RL-based Adaptive Controller for High Precision Reaching in a Soft Robot Arm

Muhammad Sunny Nazeer *IEEE Student Member*, Cecilia Laschi *IEEE Fellow*, Egidio Falotico *IEEE member*

***Abstract*—High precision control of soft robots is challenging due to their stochastic behavior and material-dependence nature. While RL has been applied in soft robotics, achieving precision in task execution is still a long way off. Traditionally, RL requires substantial data for convergence, often obtained from a training environment. Yet, despite exhibiting high accuracy in the training environment, RL-policies often fall short in reality due to the training-to-reality gap, and the performance is exacerbated by the stochastic nature of soft robots. This study paves the way for the implementation of RL for soft robot control to achieve high precision in task execution. Two sample-efficient adaptive control strategies are proposed, that leverage the RL-policy. The schemes can overcome stochasticity, bridge the training-to-reality gap, and attain desired accuracy even in challenging tasks such as obstacle avoidance. Additionally, deliberate and reversible damage is induced to the pneumatic actuation chamber, altering the soft robot's behavior to test the adaptability of our solutions. Despite the damage, desired accuracy was achieved in most scenarios without needing to retrain the RL-policy.**

***Index Terms*—Reinforcement Learning (RL), Machine Learning Based Control, Bayesian Optimization (BO), Soft Robots, Imitation Learning by Coaching, Cerebellum Inspired Compensator for Motor Control.**

I. Introduction

ACCURATE modeling of soft robots poses a significant challenge due to their highly deformable mechanics [1]. Various solutions have been suggested in this domain (Sect. II-A). One such modeling approach is Machine Learning (ML)-based, chosen for its capacity to learn from real robot data [2]. ML-based strategies have showcased superior performance in enabling training of complex tasks (see Table I). However, despite the incorporation of ML-based schemes, there is a reported decline in control solution performance when tested on the actual soft robot compared to performance within training environments. This decline can be attributed to the performance gap between the learned model and the soft robot behavior, commonly known as the training-to-reality gap.

The gap results from two primary factors: (1) Data-driven models, such as trained recurrent neural networks, forecast future states of the soft arm based on current actions and past predictions. As these predictions are approximations, small errors in current and past predictions accumulate over time, contributing to the performance disparity. (2) Inherent stochasticity, as illustrated in Fig. 1a) , in the behavior of a pneumatically actuated soft arm (see Sect. III-B), results from intrinsic factors like nonlinear material properties (e.g., hysteresis) and other elastic properties under varying environmental conditions, and extrinsic factors related to the soft arm's design characteristics, such as length, number of modules, variable moment of inertia, and initial positions.

Reinforcement Learning (RL)-based algorithms offer the advantage of training inherently stable control solutions [3] for complex tasks without an in-depth understanding of the underlying platform. This makes them promising for addressing the challenges related to the control of soft robots. However, their adaptability to variations in the evaluation environment compromises task execution accuracy [4], making the recovery of desired task accuracy while overcoming the training-to-reality gap an active area of research in the robotics community. In the context of soft robot control, this challenge is exacerbated due to the stochastic nature of the systems.

In this study, we applied Proximal Policy Optimization (PPO), an RL algorithm [5], to train a policy for a high-precision control problem using a data-driven dynamics model of a three-module pneumatically actuated soft continuum arm. The policy was trained for a reaching task with obstacle avoidance, as depicted in Fig. 1b). Experimental evaluation revealed a significant decrease in policy performance when applied to the soft arm compared to its performance in the training environment. This difference is attributed to the training-to-reality gap, worsened by the inherent stochasticity in the soft arm, negatively impacting task repeatability with the desired accuracy.

