Reinforcement Learning Enables Real-Time Planning and Control of Agile Maneuvers for Soft Robot Arms

Anonymous Author(s)
Affiliation
Address
email

Abstract: Control policies for soft robot arms typically assume quasi-static motion or require a hand-designed motion plan. To achieve real-time planning and control for tasks requiring highly dynamic maneuvers, we apply deep reinforcement learning to train a policy entirely in simulation, and we identify strategies and insights that bridge the gap between simulation and reality. In particular, we strengthen the policy's tolerance for inaccuracies with domain randomization and implement crucial simulator modifications that improve actuation and sensor modeling, enabling zero-shot sim-to-real transfer without requiring high-fidelity soft robot dynamics. We demonstrate the effectiveness of this approach with experiments on physical hardware and show that our soft robot can reach target positions that require dynamic swinging motions. This is the first work to achieve such agile maneuvers on a physical soft robot, advancing the field of soft robot arm planning and control. Our code is publicly available at [hidden for double-blind review].

Keywords: Soft Robotics, Reinforcement Learning, Sim-to-Real Transfer, Dynamics and Controls

1 Introduction

Compared to traditional rigid robot manipulators, soft robot arms with inherent material compliance offer improved adaptability and safety in tasks requiring contact with the environment [1, 2, 3]. However, soft robots have yet to achieve the high-speed capabilities of rigid robots, which can leverage dynamics to achieve difficult tasks [4, 5, 6, 7, 8]. We aim to enable dynamic control for a class of soft robot arms made from an inflated fabric tube with a fabric Pneumatic Artificial Muscle (fPAM), which has near-linear dynamics and high-bandwidth dynamic response with minimal hysteresis [9]. We mount this inexpensive and lightweight inflated-beam soft robot arm on a mobile platform that allows 1D linear actuation of the arm's base (Fig. 1). Integrating the soft arm with a sliding base increases the robot's workspace and allows additional dynamics that can be leveraged by a controller.

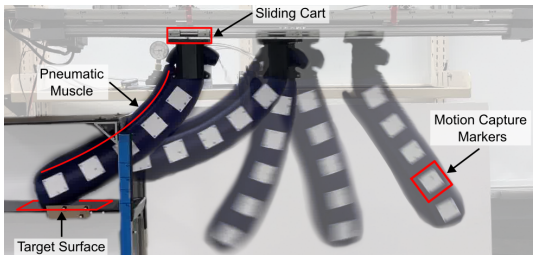

Figure 1: We achieve soft robot arm control tasks that require leveraging robot dynamics. Our inflated-beam robot uses a pneumatic bending actuator and a 1D mobile base to perform agile maneuvers such as swinging onto a target surface. We use a motion capture system to measure the robot's state and compute control actions with a policy trained using reinforcement learning.

Most prior work in soft robot arm control assumes quasi-static motion [10, 11, 12, 13]. This assumption is enticing because soft robots have complex dynamics resulting from their compliance and high dimensionality, so using quasi-static models for soft robot motion makes them easier to control. However, this approach is unable to exploit dynamic capabilities, such as leveraging swinging motions to generate momentum. More recent works have incorporated dynamic models [14, 15], but limit their applications to tasks that are simple enough to use hand-designed motion plans, such

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

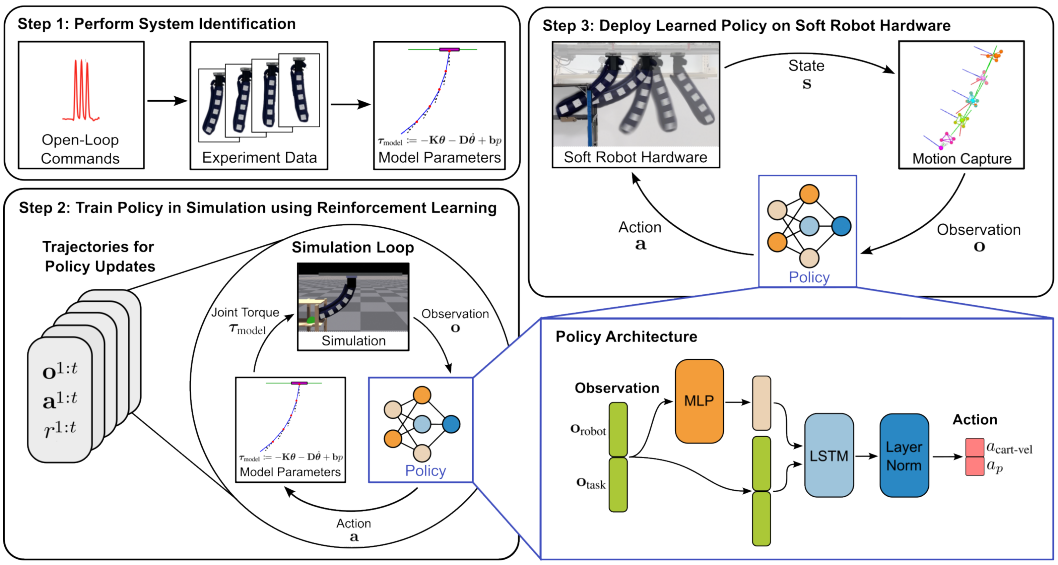

Figure 2: In our proposed framework, we first fit parameters for a physics-based dynamic model using about 30 seconds of real experimental data. Next, we train a control policy using deep reinforcement learning entirely in simulation using the fit model from the previous step. Finally, we deploy our learned control policy directly on real hardware using a motion capture system to measure the robot's state.

as tip trajectory following. In contrast, our learned policies eliminate the need for a pre-defined motion plan because they are goal-conditioned and handle planning and control jointly.

