A Reinforcement Learning Approach to Non-Prehensile Manipulation Through Sliding

Hamidreza Raei, Elena De Momi, *Senior Member, IEEE*, and Arash Ajoudani, *Member, IEEE*

***Abstract*—Although robotic applications increasingly demand versatile and dynamic object handling, most existing techniques are predominantly focused on grasp-based manipulation, limiting their applicability in non-prehensile tasks. To address this need, this study introduces a Deep Deterministic Policy Gradient (DDPG) reinforcement learning (RL) framework for efficient non-prehensile manipulation, specifically for sliding an object on a surface. The algorithm generates a linear trajectory by precisely controlling the acceleration of a robotic arm rigidly coupled to the horizontal surface, enabling the relative manipulation of an object as it slides along the surface. Furthermore, two distinct algorithms have been developed to estimate the frictional forces dynamically during the sliding process. These algorithms dynamically provide friction estimates online after each action, serving as critical feedback to the actor model. This feedback mechanism enhances the policy's adaptability and robustness, ensuring more precise control of the platform's acceleration in response to varying surface conditions. The proposed algorithm is validated through simulations and real-world experiments. Results demonstrate that the proposed framework effectively generalizes sliding manipulation across varying distances and, more importantly, adapts to different surfaces with diverse frictional properties. Notably, the trained model exhibits zero-shot sim-to-real transfer capabilities.**

***Index Terms*—In-hand manipulation, reinforcement learning (RL), transfer learning.**

I. Introduction

FOR years, advancements in robotic manipulation have focused on enhancing grasp dexterity and generalizing manipulation skills across a wide range of objects and tasks. These developments have significantly narrowed the gap between robotic manipulators and skilled humans, particularly in prehensile manipulation which is performed in slow and quasi-static operations. The ability to manipulate objects without grasping them is a decisive aspect of human dexterity. Despite the growing demand for involving robots in various applications where non-prehensile manipulation is the most effective way to perform the task, current algorithms developed for non-prehensile manipulation still fall short of achieving human-level performance [1]. This limitation primarily stems from their dependence on oversimplified models and approximated physical parameters, which compromise the robustness of these methods when dealing with uncertainties, variations in object properties, and dynamic environmental conditions [2]. Non-prehensile manipulation tasks are essential in various fields, such as logistics, cooking, and food delivery. Examples include sliding a pizza into an oven or flipping a burger in a pan.

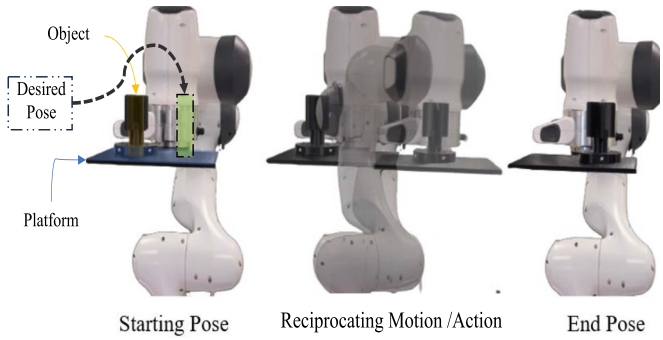

Fig. 1. Illustration of the non-prehensile manipulation explored in this study: A robotic arm sliding an object accurately through a reciprocating motion.

To address these limitations in non-prehensile manipulation, this study investigates precise sliding control of an object on a horizontal surface using a robotic arm. This approach differs from other studies on non-prehensile manipulation methods that focus on preventing objects from sliding across a horizontal surface during the transport [3], [4], as well as from previous work that uses external barriers to slide objects on the manipulator's palm [5]. Our study aims to achieve accurate sliding displacements by controlling surface acceleration and deceleration (see Fig. 1), similar to moving a glass on a tray, without requiring prior knowledge of the precise friction coefficient and only by maneuvering the tray. To perform this, an actor-critic reinforcement learning (RL) framework is implemented in a simulation environment. The RL framework generates sequences of actions in a continuous action space to create precise linear trajectories, achieving the desired sliding displacement. Once trained, the actor model is transferred to real-world setup to be evaluated.

Deploying simulation-trained policies to real-world robots is challenging due to the sim-to-real gap. Many simulators struggle to accurately capture material properties, deformable objects,

Received 27 January 2025; accepted 3 June 2025. Date of publication 9 June 2025; date of current version 20 June 2025. This article was recommended for publication by Associate Editor M. Liarokapis and Editor J. Borrs Sol upon evaluation of the reviewers' comments. This work was supported in part by the European Commission's Marie Skłodowska-Curie Actions (MSCA) Project RAICAM under Grant 101072634 and in part byHorizon Europe TORNADO under Grant 101189557. *(Corresponding author: Hamidreza Raei.)*

Hamidreza Raei is with Human-Robot Interfaces and Interaction Lab, Istituto Italiano di Tecnologia, 16163 Genoa, Italy, and also with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, 20133 Milan, Italy (e-mail: hamidreza.raei@iit.it).

Elena De Momi is with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, 20133 Milan, Italy (e-mail: elena.demomi@polimi.it).

Arash Ajoudani is with Human-Robot Interfaces and Interaction Lab, Istituto Italiano di Tecnologia, 16163 Genoa, Italy.

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3578238, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3578238

and detailed robot-environment interactions, making it essential to choose an environment that balances computational efficiency with high-fidelity physics [6]. Moreover, critical variables like surface roughness and contact mechanics are often oversimplified, which can adversely affect real-world performance [7].

In non-prehensile manipulation, dynamic friction is crucial, while grasping mainly depends on static friction. Yet, most simulators (MuJoCo, Bullet, ODE, Vortex, PhysX) rely on the Coulomb friction model, which does not accurately capture dynamic friction [8], [9], [10]. To address this known inaccuracy, a common approach is domain randomization, which varies environment dynamics to expose the RL agent to diverse scenarios during the training [11], [12]. Additionally, we further enhance robustness by estimating and incorporating real-time friction feedback into the control algorithm. In summary, the contributions of this work can be listed as follows:

- Training an actor-critic RL model which is robust to surface friction and can slide an object on a surface by generating linear trajectories.
- Training a long short-term memory (LSTM) model to infer surface friction between two surfaces from time-series kinematic data.
- Developing an analytical friction inference approach using kinematic data to robustly estimate friction.
- Assessing the effect of domain randomization and friction inference methods on the performance of zero-shot sim-to-real transfer.