Two distinct control strategies were devised to successfully bridge the performance gap and restore task accuracy in the robot dynamics domain within seconds, as illustrated in Fig. 1c). The first approach, Bayesian Optimization Assisted Coaching (BOAC), draws inspiration from Imitation Learning (IL) by coaching [6], a variant of traditional IL. Traditional IL trains a task policy based on an expert's task demonstrations and action predictions from the same [7] or different supervisor or oracle [8]. IL by coaching trains a policy based on a coach and an oracle, with the coach presenting easy-to-reach goals followed by gradual improvements to reach the final goals, and the oracle predicting actions to reach the respective goals. The second approach, Gaussian Process-based Recurrent Cerebellar Architecture (GPRCA), trains an online compensator using errors between observations made by the robot in the task space and those made in the training environment. This scheme is inspired by the recurrent

Manuscript submitted on January 15, 2024

Muhammad Sunny Nazeer and Egidio Falotico are with BRAin-Inspired Robotics Lab, The BioRobotics Institute, Scuola Superiore Sant'Anna, Pontedera, 56025, Italy, and Department of Excellence in Robotics and AI, Scuola Superiore Sant'Anna, Pisa 56025, Italy Email: (muhammadsunny.nazeer, egidio.falotico)@santannapisa.it. Cecilia Laschi is with Department of Mechanical Engineering, National University of Singapore, Singapore 127575, Singapore Email: mpeclc@nus.edu.sg

TABLE I
LITERATURE FOCUSED ON TASK ACCURACY

Modeling Scheme	Task	Robot Dimension (d_r) $[mm]$	No. of Modules (Control Signals)	Error (e_r) $[mm]$	Accuracy $(\frac{e_r}{d_r}\text{x}100)\%$
Non-data-driven	Trajectory tracking [10]	110	3 (12)	27.1	24.6
Data-driven (Open-loop)	line following [11] circle following [11] infinity following [11] hypotrochoid following [11]	400	2 (6)	20 ± 25 51 ± 32 21.7 ± 15.3 49.3 ± 23.2	5 ± 6.25 12.75 ± 8 5.4 ± 3.8 12.3 ± 5.8
Data-driven (Closed-loop)	dynamic reaching [12] infinity following [13] wavy circle following [13] circle following [13]	400 440	2 (6) 2 (6)	26 ± 32 ~ 12.4 ~ 15.5 ~ 11.8	6.5 ± 8 2.8 3.5 2.7

architecture proposed in [9]. Both approaches utilize Bayesian Optimization for improved sample efficiency.

These strategies were evaluated not only for bridging the training-to-reality gap with task repeatability and desired accuracy but also for their adaptability to scenarios that the training environment could not account for, such as various damage incidents to the soft arm (see Sect. V-A) and external loads (details in the supplementary materials). The adaptive nature of the control strategies, coupled with sample efficiency and task repeatability, contributes to the overall reliability of soft robots, complementing the existing literature on overcoming the training-to-reality gap.

The main contributions of this study, as summarized in Fig. 1, include:

- Deploying a soft arm with nine pneumatic chambers for obstacle avoidance and a high-precision reaching task using RL in the robot dynamics domain;
- Bridging the training-to-reality gap by addressing the soft arm behavioral stochasticity using two online control strategies (Sect. V-C);
- Successfully performing the reaching task with the soft arm, even after deliberately damaging it in various ways, using the proposed control schemes (Sect. V-D);
- Achieving sample efficiency through the use of Bayesian optimization in the control strategies;
- Demonstrating task repeatability with desired accuracy, despite stochasticity (Sect. III-B), damage incidents (Sect. V-A) or external loading (supplementary materials).

II. RELATED WORK

This section lists the available literature on soft robot modeling and control using learning and non-learning-based schemes, and comparison of proposed control strategies with similar literature regarding overcoming the performance gap.

A. Non-learning vs Learning-based Methods for Modeling

When it comes to modeling soft robots, there have been numerous advancements dealing with a range of challenges highlighted in the literature, including Constant Curvature (CC) [14], [15], Piecewise Constant Curvature (PCC) [16], [17], Piecewise Smooth Curvature (PSC) [18], Cosserat rod theory [19], [20], and Finite Element Method (FEM)-based [21], [22]. The approaches explain soft robots' behavior by either approximating the curved geometry of a soft uniform robot using fixed geometrical parameters for single or multiple

