Learning a Controller for Soft Robotic Arms and Testing its Generalization to New Observations, Dynamics, and Tasks

Carlo Alessi[1,2], Helmut Hauser[3], Alessandro Lucantonio[4], and Egidio Falotico[1,2] *(Member, IEEE)*

***Abstract*— Recently, learning-based controllers that leverage mechanical models of soft robots have shown promising results. This paper presents a closed-loop controller for dynamic trajectory tracking with a pneumatic soft robotic arm learned via Deep Reinforcement Learning using Proximal Policy Optimization. The control policy was trained in simulation leveraging a dynamic Cosserat rod model of the soft robot. The generalization capabilities of learned controllers are vital for successful deployment in the real world, especially when the encountered scenarios differ from the training environment. We assessed the generalization capabilities of the controller in silico for four tests. The first test involved the dynamic tracking of trajectories that differ significantly in shape and velocity profiles from the training data. Second, we evaluated the robustness of the controller to perpetual external end-point forces for dynamic tracking. For tracking tasks, it was also assessed the generalization to similar materials. Finally, we transferred the control policy without retraining to intercept a moving object with the end-effector. The learned control policy has shown good generalization capabilities in all four tests.**

***Index Terms*— Modeling, Control, and Learning for Soft Robots, Learning and Adaptive Systems, Soft Robot Applications.**

I. Introduction

The modeling and control of continuum and soft robotic arms are still challenging problems due to hyper-redundancy, complex dynamics, and non-linear properties of soft materials [1], [2]. Researchers have proposed several modeling techniques [3], which lead to the development of a variety of model-based and model-free control strategies [4].

The most used approaches for deriving forward dynamics or kinematics models for continuum and soft robots have been geometrical models like the piece-wise constant curvature (PCC) approximation [5]. These models were used within proportional-derivative (PD) control laws for dynamic task space control of a soft manipulator performing a variety of real-world tasks [6]. As an alternative [7] described the shape of a synthetic planar soft robot analytically by a polynomial curvature model, which was used within an extended PD regulator to achieve perfect steady state control in generic curvature conditions. However, the suitability of these model approximations degrades when the robot is subject to non-negligible external forces and unpredictable interactions with the environment.

Data-driven modeling and control are also viable approaches [8]. In the context of supervised learning, reservoir computing was used to emulate the nonlinear dynamics of a pneumatic soft robotic arm and learning to reproduce trajectories [9]. Moreover, continual learning was proposed to tune the weights of a neural network-based controller to adapt to changes in the dynamics of a soft robotic arm due to loading conditions without catastrophic forgetting [10].

The success of reinforcement learning (RL) for behavior generation in rigid robots prompted interest in its application to continuum and soft robots [11]. The first applications of RL for soft robot control relied on discretized state-action spaces. The well-known Q-Learning algorithm was applied to train a model-free, open-loop, static controller for a multi-segment planar pneumatic soft arm [12]. The same algorithm was used to compare control policies learned in simulation and directly on the real soft robot subject to tip loads [13]. The SARSA algorithm was applied to obtain a static controller of position and stiffness for a hybrid soft robotic arm in a multi-agent setting, which considered the actuators as individual agents cooperating in a shared environment [14]. Following the recent advances of Deep RL, now it is also possible to consider continuous states and actuations. For example, Satheeshbabu et al. [15] learns via Deep Q-Learning transition between way-points in quasi-static conditions with an open-loop position controller for a pneumatic soft robotic arm capable of bending and twisting that was trained in simulation, leveraging a Cosserat rod model. The same authors extended the work by increasing the dexterity of the soft robotic arm and attaining a closed-loop controller for precise quasi-static positioning via Deep Deterministic Policy Gradient approach [16]. The authors validated both controllers on unseen payloads. Our work presented here, however, uses a dynamic (not just quasi-static) Cosserat rod model, subject to pressure-induced stretching and bending.

In a similar work, Centurelli et al. [17] learned a closed-loop controller for dynamic trajectory tracking for a soft robotic arm via Trust Region Policy Optimization leveraging an approximation of the robot forward dynamic model obtained by a recurrent neural network. In Naughton et al. [18], the authors applied several deep reinforcement learning algorithms to learn in simulation various control policies using a synthetic soft arm based on Cosserat theory. All these approaches confirm that Deep RL algorithms are suitable candidates for generating control policies for soft robots.

*This work was supported by the European Union's Horizon 2020 Research and Innovation Programme under the Specific Grant Agreement No. 945539 (Human Brain Project SGA3).

[1]The BioRobotics Institute, Scuola Superiore Sant'Anna, Pisa, Italy (email: {c.alessi, e.falotico}@santannapisa.it).

[2]Department of Excellence in Robotics and AI, Scuola Superiore Sant'Anna, Pisa, Italy.

[3]Department of Engineering Mathematics, University of Bristol, Bristol, UK (email: helmut.hauser@bristol.ac.uk).

[4]Department of Mechanical and Production Engineering, Aarhus University, Aarhus, Denmark (email: a.lucantonio@mpe.au.dk).

2023 IEEE International Conference on Soft Robotics (RoboSoft) | 979-8-3503-3222-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ROBOSOFT55895.2023.10121988

However, these works do not explore common problems for RL-based controllers: the abilities to generalize to new observations, environment dynamics, and tasks [19]. In this work, we adopt an approach similar to [17], [18]. We propose a closed-loop controller for dynamic trajectory tracking tasks using a pneumatic soft robotic arm trained via deep reinforcement learning using Proximal Policy Optimization. The control policy is learned in simulation leveraging a dynamic Cosserat rod model of the soft robot. However, we diversify the validation producing different target velocities profiles to investigate in silico the generalization capabilities and limitations of the controller in various conditions and tasks.