II. Related Works

1) Non-Prehensile Manipulation

Non-prehensile manipulation is the ability to manipulate objects without grasping and is crucial for tasks involving hard-to-grasp objects or cluttered workspaces [1], [13]. This type of manipulation encompasses a range of techniques such as throwing [14], pushing [2], rolling [15], and sliding [5], [16], among others. Several studies have explored the non-prehensile manipulation of an object on horizontal surfaces [3], [5], [17]. One of these studies [3] primarily focused on transporting an object on a horizontal surface while ensuring no sliding or slippage occurs during the task. In contrast, our work deliberately leverages sliding as the primary method for moving the object across the surface. Another study [5] investigates a control approach for sliding an object on a surface, demonstrating impressive controllability and performance. However, it assumes the presence of external barriers to assist with the sliding task and incorporates precise knowledge of the surface friction coefficient (μ) into its model-based controller. Although effective, this inherently limits the approach's ability to generalize to scenarios without external constraints or unknown friction properties. Another study in this domain [17] avoided external barriers, employing repetitive sliding maneuvers to reposition the object to its desired pose. However, due to a lack of accurate friction knowledge, very conservative solutions were implemented that relied on short sliding increments and numerous corrective moves, and the control policy lacked real-time friction adaptation.

2) RL for Continuous Action Space

Reinforcement learning algorithms have long been studied and applied in simplified environments, such as the Atari games discussed in [18]. In these settings, they have achieved notable success, often surpassing human experts in performance. However, these environments typically rely on discrete action spaces, whereas most real-world control problems require continuous action spaces. To address this, several RL algorithms have been developed to operate in continuous action spaces. Among the most prominent are actor-critic models which have been frequently utilized for non-prehensile manipulation. Notably, in a recent study, Proximal Policy Optimization (PPO), and Soft Actor-Critic (SAC) are implemented to perform pushing through non-prehensile manipulation [19]. Several other studies have explored the development of Twin Delayed Deep Deterministic Policy Gradient (TD3) or similar algorithms for non-prehensile manipulation, such as throwing [20], and 6D object pose alignment [1]. These findings underscore the value of actor–critic RL methods, particularly in achieving effective sim-to-real transfer.

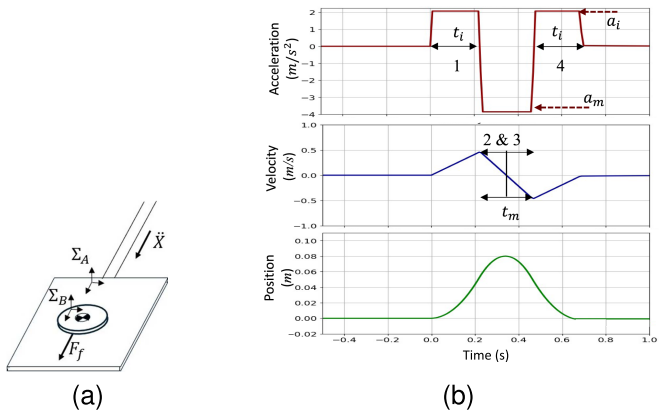

Fig. 2. (a) Simple schematic of the platform and sliding object with body-fixed frames. (b) Four phases of reciprocating motion in terms of platform acceleration, velocity, and position.

III. Methodology

In this study, we utilize an iterative framework for sliding where the agent generates sequential actions. Following each action, if the cumulative displacement has not reached the target accuracy, a subsequent action is generated. This decision uses an updated state incorporating the feedback, including remaining distance (D_{des}), previous actions, and friction estimate (μ_e), to generate the next action. Iteration continues until the cumulative displacement meets the accuracy requirement. For this purpose, we present the RL framework, providing details about the simulation environment and the architecture of the actor and critic models. Finally, we introduce two distinct online algorithms for friction estimation: an analytical approach and a data-driven LSTM approach, both designed to infer friction based on the action given to the manipulator and the kinematic data of relative motion between the object and the surface.

A. Problem Statement and Assumptions

We propose a strategy for sliding an object B on a surface A via a linear maneuver, with relevant parameters illustrated in Fig. 2(a). Sliding begins when the inertial force, induced by the surface's linear acceleration ($\ddot{X}$) exceeds the maximum static frictional force as shown in the following:

$$F_{f_{max}} = \mu_s\, mg \qquad |\ddot{\mathbf{X}}| > \mu_s g. \tag{1}$$

Once this condition is satisfied, the relative acceleration for the object with respect to the surface is illustrated in the following:

$$\ddot{x}_A^B = -\ddot{\mathbf{X}} - \frac{\dot{x}_A^B}{|\dot{x}_A^B|}\mu_k g. \tag{2}$$

To simplify the problem of the sliding object, assumptions are made regarding the input command $\ddot{\mathbf{X}}$ which defines the linear acceleration of the supporting surface. The surface starts from a stationary position, accelerates, decelerates, and returns to its initial resting pose. This reciprocating motion defines an action in the DDPG RL framework. The relative displacement of object B with regard to the surface A is given by

$$x_A^B = \begin{cases} 0 & |\ddot{X}| \le \mu_s g \\ -\iint \ddot{X}\, dt - \iint \frac{|\dot{x}_A^B|}{\dot{x}_A^B}\mu_k\, g\, dt & |\ddot{X}| > \mu_s g \end{cases}. \tag{3}$$

B. Generated Action Into Linear Trajectory

The action generated by the RL agent comprises three parameters, which are two accelerations and one time duration defined in Fig. 2(b) denoted as (a_i, a_m, t_m). To elaborate the maneuver using these parameters, we need to write down the velocity of the surface as a function of time derived from this command, which is given by

$$\dot{x}_A(t) = \begin{cases} a_i\, t & 0 \le t \le t_i \\ a_i\, t_i + a_m(t - t_i) & t_i \le t \le t_i + \frac{t_m}{2} \\ a_i\, t_i + a_m \frac{t_m}{2} + a_m(t - t_i - \frac{t_m}{2}) & t_i + \frac{t_m}{2} \le t \le t_i + t_m \\ a_i\, t_i + a_m\, t_m + a_i\, t & t_i + t_m \le t \le 2t_i + t_m \end{cases}. \tag{4}$$