In our work, we use a dynamic model that approximates our soft robot arm as a multi-link rigid body system. This allows us to leverage robot dynamics, as well as existing rigid-body simulators and control frameworks, to achieve tasks requiring high-speed motion. With this model, we employ reinforcement learning (RL) to train a control policy and deploy this policy directly onto a real, physical soft robot system. However, naive transfer to the real world is unsuccessful, which is a common challenge encountered when deploying control policies learned in simulation [16, 17]. We overcome this challenge by implementing key adjustments to the simulator's actuator and sensor models as well as improving the policy's robustness to modeling inaccuracies using domain randomization. To our knowledge, this is the first work that achieves real-time motion planning and control for high-speed behaviors on a physical soft robot arm. This work also provides evidence that simple physics-based models can be used to build a simulator that enables learning of agile maneuvers for soft robots with zero-shot sim-to-real transfer.

To evaluate our system's performance, we optimize trajectories for each goal in our test set as an approximate upper bound of performance. We find that the policy from our RL framework performs similarly to the approximate upper bound while achieving real-time planning and control. In contrast, trajectory optimization requires an offline planning stage that takes several minutes to perform, which is impractical for real-world use.

2 Related Work

Here we discuss the most relevant prior work and provide a more thorough discussion in Supplementary Material (SM) A.

Soft Robot Arm Modeling. Our work requires motion that is impossible under quasi-static assumptions, preventing the use of kinematic models such as the piecewise constant curvature model [13]. However, we find that we do not require the higher model fidelity offered by the augmented rigid body formulation [18], Cosserat rod theory [19], Koopman [20] models, or Long-Short Term Memory models [14]. Instead, we choose a model similar to the n-link pendulum model used by Jitosho et al. [21] since it is cheap to compute, simple to implement, and enables fast simulation.

Soft Robot Arm Control. Recent works have demonstrated dynamic behaviors on soft robot arms. Centurelli et al. [14] train a control policy using reinforcement learning to perform tip path following under light payloads. Fischer et al. [15] use an operational space controller and proportional-derivative control to perform tip path following for pick and place, throwing, and drawing. Bruder et al. [20] use a Koopman-based system identification method to construct explicit linear dynamical models of soft robots and use model-predictive control to perform trajectory following tasks. Dynamic models improve the speed and accuracy of tip path following, but these works require hand-designed tip trajectories, limiting the capabilities of their robots. In our work, we achieve autonomous motion planning. This enables agile maneuvers to emerge during training, such as high-speed swinging motions, that would be difficult to design by hand.

Reinforcement Learning for Controlling Physical Robots. A common training paradigm when learning policies for physical robots is to train control policies entirely in simulation and then deploy them in the real world [22]. However, policies trained in simulation often fail in the real world due to model discrepancies between the simulated and real systems [16, 17]. One approach to overcoming this sim-to-real gap is training the policy under domain randomization so that the resulting policy will be robust to simulation inaccuracies [23]. This approach has been successfully applied to robotics domains such as legged locomotion [24, 25, 26, 27, 28], in-hand manipulation [29, 30, 31], and robot grasping [23, 32, 33]. In addition to this randomization, Tan et al. [34] overcome the sim-to-real gap by improving their simulator with system identification and actuator modeling for their quadruped robot. Our approach to actuator modeling achieves sim-to-real transfer despite lacking access to the underlying dynamics of our actuators.

3 Method

3.1 Soft Robot Modeling

We consider a class of inflated-beam soft robots that utilize fPAMs as their bending actuators [9]. Our hardware implementation features one beam as the main body and a second inflated beam with a smaller radius (the fPAM) attached along the length of the main body. Pressurizing the fPAM causes one side of the main body to shorten, forming a bend (Fig. 3a,b). We model the soft robot arm as an n-link pendulum with hinge joints between links and between the base link and the sliding cart (Fig. 3c). This allows us to model the soft robot arm with a finite number of state variables and use rigid-body simulators. The cart is attached to fixed rails with a prismatic joint. We model each hinge joint as a rotational spring and damper to capture the stiffness and damping of the inflated beam. We assume a linear mapping between the pressure inside the bending actuator and the torque applied at each hinge joint. This results in the following definition for the hinge joint torques:

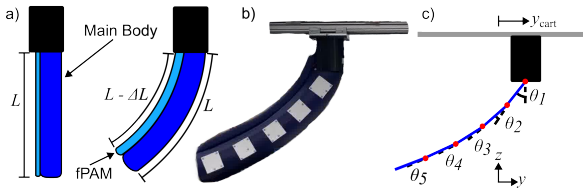

Figure 3: a) Illustration of the fabric Pneumatic Artificial Muscle (fPAM) actuation. Inflating the fPAM causes it to shorten by ΔL. This enforces an asymmetric shortening of the robot's main body, causing it to bend. b) Image of our physical soft robot arm mounted to the mobile cart of a linear actuator. c) Illustration of the 5-link pendulum model. Rigid links (blue) are connected via hinge joints (red) to each other and to the cart (black). This cart is attached to the rails of the linear actuator (gray) with a prismatic joint. We define the cart position y_{cart} and relative joint angles θ_i. We also model rotational springs and dampers at each hinge joint (not visualized).

$$\boldsymbol{\tau}_{\text{model}} := -\mathbf{K}\boldsymbol{\theta} - \mathbf{D}\dot{\boldsymbol{\theta}} + \mathbf{b}p, \tag{1}$$

where $\boldsymbol{\tau}_{\text{model}} \in \mathbb{R}^n$ are the joint torques, $\boldsymbol{\theta} \in \mathbb{R}^n$ are the joint angles, $\dot{\boldsymbol{\theta}} \in \mathbb{R}^n$ are the joint velocities, $p \in \mathbb{R}$ is the fPAM pressure, $\mathbf{K} \in \mathbb{R}^{n\times n}$ is a diagonal matrix for stiffness, $\mathbf{D} \in \mathbb{R}^{n\times n}$ is a diagonal matrix for damping, and $\mathbf{b} \in \mathbb{R}^n$ is a linear mapping from bending actuator pressure to joint torques. We set $n = 5$ and fit $\mathbf{K}$, $\mathbf{D}$, and $\mathbf{b}$ using data from experiments with open-loop commands. We use measurements from the physical hardware for mass and inertia values. We describe these measurements as well as our system identification method, results, and validation in SM-B.

3.2 Learning Dynamic Soft Robot Control Policies

In this work, we focus on two soft robot arm control tasks:

1. ***Free Space Target Reaching:*** Reaching tip position targets in free space. We are particularly interested in targets that are high enough that the soft arm requires swinging to achieve the task.
2. ***Shelf Target Reaching:*** Swinging onto the surface of a shelf (Fig. 1). This task is difficult because the robot needs to coordinate the swinging of the arm with the sliding of the base such that 1) the tip swings above the shelf surface while avoiding a collision, and 2) the tip moves laterally over the shelf edge before the tip swings back down. This task also demonstrates that we can readily apply our framework to tasks involving contact.

We formulate these as RL problems, in which an agent sequentially chooses actions to interact with an environment with the goal of maximizing cumulative reward [35]. For each task, we learn a single policy that can reach a diverse range of target positions, enabling real-time planning. We describe our RL framework here, and provide problem formulation and policy training details in SM-C.

Action Space. The RL agent's action $\mathbf{a} \in \mathbb{R}^2$ includes the commands for our robot's two actuators. The first actuator is a pressure regulator that controls the fPAM pressure. We define the commanded fPAM pressure in kPag as $a_p \in \mathbb{R}$, which is different from the fPAM pressure p in Eq. 1 because it takes time for the regulator to inflate or deflate the fPAM. The second actuator is the linear actuator controlling cart motion. The linear actuator has an internal controller which operates in "Velocity Mode", which requires the user to send velocity commands over time. We define the commanded cart velocity in m/s as $a_{\text{cart-vel}} \in \mathbb{R}$, which is different from the actual cart velocity $v_{\text{cart}} \in \mathbb{R}$ because it takes time to accelerate the cart to a new velocity. Section 3.3 provides a comprehensive description of the actuator dynamics in our simulator, emphasizing the crucial additional details concerning the real actuator dynamics that needed to be accurately modeled in the simulation.

Observation Space. The RL agent's observation $\mathbf{o}$ consists of two parts: $\mathbf{o}_{\text{robot}}$ and $\mathbf{o}_{\text{task}}$. $\mathbf{o}_{\text{robot}} \in \mathbb{R}^8$ includes the cart position $y_{\text{cart}} \in \mathbb{R}$, the cart velocity $v_{\text{cart}} \in \mathbb{R}$, the tip position $\mathbf{x}_{\text{tip}} \in \mathbb{R}^2$, the tip velocity $\mathbf{v}_{\text{tip}} \in \mathbb{R}^2$, the previous cart velocity command $a_{\text{prev-cart-vel}} \in \mathbb{R}$, and the current fPAM pressure $p \in \mathbb{R}$. We found that including $a_{\text{prev-cart-vel}}$ and p improves system performance, and we hypothesize these observations encode critical information for understanding actuator dynamics and generating coordinated actions over time. $\mathbf{o}_{\text{task}}$ is defined specifically for each task. For the free space target reaching task, we define $\mathbf{o}_{\text{task}} := \mathbf{x}_{\text{target}}$, where $\mathbf{x}_{\text{target}} \in \mathbb{R}^2$ is the target position. For the shelf target reaching task, we define $\mathbf{o}_{\text{task}} := (\mathbf{x}_{\text{target}}, d_{\text{target}})$, where $\mathbf{x}_{\text{target}} \in \mathbb{R}^2$ is the target position and $d_{\text{target}} \in \mathbb{R}$ is the difference between the y-coordinates of the target and edge of the shelf (Fig 4).