Specifically, we evaluate the generalization capabilities of the controller in silico on four tests: (i) tracking trajectories of different geometries and velocities; (ii) tracking trajectories subject to constant external forces applied to the end-effector; (iii) tracking trajectories using different material properties; and (iv) intercepting an object moving at various velocities towards the workspace. Note that all of them are carried out without retraining. The rest of the paper is as follows. Section II describes the soft robotic platform, the Cosserat rod model, and the training process to solve the control problem of dynamic tracking. Section III reports and discusses the results obtained for the four generalization tests. Section IV concludes with an insight into future works.

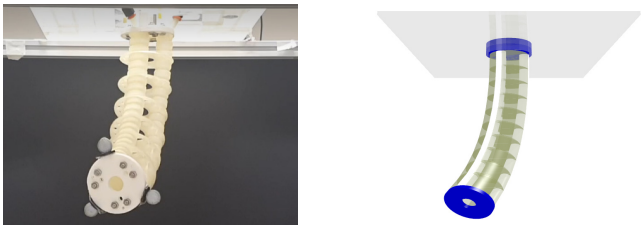
II. Materials and Methods

A. Robotic Platform: The AM I-Support

The AM I-Support is a 3D-printed soft robotic arm with three elliptical pneumatic chambers that can generate large movements by combining stretching and bending [20]. As shown in Fig. 1a, two terminal plates (top and bottom) confine the modules, six rings distributed along the body constrain the chambers, while nuts and bolts assemble the parts. The soft robotic arm is characterized by a cross-section of radius 30 mm, an overall length of ∼202 mm, and ∼183 g overall weight. The pneumatic chambers are ∼180 mm long, while the top and bottom terminal are ∼20 mm and ∼5 mm long, respectively. The actuators are distributed axially, at a radial distance of 20 mm from the cross-section centroid, and equally spaced by 120° around the centre. The arm was fabricated using the soft material thermoplastic polyurethane with 80 Shore A hardness (TPU 80 A LF, by BASF^{TM}), which is characterized by 17 MPa tensile strength and elongation at break of 471%.

B. Cosserat Rod Model

The soft robotic arm was modeled as a Cosserat rod with constant cross-section and homogeneous material properties by extending the Cosserat theory introduced in [21] to account for the pneumatic actuation. A rod is described by a center-line $\bar{\boldsymbol{x}}(s,t) \in \mathbb{R}^3$ and orthogonal rotation matrix $\boldsymbol{Q}(s,t) = \{\bar{\boldsymbol{d}}_1, \bar{\boldsymbol{d}}_2, \bar{\boldsymbol{d}}_3\}^{-1}$. Here, t is time and $s \in [0, L]$ is the material coordinate of a rod of length L. $\boldsymbol{Q}$ transforms vectors from global frame to the local frame via $\boldsymbol{x} = \boldsymbol{Q}\bar{\boldsymbol{x}}$, and vice versa $\bar{\boldsymbol{x}} = \boldsymbol{Q}^{\intercal}\boldsymbol{x}$. If the rod is unsheared, $(\bar{\boldsymbol{d}}_1, \bar{\boldsymbol{d}}_2)$ spans the normal-binormal plane of the cross-section, and



(a) (b)

Fig. 1: Robotic platform. (a) AM I-Support. (b) Rendering of the used computational model.

$\bar{\boldsymbol{d}}_3$ points along the center-line tangent ($\partial_s \bar{\boldsymbol{x}} = \bar{\boldsymbol{x}}_s$). The deformations that the rod can undergo are expressed by the shear/stretch vector $\boldsymbol{\sigma}(s,t)$ and the bend/twist vector $\boldsymbol{\kappa}(s,t)$. Shearing and stretching deviate $\boldsymbol{d}_3$ from $\bar{\boldsymbol{x}}_s$, $\boldsymbol{\sigma} = \boldsymbol{Q}(\bar{\boldsymbol{x}}_s - \bar{\boldsymbol{d}}_3) = \boldsymbol{Q}\bar{\boldsymbol{x}}_s - \boldsymbol{d}_3$ in the local frame. The curvature vector $\boldsymbol{\kappa}$ encodes $\boldsymbol{Q}$'s rotation rate along the material coordinate $\partial_s \boldsymbol{d}_j = \boldsymbol{\kappa} \times \boldsymbol{d}_j$, while angular velocity $\boldsymbol{\omega}$ is defined by $\partial_t \boldsymbol{d}_j = \boldsymbol{\omega} \times \boldsymbol{d}_j$. The rod dynamics is then governed by the following set of nonlinear differential equations:

$$\rho A \cdot \partial_t^2 \bar{\boldsymbol{x}} = \partial_s \left(\frac{\boldsymbol{Q}^{\intercal}\boldsymbol{S}\boldsymbol{\sigma}}{e} \right) + e\bar{\boldsymbol{f}} \tag{1}$$

