2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

A Framework for Soft Robot Control: Integrating Physics-Based Modeling with Exploration Based Learning

Costanza Armanini*[1], Anthony Tzes[1,2], Andrea Del Prete[3], Fares Abu Dakka[1,4]

***Abstract*— Soft robots present unique challenges for accurate modeling and control due to their virtually infinite degrees of freedom and highly nonlinear deformations. High-fidelity continuum models offer accuracy but are often computationally prohibitive for real-time control, while purely learning-based policies are efficient yet frequently lack robustness and require extensive data collection. In this paper, we propose a hybrid control framework that trains Reinforcement Learning (RL) policies using the physics-based Geometric Variable-Strain (GVS) formulation. This integration enables a Proximal Policy Optimization (PPO) agent to learn on a compact, physically exact state parameterization within the SoRoSim environment, leveraging continuum mechanics for accuracy without the need for real-world data collection. We validate our approach in simulation through two tasks: a basketball throw task and a multi-step pick-and-place scenario. In the throwing task, the agent achieved a 100% success rate within a defined tolerance, with 55% of trials reaching the target with 1cm precision. In the multi-step scenario, the controller maintained high accuracy with a maximum relative error of approximately 8.5%. These results demonstrate that combining GVS-based modeling with PPO yields robust, data-efficient control policies, providing a scalable solution for controlling soft robotic systems across diverse applications.**

I. INTRODUCTION

Soft robots, with their compliant and deformable structures, are characterized by virtually infinite degrees of freedom, granting them an inherent embodied intelligence that rigid systems lack [1], especially when operating in unstructured or unknown environments where adaptability and generalization are critical. As a result, soft robots have been employed in numerous technological fields, from underwater exploration and biomedical applications to agriculture and manufacturing [2], [3]. However, the very properties that make soft robots versatile also make their modeling and control exceptionally challenging. Their continuous, non-linear, and deformable bodies violate the assumptions of rigid-body linear robotic models, creating a highly complex modeling [4] and control [5], [6] problems.

Model-based control approaches rely on physical and continuum mechanics models of soft robots, often providing high accuracy in capturing their behavior, but struggle with computational demands that limit real-time applicability [5]. On the other hand, Machine Learning-based (ML) policies offer computational efficiency and adaptability by learning surrogate models from data; however, they also tend to suffer from limited generalization and can require extensive data collection [7], [8].

Traditionally, these two paradigms—continuum-mechanics-based models and data-driven learning—have been viewed as distinct and sometimes opposing approaches. Recent advances indicate that hybrid integration of these methodologies can exploit their complementary strengths. By incorporating physical model fidelity and the adaptive capabilities of learning algorithms, it is possible to develop control systems for soft robots that are both robust and versatile, capable of handling diverse tasks with improved efficiency [6], [9].

Contribution. We propose an RL control framework that integrates the high-fidelity Geometric Variable-Strain (GVS) approach [10], [11] with a PPO agent [12]. This modeling choice is strategic: while standard numerical continuum models are often computationally prohibitive for online control, GVS provides a reduced-order parametrization that remains physically exact. Consequently, the RL agent operates on a state representation that inherently characterizes soft-body nonlinearities, leading to enhanced training efficiency and robust generalization without the accuracy trade-offs typical of data-driven or simplified geometric models.

Using the GVS-based SoRoSim environment for training, we generate physics-consistent synthetic data, eliminating the need for exhaustive real-world data collection. The resulting RL policy enables real-time execution that analytical Cosserat Rod solvers typically struggle to achieve. To demonstrate these advantages, we tested our framework through two experiments, a single-step "basketball" projectile problem and a multi-step pick-and-place scenario. Both experiments led to high success rates, demonstrating that the integration of GVS and PPO provides a robust solution for soft robotic manipulation. This makes the possibilities for modeling different problems and robots virtually infinite, without requiring the time-consuming data collection needed by data-driven approaches.

The remaining parts of the paper are organized as follows. Section II introduces the related literature on the topic. In Section III we present our approach with the combination of the GVS model with the RL framework and in Section IV we present the experiments that we employed to train and test this framework. Finally, Section V draws the conclusions and discusses the possible future development of this research.

*This work is supported in part by the NYUAD Center for Artificial Intelligence and Robotics, funded by Tamkeen under the NYUAD Research Institute Award CG010.

* Corresponding author costanza.armanini@nyu.edu.
[1] Center for AI and Robotics (CAIR), New York University Abu Dhabi, Abu Dhabi, UAE
[2] Electrical Engineering, New York University Abu Dhabi, 129188, Abu Dhabi, UAE
[3] Department of Industrial Engineering, University of Trento, Trento, Italy
[4] Mechanical Engineering Program, New York University Abu Dhabi, Abu Dhabi, UAE