Reward Function. We define our reward at each timestep, $r \in \mathbb{R}$, to be a weighted sum of rewards from individual reward functions $r = \sum_i w_i r_i$, where $r_i \in \mathbb{R}$ is the reward from the i-th reward function and $w_i \in \mathbb{R}$ is weight associated with the i-th reward function. Specifically, we use:

$$w_{\text{pos}} = 1.0, \quad r_{\text{pos}} = \begin{cases} 1000, & ||\mathbf{x}_{\text{tip}} - \mathbf{x}_{\text{target}}|| < d_{\text{success}} \\ 0, & \text{otherwise} \end{cases}$$

$$w_{\text{vel}} = 0.1, \quad r_{\text{vel}} = ||\mathbf{v}_{\text{tip}}||$$

$$w_{\text{limit}} = 1.0, \quad r_{\text{limit}} = \begin{cases} -100, & |y_{\text{cart}}| > \bar{y} \\ 0, & \text{otherwise} \end{cases}$$

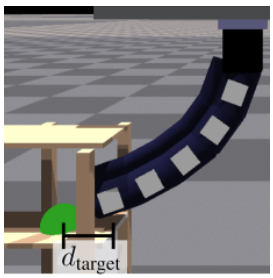

Figure 4: Shelf target reaching task in simulation. $\mathbf{x}_{\text{target}}$ is the target position, which is visualized as a green sphere with radius $d_{\text{success}} = 0.04$ m. d_{target} is the difference between the y-coordinates of $\mathbf{x}_{\text{target}}$ and the edge of the shelf.

where $d_{\text{success}} = 0.04$ m is the maximum tip-to-target distance to count as a success and $\bar{y} = 0.25$ m is the rail limit. The small velocity reward improves training speed because it encourages swinging, which is required for higher target positions. An episode terminates when the cart exceeds the rail limits, the tip position reaches the target position, or the episode exceeds 100 timesteps (3 seconds).

3.3 Sim-to-Real Policy Transfer

One of the main contributions of this work is overcoming the discrepancy between the robot's simulated training environment and the real robot hardware. We find that matching the actuation and sensing in simulation to the behaviors of their real-world counterparts is crucial for achieving zero-shot sim-to-real transfer, while a high-fidelity dynamic model of the soft robot is not necessary. In addition, we concur with previous work that modeling latency in actuators as well as domain randomization also play important roles in the success of sim-to-real transfer [34].

Pressure Regulator Dynamics. Physical pressure regulators have finite air flow, resulting in reduced tracking performance for high-frequency signals. This issue is apparent in our hardware, where the pressure regulator struggles to track abrupt pressure command changes (ex. black and red signals in Fig. 5). We model this behavior in simulation by applying an infinite impulse response (IIR) filter to the pressure command. We implement this using the update rule:

$$p^{k+1} \leftarrow \alpha a_p^k + (1-\alpha)p^k,$$

where $\alpha \in \mathbb{R}$ is an empirically-chosen parameter. This model closely approximates reality (Fig. 5) and serves as a general approach for modeling pressure regulators in soft robot RL frameworks.

Linear Actuator Dynamics. Many actuators come with internal controllers that make them easier to use (e.g. position control for motors), but introduce modeling challenges for sim-to-real transfer. For example, our linear actuator's "Velocity Mode" controller hides the actual cart physics from the user. This controller first requires the user to specify a fixed acceleration $\dot{v}_{\text{fixed}} \in \mathbb{R}$ before running the control policy. Then, the user sends velocity commands over time, and the actuator accelerates the cart at the fixed acceleration until the velocity reaches the latest commanded velocity. This results in a piece-wise linear velocity profile (e.g. red signal in Fig. 6) consisting of constant slopes (constant acceleration) of either 0 or $\pm\dot{v}_{\text{fixed}}$ (e.g. we set $\dot{v}_{\text{fixed}} = 6$ m/s^2).

