Friction-Aware Reinforcement Learning for Pick-and-Place Robots

Roger Wienaah
CoRIS Institute
Oregon State University
Corvallis, OR, USA
wienaahr@oregonstate.edu

Dax Lynch
CoRIS Institute
Oregon State University
Corvallis, OR, USA
lynchdec@oregonstate.edu

Jamie Santos
CoRIS Institute
Oregon State University
Corvallis, OR, USA
santjami@oregonstate.edu

***Abstract*—Robot manipulation is central to applications such as warehouse automation, home robotics, and manufacturing, where reliable grasping is essential. However, these tasks remain challenging because robots must simultaneously manage perception and control while adapting to object-specific surface properties to prevent slippage. Traditional approaches often rely on pre-programmed motions or learning-based methods that fail to account for the physical interactions between the gripper and the object. These methods typically assume fixed or simplified contact conditions, leading to failure when object textures differ from those used during planning or training**

To address this gap, we investigate three strategies to improve the robustness of a simulated robotic arm against varying friction by: 1) introducing the coefficient of friction as an input channel to a Truncated Quantile Critics (TQC) algorithm, 2) outputting joint-level commands rather than end-effector targets, and 3) implementing a training curriculum that gradually increases task difficulty. Our key insight is that curriculum learning is the most effective strategy for ensuring robustness in low-friction conditions, enabling robots to manipulate objects with diverse textural properties. This curriculum-based approach yields a 39higher success rate over the baseline TQC algorithm in low-friction scenarios ([0.05, 0.25]). Overall, this work provides a systematic evaluation of robustness strategies across a range of frictional conditions and identifies curriculum learning as a simple, effective mechanism for enhancing grasp stability.

***Index Terms*—Reinforcement learning, grasping, friction, tactile, manipulation, manufacturing, warehouse automation, home robotics**

I. Introduction

Robotic manipulation is widely deployed in structured environments such as manufacturing and warehouse fulfillment, and is beginning to emerge in domestic settings with robots like Tesla's Optimus. Though ubiquitous, manipulation remains a fundamental yet challenging problem in robotics. To operate reliably in highly unstructured environments, especially when handling potentially slick objects such as greased gears or smooth plastics, robot controllers must adapt to a broad range of contact conditions. While humans perform grasping tasks effortlessly and with great dexterity, robots operating in unstructured environments face significant hurdles due to uncertainties in perception, control, and physical interaction between the gripper and the object surfaces. A successful robot grasp requires precise coordination across sensing, planning, and control to effectively handle the complexities of varying material properties, such as texture and surface friction.

Traditionally, robotic grasping approaches rely on pre-programmed plans that involve perceiving an object, computing a grasp pose, and executing a grasp motion, without accounting for environmental changes or the physical interaction between the gripper and the object. Researchers have also proposed learning-based grasping techniques that enable robot arms to learn and adapt to different object shapes, avoid collisions, and improve grasp success rates in dynamic environments.

Most learning-based methods are vision-based (e.g., [1], [2]) and do not consider the contact dynamics between the gripper and the object surface. Only a small number of works have integrated tactile feedback into learning approaches to enhance grasp stability and control, and these primarily focus on estimating physical properties, such as contact forces, pressure, and temperature (e.g., [3]–[5]) rather than accounting for frictional properties of the gripper and the object surface.

Despite these advances, a key limitation persists: most robotic grasp planning methods implicitly assume uniform friction properties across objects. In practice, however, this is often not the case in real-world environments. An order fulfillment robot may be tasked with retrieving sandpaper in one moment and smooth metal the next, or a robot outdoors may encounter objects that become wet over time, all of which can significantly affect grasp stability and success rates.

To address this limitation, in this paper we analyze three approaches to improve robustness for successful pick and place across an extensive range of friction coefficients between the gripper and object. The first method investigates the effect of introducing the friction coefficient directly to the controller as an input channel; this experiment examines if using information from a tactile sensor directly on the gripper alone can improve performance. The second method compares

controlling the arm via joint-level torque control and end-effector Cartesian coordinates. Lastly, a curriculum is applied when training the controller: after training across the whole range of frictions randomly, how are results affected when training continues on the narrow band of (low) coefficients where all other methods tend to fail? These analyses are conducted using the Franka Emika Panda arm simulated in Panda Gym, and with Truncated Quantile Critics (TQC) as the base algorithm used to train the controller. Though there exist several commonly used reinforcement learning algorithms such as TD3, DDPG, or SAC, TQC was chosen for its performance and faster convergence time [20] [21].

These experiments yield a method for training a controller that achieves higher grasp success rates in low-friction conditions than the baseline TQC algorithm and standard training regime across input configurations and control modes. This method can be applied directly to real robotic controllers for enhanced grasping robustness.