II. RELATED WORKS

Soft robot control techniques fall into two main categories: model-based and model-free approaches. Model-based techniques are based on a mathematical representation of the kinematics of the soft robot, making this approach effective when a reliable model is available [5]. To overcome the intrinsic nonlinearities of soft robots, various modeling approaches have been developed, mostly by numerically solving traditional continuum mechanics theories (physics-based approaches) or by simplifying the robot's geometry. Physics-based methods usually offer the highest level of accuracy, but come with significant computational complexity. Examples include the Finite Element Method (FEM) [13] and those based on the Cosserat rod theory. The latter, in particular, relies on the assumption that the soft arm can be represented as a continuous curve (the midline) and a moving reference frame that defines the orientation of the cross-section at each point. Different numerical solutions have been proposed using this formulation [14], [11], [15], but due to their high dimensional nature, their direct application to real-time control is often difficult. Other models rely on the assumption that the soft arm deforms following a specific geometrical shape. The most popular example is the Piecewise Constant Curvature (PCC) approach, where the soft arm is discretized in a finite number of constant curvature elements [16]. This approach has been extensively applied in control applications [17], [18], and has also been extended to employ polynomial curvature models, where the curvature varies along the length of each segment following a polynomial curve [19], [20], [21]. These approaches are usually seen as a good compromise between accuracy and computational complexity, but their application is necessarily restricted to problems where the geometrical assumption is valid.

As the name suggests, model-free control, or data-driven control, does not require a pre-existing mathematical model, thus bypassing the complexity associated with intrinsic nonlinearities of soft robots. These approaches use machine learning techniques to directly learn the relationship between the control inputs and the robot's behavior from experimental or simulated data. They often employ Recurrent Neural Networks (RNNs) [22] or Reinforcement Learning (RL) [23], [24], [8]. Although these approaches offer a valuable tool in terms of computational efficiency, they often lack generalizability and require extensive data collection.

Finally, recent literature also provides examples of hybrid approaches, combining model-based and model-free elements, with learning-based controllers grounded in physics-based models. This involves frameworks where the models are employed to generate the synthetic training data for the RL agents to master the soft robotic behaviors prior to physical deployment [25], or hybrid architectures that use physics-based baselines to guide or constrain the learning process [26], [27]. These frameworks combine the interpretability of physical models with the adaptive power of data-driven control.

III. PROPOSED APPROACH: MODELING AND REINFORCEMENT LEARNING

In this paper, we aim at providing a general framework for solving soft robotics control problems using Reinforcement Learning (RL) combined with physics-based modeling techniques.

A. Modeling

Our approach is based on the use of SoroSim, a MATLAB toolbox for simulating various families of robots, including soft, rigid, and hybrid soft-rigid configurations. It was originally presented in [28] and was recently further generalized in [29] and in [30], with an analytical differentiable formulation. It is based on the Geometric Variable-Strain (GVS) approach presented in [11] (statics) and in [10] (dynamics).

The GVS framework characterizes the soft body deformation through the Cosserat rod theory. In this formulation, the configuration of the system is represented by the position of its centerline, $\boldsymbol{r}(X,t) \in \mathbb{R}^3$, and the orientation of the local cross-section, represented by the rotation matrix $\boldsymbol{R}(X,t) \in SO(3)$. Here, $X \in [0, L]$ denotes the material coordinate along the robot's length L. In this way, the spatial configuration of the soft body is mapped to the Lie group $SE(3)$ via the homogeneous transformation matrix $\boldsymbol{g}(X,t)$:

$$\boldsymbol{g}(X,t) = \begin{pmatrix} \boldsymbol{R}(X,t) & \boldsymbol{r}(X,t) \\ \boldsymbol{0}^T & 1 \end{pmatrix} \tag{1}$$

The kinematics is then obtained by evaluating the spatial and temporal variations of $\boldsymbol{g}(X,t)$:

$$\boldsymbol{g}'(X) = \boldsymbol{g}\widehat{\boldsymbol{\xi}}(X)\,, \quad \dot{\boldsymbol{g}}(X) = \boldsymbol{g}\widehat{\boldsymbol{\eta}}(X)\,, \tag{2}$$

where the vectors $\boldsymbol{\xi}(X)$ and $\boldsymbol{\eta}(X)$ represent the local strain twist and velocity twist, while $(\cdot)'$ and $\dot{(\cdot)}$ are the partial derivatives with respect to the curvilinear abscissa and time, respectively.

In the GVS framework, the configuration space is defined through a discretization of the rod's strain fields. By utilizing a truncated modal basis, the continuous strain distribution is approximated as:

$$\boldsymbol{\xi}(X) = \boldsymbol{\Phi}_{\xi}(X)\boldsymbol{q} + \boldsymbol{\xi}^*(X)\,. \tag{3}$$

In (3), the matrix $\boldsymbol{\Phi}_{\xi}(X) \in \mathbb{R}^{6\times n}$ contains the basis functions as its columns, while the vector $\boldsymbol{q} \in \mathbb{R}^n$ denotes the generalized coordinates corresponding to the n degrees of freedom. The term $\boldsymbol{\xi}^*(X)$ accounts for the strain of the system in its undeformed state, derived from the reference configuration $\boldsymbol{g}^*(X)$. This mapping allows the strain coefficients to function as generalized coordinates, allowing the homogeneous transformations along the soft body in a manner mathematically consistent with the joint-space representations found in rigid-link robotics [31].