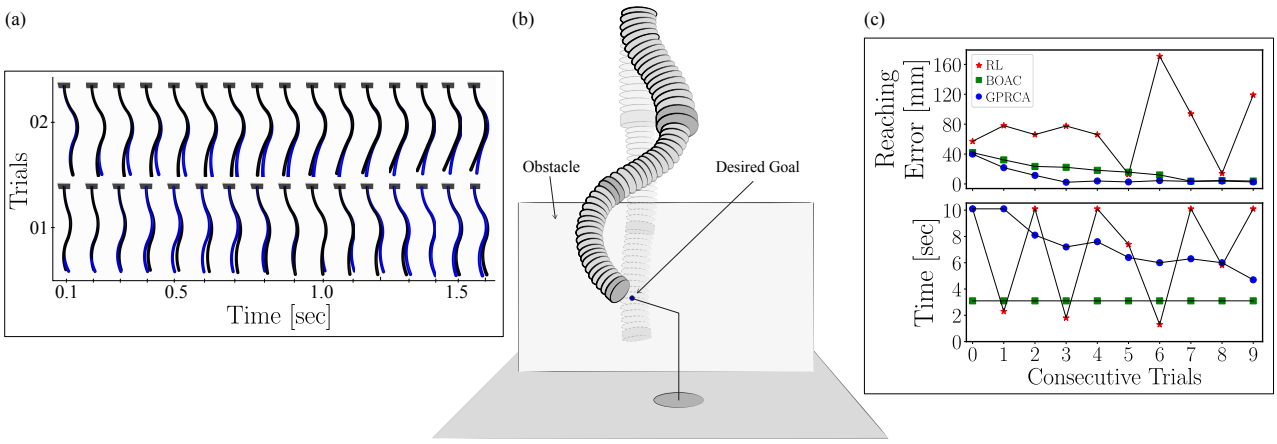

Fig. 1. The figure presents the main contributions of this work. Figure (a) illustrates stochasticity exhibited by the three-module soft arm, where two independent random actuation-space trajectories were used to actuate the soft arm twice. In each trial, the black and blue represent the first and second trajectory for the respective actuation-space trajectory, respectively. Figure (b) describes the reaching task with obstacle avoidance with the soft arm in initial and final positions. Finally, figure (c) shows the comparison between the accuracy, task repeatability and sample efficiency achieved with the policies resulted from RL, BOAC and GPRCA. Please note, trials 1, 3, and 6 in figure (c) represent collision in RL-policy testing.

curved segments, or low-order polynomials describing the flexure motion by assuming smooth bending in an elastic beam, or deriving a set of nonlinear partial differential equations to compute differential displacements around a set of boundary conditions for an elastic rod, or lastly computing non-linear and non-uniform deformations in a soft body, respectively.

Although the underlying behavioral description of soft robots has improved through new modeling schemes, acquiring the generic behavior of soft robots even in a controlled environment is still a long way off [23]. In [24]–[26] soft robot behavior was emulated by training an Artificial Neural Network (ANN) on data collected directly on the soft robot. The behavior was reasonably approximated, although the stochasticity in the soft platform due to inherent material properties or extrinsic stimuli still poses a performance discrepancy between the model and the soft robot.

B. Non-learning vs Learning-based Methods for Control

Controlling soft robots using non-learning-based control methods is a challenging area mainly due to intrinsic difficulties in deriving controllers for systems with virtually infinite degrees of freedom. Most of the literature attempts to approximate the Inverse Statics (IS) or Kinematics (IK) of the soft robot to derive controllers. Chirikjian et al. [27] proposed a modal-approach based on a set of time-varying backbone curve functions for a hyper-redundant continuum robot for planar and spatial movements. Coevoet et al. [28] presented an interactive or contact handling controller based on an IK model approximation using FEM. Models based on non-learning have also been used, such as CC, PCC, PSC and Cosserat models with a closed-loop control scheme in 3D trajectory tracking [29], curvature and bending control [17], a multi-contact point handling framework for contact force estimation, end-effector path planning and navigating obstacles through planar structured environments [30], and finally planer motion control using sliding mode control [31], respectively. On the account of restricted capability of soft robots exhibited under the umbrella of non-learning based approaches in sophisticated situations, a comparative analysis of non-learning to learning based was used in [32].

1) Supervised Learning (SL) for Control: This is the most explored area for the control of soft robots [33]. Among the pioneering works, [1], [11], [24], [34] use an ANN trained using SL to approximate IK or forward dynamics model for position control, quasi-static tracking, dynamic reaching and self-stabilizing open loop dynamic control, respectively. Similar approaches also include [25], [26] for position control using model predictive control and open-loop trajectory tracking, respectively, on a data-driven model using SL. In this class of algorithms, either an SL-trained model with an external controller running feedback optimization has to be used [35] or a controller trained on a task-specific data [36]. In both cases, the solution may either lack robusticity/adaptability or will perform merely qualitatively as in dynamic movement primitives [37] or probabilistic movement primitives [38] assisted adaptive controllers to approximate trajectory control.