The maneuver is divided into 4 phases. Transitions between phases occur upon changes in the commanded acceleration or in the direction of motion of the platform, as illustrated in Fig. 2(b). At $t = t_i + \frac{t_m}{2}$ the platform changes its direction of motion and that means $\dot{x}_A(t) = 0$. Although t_i does not explicitly appear among the generated action parameters, it can be calculated as follows:

$$t_i = \frac{t_m}{2}\left|\frac{a_m}{a_i}\right|. \tag{5}$$

Furthermore, the maximum motion range required for an action, occurring at $t = t_i + \frac{t_m}{2}$, is obtained by integrating (4):

$$\int_0^{t_i + \frac{t_m}{2}} \dot{x}_A(t)\, dt = \frac{a_i {t_i}^2}{2} + \frac{a_m\, t_m^2}{8}. \tag{6}$$

The formulation presented for t_i shown in (5), ensures that the robot begins and concludes the maneuver at the same pose with $\dot{x}_A = 0$, adhering to the initial assumption discussed in Section III-A. The calculated $\dot{x_A}$ will be commanded to the franka cartesian velocity controller to be executed.

C. Simulation Environment and Physics Engine

Several physics simulation engines are commonly used in robotics, but in RL, both simulation speed and physics accuracy are critical [21]. One of those simulation engines namely **Mu**lti-**Jo**int dynamics with **Co**ntact (MuJoCo) outperforms alternatives such as ODE (used in Gazebo), PhysX (used in Isaac Sim), and Bullet, particularly in terms of speed and accuracy for constraint-based systems [21]. MuJoCo's design leverages generalized coordinates for efficient and accurate contact dynamics along with contact solvers such as implicit nonlinear complementarity, whereas the other engines rely on Linear Complementarity Problem (LCP) solvers and cartesian representations. This can lead to numerical instability and reduced precision in fine-grained contact tasks [10]. These findings motivated our choice of MuJoCo as the simulation physics engine for modeling the surface contacts essential for sliding manipulation.

D. Reinforcement Learning Framework

The choice of learning algorithm, capable of generating continuous actions is another important factor for RL. Off-policy algorithms such as DDPG, TD3 and SAC benefit from a replay buffer which allows training on any form of past experience without requiring constant interaction with the simulation environment, which results in faster learning from previous interactions [22]. Among off-policy algorithms, considering the curriculum training and reward shaping strategies detailed in Section III-D3, and given evidence that DDPG achieves faster early convergence compared to TD3 and SAC [23], we employ DDPG in our approach. The aforementioned training strategies are expected to mitigate the overestimation bias commonly observed in DDPG [24], [25], thereby enhancing learning stability. In this method, the objective function is defined as following:

$$J(\theta^\pi) = \mathbb{E}_{s\sim\mathscr{D}}\left[Q(s, \pi(s|\theta^\pi)|\theta^Q)\right]. \tag{7}$$

The actor network, represented as $\pi(s|\theta^\pi)$, is updated by applying the gradient of the objective function $J(\theta^\pi)$ with regard to its actor network parameters (θ^π), shown below:

$$\nabla_{\theta^\pi} J(\theta^\pi) = \mathbb{E}_{s\sim\mathscr{D}}\left[\nabla_a Q(s, a|\theta^Q)\nabla_{\theta^\pi}\pi(s|\theta^\pi)\right]. \tag{8}$$

This update involves two components: the gradient of the critic $Q(s, a|\theta^Q)$ with respect to the action a, evaluated at the policy output $a = \pi(s|\theta^\pi)$, and the gradient of the policy's output $\pi(s|\theta^\pi)$ with respect to its parameters θ^π. The critic $Q(s, a|\theta^Q)$ learns to approximate the expected cumulative reward using the Bellman equation given by

$$Q(s, a) = r(s, a) + \gamma\mathbb{E}_{s'}\left[Q(s', \pi(s'))\right]. \tag{9}$$

This defines the relationship between immediate and future rewards. Together, these components ensure the policy is improved to maximize the expected return. It is worth mentioning that this framework is designed for zero-shot sim-to-real transfer of the learned policy. Below, we detail the components of this RL framework.

1) Data Structure: We are using a symmetric implementation of the actor-critic model, where both have access to the same set of states from the environment. The state for this structure includes desired remaining distance for displacement denoted as D_{des}, up to three previous actions, $(A_{i-1}, A_{i-2}, A_{i-3})$ and the relative displacement caused by each action $(D_{i-1}, D_{i-2}, D_{i-3})$, and an estimation of the friction coefficient shown as μ_e. In simulation, all states are accurately available, whereas in real-world implementation, relative position is tracked via a motion capture system, and friction is estimated through extensive experimentation.

2) Action Space: At each step, the learned policy generates a three-dimensional action that defines the trajectory using three

parameters a_i (initial acceleration), a_m (maximum acceleration), and t_m (time duration). This action turns into a linear trajectory as stated in Section III-B. The generated action is constrained by acceleration and motion range limits to comply with the robotic arm's physical capabilities, as stated in the following:

$$A = \left\{ \begin{bmatrix} a_i \\ a_m \\ t_m \end{bmatrix} \;\middle|\; |a_i|, |a_m| \leq 4.2\,\text{m/s}^2, \quad t_m < 2.0\,\text{s}, \; a_i\, a_m \leq 0 \right\}$$

3) Reward Function and Training: The sliding task is formulated under the assumption of precise knowledge of the friction coefficient within the Coulomb friction model, enabling the derivation of an optimal solution. To formulate the analytical solution, we need to expand (3) for all the phases as detailed in the following.

$$\ddot{x}_A^B(t) = \begin{cases} a_i - \mu_k\, g & 0 \leq t \leq t_i \\ a_m - \frac{|\dot{x}_A^B(t)|\mu_k\, g}{\dot{x}_A^B(t)} & t_i < t \leq t_i + \frac{t_m}{2} \\ a_m - \frac{|\dot{x}_A^B(t)|\mu_k\, g}{\dot{x}_A^B(t)} & t_i + \frac{t_m}{2} < t \leq t_i + t_m \\ a_i - \frac{|\dot{x}_A^B(t)|\mu_k\, g}{\dot{x}_A^B(t)} & t_i + t_m < t \leq 2t_i + t_m \end{cases} . \quad (10)$$

