2.

[31] T. Lequy and A. M. Menzel, "Stochastic motion under nonlinear friction representing shear thinning," *Physical Review E*, vol. 108, no. 6, p. 064606, 2023.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[33] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[34] K. Ogata, *Modern control engineering*. Prentice hall, 2010.

[35] S. Skogestad and I. Postlethwaite, *Multivariable feedback control: analysis and design*. john Wiley & sons, 2005.