By mapping these relative strain parameters onto the Cosserat kinematic chain, the system's governing equations are projected into a compact set of ordinary differential equations. The resulting dynamic model follows the standard Lagrangian form:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} = \boldsymbol{\tau} + \boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}}), \tag{4}$$

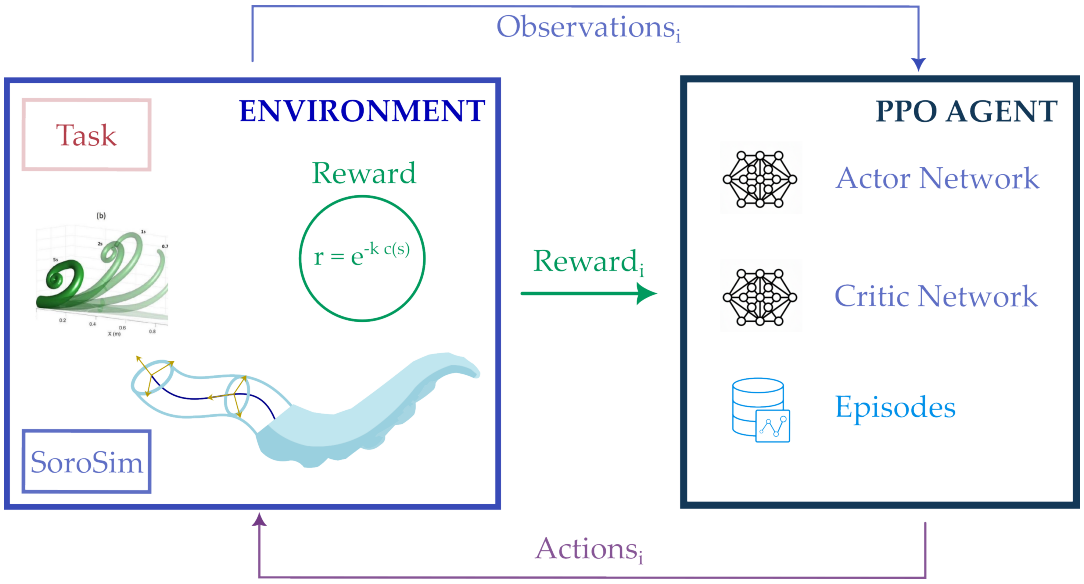

Fig. 1: Scheme of the RL framework. At each i-th step of the episode, the agent receives an observation that can include the task specification and the robot's current state. Based on this, it generates a continuous action vector, which is then denormalized by the environment to control the robot's physical motion. The environment, simulated in SoroSim, calculates the new robot state and provides a reward signal to the agent, a dense exponential function that sharply penalizes task-specific costs and encourages high-precision actions, driving the agent to learn accurate task completion over a series of episodic interactions.

where $\boldsymbol{M}(\boldsymbol{q})$, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\boldsymbol{K}$ are the respectively the generalized mass, Coriolis and stiffness matrices, while $\boldsymbol{\tau}$ and $\boldsymbol{F}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ represent the generalized vectors for internal actuation and the external forces.

B. Reinforcement Learning Framework

In our Reinforcement Learning (RL) framework, the agent learns to control the simulated soft robot by interacting with the physics-based environment powered by SoroSim. The workflow of the RL framework is given in Figure 1. The learning process follows a standard Reinforcement Learning (RL) loop: the agent selects an action (the actuation command), which the environment uses to simulate the robot's dynamics. From the resulting robot state, the environment computes an observation and a reward, which are then returned to the agent to guide its next decision. This interaction is formalized as a Markov Decision Process with the following components:

- **Observation Space**: The observation vector characterizes the task requirements and the system's state. It includes the target coordinates and the Cartesian coordinates of the robot's tip, both normalized within the range $[-1, 1]$. Depending on the complexity of the task, the vector can also be augmented with the robot's current configuration and velocities.
- **Action Space**: The agent's output is a continuous action vector, whose dimensions corresponds to the independent control parameters. Each component is normalized within the range $[-1, 1]$, and subsequently denormalized by the environment to map it to the physical actuation constraints (such as the motor angles or the tendon tensions).
- **Episode Flow**: The agent learns through a series of episodic tasks, which can be single-step (one action vector for each episode) or multi-step when multiple actions are generated in sequence. The environment receives the action and performs in SoroSim the dynamics simulation of the soft arm's motion over the full duration of the step. It then calculates the robot state and the information that are required to estimate the reward.
- **Reward**: To guide the agent toward precise task completion, we use an exponential reward function:

$$R = e^{-k \cdot c(s)}, \tag{5}$$

where $c(s)$ is the task-specific cost function based on the robot's state s, and k is a positive scaling factor. This function provides a dense reward signal that increases sharply as the cost approaches zero, strongly incentivizing high-accuracy actions.

The agent's policy is trained using Proximal Policy Optimization (PPO) [12], a reinforcement learning algorithm renowned for its stability and sample efficiency, making it particularly well-suited for the high-dimensional, continuous action spaces encountered in soft robotics. At its core, PPO optimizes a clipped surrogate objective function, which constrains the size of the policy updates at each training step. This ensures stable learning by preventing the agent from making overly aggressive and potentially catastrophic changes based on a single batch of experience. By ensuring the new policy does not deviate drastically from the prior one, PPO facilitates a smooth learning process that is often less sensitive to hyperparameter tuning than other on-policy

methods.