The key insight of this paper is that of the three strategies investigated, implementing a training curriculum is the most effective at improving success rates for lower friction coefficients, while using frictional information as input also improves success and training convergence time. We also explore several curriculum designs, and conclude that performance is highly sensitive to their hyper-parameters, particularly the number of additional training cycles and the chosen friction range. By only training for 8% longer on the lower range, the success rate at this range increased by 10-20% while avoiding the collapse in high-friction performance observed when retraining for the same number of time steps as the original training run.

In summary, the contributions of this paper are:

1) a systematic evaluation of three strategies for improving robustness across a wide range of friction conditions in simulation; and
2) from one of these strategies, a curriculum-based training approach that significantly improves low-friction grasping success.

II. Related Work

A. Reinforcement Learning for Robotic Manipulation

Industrial settings are ideal conditions for the use of robotic manipulators; they offer repetitive tasks that robotic arms excel at, such as picking and placing items or assembly [7]. The implementation of robotic arms in these settings, integral to the technological wave known as "Industry 4.0", have reduced manufacturing cost and improved product quality [8].

1) Current Robotic Manipulation Solutions: Traditional robotic arm manipulation is performed using forward and inverse kinematics, which are calculated based on input from graphical or text-based interfaces and translated to the manipulator's commands [9]. Simple motions for repetitive tasks can be programmed online, but this presents a challenge of needing to reprogram the robot for slight variations in task [9], [10]. While robotic arms have been used since the 1970s in industrial environments for repetitive tasks, many tasks require more adaptive solutions, such as pick-and-place for online order fulfillment [9]. Another example may be to sort items in a cluttered environment [9]. The inflexibility of traditional control methods makes reinforcement learning an attractive solution as policies learned may be able to adapt to unseen variations of a given task.

Reinforcement learning offers a powerful alternative to these traditional control algorithms. At its core, the agent learns to make intelligent decisions by iteratively interacting with its simulated environment. The agent seeks to maximize its reward function and learns actions that maximize that over time. Instead of being explicitly programmed, the robot learns a policy which maps current state information (e.g., joint positions, friction coefficients) to the best possible actions. This learning-based approach is particularly well-suited to address the challenge of inflexibility. By training on a wide variety of scenarios, an RL agent can learn a policy that maps any arbitrary sensor input (e.g. tactile data or camera input) to control commands. This allows it to handle variations in objects position, shape, or orientation that might require significant reprogramming using classical methods. While classical methods might have a robot arm pick up an item on an assembly line at the same contact point every time, reinforcement learning could be used in more adaptive settings such as picking up moving items at different contact points on an assembly line.

2) Deep RL Algorithms for Robotic Manipulation: Many deep reinforcement learning algorithms have been used to train controllers for robotic manipulation (e.g. Deep Deterministic Policy Gradient (DDPG) [17], Twin Delayed Deep Deterministic Policy Gradient (TD3) [22], Proximal Policy Optimization (PPO) [24] and Soft Actor-Critic (SAC) [18]). DDPG demonstrates the viability of end-to-end learning for continuous control but suffers from overestimation bias and instability. TD3 addressed the stability issues in DDPG by introducing twin critics and delayed policy updates. However, it relies on estimating a single mean Q-value and often converges to policies that are not robust to variance in environment dynamics [22] [17]. On-policy methods such as PPO, while more stable than DDPG and TD3, are often less sample-efficient, making them computationally expensive for complex manipulation tasks. SAC improves sample efficiency and exploration by maximizing entropy alongside expected return; however, it estimates a single mean Q-value and therefore suffers from a similar limitation to TD3.

To address these challenges, we used Truncated Quantile Critics (TQC) [21]. TQC combines the maximum entropy framework of SAC with a distributional representation of the critic. It employs a truncation mechanism to drop the most optimistic quantile estimates during target calculation. In our context (i.e., friction-aware grasping), this mechanism is particularly valuable, as it can prevent the agent from overestimating the value of grasps that succeed only under high-friction conditions and encourage robust learning under lower-friction conditions. Additionally, TQC was selected for its stability and training speed compared to other algorithms

[21].

B. Friction-Awareness in Robotic Manipulation

Effective manipulation requires managing the physical interaction between the robot gripper and the object. This poses complex control questions such as how much pressure to apply to avoid crushing an object, what is the right approach or angle to pick an object, and how to adapt to slippery or soft surfaces [12]. To address this, researchers have used tactile sensors to estimate surface properties (e.g., friction, temperature, pressure). Recent Deep RL works have integrated tactile feedback directly into the control loop. For instance, Hu et al. [3] demonstrated that combining tactile sensing with adaptation skills allows agents to learn robust grasping strategies that adjust to contact conditions. Similarly, Zhou et al. [5] proposed T-TD3, a framework utilizing tactile priors to enhance the grasping of deformable objects. While these methods validate the importance of friction-awareness, they often rely on complex tactile image processing or are tailored to specific object types (e.g., deformable objects). Additionally, these methods do not consider varying friction dynamics as state inputs.