Integrating (10) twice under the boundary conditions yields the phasewise displacements given by

$$\Delta\, x_A^B = \begin{cases} \frac{(\ddot{x}_A^B(t))}{2}\, {t_i}^2 & |a_i| > \mu_s\, g \\ \dot{x}_A^B(t_i)\, \frac{t_m}{2} + (\ddot{x}_A^B(t))\, \frac{t_m^2}{8} & |a_m| > \mu_s\, g \\ \dot{x}_A^B(t_i + \frac{t_m}{2})\, \frac{t_m}{2} + (\ddot{x}_A^B(t))\, \frac{t_m^2}{8} & |a_m| > \mu_s\, g \\ \dot{x}_A^B(t_i + t_m)\, t_i + \frac{(\ddot{x}_A^B(t))}{2}\, {t_i}^2 & |a_m| > \mu_s\, g \end{cases} . \quad (11)$$

We address the Coulomb model's limitations and accelerate training by employing a progressive reward shaping [26] as the agent follows a curriculum training path. The stages in the training are detailed in the following:

- *Behavioral cloning:* The agent learns to mimic the action parameters (a_i, a_m, t_m) derived from the analytical optimal solution (11) for a fixed displacement distance and friction coefficient. The reward function transition to the performance-based occurs when the generated action parameters consistently remain within predefined tolerance thresholds (± 0.1 m/s^2 for accelerations and ± 0.1 s for time duration) relative to the analytical solution.
- *Sliding distance generalization:* Operating under the fixed friction coefficient from previous stage, the actor network is trained using a performance-based reward function, which incorporates sliding accuracy, the platform range-of-motion, required time, and number of steps. The objective is to achieve accurate sliding across various desired displacements ($0.03 < D_{des} < 0.2$). Transition to the next stage occurs once the agent consistently achieves displacement errors below 1 cm across the various desired displacements.
- *Friction Generalization:* The true environment friction coefficient (μ_k) is varied per episode, sampled from a specified range ($0.05 < \mu_k < 0.4$). The agent learns to adapt its actions by utilizing the estimated friction coefficient (μ_e) provided within its state input. Crucially, during this specific training stage, μ_e is set equal to μ_k . The objective is to maintain high sliding accuracy, targeting displacement errors below 1 cm, across the full range of tested distances and friction conditions.

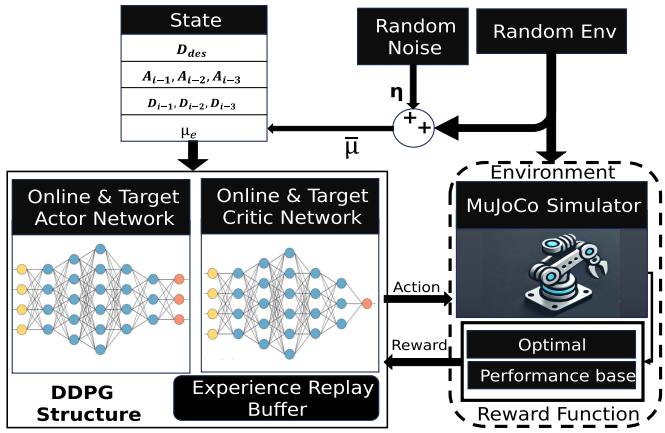

Fig. 3. This figure illustrates the states, the environment random initialization and domain randomization on friction parameter, in the DDPG training framework.

4) Domain Randomization: Upon completing the training phase, where the model successfully generalizes its actions across varying distances and friction coefficients (μ_k), domain randomization is introduced to enhance robustness and sim-to-real transferability [12]. For this problem, friction is identified as the most critical parameter influencing the sliding task. By introducing variations between μ_e and μ_k, the goal is to enable the agent to handle uncertainty in frictional conditions and prepare it for deployment in real-world environments.

To achieve this, we added a noise in two phases to the friction coefficient, provided in the state input to the actor and the critic network as shown below:

$$\bar{\mu} = \mu \pm \eta \quad (12)$$

In the first phase, $\eta \in [0.05, 0.1]$ and in the second stage $\eta \in [0.05, 0.15]$. These ranges were determined experimentally by comparing the agent's performance before and after applying domain randomization, following completion of the **friction generalization** stage. As noted in [27], excessive randomization can result in overly conservative policies, a trend confirmed by our comparisons. For this reason, the range was adjusted incrementally. Fig. 3 shows the training structure for the DDPG RL framework.

E. Friction Inference Algorithms

To provide crucial feedback for subsequent decision-making within our iterative control strategy, this section introduces two online algorithms designed to estimate the kinetic friction coefficient (μ_e). These estimations are performed following the completion of an action, utilizing the kinematic and command data collected during that specific action.

1) Analytical Approach: In this approach, the relative equations of motion stated in (10) and (11) are utilized to solve μ_k for the first three phases of the maneuver, given the known x_A^B from the simulation data or motion capture system in real

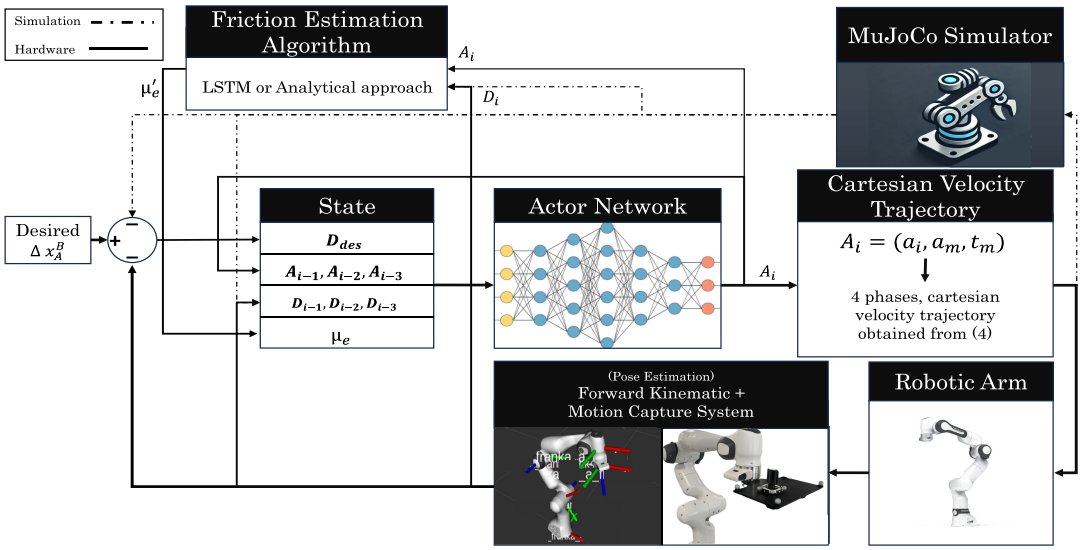

Fig. 4. A schematic of the evaluation framework. Connections between components for real-world experiments (solid lines) and simulations (dashed lines) are shown. The key difference lies in pose estimation: real-world setups rely on motion capture systems and forward kinematics to determine the poses of the end-effector and the object, while simulations provide these poses directly.