$$\frac{\rho \boldsymbol{I}}{e} \cdot \partial_t \boldsymbol{\omega} = \partial_s \left(\frac{\boldsymbol{B}\boldsymbol{\kappa}}{e^3} \right) + \frac{\boldsymbol{\kappa} \times \boldsymbol{B}\boldsymbol{\kappa}}{e^3} + \left(\boldsymbol{Q}\frac{\bar{\boldsymbol{x}}_s}{e} \times \boldsymbol{S}\boldsymbol{\sigma} \right) + \left(\rho \boldsymbol{I} \cdot \frac{\boldsymbol{\omega}}{e} \right) \times \boldsymbol{\omega} + \frac{\rho \boldsymbol{I}\boldsymbol{\omega}}{e^2} \cdot \partial_t e + e\boldsymbol{c}, \tag{2}$$

where $\boldsymbol{B}$ is the bend/twist stiffness matrix, $\boldsymbol{S}$ is the shear/stretch stiffness matrix, ρ is the constant material density, A is the cross-sectional area, $\boldsymbol{I}$ is the second area moment of inertia, $\bar{\boldsymbol{f}}$ is the external force, $\boldsymbol{c}$ is the external couple, and $e = |\bar{\boldsymbol{x}}_s|$ is the local stretching.

The pressurization of the pneumatic chambers of the AM I-Support produces an internal force along $\boldsymbol{d}_3$ normal to the rod cross-section and a bending moment. To describe the deformations of the robot, we modeled pressure-induced strains as spontaneous stretching and bending, modifying the rest configuration of the arm dynamically. The rod is subject to gravity, viscous forces, viscous torques, and external forces applied to the free-end which can be integrated into body dynamics via $\bar{\boldsymbol{f}}$ and $\boldsymbol{c}$ in (1)-(2). The rest length of the rod and the cross-section radius were directly measured from the physical prototype [20]. We computed the effective cross-sectional area and the second area moment of inertia considering the actuator geometry. The material density was taken from the TPU datasheet, the Young Modulus was fitted from experimental stretching data, and the damping coefficients were optimized on dynamic stretching and bending data. This computational environment was used for training and testing the controller.

C. Control Architecture

To solve a trajectory tracking task using this soft robotic arm, we adopt the closed-loop control scheme shown in Fig. 2. An arbitrary trajectory generator provides reference positions

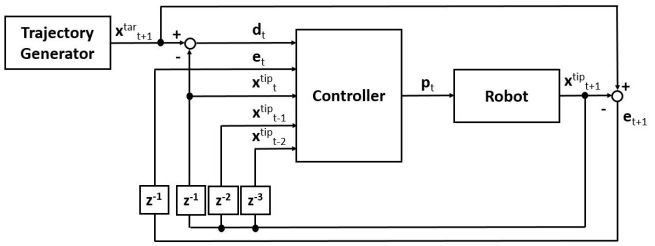


Fig. 2: Control scheme with z^{-1} discrete time delay operator.

$\boldsymbol{x}_{t+1}^{tar} \in \mathbb{R}^3$, i.e., the desired position in Cartesian coordinates of the robot end-effector for the next time step $t+1$. The controller is implemented as a feed-forward neural network with two hidden layers, each with 64 neurons with $tanh$ activation function. The output layer has a linear activation function. The controller takes as input

$$\boldsymbol{x}_t = \left[\boldsymbol{d}_t, \boldsymbol{e}_t, \boldsymbol{x}_t^{tip}, \boldsymbol{x}_{t-1}^{tip}, \boldsymbol{x}_{t-2}^{tip}\right] \in \mathbb{R}^{15}, \tag{3}$$

where $\boldsymbol{d}_t = \boldsymbol{x}_{t+1}^{tar} - \boldsymbol{x}_t^{tip}$ is the distance vector between the current desired position and the current free-end position of the arm $\boldsymbol{x}_t^{tip}$ measured before the actuation, $\boldsymbol{e}_t = \boldsymbol{x}_t^{tar} - \boldsymbol{x}_t^{tip}$ is the current tracking error, and $\boldsymbol{x}_{t-1}^{tip}$ and $\boldsymbol{x}_{t-2}^{tip}$ are the two previous positions of the robot end-effector. The vector $\boldsymbol{d}_t$ therefore provides the controller with a minimal prediction of the future target position. The error vector $\boldsymbol{e}_t$ measures how well the tracking proceeds in line with standard closed-loop controllers. Finally, $\boldsymbol{x}_t^{tip}$, $\boldsymbol{x}_{t-1}^{tip}$, and $\boldsymbol{x}_{t-2}^{tip}$ provide the controller with a simple short-term memory that allows it to infer the velocity and acceleration of the soft robotic arm. The controller outputs three pressure commands for the three chambers $\boldsymbol{p}_t = [p_1, p_2, p_3]$ limited between 0 and 3.5 bar.

The initial values are $\boldsymbol{d}_0 = \boldsymbol{x}_1^{tar} - \boldsymbol{x}_0^{tip}$, $\boldsymbol{e}_0 = \boldsymbol{0}$, and $\boldsymbol{x}_0^{tip} = \boldsymbol{x}_{-1}^{tip} = \boldsymbol{x}_{-2}^{tip} = [0, 0, -L]$. The control loop operates at 10 Hz frequency, actuating the robot every Δt=0.1 s.