Raei et al. [12] implemented a discrete-time control strategy that adapts to surface friction by iteratively incorporating results from past actions and real-time observations. Their approach utilized a DDPG actor-critic framework trained in the MuJoco simulator, achieving a zero-shot Sim2Real transfer that outperformed analytical baselines. While this demonstrates the potential of RL for friction adaptation, the method focused on sliding objects (i.e., non-prehensile tasks) rather than grasping them. Also, the use of DDPG makes the approach susceptible to overestimation bias and instability, limiting its reliability in scenarios with stochastic contact dynamics compared to distributional approaches like TQC.

III. Methods

This section outlines the reinforcement learning environment and details the three experimental designs evaluated in this work.

A. Problem Formulation

1) Task Definition and Simulation Setup: The aim of this paper is to evaluate methods for improving grasping success rates for a robotic manipulator operating under a broad spectrum of friction dynamics (coefficients spanning from 0.05 to 1.0).

We used the PandaGym simulation package [14] to simulate a Franka Emika Panda robot arm. This software package was selected because it is lightweight, easy to use, and designed specifically for implementing reinforcement learning algorithms. It also allows for tuning the friction coefficients of objects and specifying either joint control or end effector control modes. The complete setup consists of a Panda arm mounted on a table, with objects of varying textures spawning in front of it. To isolate friction dynamics, object size and shape were held constant. For this project, the task was to locate the object, successfully grasp it, and move it to a target location also randomly spawned in the environment.

2) Robotic Arm MDP: We considered two options for the model's input: 1) static camera shots of the scene at every time step, or 2) direct input of the state space variables. The first option, using static camera images, could simplify transitioning these methods from simulation to the real world in future applications; i.e., a camera sensor would be sufficient for all input information except for object-gripper friction. However, providing the state space variables directly for training is much more efficient, as it eliminates the need for the model to learn these details from images initially. We follow the input mapping outlined in [19], which includes the following:

- The end-effector position (x, y, z)
- The end-effector velocity
- The gripper finger width and each finger's velocity
- The object's position (x, y, z)
- The object's rotation (Euler)
- The object's angular velocity (z)
- For certain experiments, the friction coefficient between the fingers and the object (0.05 - 1.0)
- The target location (x, y, z)

3) Friction Modeling: These experiments assume that the controller can be made aware of the object's frictional properties. In practice, we imagine a shear force tactile sensor lining the inside of the gripper, such as the Friction/Shear Force Sensor by Nissha Co., Ltd. [15]. The friction coefficient can then be calculated from the shear force and fed directly to the controller. Whether this measurement loop is actually an effective method will need further investigation. For the purposes of the simulation, the environment randomly generates objects of varying friction coefficients, and the model is provided the friction coefficient of the object directly as an input channel depending on the method under analysis.

B. Reinforcement Learning Approach

1) RL Algorithm: For all experiments, the Truncated Quantile Critics (TQC) algorithm was used for training the robotic arm. The Stable Baselines 3 Python package was the implementation used for all instances [25]. This method was selected for its ability to effectively mitigate overestimation bias in continuous control settings, resulting in better stability and performance and faster convergence compared to SAC, DDPG, or TD3 [20] [21].

TQC builds upon SAC by incorporating distributional reinforcement learning. While standard SAC estimates a mean Q-value, TQC estimates the continuous distribution of returns using an ensemble of N critic networks, where each critic outputs M quantiles (atoms), resulting in $N \times M$ atoms approximating the return distribution. To prevent the propagation of overestimated Q-values, TQC employs a truncation mechanism. During the target calculation, the algorithm pools all $N \times M$ quantiles from the target critics, sorts them in ascending order, and truncates the top d atoms per network (removing the $d \times N$ most optimistic estimates). This truncation allows for granular control over the bias, ensuring the agent

learns aggressively without diverging. The critic is trained by minimizing the quantile Huber loss between the current quantile estimates and the truncated target distribution. The algorithm for TQC is outlined in Algorithm 1.