Algorithm 1: Analytical Friction Estimation.

Input: Relative displacement $x_A^B(t)$ (measured via OptiTrack or simulation), generated action (a_i, a_m, t_i, t_m)
Output: Friction coefficient μ_e
if $x_A^B(t = t_i) > 1.0\,\text{cm}$ **then**
  $\mu_e \leftarrow$ **Solve** $[\frac{1}{2}(a_i - \mu_k\, g)\, t^2 = x_A^B(t_i)]$ for μ_k
else if $x_A^B(t_i + t_m) - x_A^B(t_i) > 1.0\,\text{cm}$ **then**
  $\mu_e \leftarrow$ **Solve** $[\dot{x}_A^B(t_i)\, t_m + \frac{(a_m - \mu_k\, g)\, t_m^2}{2} = x_A^B(t_i + t_m) - x_A^B(t_i)]$ for μ_k
else
  $\mu_e \leftarrow 1.1\, \frac{a_m}{g}$
end if

world experiments. With the generated action provided, the only unknown parameter is the friction coefficient, μ_k, as detailed in the following.

2) Data-Driven LSTM Approach: In this approach, the training data for the LSTM were generated from the simulation environment. The input to the LSTM consists of a time series of commanded accelerations applied to the surface and the relative velocity between the object and the surface for a duration of 2.0 s. The target output is the friction coefficient corresponding to the simulation environment. The network consists of four recurrent layers with 1024, 512, 256, and 128 neurons, each utilizing kernels, recurrent kernels, and biases to process inputs and capture temporal dependencies. Fully connected layers follow, progressively reducing dimensionality, with the final output layer predicting a single friction coefficient value.

F. Evaluation Setup

The DDPG framework is evaluated in both simulation and real-world experiments, as shown in Fig. 4. Although in the simulation setup, all required parameters can be obtained through the simulation itself, for the real-world implementation, parameters related to the motion of the surface are obtained through forward kinematics of the Franka arm and the parameters regarding the relative motion of the object on the surface are obtained through motion capture system. The evaluation setup is capable of including or excluding the friction estimation algorithms, and its effect is examined in the following section.

IV. Experiment and Results

The experiment in this study consists of two main phases to comprehensively evaluate the proposed framework. In the first phase, simulations are conducted to assess the actor network's ability to generalize the sliding task across varying environmental parameters, such as displacement distances, surface friction, and inaccuracies in estimated friction coefficients. This simulation phase benchmarks the RL model's performance against analytical solutions, highlighting its adaptability under uncertainties. Additionally, it evaluates the friction estimation algorithms, and their effect on the overall proposed framework. The second phase, evaluates the trained model zero-shot sim-to-real transfer capabilities and robustness to friction estimation error. This phase involves testing the model on hardware as shown in Fig. 1 without additional fine-tuning.

A. Simulation Evaluation

This experiment aims to evaluate how effectively the actor network performs a sliding task over a fixed distance when exposed to varying friction coefficients. The friction coefficients are provided as inputs to the network's state (shown in Fig. 4). The evaluation focuses on analyzing the error in the initial action generated by the actor network and its dependency on the provided friction coefficient. Additionally, it examines the network's ability to generalize across diverse friction conditions.

To benchmark its performance, the DDPG model is compared to an analytical approach that provides robust control for the

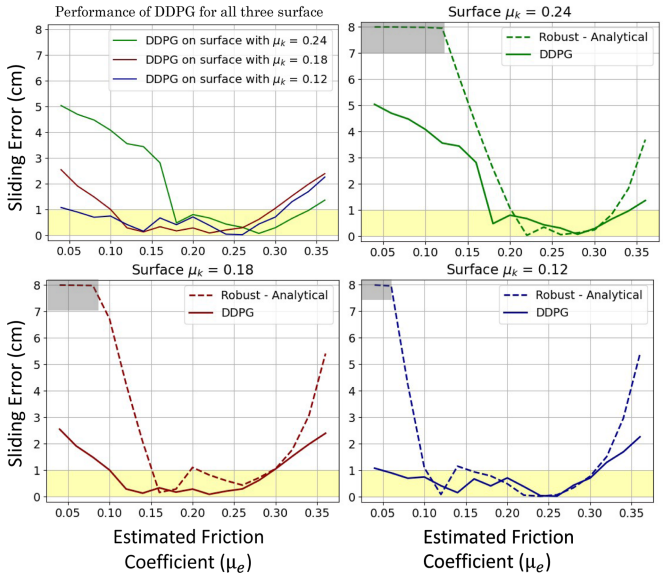

Fig. 5. This figure illustrates simulation performance of the actor network in sliding displacement under friction coefficient errors, compared to the robust analytical baseline solution. Both the actor network and the analytical solution receive μ_e as input to determine their actions. Yellow highlights show the acceptable margin for errors, while gray highlights denote complete action failure to move the object. This experiment is performed on three different surfaces, where μ_k represents the friction coefficient of each surface, and μ_e is the estimation given as input to both the actor network's state and the analytical model for this comparison.