The agent is implemented using two distinct Actor-Critic architectures, each tailored to the episodic nature of the task: a shared-layer network for single-step problems and separate networks for multi-step problems.

- For single-step tasks, the actor and critic networks share a common set of initial layers, which processes the input observation to generate a common feature representation. The network then branches into two specialized heads: the actor head outputs the parameters for the action distribution; the critic head outputs a single scalar value representing the state's estimated worth. This design is computationally efficient, as the core feature extraction is performed only once.
- For more complex multi-step tasks, the actor and critic are implemented as two fully independent networks. This separation prevents the learning updates from the policy (actor) and the value function (critic) from interfering with each other, which can lead to more stable training in sequential tasks. The actor is a complete, standalone network that takes the state and outputs the action distribution parameters. The critic is another standalone network that independently processes the state to output its value estimate.

In both architectures, the network bodies consist of fully connected hidden layers with ReLU activation functions. The actor consistently defines a continuous policy by parameterizing a Gaussian distribution over the action space, outputting both a mean and a standard deviation for each action dimension.

IV. EXPERIMENTS

To test our framework, we perform two experiments: a single-step throwing task and a multi-step "pick-and-place" scenario. In both cases, we consider a soft robotic arm with a 1 m length, a 5 cm radius circular cross-section. The material properties are defined by a Young's modulus $E = 5$ MPa, density $\rho = 1000kg/m^3$ and a Kelvin-Voigt damping $\eta = 0.1$ MPa·s. The arm is actuated at its base by a revolute joint, with the undeformed configuration aligned with the vertical gravitational axis. Gravitational acceleration is set to $g = 9.81$ m/s^2 where applicable. The system is modeled in SoRoSim using a linear polynomial strain basis for planar bending, resulting in two degrees of freedom (DoF) for the soft arm and one DoF for the revolute base joint. The choice of the polynomial order was validated through preliminary simulations compared against the Finite Element Analysis described in IV-A.1.

The two experiments are as follows:

1) **Problem 1 (Basketball Throw) - Single Step**: The soft arm carries a mass at its tip and is subject to gravity. The revolute joint is rotated counter-clockwise with a given angle law and, at a specific time, the mass is released to hit a target. This problem mimics a basketball game scenario, where the mass represents the ball and the target is the basketball hoop.
2) **Problem 2 (Pick and Place) - Multi-Step**: The arm is actuated by both the revolute joint and a single internal tendon. The task is split into two phases, where the tip must reach two sequential targets.

A. The basketball game (single step)

In this experiment, the soft robotic arm simulates a basketball throw, carrying an object and launching it toward a target. To better represent this specific application, we considered the target (hoop) positions that are representative of the distances encountered in a NBA basketball field and the success rate is defined based on the dimensions of the ball and the hoop.

The revolute joint at the arm's base rotates with a constant angular acceleration. This is described by the angular position equation $\theta(t) = \theta_F(\frac{t}{T_{\text{tot}}})^2$, where θ_F is the final angle and T_{tot} is the total duration of the motion. This profile creates a whip-like effect, resulting in long, high trajectories for the ball. For our simulations, we set the final angle $\theta_F = 210°$ and varied the total time T_{tot} to produce different arm accelerations and trajectories.

For a given T_{tot}, the ball is released at time $t_{\text{rel}} = \delta T_{\text{tot}}$, where the coefficient δ represents the fraction of the total motion time that elapses before release. Before this point, the ball is modeled as a point mass exerting a constant vertical force on the arm's tip. After release, its initial position and velocity are determined by the arm's dynamics at t_{rel}, and it subsequently follows a simple ballistic trajectory governed by gravity.

To maintain the basketball analogy, we used an average basketball mass of 600 g, corresponding to a gravitational force of 5.886 N. It should be noted that a similar robotic system was previously investigated in [32] and [21], showing that, for specific critical values of the revolute angle and the load applied to the tip, the arm presents a snap-back behavior with a sudden release of elastic energy. The load that is considered in these simulations is lower than this critical value, so this instability phenomenon is not observed.

We considered values of T_{tot} ranging from 0.5 to 3.0 seconds and of δ ranging from 0.85 to 1.0. These ranges were selected based on preliminary simulations to generate trajectories corresponding to the three point field goal, where the hoop is 6.75 m away from the line and 3.05 m high from the ground (approximately 1.1 m above the arm's base, accounting for a player's height). Figure 2 shows the resulting workspace and four example trajectories for $T_{\text{tot}} = 2$ s with varying release times (δ values).

RL Framework specification

The problem uses the RL framework introduced in Section III-B, with the following dimensions:

- **Observation Space**: For this problem, the observation space is restricted to the 2-D coordinates of the random target generated for each episode. The agent does not receive any information about the arm's physical state, to keep the agent's observation minimal and focused solely on the goal (similarly to the actual information that the basketball player has before shooting the ball).