Algorithm 1 Truncated Quantile Critics (TQC)

Input: Replay buffer $\mathcal{D}$, Actor π_ϕ, Ensemble of N Critics $\{Z_{\psi_n}\}_{n=1}^N$
Input: Target networks $\pi_{\bar{\phi}}$, $\{Z_{\bar{\psi}_n}\}_{n=1}^N$
Input: Discount factor γ, Target smoothing τ, Learning rates λ
Input: Number of atoms M, Number of dropped atoms per net d (keep $k = M - d$)
Output: Optimized parameters ϕ and $\{\psi_n\}_{n=1}^N$

Initialize parameters $\phi, \psi_1, \ldots, \psi_N$
Initialize target parameters $\bar{\phi} \leftarrow \phi$, $\bar{\psi}_n \leftarrow \psi_n$ for $n = 1..N$
for each iteration **do**
  for each environment step **do**
    Sample action $a_t \sim \pi_\phi(s_t)$
    Step environment: $s_{t+1}, r_t, done \leftarrow \text{Env}(s_t, a_t)$
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
  end for
  for each gradient step **do**
    Sample batch of transitions $B \sim \mathcal{D}$
    // 1. Target Distribution Calculation
    Sample next actions $a' \sim \pi_{\bar{\phi}}(s_{t+1})$
    Compute atoms $z_{n,m} = Z^m_{\bar{\psi}_n}(s_{t+1}, a')$ for all $n \in \{1..N\}, m \in \{1..M\}$
    Pool all $N \times M$ atoms into set $\mathcal{Z}_{pool}$
    Sort $\mathcal{Z}_{pool}$ in ascending order
    Truncate: Keep the smallest $K = N \times k$ elements to form set $\mathcal{Z}_{trunc}$
    Compute target atoms: $y_j = r_t + \gamma(z_j - \alpha \log \pi_{\bar{\phi}}(a'|s_{t+1}))$ for $z_j \in \mathcal{Z}_{trunc}$
    // 2. Update Critics
    for $n = 1$ to N **do**
      Compute Quantile Huber Loss $\mathcal{L}_Z(\psi_n)$ against target distribution $\{y_j\}_{j=1}^K$
      $\psi_n \leftarrow \psi_n - \lambda_Z \nabla_{\psi_n} \mathcal{L}_Z(\psi_n)$
    end for
    // 3. Update Actor and Alpha
    Compute actor loss $\mathcal{L}_\pi(\phi)$ (using non-truncated critics)
    $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi \mathcal{L}_\pi(\phi)$
    Update entropy temperature α automatically
    // 4. Soft Update Targets
    $\bar{\psi}_n \leftarrow \tau\psi_n + (1-\tau)\bar{\psi}_n$ for $n = 1..N$
    $\bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi}$
  end for
end for
return Optimized Policy π_ϕ

2) Reward Structure: A critical decision that shapes robot behavior during and after training is the reward structure used. In this environment, there are two main options. The first option is to use sparse rewards, which gives small negative rewards (penalties) for each time step that passes before successful completion, and zero (non-negative) rewards for completing the task [23]. The other option is to use dense rewards, which gives a reward proportional to the negative of the distance from the achieved position of the object at the end of the episode compared to the target location [23]. In these experiments, sparse rewards are used. Though training with sparse rewards is more difficult due to the lack of feedback over the training process, we chose to use sparse rewards as it prioritizes successful completion of the goal.

The reward function is defined as:

$$R = \begin{cases} 0 & \text{if } d \leq \epsilon \\ -1 & \text{if } d > \epsilon \end{cases} \tag{1}$$

where $d = \|\mathbf{p}_{\text{object}} - \mathbf{p}_{\text{goal}}\|_2$ is the Euclidean distance between the object position and the goal position, and $\epsilon = 0.05$ m is the distance threshold for success.

3) Environment Details: For all experiments, the environment used was based on "PandaPickAndPlace-v3" in which an object is randomly spawn in the environment on the table in front of the arm, and the robot is tasked with relocating the object to a randomly generated target location. The task takes place on a rectangular table measuring 1.1 m × 0.7 m × 0.4 m (length × width × height), positioned in front of the Panda arm. At the start of each episode, a cubic object with side length 0.04 m is randomly spawned on the table surface within a 0.3 m × 0.3 m region centered in front of the robot. The robot is tasked with relocating this object to a randomly generated target location within a 0.3 m × 0.3 m × 0.2 m goal region, where approximately 30% of goals are placed directly on the table surface while the remainder require lifting the object to an elevated position. An episode is considered successful when the object is within 0.05 m of the target location.

C. Experiments: How Can Gripping Robustness Be Improved?

The simulation is run off of Py-Bullet, with max episode lengths of 50. For our experiments, the controller is trained using TQC on objects of randomly selected friction coefficients between 0.05 and 1.0, which encompasses all but the most extreme materials. This range is then used for evaluation on all experiments.