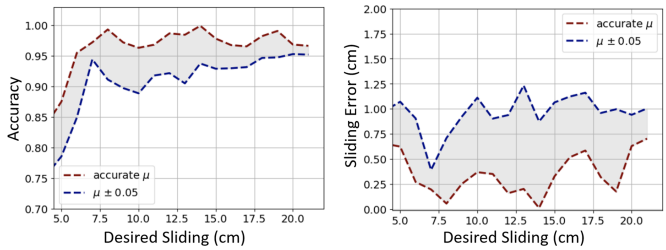

Fig. 6. The simulation performance of the actor model across different desired sliding distances (D_{des}), evaluated under conditions of accurate and perturbed friction coefficient estimation.

sliding task, as defined by (11). The experiment is conducted across three surfaces with distinct friction coefficients (μ_k), where the friction values provided to the network's state range between $\mu_e = (0.04, 0.32)$. Fig. 5 highlights the comparative performance between the DDPG model and the analytical approach. The actor trained within the DDPG framework demonstrates robust performance, showing significantly reduced sensitivity to variations in friction coefficients compared to the analytical optimal solution. Additionally, the results indicate that across all three surfaces, for $D_{des} = 8\,cm$, the actor network is robust to friction estimation errors as long as the discrepancy between μ_k and μ_e does not exceed 0.05. Notably, the actor avoids situations where the action fails to slide completely.

The second experiment evaluates the model's ability to generalize across various desired displacements (D_{des}) in a simulation environment with a friction coefficient of $\mu_k = 0.24$. As shown in Fig. 6, the results highlight the model's accuracy ($1 - \frac{|\Delta x_A^B - D_{des}|}{|D_{des}|}$) and the absolute error ($|\Delta x_A^B - D_{des}|$) for both accurate ($\mu_e = \mu_k$) and slightly deviated ($\mu_e = \mu_k \pm 0.05$) friction estimates. The model demonstrates high accuracy, with errors of $|\Delta x_A^B - D_{des}| \leq 0.75\,cm$ for accurate μ_e and $|\Delta x_A^B - D_{des}| \leq 1.25\,cm$ for slightly deviated μ_e. The sliding accuracy for accurate and slightly deviated μ_e has respectively remained above 85% and 76%. It is important to note that all results presented in Figs. 5 and 6 focus exclusively on evaluating the performance of the actor network during its first reciprocating motion or action.

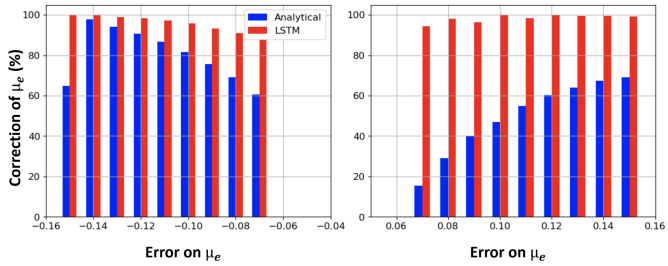

Fig. 7. Indicates the simulation results on how each friction estimation algorithm reduces the error in the estimated friction coefficient (μ_e), across a range of errors $(\mu_e - \mu_k) \in [-0.15, 0.15]$. The y axis shows the percentage improvement in the accuracy of the estimated friction coefficient for each model (LSTM and Analytical), where higher values indicate better performance.

Finally, the friction estimation algorithms are assessed for their accuracy and influence on the overall performance of the evaluation framework. The assessment begins with the DDPG actor executing a sliding task with a desired displacement of $D_{des} = 8\,cm$ in an environment characterized by $\mu_k = 0.24$. Following this, the kinematic data of the relative motion between the object and the surface, along with the command data, are provided to each algorithm according to Section III-E. The friction coefficient corrections of the LSTM and the analytical approach are calculated as the ratio of the final error ($|\mu'_e - \mu_k|$) to the initial error ($|\mu_e - \mu_k|$), as shown below:

$$\text{Correction of } \mu_e = \left(1 - \frac{|\mu_k - \mu'_e|}{|\mu_k - \mu_e|}\right) \times 100\,. \qquad (13)$$

where μ_k is the true friction, μ_e is the initial estimate given to the agent, and μ'_e is the estimate produced by the algorithms after the action. Fig. 7 plots this correction against the initial estimation error $(\mu_e - \mu_k)$ for both algorithms across the range of errors tested, specifically for $(\mu_e - \mu_k) \in [-0.15, 0.15]$.

In simulation, the analytical and LSTM methods achieved overall correction accuracies for μ_e based on (13) of 64.5% and 95.6%, respectively. Experiments across various simulated surfaces ($\mu_k \in [0.2, 0.4]$) confirmed that this key trend persisted: performance was comparable when friction was underestimated ($\mu_e < \mu_k$), whereas the LSTM consistently outperformed the analytical method during overestimation ($\mu_e > \mu_k$). Furthermore, Fig. 8 examines two scenarios of significant gap between μ_e and μ_k, performed by the robust analytical solution and the DDPG actor. The results indicate that the DDPG actor adapts better than the analytical solution in sequence of actions by learning from its past experience and adding online friction estimation yields further improvements. Both estimation methods reduce the number of actions for task completion and enhance the accuracy of subsequent actions compared to the standalone DDPG actor.

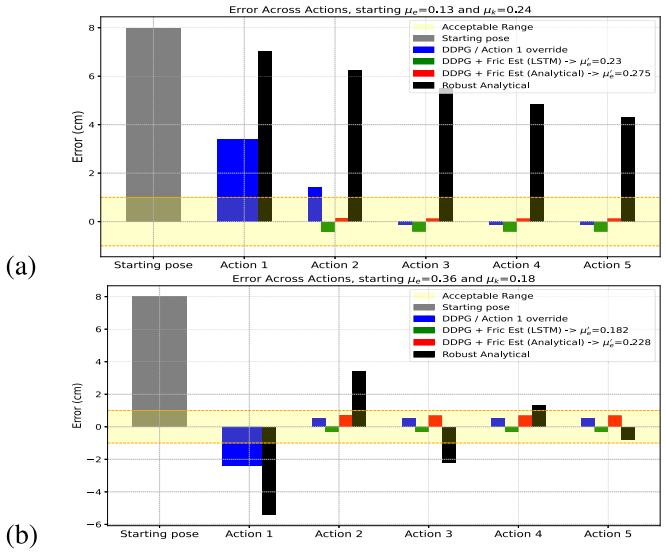

Fig. 8. Comparison of sequential actions performance between the analytical solution and the DDPG actor in simulation under significant gap between μ_e and μ_k. Performance shown with and without online friction estimation updates. (a) Underestimation case ($\mu_e = 0.13, \mu_k = 0.24$). (b) Overestimation case ($\mu_e = 0.36, \mu_k = 0.18$).