2) Beyond Supervised Learning for Control: RL for control [39] has attracted more attention of soft roboticists than other classes of ML algorithms, using non-learning-based and learning-based models. Some examples of the former category are: A Cosserat model simulator in [40] and [41] employed to follow different trajectories in 2D and 3D under cluttered environments using RL [42]. SoMo [43], a framework able to approximate continuum manipulators through rigid link systems with spring-loaded joints, deployed for a variety of tasks [44], and also to benchmark RL-controllers. An FEM simulator [45], which accounts for material properties in a soft robot deformation, has been exploited for tasks where interactions with the external environment are required [46]. Commercial simulation engines are also used with a simplified continuum manipulator for feedback, such as Gazebo to find an unknown object in the robot workspace [47], and MuJoCo to reach target-positions with the end effector tip [48] and distance maintenance for minimally invasive surgery [49].

For the second category, the kinematics or dynamics of a soft robot are learned using an ANN and employed in an open loop [11], [50] for self-stabilizing trajectory and position control, respectively, or closed loop [12], [13] for dynamic reaching and trajectory following, respectively. As opposed to conventional or data-driven modeling methods, in [51] Oikonomou et al. presented a modified version of continuous actor-critic learning automaton to learn a policy capable of passing through a series of target waypoints generated using dynamic movement primitives (and proposed probabilistic movement primitives for soft robots with stochastic performance). Approaches other than RL-based schemes may also follow a similar trend, such as an adaptive controller, using a cerebellum-inspired approach built on top of a data-driven IK model [52] to enable desired trajectory tracking.

C. Overcoming Performance Gaps in Controls

The studies outlined in Table I present controllers in the soft robot's dynamic domain, revealing significant declines in control solution performance when they are tested in evaluation environments. The declines are a result of the controllers' inability to account for the stochasticity, training-to-reality gap or any other factor affecting soft robot's behavior control.

The studies that focus on addressing the behavioral gaps are either applied directly to the model [53], [54] or to the derived control solution [55]–[57]. Our proposed schemes belong to the latter category. For the first category, Fang et al. [53] proposed learning forward and inverse kinematic models using neural networks on a simulator. Then, they used fewer samples directly from the hardware to train additional layers, which thus mitigated the performance gap. Similarly, Dubied et al. [54] optimized different elements associated with the Finite Element Method (FEM), such as meshing elements and resolution, and numerical damping, to improve the performance disparity between the simulation environment and the soft robot's performance.

For the second category, Johnson et al. [55] combined a deep neural network with a first-principles model to improve the overall accuracy of a non-linear model predictive controller (MPC). Despite being quite similar to our proposed

approaches, this solution does not account for the stochasticity in the robot's behavior. The solution in [55] takes 88% more data samples than our approaches, on average, to reduce the error in MPC performance by 52%. Our approaches take fewer samples to reduce the error in RL-policy performance by 67% on the soft robotic arm while accounting for the training-to-reality gap and stochasticity, ensuring task repeatability. Similar approaches are also presented in [56] and [57] to overcome the performance disparity. Neither of these approaches accounts for the training-to-reality gap. The scheme proposed in [56] learns to perform a trajectory in a simulation environment and validates it (also in the same simulation environment) by overcoming the uncertainty introduced in the inverse kinematics. The adaptor is a cerebellar-inspired control architecture, which takes approximately 75% more data samples than our solutions to reduce the error by 70%. On the other hand, [57] uses an RL-agent to learn to compensate for performance discrepancy. This approach may not be practical for soft robots as it takes approximately $10\ hrs$ to learn to provide the compensation.