D. Reinforcement Learning Algorithm

We solve the control problem using deep Reinforcement Learning. In general, in RL an agent receives at each time step t an observation o_t from the environment, which is a subset of the full environment state s_t. The agent acts according to a policy π mapping states/observations to actions, which can be deterministic or stochastic. The agent receives a scalar reward $r(s, a)$ indicating the current task performance. Let the return $G_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$ be the discounted sum of future rewards, with discount factor $\gamma \in [0, 1]$. The agent aims to maximize the expected return $\mathbb{E}_\pi [G_0|s_0]$. The state-value function is defined as $V_\pi(s) = \mathbb{E}_\pi [G_t|s_t]$. The action-value function is defined as $Q_\pi(s, a) = \mathbb{E}_\pi [G_t|s_t, a_t]$. The advantage value function $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$ expresses whether the action a is better or worse than an average action the policy π takes in the state s.

In our setting, the agent is the controller implemented as a neural network, and the environment is the soft robotic arm modeled as a Cosserat rod. Therefore, the agent receives observations $s_t = \boldsymbol{x}_t$ and outputs actions $a_t = \boldsymbol{p}_t$ (see Fig. 2). State and action spaces are each normalized between -1 and 1 to increase numerical stability of the training process. The reward is defined as

$$r_t = \begin{cases} -10 & \text{if NaN} \\ -e_t + b(e_t) & \text{otherwise,} \end{cases} \tag{4}$$

where the penalty term of -10 was applied to discourage actions that would cause numerical instabilities as proposed by [18], $e_t = ||\boldsymbol{e}_t||$ is the norm of the tracking error, and an inductive bias $b(\cdot)$ is provided as incentive to explore

$$b(e) = \begin{cases} 0.05 & 0.03 < e \leq 0.05 \\ 0.1 & 0.01 < e \leq 0.03 \\ 0.2 & e \leq 0.01. \end{cases} \tag{5}$$

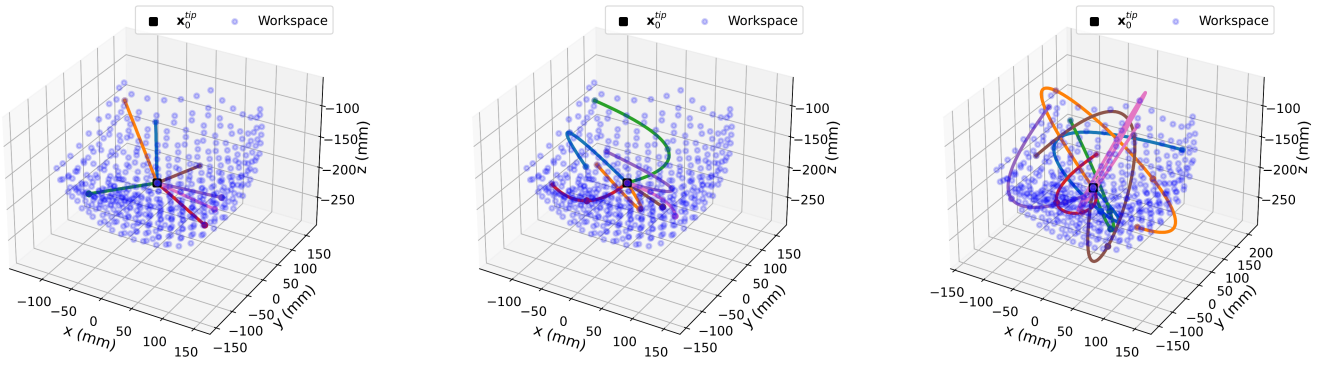
1) Proximal Policy Optimization: The controller is learned via Proximal Policy Optimization (PPO), a policy-gradient method appropriate for continuous control tasks [22]. In particular, we adopt the reliable implementation provided by [23]. The algorithm jointly optimizes a stochastic policy $\pi(a|s)$ and a value-function approximator. PPO alternates between sampling data from the policy through interaction with the environment and performing optimization on the sampled data using stochastic gradient descent (SGD) to maximize the objective

$$\mathbb{E}\left[\min\left(\rho_t(\pi) \cdot \hat{A}_t, \text{clip}(\rho_t(\pi), 1-\epsilon, 1+\epsilon) \cdot \hat{A}_t\right)\right], \tag{6}$$

where $\rho_t(\pi) = \frac{\pi(a_t|s_t)}{\pi_{old}(a_t|s_t)}$ is the ratio of the probability of selecting an action under the current policy π and the probability of taking it with the policy π_{old} that collected the current batch of data, ϵ=0.2 is the clipping parameter, and $\hat{A}$ is an estimator of the advantage function. This loss encourages the policy to select actions with a positive advantage while discouraging large policy updates via clipping.

E. Training Process

The control policy was optimized using PPO. For each episode, a random trajectory was produced. The starting point was the resting tip position $\boldsymbol{x}_0^{tip}$ and two additional way-points were uniformly sampled from 512 positions in the workspace. Target trajectory $\boldsymbol{x}^{tar}$ was then produced through interpolation of these three points using a cubic spline (see Fig. 3b). This was redone for each training episode. This ensured that the controller visited different parts of the workspace. The duration of the training episode was fixed at T=10 s for each target trajectory. Note that since the space traveled in these 10s depended on how far apart the sampled way-points were, the training trajectories had different velocity profiles. This approach ensured that the controller experienced a wide range of velocities $\Delta \boldsymbol{x}^{tar}$. Through this process, we obtained learning data points with velocities in the range $[0, 0.10]$ m/s with a mean of 0.025 ± 0.016 m/s (see also Fig. 5a orange curve). Each episode starts with the robot at rest facing vertically downward (see Fig. 1). The episode terminated when the



(a) (b) (c)

Fig. 3: Example of target trajectories. The starting point is the resting tip position $\boldsymbol{x}_0^{tip}$. Additional waypoints are sampled uniformly from the workspace. Target trajectories $\boldsymbol{x}^{tar}$ are generated by interpolating $\boldsymbol{x}_0^{tip}$ and the waypoints using a cubic spline. (a) 3D straight lines (1 waypoint, T=10 s); (b) 3D curves (2 waypoints, T=10 s); (c) 3D curves (3 waypoints, T=15 s). The controller was trained on curves with two waypoints and tested on straight lines and curves with three waypoints.