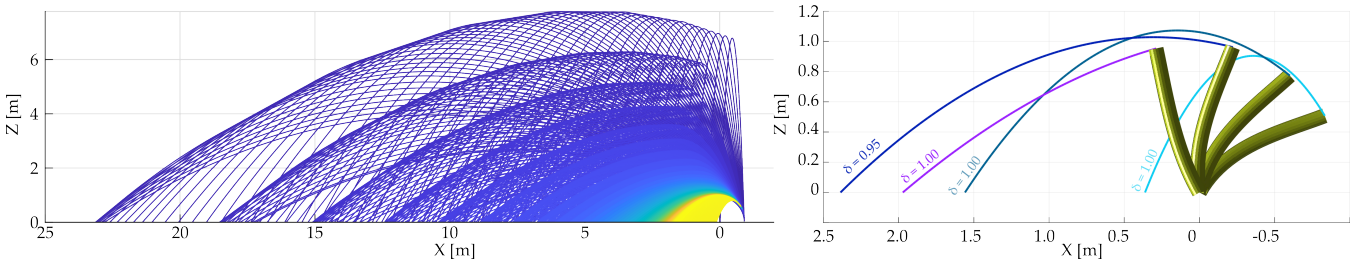

Fig. 2: (Left) Examples of the ball trajectories that can be obtained for $T_{\text{tot}} \in [0.5, 3]$ and $\delta \in [0.85, 1]$. (Right) Deformed shapes and trajectories obtained for $T_{\text{tot}} = 2$s at varying time of δ.

- **Action Space**: The action space is defined by the 2-dimensional continuous vector $[a_T, a_\delta]$, where a_T represents the action on the total time of the simulation and a_δ represents the release moment.
- **Episode Flow**: The agent learns through a series of single-step episodic tasks, where each episode constitutes a single throw attempt. The environment receives the action and performs the dynamics simulation of the soft arm's motion over the full duration of T_{tot}. It then calculates the arm's tip position and velocity at the precise moment of release t_{rel}, and computes the subsequent ballistic trajectory of the ball under gravity.
- **Reward**: Considering the dimension of a basketball (having a radius of around 12 cm) and the hoop (radius 22.86 cm), the maximum distance between the two that results in a successful throw is estimated to be 10cm. The parameter k in the reward function (5) is thus fixed to 2.2, yielding a 0.8 reward at the success threshold..

The agent employs the Proximal Policy Optimization (PPO) algorithm, implemented with the Actor-Critic network shared architecture. The network's shared path begins with an input layer that accepts the 2D normalized target position. This is immediately followed by three sequential, fully connected hidden layers with 128, 32 and 16 neurons, each using a ReLU activation function. The critic network processes the shared features through a fully connected linear layer with just one neuron, which outputs a single scalar value representing the estimated total future reward from the current state.

Figure 3 shows the learning progression of the PPO agent, showing the average reward as a function of training episodes. The top plot shows the initial training phase with a learning rate of 10^{-4} and a clip of 0.1. The bottom plot shows the fine-tuning phase with a reduced learning rate of 10^{-5} and a clip of 0.02. All simulations are performed in MATLAB using the Reinforcement Learning Toolbox.

1) Results

After training and fine-tuning, the PPO agent is tested on 1000 random targets obtained within the considered workspace.

Figure 4 presents one example of the ball trajectory, with the corresponding deformed shape of the arm at the releasing moment, under the actuation provided by the agent for a specific target. For comparison, the figure also shows the trajectory that is obtained performing an FEM simulation

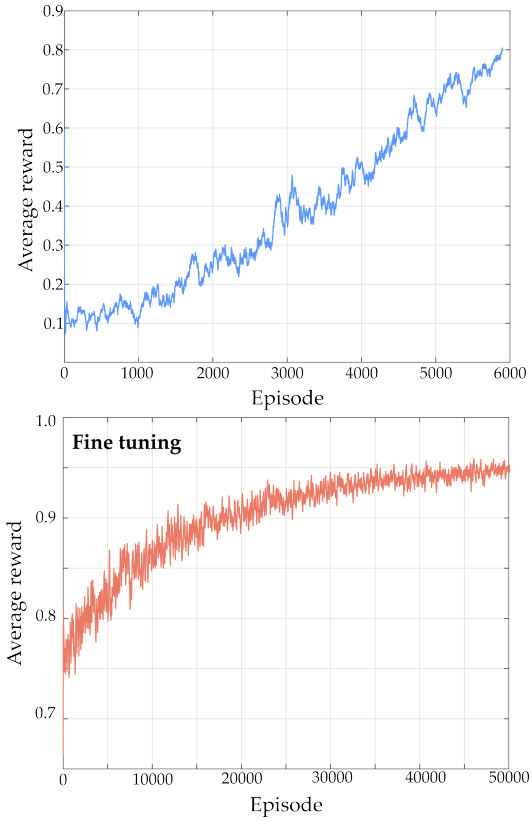

Fig. 3: Agent's learning progress: average reward during the initial training (top) and the continued learning during the fine-tuning phase (bottom). The plot use a 100-episode moving average to smooth the data and highlight the learning curve.

in Abaqus. To simulate the soft arm, we utilized 100 linear elastic beam elements, with the first element constrained to a revolute joint and a vertical force (representing the ball) applied on the terminal edge of the 100th element. A large displacement analysis was performed in two steps: 1) a static simulation to apply all the external forces (gravity and weight of the mass) and 2) a dynamic implicit simulation where the revolute rotation is imposed as a boundary condition.