We design a cart controller in simulation that approximates this behavior without knowing the details of the physical linear actuator controller. Because there are forces on the cart from the rest of the robot, we cannot simply apply a constant force to the cart to achieve constant acceleration. We address this challenge by implementing a low-level cart controller in simulation that performs acceleration tracking when the velocity error is high and velocity tracking when the velocity error is low. We implement this by applying a cart force (in Newtons) $F_{\text{cart}} \in \mathbb{R}$ defined as:

$$F_{\text{cart}} = \begin{cases} m\dot{v}_{\text{fixed}} + k_{\dot{v}}(\dot{v}_{\text{fixed}} - \dot{v}_{\text{cart}}), & |e_v| > 0.1 \text{ (track acc)} \\ k_v e_v, & |e_v| \leq 0.1 \text{ (track vel)} \end{cases}$$

$$e_v = a_{\text{cart-vel}} - v_{\text{cart}} \quad \text{(Note: } a \text{ is for action, not acceleration)}$$

where $e_v \in \mathbb{R}$ is the velocity tracking error, $m \in \mathbb{R}$ is the mass of the robot, $\dot{v}_{\text{fixed}} \in \mathbb{R}$ is the fixed cart acceleration, $\dot{v}_{\text{cart}} \in \mathbb{R}$ is the actual cart acceleration, and $k_{\dot{v}} \in \mathbb{R}$ and $k_v \in \mathbb{R}$ are the acceleration tracking gain and velocity tracking gain, respectively. In the acceleration tracking case, $m\dot{v}_{\text{fixed}}$ is a feedforward term based on $F = ma$, and $k_{\dot{v}}(\dot{v}_{\text{fixed}} - \dot{v}_{\text{cart}})$ is a feedback term to improve tracking on this fixed acceleration. In the velocity tracking case, the velocity is already close to the commanded velocity, so only a small feedback term is required. Our simulated cart motion closely

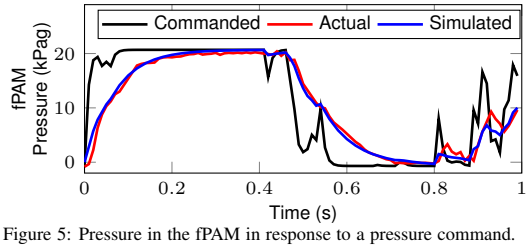

Figure 5: Pressure in the fPAM in response to a pressure command. The fPAM pressure regulator is bandwidth-limited, so we simulate this behavior by applying an infinite impulse response (IIR) filter to the RL command. The RMS error between the actual and simulated pressure is 0.94 kPag, which is small compared to the range of pressure commands (-0.7 to 20 kPag).

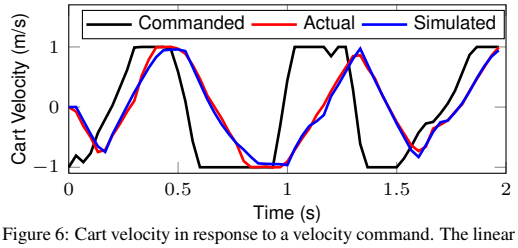

Figure 6: Cart velocity in response to a velocity command. The linear actuator cannot respond immediately to changes in commanded velocity, so we simulate the linear actuator as an acceleration tracker for large velocity errors and a velocity tracker for small errors. The RMS error between the actual and simulated velocity is 0.08 m/s, which is small compared to the range of velocity commands (-1.0 to 1.0 m/s).

approximates the real cart's piece-wise linear velocity (Fig. 6). This strategy can be adapted to overcome sim-to-real difficulties for a broad range of actuators.

Linear Actuator Latency. Physical actuators exhibit latency, which can hinder sim-to-real transfer when left unmodeled in simulation [34]. We model the linear actuator latency in our simulator by adding a time delay between the time the policy outputs a velocity command and the time that the command is sent to the simulator's lower-level controller. Fig. 6 shows that modeling the delay helps our simulated cart's velocity profile closely match the real cart's velocity profile.

Velocity Measurements. During physical experiments, we compute tip velocity with first-order finite-differencing, so we replicate this in our simulated observations.

Domain Randomization. We use domain randomization while training our policy in simulation so that it can handle the remaining sim-to-real gap not captured by our other adjustments. This allows us to run this policy directly on the real robot with no real-world training. See SM-D for details.

4 Results

We evaluate the capabilities of our RL framework by performing demonstrations and experiments on real soft robot arm hardware (hardware details in SM-E). Our learned policies achieve real-time planning and control of agile maneuvers not previously demonstrated on soft robot arms.

4.1 Characterizing Performance for Free Space Target Reaching

In this section, we assess the quality of behaviors generated by our RL framework by showing that its performance is close to that of an approximate upper bound method. The upper bound method uses trajectory optimization (TO) for motion planning and a real-time tracking controller for control (details in SM-F). We emphasize that trajectory optimization is not a practical choice for our task since it requires a time-consuming offline planning stage.

We compare our RL method with the TO method on the free space target reaching task. When training our RL method, we sample target positions $\mathbf{x}_{\text{target-train}} \in [-0.4\text{ m}, 0.0\text{ m}] \times [0.55\text{ m}, 0.7\text{ m}]$. We test our RL method on an array of 54 target positions $\mathbf{x}_{\text{target-test}} \in [-0.4\text{ m}, 0.0\text{ m}] \times [0.55\text{ m}, 0.8\text{ m}]$ (Fig. 7). Targets with z-coordinates above 0.7 m are outside the training distribution. To compare with the approximate upper bound, TO, we precompute 54 trajectories in an offline planning stage, each specifically optimized to reach its respective target position ($\sim$4.7 hours to precompute). All targets are on the left side of the soft robot because the fPAM is designed to bend towards this side.