1) Experiment 1: Friction Awareness Only: The first experiment examines the effect of explicitly supplying the friction coefficient between the gripper and the object to the policy. This idea stems from the notion that there exist tactile sensors, such as the Nissha Co. sensor described previously, that line the inside of robotic grippers and can measure shear force and thereby directly provide information about object surface properties to the controller. Our hypothesis was that this information would allow the controller to learn more specific gripper width and finger velocities. In this experiment, the baseline algorithm is TQC from Stable Baselines which is trained on objects of varying friction but is not provided the

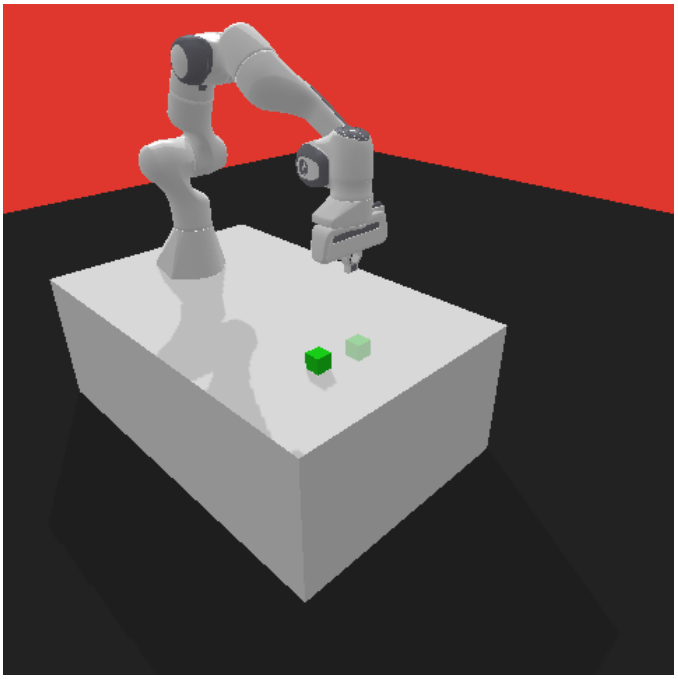


Fig. 1: The PandaPickAndPlace-v3 simulation environment showing the Panda robotic arm, table workspace, and target object.

friction information as input. We then train a second model in the same way, with the only difference being that the friction coefficients are included in the input space.

2) Experiment 2: Training for Joint-Level Control: In the realm of robot manipulation, there are two common methods for control: specifying end-effector target poses that are solved for via inverse kinematics, and giving joint-torque commands. Unless explicitly mentioned, all controllers in this paper were trained using target poses. This method is simpler for the model to learn as it only needs to learn the placement of the end effector and the finger torque, using inverse kinematics to solve for the rest of the arm. With the second method, the model would need to learn control for each joint.

However, one limitation of learning only end-effector position commands is that the orientation of the end-effector remains locked, commonly known as *fixed-orientation control*. This second experiment investigates whether allowing the end-effector to vary yaw-pitch-roll orientations will yield more successful outcomes, since more gripping strategies are possible. In this experiment, three models are compared:

1) Baseline: End-effector control, varying friction during training
2) Joint control, trained without friction as input
3) Joint control, trained with friction as input

3) Experiment 3: Implementing Training Curriculum: The final experiment implements a curriculum-based training strategy: starting with the model trained with friction-awareness in Experiment 1, we continue training on a narrower, low-friction region where most models tend to struggle (i.e., success rates below 80%).

Initially, training on the model was continued for the same number of time steps as the initial training: 3 million. This was done to understand the effects of re-training an already trained model on a different dataset. The range of friction coefficients used in this training was [0.001, 0.15], informed by success rates from the prior two experiments. After understanding the effects, which will be discussed in the next section, this curriculum procedure was reapplied to the same Experiment 1 friction-aware model, but for 250,000 time steps and for a range of [0.05, 0.25].

The second range was shifted and expanded to overlap more with the region of friction coefficients that produce higher success rates. As multiple variables were changed between the first and second runs, the differences cannot be directly attributed to one change, but we include both sets of results as they offer valuable insight. To summarize, the three models compared in this experiment are:

1) Baseline: Friction-aware trained model from Exp.1
2) Baseline model + 3 million time steps of training on objects of random friction, coefficients: [0.001, 0.15]
3) Baseline model + 250,000 time steps of training on objects of random friction, coefficients: [0.05, 0.25]

D. Evaluation Metrics

For each experiment, the models trained with adaptations are compared to the corresponding baselines described above, using success rate as the primary evaluation metric. After training is complete, each model is evaluated for 100 rollouts at every 0.05 increment of the friction coefficient within the range [0.05, 1.0]. Training time (in time steps) is also considered in Experiments 1 and 2; however, because additional training steps are the primary variable manipulated in Experiment 3, this metric is not included for that experiment.