Figure 5 presents the performance evaluation of the RL agent based on its prediction accuracy. The histogram categorizes the 1000 targets by their final distance from the goal, with the labels indicating the percentage of total targets

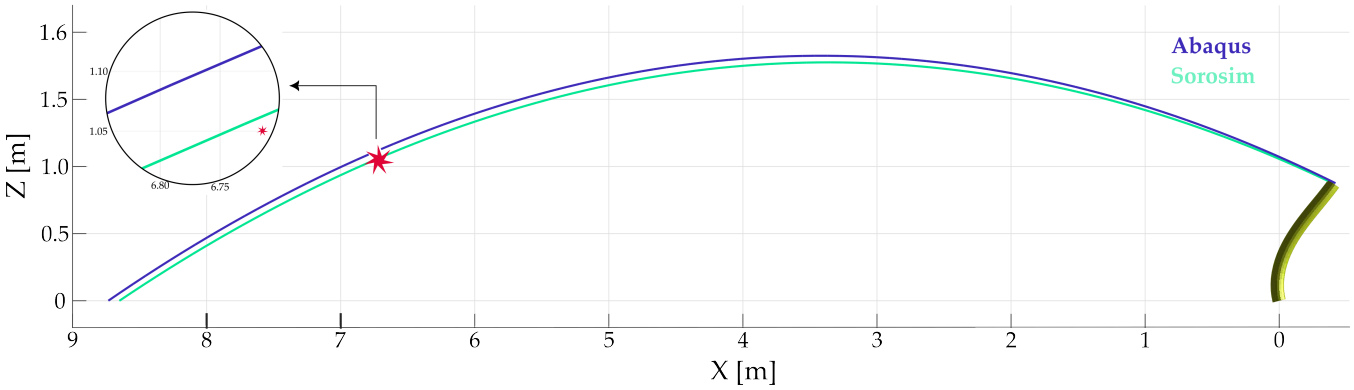

Fig. 4: Example of the trajectory obtained using the actuations provided by the PPO agent, for one of the random tested targets. For completeness, the trajectory obtained by FEM simulations performed in Abaqus is also presented in blue.

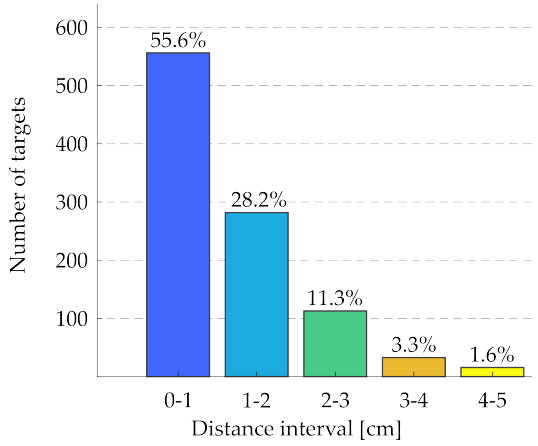

Fig. 5: Distribution of the RL agent's prediction error across 1000 test targets. Each bar represents the percentage of targets whose final position was within a specific distance interval from the intended goal, demonstrating the agent's overall accuracy. Success is considered when the distance is lower than 10cm (the ball enters the hoop).

that fall into each accuracy bracket. More than 55% of the predicted trajectories were located between 0 and 1 cm of the goal target. As mentioned before, we can estimate that the ball enters the hoop when the distance is lower than 10cm, thus the success score on these experiments is 100%.

B. Pick and place (double step)

The second problem focuses on a multi-step episode involving a two-phase 'pick and place' task. The soft arm is actuated by both the revolute joint at the base and one tendon routed inside the arm, maintained at a constant distance of 2.5 cm from the midline. The task consists of two phases. Phase 1 (Pick): The joint rotates and the tendon is pulled to guide the arm's tip to a target object. Phase 2 (Place): The arm then rotates in the opposite direction to reach a distant drop-off position. To simulate a smooth approach, the revolute joint's rotation follows a sinusoidal velocity profile, accelerating during the first half of the movement and decelerating as it approaches the target.

The agent is trained to learn the final rotation angle of the base, $\alpha_{F_{i,j}}$ and the required tendon actuation force $T_{i,j}$ for the two steps $(i, j = 1, 2)$.