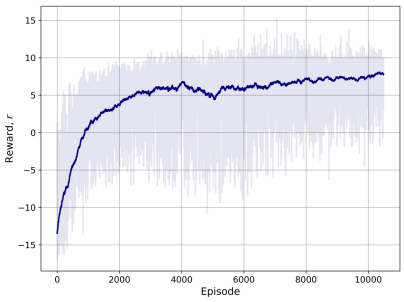
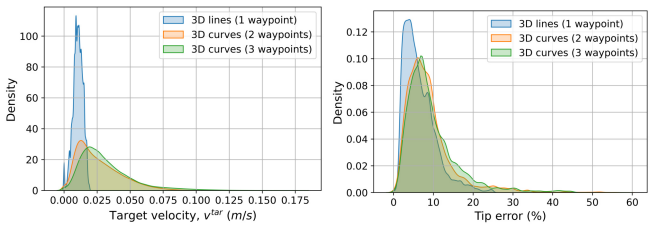


Fig. 4: Learning curve showing cumulative episode reward.



(a) (b)

Fig. 5: Statistics for the trajectory tracking tasks. (a) Distribution of target velocities for each task. (b) Error distribution for each task. The control policy trained on 3D curves (2 waypoints) generalizes to different trajectories and velocities.

entire target trajectory was done (i.e., the time limit of 10s was reached) or when numerical problems occurred.

We trained a stochastic policy to solicit exploration in the environment. After training, we used a deterministic and greedy policy to exploit the best actions learned. After an empirical model selection and hyper-parameter tuning, the learning took place over 1.2 million time steps (i.e., $\sim$10k episodes), equivalent to about 33 hours of learning experience in silico. The training episodes were collected using N=8 parallel agents interacting with the environment for M=64 time steps per policy update. At each iteration, the policy was optimized on the current $N{\cdot}M$ samples with SGD for ten epochs using four mini-batches and a learning rate of 0.00025. The training lasted about 14 hours on a standard laptop (Intel i7-7500U Processor, 8 GB RAM).

The learning curve in Fig. 4 shows the sum of the rewards the agent received in each training episode. The light blue curve is noisy because of the intrinsic explorative behavior of training a stochastic policy and the fact that each episode generates a new target trajectory $\boldsymbol{x}^{tar}$. Nonetheless, the trend of the exponential moving average (dark blue) is increasing.

III. Results and Discussion

After learning the stochastic policy, we evaluated its greedy (deterministic) version. We conducted four tests to investigate its generalization abilities. In particular, first, we tested how the controller could track trajectories of different geometries and velocities profiles. Second, we assessed the robustness of the controller to external forces of various magnitudes and directions applied to the robot's end-effector during trajectory tracking. Third, we evaluated the generalization of the policy to different material stiffnesses for dynamic tracking. As a performance metric, we adopted the mean and standard deviation of the normalized tip error e/L, i.e., the error in percentage of the robot length L. In addition, we measured the spread of the normalized tip error using the interquartile range (IQR), which is robust to extreme outliers. Finally, we deployed the controller to make the soft robot tip intercept the trajectory of a moving object.

TABLE I: Tip error e/L (%) on dynamic trajectory tracking.

Trajectory	mean $\pm$ std (%)	IQR (%)
3D curves (2 waypoints)	8.86 $\pm$ 6.22	5.60
3D lines (1 waypoint)	6.56 $\pm$ 4.03	5.11
3D curves (3 waypoints)	9.61 $\pm$ 6.61	6.56