IV. Results and Discussion

In this section, we present the results of the three experiments described above, evaluating performance across a comprehensive range of friction coefficients, [0.05, 1.0]. For each experiment, we report the success rates of all models including baselines, and when applicable, we include the training times for comparison.

A. Experiment 1: Friction Awareness Only

This experiment compares the success rates of the pick-place-tasks for two controllers across friction coefficients: one trained with friction as input and one without (the baseline). The control output is end-effector locations. The results are shown in Figure 2.

These results demonstrate that directly feeding the controller friction dynamics information increases or retains pick-and-place success rate when compared to the model trained by the baseline algorithm. The greatest difference appears between coefficients [0.1, 0.6], with the highest difference at 0.15: 30% better performance.

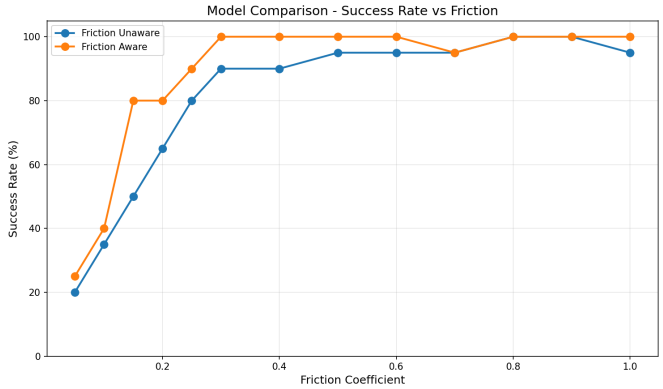


Fig. 2: Experiment 1 (friction-awareness) success rates as a function of the friction coefficient

Figure 3 shows the training steps until convergence of the success rate for Experiment 1. As demonstrated by this figure, the friction-aware model converges to near 100% success faster than the non-friction-aware model. For this reason, the friction-aware model is used as the baseline for the next two experiments.



Fig. 3: Experiment 1 (friction-awareness) success rates as a function of time steps trained

B. Experiment 2: Training for Joint-Level Control

In this experiment, two modes of control are used: 1) torque commands for each joint, and 2) end-effector locations. As previously mentioned, the baseline for this test was the friction-aware model from Experiment 1, which outputs end-effector positions to be solved by inverse kinematics. Two variations of the joint-level control model were generated: one trained on objects of varying friction and one on objects of constant friction. The success rates are shown in Figure 4. Over most friction coefficient values, the end-effector model matched or outperformed the other two models. The exception is at 0.15, where both joint-level control models outperformed the baseline end-effector control model. This was the only lower friction value where this occurred, so conclusions cannot be drawn about the superiority of joint-level control at lower values.

Furthermore, Figure 5 shows training success as a function of time steps trained for the two joint-control models. These models converged approximately 50% slower than the baseline end effector control model shown in Figure 3. This corresponds to our intuition that learning joint-level control is more difficult than end-effector control, discussed earlier. Because of these factors, the performance loss across the range of friction does not compensate for any possible performance gains at the lower end.

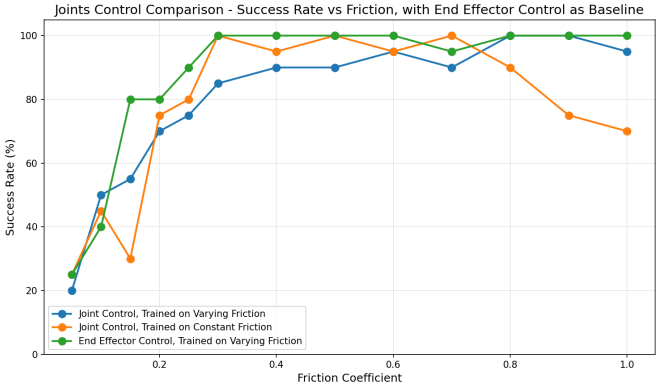


Fig. 4: Experiment 2 (joint-level control) success rates as a function of the friction coefficient



Fig. 5: Experiment 2 (joint-level control) training success rate over time steps

C. Experiment 3: Implementing a Training Curriculum

As clearly demonstrated in the previous two experiments (Figures 2 and 4), all methods suffer from low success rates at the lowest end of the friction coefficient range, namely [0.05, 0.25]. This is expected as these values correspond to very slick interactions, such rubber on wet surfaces. The final experiment evaluates the effects and efficacy of implementing a training curriculum to promote higher success rates in the lower region of friction coefficients.