ML-based schemes can emulate a soft robot's performance with a good degree of accuracy and control for a desired task. However, adapting to the range of soft robot's behaviors over time while continuing to perform the intended task with a similar proclivity is still an active area of research. In this study, we have targeted this area and successfully demonstrated overcoming stochasticity, bridging the training-to-reality gap, and ensuring accuracy in the task execution, even for scenarios the trained model is incapable of accounting for, such as damage incidents and external loadings. In the following sections we attempt to quantify the stochasticity (Sect. III-B), model the robot dynamics (Sect. III-C), derive a control policy offline using the dynamics model with an RL algorithm (Sect. IV-A), present two control schemes (Sects. IV-B and IV-C), and the obtained results (Sect. V) along with the policy evaluation scenarios on a soft arm (Sect. III-A). Finally, we discuss the findings (Sect. VI) and conclude (Sect. VII).

III. Preliminaries

A. Experimental Setup

The robot in question is a pneumatically actuated three-module soft continuum arm as shown in Fig. 2b. This platform was initially designed to provide support to the elderly in taking shower as presented in [58], [59]. Each module in the soft arm is independently actuated using three pneumatic chambers placed at 120° in a circular arrangement. Each pneumatic chamber consists of two McKibben-based flexible fluidic muscles. The chambers are constrained by thin disks made of polypropylene. This arrangement ensures bending in all directions by actuating the chambers individually or in pairs. Once all three chambers have been actuated simultaneously with equal pressure, it produces whole arm extension. A collection of such behaviors ensure that the space around the robot is accessible. Since each module is actuated independently, the whole arm is capable of exhibiting redundant behavior, up to a certain degree. This redundancy, along with adaptive decision-making capability, can be exploited to elicit recovery from behavior-altering factors, such as repeatable and reversible damage incidents and external loadings. The soft arm (with three modules) is operated using nine pneumatic control signals.

To pneumatically actuate the chambers we used an electronic proportional microregulator series K8P with an operating pressure from 0 to $4\ bars$ ($400\ KPa$). For the safety of the soft arm, the pressure ceiling was set to $1.5\ bars$ ($150\ KPa$). There were a total of nine regulators responsible for the low-level control of nine chambers. As a result of the nine pneumatic signals, we tracked the tip of all three modules using a motion capture system (Vicon system) with eight Bonita cameras. We placed three markers arranged at 120° on the tip of each module. The markers acted as the three corners of an equilateral triangle, the center of this triangle represented the tip position of that module. Additional three markers were placed on the base of the soft arm to generate the origin plane for the soft arm which also served as the origin frame. The cameras of motion capture system were set to capture different perspectives of the soft robot with redundancy (i.e., each marker is tracked by at least two cameras) in order to recreate the entire network of markers. The Vicon system was set to track the markers at $100\ Hz$; additional delays were introduced to synchronize the tracking with our control/optimization loop. The positions of all the markers (with respect to the robot origin frame) were published via ROS to the Python environment for closed-loop control with the RL-policy.

For the task setting, a fixed cuboid obstacle ($5\ mm$ thick rectangle) was placed with a dimension (in mm scale) extending from -60.0 to 60.0 in the x direction, -50.0 on the y axis (with a thickness of $5\ mm$) and -500.0 to -700.0 on the z axis (according to the robot's origin plane). The obstacle was stationary, and mostly restricted the workspace of the third module and partially the second module of the soft arm, and completely blocked direct access to the goal-point. The tip of the third module was required to reach the goal-point in the 3D robot reference frame, while avoiding a collision with the obstacle. The learning agent must learn to use the unrestricted modules of the soft arm, as assistive limbs, to ensure the tip of the third module reaching the desired goal-point. Additionally, for faster convergence, the search space of the policy was restricted by introducing a boundary (in mm) that extended from -130.0 to 90.0 along x-axis, -150.0 to 50.0 along the y-axis, and -700.0 to -570.0 along the z-axis. Different environments with varied goal-points in all three axes (in the third module) were trained and tested with proposed schemes. With each of these goal-points, the obstacle location was also varied, mostly along the z axis, to see different ways in which the policy enabled reaching desired goal-point with sufficient accuracy. The placement of the goal-points behind the obstacle made some of the goal-points more difficult to reach than others. The results with different goal-points are compiled in Table II along with the evaluation scenarios as listed in Sect. V-A.

B. Stochasticity Analysis

The stochasticity in the soft robot's performance could be linked to either intrinsic factors (inherent to the material) such as material hysteresis and its variable elastic properties due to environmental conditions, etc., or extrinsic factors (related to the characteristic length and mode of actuation) such as variable initial positions due to flexible shape, variable moments of inertia due to imbalanced morphology resulted from manufacturing inaccuracies, and incomplete depletion of pneumatic channels during operation.