Fig. 7 shows that our learned policy reaches a wide range of target positions, missing only the most difficult ones. Specifically, targets towards the upper left are difficult because they require the most aggressive swinging to reach, and targets with y = 0 are difficult because the robot needs to swing aggressively to the sides in order to swing the tip towards the rail center (y = 0) while avoiding the rail limits. Despite this, our policy achieved a success rate of 94% for targets within the training distribution, 50% for targets outside the training distribution, and 80% overall. The TO method achieved success rates of 94%, 83%, and 91%, respectively. This suggests that our RL method matches the performance of the TO method within its training distribution, and also has some level of generalization to targets outside of this range, though at a much lower success rate.

Fig. 8 compares tip trajectories achieved on hardware with our RL method and the TO method. These plots show similar trajectories for both methods, suggesting that our RL method approaches the approximate upper bound of performance. We see that the robot reaches the target tip position by building up momentum over multiple swings to increase its tip height. This demonstrates that the free space target reaching task is not possible without well-timed actuation that leverages swinging motion. Such a motion plan would be difficult to design by hand, but our RL method enables real-time planning and control of these types of agile maneuvers for soft robot arms.

We further demonstrate that leveraging swinging is required for task success by comparing our policy to a PID controller, which does not reason about swinging (method and results in SM-G).

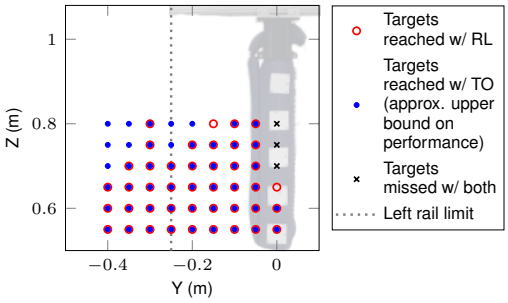

Figure 7: Free space target reaching on hardware for our learned policy (RL) vs. trajectory optimization (TO). Targets above 0.7 m are outside our learned policy's training distribution. We overlay the starting position of the soft robot arm used for each trial.

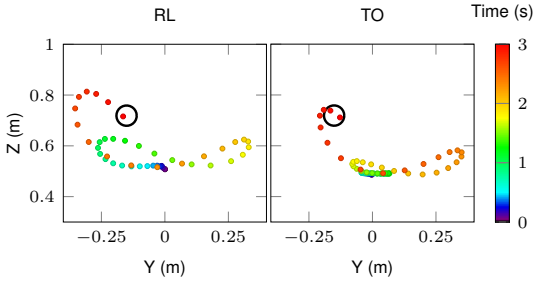

Figure 8: Comparison of learned policy (RL) vs. trajectory optimization (TO) for reaching a target tip position in free space. The black circle is centered at the target position with a radius of 4 cm. Both methods perform high-speed swinging to reach the target position, showing that RL is able to produce motions that are comparable to TO.

4.2 Ablation Study on Shelf Target Reaching

We assess the significance of individual components of our framework through an ablation study with the shelf-reaching task. During training, we sample target positions $\mathbf{x}_{\text{target-train}} \in [-0.48\text{ m}, -0.4\text{ m}] \times [0.58\text{ m}, 0.67\text{ m}]$ and position the shelf such that $d_{\text{target}} \in [-0.05\text{ m}, 0.2\text{ m}]$ (Fig 4). To evaluate each ablation, we deploy the policy on physical hardware and run 3 trials for each of the following shelf target positions (y, z): $(-0.48\text{ m}, 0.6\text{ m})$, $(-0.48\text{ m}, 0.63\text{ m})$, $(0.48\text{ m}, 0.66\text{ m})$, and $(-0.53\text{ m}, 0.63\text{ m})$. We then compute the success rate across these 12 trials and compare the results of these policies with our best-performing policy (Table 1).

Actuator and Sensor Models. We find that our actuator and sensor modeling improvements in our simulator (Sec. 3.3) were crucial for success. We verify this with the following ablations:

- ***No fPAM Smoothing:*** We remove the IIR filter that models the fPAM pressure regulator's dynamics, so pressure commands instantly update the pressure.
- ***No Cart Acceleration Tracking:*** We remove the acceleration tracking mode in our low-level controller, leaving only the velocity tracking mode.
- ***No Cart Action Delay:*** We remove the delay in sending the RL command to the simulated low-level cart controller.
- ***Ground Truth Velocities:*** We use ground truth velocities in our observations instead of velocities computed using first-order finite differencing.

Table 1: Shelf Target Reaching Task Ablation Study

Policy	**Success Rate**
Ours	92%
Actuator and Sensor Model Ablations	
No fPAM Smoothing	0%
No Cart Acceleration Tracking	8%
No Cart Action Delay	0%
Ground Truth Velocities	0%
Domain Randomization Ablations	
Domain Randomization on Observations	25%
Domain Randomization on Dynamics	25%
No Domain Randomization	17%
Architecture Ablations	
MLP Only	0%
LSTM Before MLP	0%
fPAM Actuation Ablation	
Cart Actions Only (No fPAM)	0%

While these policies obtain high success rates in simulation, these ablations result in poor performance on the physical robot due to the larger sim-to-real gap. The failures appear similar to those we saw during initial development before we added these sim-to-real adjustments. We found the underlying causes of these failures by methodically evaluating the failures and comparing the sim and real observations, actions, and dynamics.