Using the simple curriculum of training the baseline algorithm 3 million time steps across [0.001, 0.15] and again from the baseline [0.05, 0.25] for 250,000 steps, the results from Figure 6 are obtained. The 3 million time step model was trained first. As expected, this model performs significantly better than the baseline algorithm that was only trained across the entire range of coefficients when given objects with low coefficients: at 0.05, the difference is 40%, and at 0.1, the difference is 55%. However, after 0.25, the results are linearly driven toward 0% as the friction is increased between the gripper and the object. The insight gained from this experiment is that increasing training difficulty can be used to improve results where previous models fail, but a balance must be struck to prevent overfitting and catastrophic failure for the rest of the dataset.

For this reason, in the second iteration the friction range was shifted higher to [0.05, 0.25], where the baseline had higher success rates. The baseline model was also only retrained for 250,000 time steps, to retain the training for the higher friction coefficients. The success rate for this model, though lower than the 3 million time-step-retrained model across the lower friction regime, retains higher performance across the entire range. More significantly, this model outperforms the baseline algorithm in the lower regime: a difference of approximately 20% from [0.05, 0.1], and remains higher until performing the same or marginally worse (5%) from [0.3, 0.8]. Performance is lost in the highest regime, [0.9, 1.0], with the baseline outperforming this model by 10-20%. Over the entire range, this second curriculum-trained model outperformed the baseline: 88% versus 86% success.

Though the second model still traded higher performance at the lower end for lower performance at the higher end, these results demonstrate that hyperparameter tuning, namely number of time steps and the difficulty parameters (e.g., friction range) of the environment, can improve outcomes not only where monotonously trained models fail, but overall. Results may be further improved by implementing a more complex curriculum, i.e. by gradually narrowing the difficulty over time, rather than performing one second training block of fixed parameters.

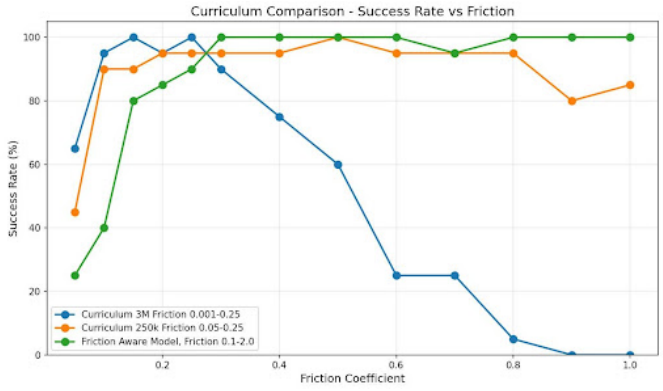


Fig. 6: Experiment 3 (training curriculum) success rates as a function of the friction coefficient

V. Future Work and Conclusion

The experiments presented in this work demonstrate that training curricula and friction-awareness can substantially improve grasping performance under low-friction conditions, but they also demonstrate that improvements from the former are highly sensitive to hyperparameter tuning.

This is highlighted by the shift in the shape of the success-rate curve between the first and second curriculum-trained models, suggesting that further tuning has the potential to generate higher success rates across the entire range of friction coefficients. Therefore, the natural next step to extend this work would be to continue to investigate hyperparameter tuning and curriculum structuring for Experiment 3.

Additionally, a fourth avenue for investigation is custom reward shaping. The reward structure used in these experiments were quite simple. Can results be improved if a slip penalty is introduced? Also, these experiments thus far have assumed a fixed rigidity of objects. Training for low friction can reward higher grasping force without repercussions. How do these models generalize to grasping delicate objects, such as an egg? Reward shaping may offer improved success rates for situations such as this.

Overall, this study demonstrates both the potential and limitations of curriculum-based training for robust grasping under varying friction dynamics. By combining targeted improvements in curriculum design, friction coefficient input channels, and possibly better-shaped rewards, more adaptable and robust robotic grasping policies can be obtained.

VI. Contribution

TABLE I: Project Contribution

Name	Writing	Coding	Organization
Jamie Santos	40 %	20 %	Deadlines and Task Delegation
Dax Lynch	20 %	60 %	Software Management
Roger Wienaah	40 %	20 %	Meeting Scheduling

Acknowledgment

We would like to thank our T.A., Ever, and Professor Kagan for their guidance throughout this course.

References

[1] D. -C. Hoang, J. A. Stork and T. Stoyanov, "Context-Aware Grasp Generation in Cluttered Scenes," 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 2022, pp. 1492-1498, doi: 10.1109/ICRA46639.2022.9811371.

[2] A. Mousavian, C. Eppner and D. Fox, "6-DOF GraspNet: Variational Grasp Generation for Object Manipulation," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 2901-2910, doi: 10.1109/ICCV.2019.00299.