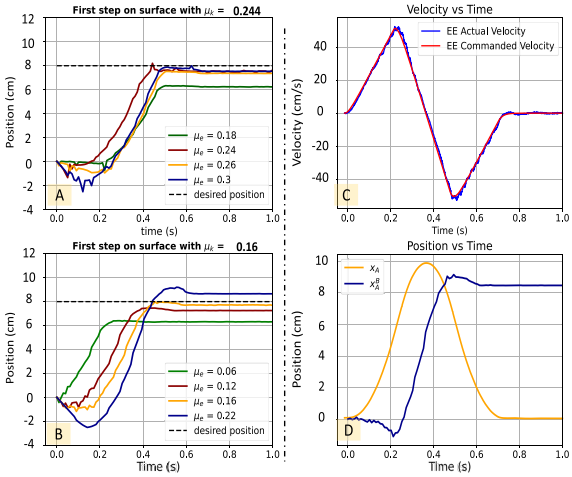

Fig. 9. **A and B:** show the actor network performance on two surfaces with different friction properties, respectively $\mu_k = 0.244$ and $\mu_k = 0.16$ to slide an object for 8 cm when various μ_e are given as state to the network. **C:** illustrates the commanded and actual velocity of the end-effector (surface) during a single action and **D:** indicates the platform displacement and relative displacement of the object on the surface within the same action as C.

B. Zero-Shot Sim-to-Real Transfer Evaluation

Initially, we test the trained model, referred to as the target actor network, within the DDPG RL framework (see Fig. 3) on a real-world setup designed for sliding an object on a horizontal surface. We conduct evaluations on two different surfaces, each characterized by dynamic friction coefficients of $\mu_k = 0.244$ and $\mu_k = 0.16$. To evaluate the model's adaptability and precision, we supply it with four different estimated friction coefficients (μ_e) to determine its capability to effectively perform the sliding task in a single step regardless of the accuracy of the provided (μ_e). Results of this experiment are illustrated in Fig. 9(A) and

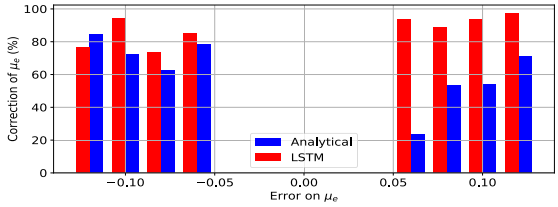

Fig. 10. Illustrates how each friction estimation algorithm reduces the error in the estimated friction coefficient in the real-world experiment. The y axis shows the correction accuracy of the estimated friction coefficient for each model (LSTM and Analytical), and the x axis shows the error between actual and estimated friction coefficient.

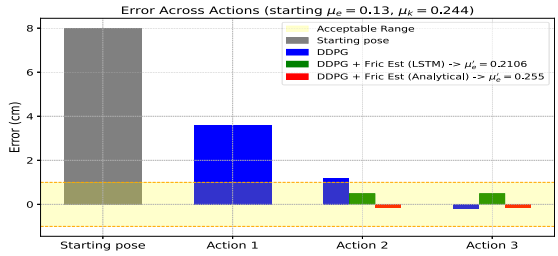

Fig. 11. Comparison of the impact of friction estimation algorithms (analytical and LSTM) on task performance when the actor network starts with significant friction estimation error in the real-world experiment.

Fig. 9(B). With a desired relative displacement of 8 cm, the results show that an accurate estimate of μ_e keeps the displacement within the acceptable error range ($|\Delta x_A^B - D_{des}| \leq 1\,cm$). Furthermore, even a slightly deviated μ_e ($\mu_e = \mu_k \pm 0.05$) still yields only a minor error—remaining below 1 cm—for the same D_{des} in the first action.

To clearly illustrate how the actor network achieves the relative displacement in a single action in the real-world setup, Fig. 9(C) presents the commanded velocity—derived from the generated action—accompanied by the end-effector's velocity tracking. Meanwhile, the displacement of the end-effector surface (x_A) and the resulting relative displacement of the object (x_A^B) during the one-step maneuver are depicted in Fig. 9(D). As shown in Fig. 9(C) and (D), the predefined conditions for the maneuver, stated in III-A, are satisfied.

Next, we seek to evaluate the accuracy of friction estimation algorithms on a real-world setup, an equivalent of the experiment shown in Fig. 7 for the simulation setup. The assessment is performed in an environment characterized by $\mu_k = 0.16$. The kinematics and command data were collected for the estimation algorithm after a single action for $D_{des} = 8\,cm$. The correction accuracy across the following error range $(\mu_e - \mu_k) \in [-0.12, 0.12]$ is calculated by (13) and depicted in Fig. 10. The results indicate, that when ($\mu_e < \mu_k$) the analytical and LSTM approach have comparable results, however, when ($\mu_e > \mu_k$), the LSTM outperforms the analytical approach in friction coefficient estimation. On average the analytical and the LSTM approach respectively achieved a correction accuracy of 62.16% and 87.2%.

Additionally, in the real-world setup, we examined a scenario where the environment had a friction coefficient of $\mu_k = 0.244$, while an estimated value of $\mu_e = 0.13$ was provided as the initial state input to the actor network. Equivalent to the experiment shown in Fig. 8 for simulation setup. The friction estimation

algorithms adjusted μ_e from 0.13 to $\mu_e = 0.2106$ using the LSTM approach and $\mu_e = 0.255$ using the analytical approach. The results shown in Fig. 11, demonstrate that while the DDPG actor is capable of performing the task despite discrepancies between μ_e and μ_k, integrating friction estimation algorithms reduces the number of required actions and improves sliding accuracy in subsequent actions. A video recording showcasing part of the real-world experiment is available at: https://youtu.be/HbvEpeCo5DU.

V. Discussion and Conclusion

The results of this study highlight the efficacy of the proposed DDPG RL framework in addressing the challenges of non-prehensile manipulation through sliding. The actor is capable of sliding an object and adapting its generated action to its experience from previous actions. These capabilities were validated in both simulation and real-world experiments. The zero-shot sim-to-real transfer capability underscores the model's adaptability, and the effectiveness of domain randomization. Additionally, by integrating friction estimation algorithms, the framework demonstrated robust performance in both simulation and real-world environments for iterative corrective action, as presented respectively in Figs. 8 and 11.