When evaluating our learned policy without fPAM smoothing, we noticed that the real fPAM actuations were not well-coordinated with the cart's motion. From there, we compared the commanded and actual fPAM pressures and found that the actual pressure did not immediately track the commanded pressure. When evaluating our learned policy without the linear actuator adjustments, we noticed that the real cart would not be able to change direction quickly enough to avoid the rail limits. We compared the sim and real cart velocity tracking profiles, which helped us model the cart controller more accurately. Finally, when evaluating our learned policy with ground truth velocities, comparing real and sim observations revealed that the velocities computed with first-order finite-differencing were noticeably different from the simulator's ground truth velocities.

Domain Randomization. We find that the best performance on hardware is achieved by training a policy with domain randomization on policy actions. We consider ablations with different domain randomization schemes to verify this:

- ***Domain Randomization on Observations:*** We remove randomization on the actions and add Gaussian noise to the observations.
- ***Domain Randomization on Dynamics:*** We remove randomization on the actions and scale dynamic parameters $(\mathbf{K}, \mathbf{D}, \mathbf{b})$ by a value from a uniform distribution.
- ***No Domain Randomization:*** We remove randomization on the actions.

Within this ablation subset, removing all domain randomization performed the worst, verifying the benefit of domain randomization in crossing the sim-to-real gap. The ablations with observation and dynamics randomization each show a small improvement in success rate compared to the ablation without any randomization, suggesting there is a small sim-to-real gap in our dynamic parameters and our observation measurements. However, there is a large difference in success rate between our best-performing policy and these ablations, which suggests that action randomization is important to address the remaining sim-to-real gap not captured by our adjustments described in Sec. 3.3.

Architecture. Our best-working policy architecture is a multi-layer perceptron (MLP) followed by a Long-Short Term Memory (LSTM) network. We consider ablations with different architectures:

- ***MLP only:*** We remove the LSTM layer so that the policy does not have memory.
- ***LSTM before MLP:*** We rearrange the order of the architecture so that the input first goes into the LSTM before going into the MLP.

Both ablations perform poorly on physical hardware. We expected the MLP without the LSTM to perform worse because memory has been shown to be useful in control tasks with partial observability [36, 37]. The higher success rate for the architecture with the LSTM after the MLP suggests that the MLP provides a useful representation for the LSTM to model temporal dependencies.

fPAM Actuation. We verify that the fPAM is necessary for task success by running an ablation in which we turn off fPAM actuation in both the simulated and real systems (i.e. the robot is limited to cart actions only). The robot had low success rates in both simulation and on physical hardware, demonstrating that both actuators (fPAM and cart) are required for the shelf-reaching task.

5 Limitations

While our work demonstrates significant progress in enabling real-time planning and control for soft robot arms, there are limitations that should be addressed in future work. One limitation is the use of motion capture, which is costly to set up and not suitable for outdoor or field use. Onboard state estimation would allow broader use of our system. Another limitation is that we do not fine-tune the policy in the real world, which could allow adaptation to changes in the environment or robot hardware. Finally, while our work demonstrates that using simple physics-based models for this class of soft robots is an effective approach, it is important to acknowledge that the effectiveness of such models may vary across different soft robot systems and that more complex systems may benefit from higher-fidelity or learning-based models.

6 Conclusion

In this study, we demonstrate that deep reinforcement learning enables real-time planning and control of dynamic maneuvers for soft robot arms. We identify key insights needed to overcome sim-to-real challenges for zero-shot sim-to-real transfer, including domain randomization and improvements to actuator and sensor modeling. Our work provides evidence that accurate actuator and sensor models are crucial for task success, and that agile behaviors can be learned without requiring a high-fidelity model of soft robot dynamics. We demonstrate the effectiveness of our framework through physical experiments that show our soft robot arm executing agile maneuvers in real time.

References

[1] W. Burgard, O. Brock, and C. Stachniss. Safety evaluation of physical human-robot interaction via crash-testing. In *Robotics: Science and Systems III*, pages 217–224, 2008.

[2] H. Jiang, Z. Wang, Y. Jin, X. Chen, P. Li, Y. Gan, S. Lin, and X. Chen. Design, control, and applications of a soft robotic arm. *arXiv preprint arXiv:2007.04047*, 2020.

[3] D. Rus and M. T. Tolley. Design, fabrication and control of soft robots. *Nature*, 521(7553): 467–475, 2015.

[4] F. S. M. Alkhafaji. Modeling and control high speed robotic arm for industrial applications. In *Global Congress on Electrical Engineering*, pages 121–130, 2021.