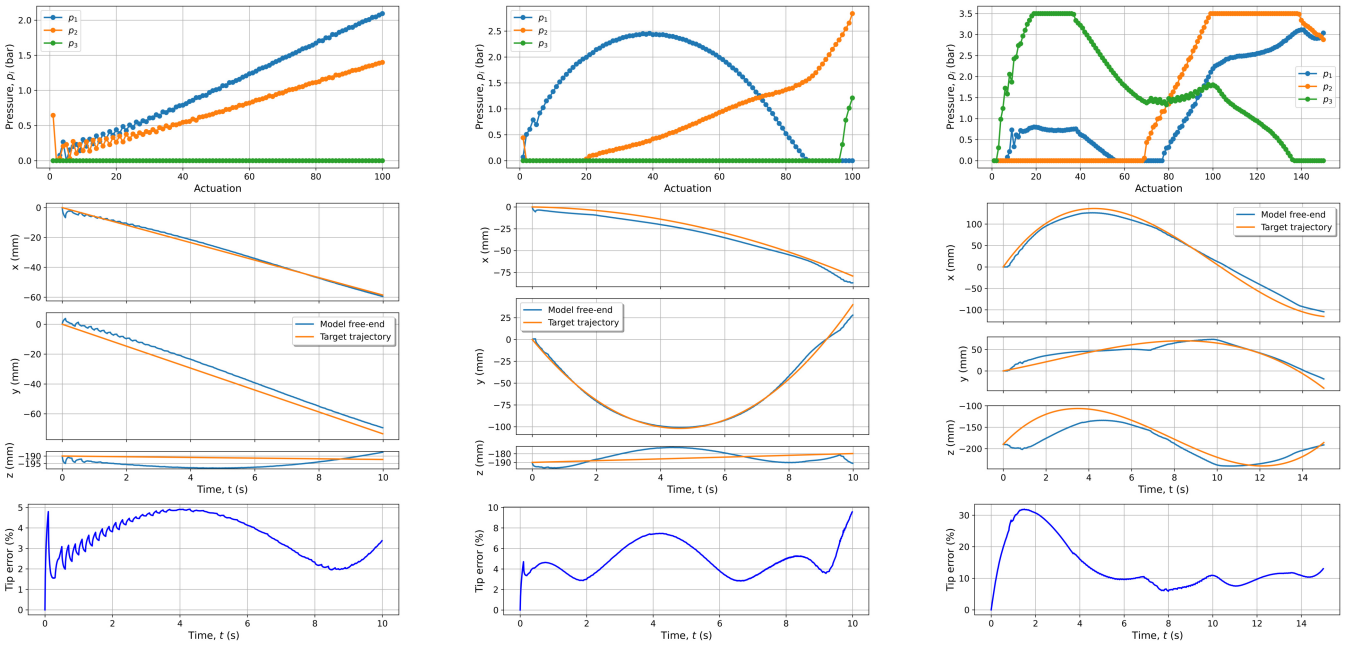


Fig. 6: Controller evaluation on three sample trajectories: (left) straight line (1 waypoint); (center) curve (2 waypoints); (right) curve (3 waypoints). The controller outputs various pressure profiles to track target trajectories geometrically different.

A. Generalization to Trajectory Tracking Tasks

We tested the control policy trained on trajectories generated from two waypoints on three different sets of 100 generated tracking tasks (see Fig. 3).

As a baseline, we assessed the performance on trajectories sampled from the same distribution as the ones used for training, i.e., 3D curves (two way-points, with the same starting point, duration T=10 s). The velocities for this task ranged in $[0, 0.10]$ m/s with a mean of 0.025 ± 0.016 m/s. On this set of trajectories, the controller achieved a mean tip error of 8.86 ± 6.22%L and 5.60%L IQR, see Table I.

For the first test of generalization capabilities, we employed trajectories that differed from the training set in geometry and velocity profile (see Fig. 5a). The first group of testing trajectories included 3D straight lines (1 way-point, T=10 s) with velocities in the range $[0, 0.019]$ m/s with a mean of 0.01 ± 0.004 m/s. The second group included 3D curves (3 way-points, T=15 s) with velocities in the range $[0, 0.178]$ m/s with a mean of 0.031 ± 0.02 m/s. The mean tip error performance attained on these tasks were 6.56 ± 4.03%L, and 9.61 ± 6.61% respectively (see Table I for a comparison). The controller tracked 3D straight lines better than the baseline (i.e., 3D curves with two way-points) and performed slightly worse on tracking 3D curves with three way-points. Fig. 5b shows the distributions of the tip errors for each task. The controller achieved good results in tracking trajectories that differed not only geometrically (e.g., lines and curves) but also in the velocity profile required to follow them. The hypothesis is that the learned policy has generalized to different tracking tasks. This was verified quantitatively by conducting a Kolmogorov-Smirnov statistical test [24] on the velocity distributions (see Fig. 5a). The test confirmed that the distributions are pair-wise different. In particular, the p-value was 0 for the three tests, rejecting the null hypothesis that the distributions are identical. Fig. 6 shows examples of actuation, dimension-wise tracking, and tracking error for each trajectory type.

TABLE II: Tip error e/L (%) for trajectory tracking subject to random perpetual external endpoint forces.

Trajectory	f_{ext} (N)	mean $\pm$ std (%)	IQR (%)
3D curves (2 waypoints)	0.0	8.25 $\pm$ 5.64	5.23
3D curves (2 waypoints)	0.1	8.31 $\pm$ 5.73	5.16
3D curves (2 waypoints)	0.5	8.75 $\pm$ 6.30	5.42
3D curves (2 waypoints)	1.0	9.89 $\pm$ 7.59	5.80

B. Generalization to External Forces

External forces applied to soft robots can significantly alter their dynamics. In this experiment, we evaluate the robustness of the controller to perpetual external forces applied to the end-effector in dynamic trajectory tracking tasks. The force $\boldsymbol{f}_{\text{ext}} = [f_x, f_y, f_z]$ includes the special case of a standard payload in which f_x=f_y=0. After sampling the force vector components from a normal distribution, $\boldsymbol{f}_{ext}$ was scaled to the desired magnitude. We investigated three different magnitudes, i.e., $f_{ext} \in \{0.1, 0.5, 1.0\}$ N. When the soft arm was at rest, the force caused an average deflection in the direction of $\boldsymbol{f}_{ext}$ of 1.16 ± 0.32%L, 5.82 ± 1.56%L, and 11.65 ± 3.06%L, respectively. The controller did not have any explicit information about the perturbations. The controller evaluated on 100 random trajectories for each of the three magnitudes, each with a different endpoint force, shows comparable performance with the no-force case (see Table

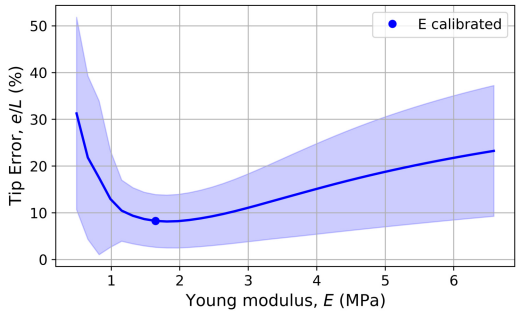


Fig. 7: Controller generalization to a range of Young Moduli.