Furthermore, the LSTM friction estimation, trained solely on simulation data, achieved notable real-world accuracy (87.2% average). Additionally, it outperformed the analytical method when friction was initially overestimated. Such overestimation typically results in stronger commanded accelerations and consequently higher relative velocities between the object and surface. In these higher-velocity regimes, the LSTM's ability to capture complex temporal dynamics, learned from the simulation, may offer an advantage over the analytical approach, which relies on assumptions inherent in simpler friction models. The LSTM approach's performance was even higher in simulation (95.6% accuracy). The performance gap between simulation and real-world deployment suggests that the real-world accuracy of this data-driven approach could be enhanced further by incorporating data gathered from physical sliding interactions and fine-tuning in the real-world setup.

In conclusion, this study demonstrates that a DDPG RL framework integrated with online friction estimation can achieve robust zero-shot sim-to-real transfer for non-prehensile manipulation tasks, laying a foundation for advancing robotic dexterity in domains traditionally dominated by human skills.

Future studies could focus on extending this framework to more complex and dynamic non-prehensile manipulation tasks, such as flipping or hitting. Additionally, improving robustness against variations in friction arising from factors like temperature changes, surface wear, or aerodynamic effects will further enhance its sim-to-real transfer capability.

References

[1] W. Zhou, B. Jiang, F. Yang, C. Paxton, and D. Held, "HACMan: Learning hybrid actor-critic maps for 6D non-prehensile manipulation," Cornell University, May 2023, *arXiv:2305.03942*.

[2] M. Bauza, F. R. Hogan, and A. Rodriguez, "A data-efficient approach to precise and controlled pushing," in *Proc. Conf. Robot Learn.*, 2018, pp. 336–345.

[3] M. Selvaggio, A. Garg, F. Ruggiero, G. Oriolo, and B. Siciliano, "Non-prehensile object transportation via model predictive non-sliding manipulation control," *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 5, pp. 2231–2244, Sep. 2023, doi: 10.1109/TCST.2023.3277224.

[4] A. Heins and A. P. Schoellig, "Keep it upright: Model predictive control for nonprehensile object transportation with obstacle avoidance on a mobile manipulator," *IEEE Robot. Automat. Lett.*, vol. 8, no. 12, pp. 7986–7993, Dec. 2023, doi: 10.1109/LRA.2023.3324520.

[5] W. Yang and M. Posa, "Dynamic on-palm manipulation via controlled sliding," Cornell University, 2024, *arXiv:2405.08731*.

[6] S. Katyara, S. Sharma, P. Damacharla, C. G. Santiago, L. Dhirani, and B. S. Chowdhry, "Benchmarking Sim2Real gap: High-fidelity digital twinning of Agile manufacturing," 2024, *arXiv:2409.10784*.

[7] Y. Sang, M. Dubé, and M. Grant, "Dependence of friction on roughness, velocity, and temperature," *Phys. Rev. E*, vol. 77, no. 3, Mar. 2008, Art. no. 036123, doi: 10.1103/physreve.77.036123.

[8] J. Yoon, B. Son, and D. Lee, "Comparative study of physics engines for robot simulation with mechanical interaction," 2024, *arXiv.2403.07947*.

[9] V. Makoviychuk et al., "Isaac Gym: High performance GPU-Based physics simulation for robot learning," 2021, *arXiv:2108.10470*.

[10] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vilamoura-Algarve, Portugal, 2012, pp. 5026–5033, doi: 10.1109/IROS.2012.6386109.

[11] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vancouver, BC, Canada, 2017, pp. 23–30, doi: 10.1109/IROS.2017.8202133.

[12] X. Chen, J. Hu, C. Jin, L. Li, and L. Wang, "Understanding domain randomization for SIMtoReal transfer," Oct. 2021, *arXiv:2110.03239*.

[13] M. T. Mason, "Progress in nonprehensile manipulation," *Int. J. Robot. Res.*, vol. 18, no. 11, pp. 1129–1141, 1999, doi: 10.1177/02783649922067762.

[14] A. Satici, F. Ruggiero, V. Lippiello, and B. Siciliano, "Coordinate-free framework for robotic pizza tossing and catching," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 3932–3939.

[15] 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 Trans. Robot.*, vol. 35, no. 2, pp. 317–329, Apr. 2019.

[16] A. G. -Giles, F. Ruggiero, V. Lippiello, and B. Siciliano, "Closed-loop control of a nonprehensile manipulation system inspired by a pizza-peel mechanism," in *Proc. Eur. Control Conf.*, Naples, Italy, 2019, pp. 1580–1585.

[17] M. Higashimori, K. Utsumi, Y. Omoto, and M. Kaneko, "Dynamic manipulation inspired by the handling of a pizza peel," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 829–838, Aug. 2009, doi: 10.1109/TRO.2009.2017085.

[18] V. Mnih, K. Kavukcuoglu, and D. Silver, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015, doi: 10.1038/nature14236.

[19] J. Del Aguila Ferrandis, J. Moura, and S. Vijayakumar, "Non-prehensile planar manipulation through reinforcement learning with multimodal categorical exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Detroit, MI, USA, 2023, pp. 5606–5613, doi: 10.1109/IROS55552.2023.10341629.

[20] A. Louette, G. Lambrechts, D. Ernst, E. Pirard, and G. Dislaire, "Reinforcement learning to improve delta robot throws for sorting scrap metal, 2024, *arXiv:2406.13453*.

[21] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, MuJoCo, ODE, and PhysX," in *Proc. IEEE Int. Conf. Robot. Automat.*, Seattle, WA, USA, 2015, pp. 4397–4404, doi: 10.1109/ICRA.2015.7139807.

[22] L. Bao, J. Humphreys, T. Peng, and C. Zhou, "Deep reinforcement learning for bipedal locomotion: A brief survey," 2024, *arXiv:2404.17070*.

[23] A. Rzayev and V. T. Aghaei, "Off-policy deep reinforcement learning algorithms for handling various robotic manipulator tasks," 2022, *arXiv:2212.05572*.

[24] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[25] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[26] E. Wiewiora, "PotentialBased shaping and QValue initialization are equivalent," *J. Artif. Intell. Res.*, vol. 19, pp. 205–208, 2003, doi: 10.1613/jair.1190.

[27] G. Tiboni, P. Klink, J. Peters, T. Tommasi, C. D'Eramo, and G. Chalvatzaki, "Domain randomization via entropy maximization," Nov. 2023, *arXiv:2311.01885*.