[5] S. Mori, K. Tanaka, S. Nishikawa, R. Niiyama, and Y. Kuniyoshi. High-speed and lightweight humanoid robot arm for a skillful badminton robot. *IEEE Robotics and Automation Letters*, 3 (3):1727–1734, 2018.

[6] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, 2008.

[7] A. Takahashi, M. Sato, and A. Namiki. Dynamic compensation in throwing motion with high-speed robot hand-arm. In *IEEE International Conference on Robotics and Automation*, pages 6287–6292, 2021.

[8] Y. Yamakawa, A. Namiki, and M. Ishikawa. Motion planning for dynamic knotting of a flexible rope with a high-speed robot arm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 49–54, 2010.

[9] N. D. Naclerio and E. W. Hawkes. Simple, Low-Hysteresis, Foldable, Fabric Pneumatic Artificial Muscle. *IEEE Robotics and Automation Letters*, 5(2):3406–3413, 2020.

[10] J. M. Bern, Y. Schnider, P. Banzet, N. Kumar, and S. Coros. Soft robot control with a learned differentiable model. In *IEEE International Conference on Soft Robotics*, pages 417–423, 2020.

[11] T. George Thuruthel, F. Renda, and F. Iida. First-order dynamic modeling and control of soft robots. *Frontiers in Robotics and AI*, 7, 2020.

[12] M. Grube, J. C. Wieck, and R. Seifried. Comparison of modern control methods for soft robots. *Sensors*, 22(23):9464, 2022.

[13] I. Robert J. Webster and B. A. Jones. Design and kinematic modeling of constant curvature continuum robots: A review. *The International Journal of Robotics Research*, 29(13):1661–1683, 2010.

[14] A. Centurelli, L. Arleo, A. Rizzo, S. Tolu, C. Laschi, and E. Falotico. Closed-loop dynamic control of a soft manipulator using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 7(2):4741–4748, 2022.

[15] O. Fischer, Y. Toshimitsu, A. Kazemipour, and R. K. Katzschmann. Dynamic task space control enables soft manipulators to perform real-world tasks. *Advanced Intelligent Systems*, page 2200024, 2022.

[16] A. Boeing and T. Bräunl. Leveraging multiple simulators for crossing the reality gap. In *International Conference on Control Automation Robotics & Vision*, pages 1113–1119, 2012.

[17] S. Koos, J.-B. Mouret, and S. Doncieux. Crossing the reality gap in evolutionary robotics by promoting transferable controllers. In *Conference on Genetic and Evolutionary Computation*, GECCO '10, page 119–126. Association for Computing Machinery, 2010. ISBN 9781450300728.

[18] R. K. Katzschmann, C. D. Santina, Y. Toshimitsu, A. Bicchi, and D. Rus. Dynamic motion control of multi-segment soft robots using piecewise constant curvature matched with an augmented rigid body model. In *IEEE International Conference on Soft Robotics*, pages 454–461, 2019.

[19] N. Naughton, J. Sun, A. Tekinalp, T. Parthasarathy, G. Chowdhary, and M. Gazzola. Elastica: A compliant mechanics environment for soft robotic control. *IEEE Robotics and Automation Letters*, 6(2):3389–3396, 2021.

[20] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan. Modeling and control of soft robots using the koopman operator and model predictive control. *Robotics: Science and Systems*, 2019.

[21] R. Jitosho, N. Agharese, A. M. Okamura, and Z. Manchester. A dynamics simulator for soft growing robots. In *IEEE International Conference on Robotics and Automation*, pages 11775–11781, 2021.

[22] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. Daydreamer: World models for physical robot learning. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 2226–2240, 14–18 Dec 2023.

[23] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation*, pages 3803–3810. IEEE, 2018.

[24] J. Dao, H. Duan, K. R. Green, J. W. Hurst, and A. Fern. Learning to walk without dynamics randomization. In *2nd Workshop on Closing the Reality Gap in Sim2Real Transfer for Robotics (Robotics: Science and Systems)*, 2020.

[25] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid Motor Adaptation for Legged Robots. In *Proceedings of Robotics: Science and Systems*, 2021.

[26] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.

[27] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 91–100, 2022.

[28] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. In *Proceedings of Robotics: Science and Systems*, 2021.

[29] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[30] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving Rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[31] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, Y. Narang, J.-F. Lafleche, D. Fox, and G. State. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. *arXiv preprint arXiv:2210.13702*, 2022.

[32] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[33] E. Tzeng, C. Devin, J. Hoffman, C. Finn, P. Abbeel, S. Levine, K. Saenko, and T. Darrell. *Adapting Deep Visuomotor Representations with Weak Pairwise Constraints*, pages 688–703. Springer International Publishing, 2020. ISBN 978-3-030-43089-4.

[34] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems*, 2018.

[35] R. S. Sutton and A. G. Barto. *Reinforcement Learning - An Introduction*. Adaptive computation and machine learning. MIT Press, 1998. ISBN 978-0-262-19398-6.

[36] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. P. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, pages 1–5, 2019.

[37] P. Zhu, X. Li, P. Poupart, and G. Miao. On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv:1704.07978*, 2017.