RL Framework

The state RL Environment for this experiment is defined as follows:

- **Observation Space**: The environment's state is a 9 element vector, comprising the arm's state (6 values from the strain basis and velocities), the 2D target coordinates, and the current step indicator.
- **Action Space**: The agent provides a 2-element action: a target angle for the base joint and the tendon pulling force. These actions are scaled to a maximum angle of π and a maximum force of 30 N. To respect physical actuation constraints, the normalized action for the tendon is mapped to a non-negative range $[0, 30]$ N, ensuring the controller accounts for the unilateral nature of tendon actuation.
- **Episode Flow:** Each episode consists of two sequential steps of 5 seconds, where the final state of Step 1 serves as the initial condition for Step 2.
- **Reward**: In the reward function (5), we set $k = 4.46$, corresponding to a 0.8 reward for each step when the distance is 5cm. The episode terminates after the second step is completed.

For this problem, the actor network has a body consisting of two fully connected hidden layers (128 and 32 neurons with ReLU activations), which is mirrored by the critic network's body: two hidden layers (also 128 and 32 neurons), followed by a single linear output neuron that estimates the state value.

1) Results

Figure 6 presents the dynamics of the soft arm for the two steps as predicted by the agent's action policy. It can be observed that, in the first step, the agent defines a control policy where the arm is initially rotated and the pulling force is increased to position the tip near the target. In the second step, the revolute joint returns toward the reference position while the tendon force is further increased. Table I details the target and the tip positions at each step. While both steps resulted in errors below 10% of the arm's length, the first step achieved

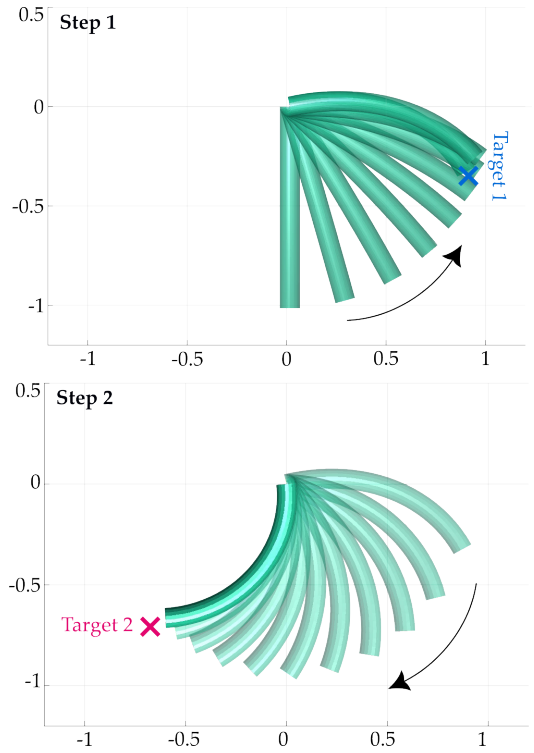

Fig. 6: Dynamics of the soft robotics arm for both the steps forming Problem 2, obtained with the actuation provided by the PPO agent.

higher accuracy (2% error). The relatively higher error in the second step is attributed to compounded error propagation, as the second phase is initialized from the terminal state of the first and any minor positional deviations or residual oscillations from the initial movement accumulate. Despite this, the results demonstrate the framework's effectiveness for sequential, multi-step tasks.

	$\mathbf{X_{target}}$ [m]	$\mathbf{Y_{target}}$ [m]	$\mathbf{X_{tip}}$ [m]	$\mathbf{Y_{tip}}$ [m]	**Distance** [cm]
Step 1	0.884	-0.344	0.866	-0.342	2.02
Step 2	-0.675	-0.707	-0.623	-0.641	8.52

TABLE I: Position of the targets and of the soft arm tip obtained by the actuation from the RL agent, and their distances.

V. CONCLUSIONS

This paper presents a reinforcement learning approach that leverages a physics-based model for the control of soft robots. It combines the accuracy of the GVS approach with the power of machine learning, offering a general framework applicable to a wide range of applications. We performed experiments in both single-step and multi-step scenarios, with a 100% succes rate for the single-step task and around 8.5% errors for the multi-step task. Current literature combining soft robotics with RL often relies on either overly simplified or purely data-driven models. While the former compromises model accuracy and widens the sim-to-real gap, the latter requires extensive, time-consuming data collection and often lacks generalization. In contrast, our use of a physics-based modeling tool enables us to accurately depict the complex nonlinear behaviour that lies the foundation of the soft robotics paradigm.

While these simulation results establish a high-performance baseline, the primary limitation remains the lack of validation on a physical manipulator, which represents the immediate next step for this research. Future work will also expand the framework to 3D scenarios, dynamic stabilization, and obstacle avoidance.

ACKNOWLEDGMENT

C. Armanini and F. Abu-Dakka thank Hozefa Jesawada for the valuable discussions on implementing the reinforcement learning algorithm.