II). Despite the wide range of force magnitudes, i.e., between f_{ext}=0.1 N and f_{ext}=1.0 N, the tip error increased only by around 1%L. Therefore, the control policy trained without disturbances generalized well to perpetual external forces applied to the end-effector post-training.

C. Generalization to Material Properties

Materials play a crucial role in soft robotics. Factors like temperature changes or material degradation could nonlinearly modify the stiffness of soft robots. Similarly, soft robots with equal geometries but different material properties could attain significantly different deformations that affect the reachable workspace. Therefore, it is interesting to evaluate the generalization of a controller to diverse materials. In particular, we tested how the control policy generalized to different Young Moduli of the Cosserat rod for tracking trajectories. The calibrated value of the Young modulus of the rod used to train the controller was E=1.65 MPa. The learned controller was tested on Young Moduli ranging from 0.49 MPa to 6.59 MPa in steps of ΔE=0.1E, respectively 0.3 and four times the calibrated value. This range is reasonable for soft robotics applications. For each value of E, we averaged the normalized tip errors over 100 target trajectories sampled from two waypoints and T=10 s. As shown in Fig. 7, the controller generalized fairly well for values of E that deviated moderately from the calibrated value. However, the mean and standard deviation of the normalized tip error increased with ΔE. Notice how the error curve was asymmetric to the calibrated stiffness. For lower values of the Young Modulus, the tracking error increased faster than for higher values. This could be because softer materials undergo larger deformations under the same applied pressure, making them harder to control. Conversely, stiffer materials reduced the reachable workspace. As a result, the controller could not track all the points along the generated trajectories as effectively. In summary, the controller generalized well (e.g., average tip error less than 9%L) to similar materials, having Young Modulus in the range $[0.9E, 1.4E]$.

D. Generalization to Trajectory Interception Tasks

After assessing that the control policy can generalize to track different trajectories at various speeds, we deployed the controller to intercept a moving object. Again, the control policy was not retrained for solving the new task. The object was a sphere of radius R_{obj}=25 mm identified by its centroid $\boldsymbol{x}^{tar}$ travelling a path towards the workspace. The object's initial position $\boldsymbol{x}_0^{tar}$ was uniformly sampled outside the workspace from a sphere of radius 0.3 m centered at the resting position of the free-end of the robot, i.e., $\boldsymbol{x}_0^{tip}$. Then one or two additional way-points were sampled uniformly from the workspace and interpolated with a cubic spline to generate the object trajectories, i.e., 3D straight lines or 3D curves. The task was successful if the end-effector contacted the object intercepting its trajectory.

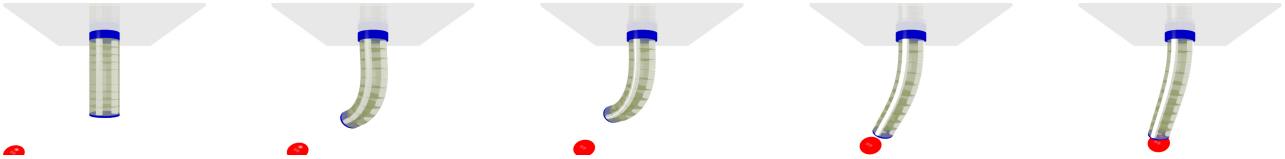
TABLE III: Accuracy of trajectory interception task.

T (s)	3D lines	3D curves
10	85%	95%
5	74%	90%
2	46%	64%
1	33%	52%
0.5	18%	27%

We evaluated the accuracy of trajectory interception for different object velocities to understand the complexity of the interception task (see Table III). The average velocity of the object ranged from 0.03 m/s (T=10 s) up to 0.51 m/s (T=0.5 s) for the linear trajectories, and from 0.05 m/s (T=10 s) up to 0.63 m/s (T=0.5 s) for the curvilinear trajectories. As expected, the percentage of successful interceptions decreased for objects moving at higher velocities for both trajectory types. This was because the average object trajectory in the interception task was up to 25 times faster (for T=0.5 s) than the average training trajectory. Mechanical limitations of the soft robot also play a role. Interestingly, the success rate for the curvilinear trajectories was higher than for the straight lines despite the higher velocities of the former. This could be because the curves stay inside the learned workspace for longer, increasing the probability of interception. Overall, the learned controller performed the object interception satisfactorily, suggesting that the knowledge learned for dynamic path following is transferrable to other tasks. Fig. 8 shows a rendering of a successful trajectory interception trial for T=5 s. From Fig. 9, observe that the object started from a remote position and quickly moved toward the workspace. The controller smoothly tracked the trajectory contributing with all three chambers reducing the tracking error.