[3] Y. Hu, M. Li, S. Yang, X. Li, S. Liu and M. Li, "Learning Robust Grasping Strategy Through Tactile Sensing and Adaption Skill," 2024 IEEE International Conference on Robotics and Biomimetics (ROBIO), Bangkok, Thailand, 2024, pp. 1197-1202, doi: 10.1109/ROBIO64047.2024.10907398.

[4] W. Chen, H. Khamis, I. Birznieks, N. F. Lepora and S. J. Redmond, "Tactile Sensors for Friction Estimation and Incipient Slip Detection—Toward Dexterous Robotic Manipulation: A Review," in IEEE Sensors Journal, vol. 18, no. 22, pp. 9049-9064, 15 Nov.15, 2018, doi: 10.1109/JSEN.2018.2868340.

[5] Y. Zhou, Y. Jin, P. Lu, S. Jiang, Z. Wang and B. He, "T-TD3: A Reinforcement Learning Framework for Stable Grasping of Deformable Objects Using Tactile Prior," in IEEE Transactions on Automation Science and Engineering, vol. 22, pp. 6208-6222, 2025, doi: 10.1109/TASE.2024.3440047.

[6] S. Amarjyoti, "Deep Reinforcement Learning for Robotic Manipulation-The state of the art," arXiv:1701.08878 [cs], Jan. 2017, Available: https://arxiv.org/abs/1701.08878

[7] D. Han, B. Mulyana, V. Stankovic, and S. Cheng, "A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation," Sensors, vol. 23, no. 7, p. 3762, Apr. 2023, doi: 10.3390/s23073762.

[8] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," Bus Inf Syst Eng, vol. 6, no. 4, pp. 239–242, June 2014, doi: 10.1007/s12599-014-0334-4.

[9] A. Lobbezoo, Y. Qian, and H.-J. Kwon, "Reinforcement Learning for Pick and Place Operations in Robotics: A Survey," Robotics, vol. 10, no. 3, p. 105, Sept. 2021, doi: 10.3390/robotics10030105.

[10] A. Iriondo, E. Lazkano, L. Susperregi, J. Urain, A. Fernandez, and J. Molina, "Pick and Place Operations in Logistics Using a Mobile Manipulator Controlled with Deep Reinforcement Learning," Applied Sciences, vol. 9, no. 2, p. 348, Jan. 2019, doi: 10.3390/app9020348.

[11] X. Hu et al., "Impact of Static Friction on Sim2Real in Robotic Reinforcement Learning," 2025, arXiv. doi: 10.48550/ARXIV.2503.01255.

[12] H. Raei, E. De Momi, and A. Ajoudani, "A Reinforcement Learning Approach to Non-prehensile Manipulation through Sliding," 2025, arXiv. doi: 10.48550/ARXIV.2502.17221.

[13] OpenAI, "Solving Rubik's Cube with a Robot Hand," 2019, arXiv. doi: ARXIV.1910.07113.

[14] Q. Gallouédec, "panda-gym ," Read the Docs, 2021. https://panda-gym.readthedocs.io/en/latest/guide/install.html

[15] "Film-type tactile sensor," Nisha Co, Ltd. https://connect.nissha.com/filmdevice/en/shear-force-sensor/robot/

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017, arXiv. doi: 10.48550/ARXIV.1707.06347.

[17] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, arXiv. doi: 10.48550/ARXIV.1509.02971.

[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in Proc. Int. Conf. Mach. Learn. (ICML), Stockholm, Sweden, Jul. 2018, pp. 1861–1870.

[19] "PPO pick-and-place," hlfshell.ai. [Online]. Available: https://hlfshell.ai/posts/ppo-pick-and-place/ (accessed Nov. 24, 2025).

[20] Kuznetsov, Arsenii, et al. "Controlling overestimation bias with truncated mixture of continuous distributional quantile critics." International conference on machine learning. PMLR, 2020.

[21] J. Gu, "Application of Robotic Arm Path Planning Based on TQC Algorithm," vol. 53, pp. 216–220, Dec. 2023, doi: https://doi.org/10.1109/auteee60196.2023.10407890.

[22] S. Fujimoto, van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," arXiv.org, 2018. https://arxiv.org/abs/1802.09477

[23] "Environments — panda-gym v3.0.8 documentation," Readthedocs.io, 2021. https://panda-gym.readthedocs.io/en/latest/usage/environments.html (accessed Dec. 05, 2025).

[24] Li C, Liu Z, Li L, Ji Z, Li C, Liang J, Li Y. Improved PPO Optimization for Robotic Arm Grasping Trajectory Planning and Real-Robot Migration. Sensors. 2025; 25(17):5253. https://doi.org/10.3390/s25175253

[25] "stable-baselines3," PyPI, Jan. 27, 2025. https://pypi.org/project/stable-baselines3/