REFERENCES

[1] M. Cianchetti, "Embodied intelligence in soft robotics through hardware multifunctionality," *Frontiers in Robotics and AI*, vol. 8, 2021.
[2] G. Alici, "Softer is harder: What differentiates soft robotics from hard robotics?" *MRS Advances*, vol. 3, no. 28, p. 1557–1568, 2018.
[3] C. Laschi, B. Mazzolai, and M. Cianchetti, "Soft robotics: Technologies and systems pushing the boundaries of robot abilities," *Science Robotics*, vol. 1, no. 1, 2016.
[4] C. Armanini, F. Boyer, A. T. Mathew, C. Duriez, and F. Renda, "Soft robots modeling: A structured overview," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1728–1748, 2023.
[5] C. Della Santina, C. Duriez, and D. Rus, "Model-based control of soft robots: A survey of the state of the art and open challenges," *IEEE Control Systems Magazine*, vol. 43, no. 3, pp. 30–65, 2023.
[6] 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.
[7] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-driven control of soft robots using koopman operator theory," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 948–961, 2021.
[8] 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.
[9] P. Hyatt, C. C. Johnson, and M. D. Killpack, "Model reference predictive adaptive control for large-scale soft robots," *Frontiers in Robotics and AI*, vol. 7, p. 558027, 2020.
[10] F. Boyer, V. Lebastard, F. Candelier, and F. Renda, "Dynamics of continuum and soft robots: A strain parameterization based approach," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 847–863, 2021.
[11] F. Renda, C. Armanini, V. Lebastard, F. Candelier, and F. Boyer, "A geometric variable-strain approach for static modeling of soft manipulators with tendon and fluidic actuation," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4006–4013, 2020.
[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347
[13] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, "Sofa: A multi-model framework for interactive physical simulation," *Studies in Mechanobiology, Tissue Engineering and Biomaterials*, 2012.
[14] M. Gazzola, L. H. Dudte, A. G. McCormick, and L. Mahadevan, "Forward and inverse problems in the mechanics of soft filaments," *Royal Society Open Science*, vol. 5171628171628, 2018.
[15] D. Trivedi, A. Lotfi, and C. D. Rahn, "Geometrically exact models for soft robotic manipulators," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 773–780, 2008.
[16] R. J. Webster 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.
[17] C. Della Santina, A. Bicchi, and D. Rus, "On an improved state parametrization for soft robots with piecewise constant curvature and its use in model based control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1001–1008, 2020.

[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 *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, 2019, pp. 454–461.

[19] C. D. 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, 2020.

[20] F. Stella, Q. Guan, C. Della Santina, and J. Hughes, "Piecewise affine curvature model: a reduced-order model for soft robot-environment interaction beyond pcc," in *2023 IEEE International Conference on Soft Robotics (RoboSoft)*, 2023, pp. 1–7.

[21] D. Caradonna, M. Pierallini, C. D. Santina, F. Angelini, and A. Bicchi, "Model and control of r-soft inverted pendulum," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5102–5109, 2024.

[22] T. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Learning dynamic models for open loop predictive control of soft robotic manipulators," *Bioinspiration & Biomimetics*, vol. 12, no. 6, p. 066003, 2017.

[23] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Transations on Robotics*, vol. 35, no. 1, pp. 124–134, 2019.

[24] D. Bianchi, M. G. Antonelli, C. Laschi, A. M. Sabatini, and E. Falotico, "Softoss: Learning to throw objects with a soft robot," *IEEE Robotics & Automation Magazine*, vol. 31, no. 4, pp. 113–123, 2024.

[25] P. Schegg, E. Ménager, E. Khairallah, D. Marchal, J. Dequidt, P. Preux, and C. Duriez, "Sofagym: An open platform for reinforcement learning based on soft robot simulations," *Soft Robotics*, vol. 10, no. 2, pp. 255–277, 2023.

[26] C. M. Best, M. T. Gillespie, P. Hyatt, L. Rupert, V. Sherrod, and M. D. Killpack, "A new soft robot control method: Using model predictive control for a pneumatically actuated humanoid," *IEEE Robotics & Automation Magazine*, vol. 23, no. 3, pp. 75–84, 2016.

[27] C. Alessi, D. Bianchi, G. Stano, M. Cianchetti, and E. Falotico, "Pushing with soft robotic arms via deep reinforcement learning," *Advanced Intelligent Systems*, vol. 6, no. 8, p. 2300899, 2024.

[28] A. T. Mathew, I. B. Hmida, C. Armanini, F. Boyer, and F. Renda, "Sorosim: A matlab toolbox for hybrid rigid–soft robots based on the geometric variable-strain approach," *IEEE Robotics & Automation Magazine*, vol. 30, no. 3, pp. 106–122, 2023.

[29] A. T. Mathew, D. Feliu-Talegon, A. Y. Alkayas, F. Boyer, and F. Renda, "Reduced order modeling of hybrid soft-rigid robots using global, local, and state-dependent strain parameterization," *The International Journal of Robotics Research*, vol. 44, no. 1, pp. 129–154, 2025.

[30] A. T. Mathew, F. Boyer, V. Lebastard, and F. Renda, "Analytical derivatives of strain-based dynamic model for hybrid soft-rigid robots," *The International Journal of Robotics Research*, vol. 45, no. 1, pp. 128–158, 2026.

[31] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, 1st ed. USA: CRC Press, Inc., 1994.

[32] C. Armanini, F. Dal Corso, D. Misseroni, and D. Bigoni, "From the elastica compass to the elastica catapult: an essay on the mechanics of soft robot arm," *Proceedings Royal Society A*, vol. 473, no. 2198, 2017.