IV. Conclusion

In this paper, we leveraged a dynamic Cosserat rod model of a soft robotic arm and trained a control policy for dynamic trajectory tracking using Proximal Policy Optimization, a deep reinforcement learning algorithm. We investigated how well the learned control policy generalized to new observations, including tracking trajectories of different geometry and velocity profiles. Moreover, the controller generalized to new environmental dynamics imposed as perpetual endpoint forces of different magnitudes in any direction. We also tested new dynamics for various material stiffnesses in trajectory tracking. Finally, the policy also generalized



(a) 0/4 (b) 1/4 (c) 2/4 (d) 3/4 (e) 4/4

Fig. 8: Rendering of object interception with a soft robotic arm. Episode lasts 3.3 seconds, max T=5 s.

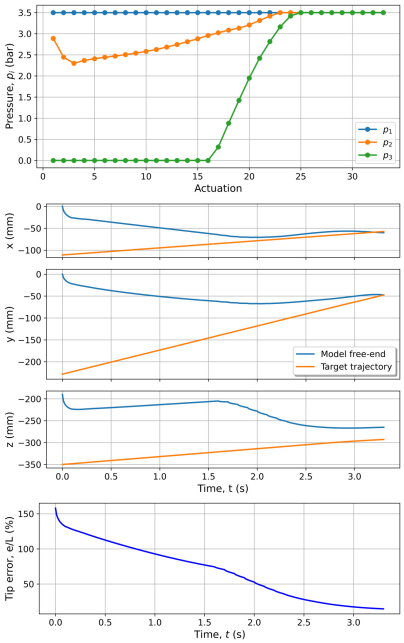


Fig. 9: Successful trajectory interception example, T=5 s.

to the new task of intercepting with the end-effector a moving object (zero-shot transfer). While, as expected, the performance dropped slightly for these new scenarios, the learned model was surprisingly robust. Cosserat rod models of soft robots are promising for learning complex control policies in simulation. Extensions to this work include the sim-to-real transfer of the controller to multi-section soft robotic arms, orientation tracking, using recurrent networks, and learning the trajectory interception policy.

REFERENCES

[1] C. Laschi and M. Cianchetti, "Soft robotics: new perspectives for robot bodyware and control," *Frontiers in bioengineering and biotechnology*, vol. 2, p. 3, 2014.

[2] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467–475, 2015.

[3] C. Armanini, F. Boyer, A. T. Mathew, C. Duriez, and F. Renda, "Soft robots modeling: A structured overview," *IEEE Transactions on Robotics*, 2023.

[4] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control strategies for soft robotic manipulators: A survey," *Soft robotics*, vol. 5, no. 2, pp. 149–163, 2018.

[5] R. J. Webster III and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, 2010.

[6] 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*, p. 2200024, 2022.

[7] C. Della Santina and D. Rus, "Control oriented modeling of soft robots: the polynomial curvature case," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 290–298, 2019.

[8] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee *et al.*, "Review of machine learning methods in soft robotics," *Plos one*, vol. 16, no. 2, p. e0246102, 2021.

[9] M. Eder, F. Hisch, and H. Hauser, "Morphological computation-based control of a modular, pneumatically driven, soft robotic arm," *Advanced Robotics*, vol. 32, no. 7, pp. 375–385, 2018.

[10] F. Piqué, H. T. Kalidindi, L. Fruzzetti, C. Laschi, A. Menciassi, and E. Falotico, "Controlling soft robotic arms using continual learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5469–5476, 2022.

[11] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[12] X. You, Y. Zhang, X. Chen, X. Liu, Z. Wang, H. Jiang, and X. Chen, "Model-free control for soft manipulators based on reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2909–2915.

[13] H. Zhang, R. Cao, S. Zilberstein, F. Wu, and X. Chen, "Toward effective soft robot control via reinforcement learning," in *International Conference on Intelligent Robotics and Applications*. Springer, 2017, pp. 173–184.

[14] Y. Ansari, M. Manti, E. Falotico, M. Cianchetti, and C. Laschi, "Multiobjective optimization for stiffness and position control in a soft robot arm module," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 108–115, 2017.

[15] S. Satheeshbabu, N. K. Uppalapati, G. Chowdhary, and G. Krishnan, "Open loop position control of soft continuum arm using deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5133–5139.

[16] S. Satheeshbabu, N. K. Uppalapati, T. Fu, and G. Krishnan, "Continuous control of a soft continuum arm using deep reinforcement learning," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2020, pp. 497–503.

[17] 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*, vol. 7, no. 2, pp. 4741–4748, 2022.

[18] 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*, vol. 6, no. 2, pp. 3389–3396, 2021.

[19] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, "A survey of generalisation in deep reinforcement learning," *arXiv preprint arXiv:2111.09794*, 2021.

[20] L. Arleo, G. Stano, G. Percoco, and M. Cianchetti, "I-support soft arm for assistance tasks: a new manufacturing approach based on 3d printing and characterization," *Progress in Additive Manufacturing*, vol. 6, no. 2, pp. 243–256, 2021.

[21] M. Gazzola, L. Dudte, A. McCormick, and L. Mahadevan, "Forward and inverse problems in the mechanics of soft filaments," *Royal Society open science*, vol. 5, no. 6, p. 171628, 2018.

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

[23] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[24] J. L. Hodges, "The significance probability of the smirnov two-sample test," *Arkiv för Matematik*, vol. 3, no. 5, pp. 469–486, 1958.