To study the stochasticity in the soft arm, we created a random trajectory in the actuator space with nine pneumatic actuators for the three-module soft arm shown in Fig. 2b. We conducted ten trials with the same actuation space trajectory. At the end of each trial, a change in the resting position of the soft arm was observed due to its flexible morphology. The difference in the initial conditions introduced visible variability in the task-space recordings. There is also a possibility that there are manufacturing inaccuracies in the soft arm, leading to an imbalanced morphology. This imbalance can cause variations in the moments of inertia along the length of the soft arm. Additionally, actuating the pneumatic chambers at high frequency leaves less time for them to inflate and deflate fully. These factors, combined with material's own hysteresis, contributes to the stochasticity in its behavior.

The first trial in the experiments was taken as the base-trial and distances of the following trials were computed with respect to the base trial. Population statistics of the trials, in the form of a dispersion in the mean distances among the trials, is shown as a boxplot in Fig. 2. We conducted similar experiments on a single-module and two-module soft arm as well that are one-third and two-thirds of the length, approximately, compared to the three-module soft arm, respectively. The stochasticity was found to be relatively insignificant in the single-module and more visible in the two-module arm.

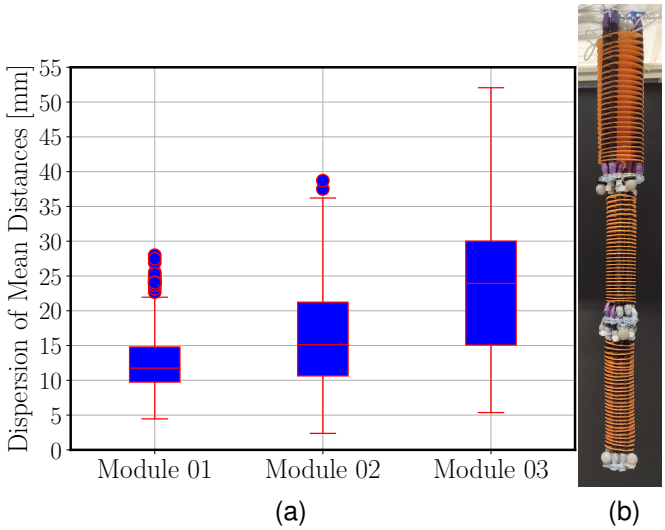

Fig. 2. The figure aims to quantify the stochasticity in the form of a distribution, shown in (a), for the soft arm, shown in (b), based on ten trials conducted using the same actuation space trajectory. The blue box (with red border) represents the middle 50% of the underlying dataset and the red horizontal line within the box represents the median of the central 50% dataset. The lower and upper extreme red whiskers, extending from the box, represent the minimum and maximum range of the data, respectively. Each blue circle (with red border) outside the extreme whiskers represent an outlier.

C. Dynamics Modeling of Three-module Soft Arm

To model the dynamics of the three-module soft arm (shown in Fig. 2b), we used a data-driven modeling technique where the data gathered on the robot were used in supervised learning (using a neural network as a function approximator) to predict the general behavior underlying the gathered dataset. For this purpose, we collected the tip movements of all three modules using the Vicon system for a dynamically saturating pressure signal for 10 $mins$ (6000 points at 10 Hz). Here the time chosen can be treated as a hyperparameter, which varies depending on the soft arm complexity, desired accuracy, and the function approximator selected for modeling. A pressure ceiling was set for safe operation. During the data gathering, the pressure ceiling was varied dynamically, while respecting the safety threshold, and saturated at that pressure for randomly chosen time instants for all nine pressure signals. This helped in acquiring a variety of non-repetitive robot movements within its workspace.

The movement recorded at an instant t is in terms of a state (x_t) and action (τ_t) vectors with a dimension of 1x9 each. The state vector includes the end-effector positions (in task space) of all three modules, while the action vector has nine signals for nine pressure chambers. We trained a recurrent neural network based architecture (using Long-Short-Term-Memory (LSTM)-type layers) with the gathered data. The architecture is a single hidden layer with 128 nodes, followed by a Dense output layer, and a 30% neuron-drop dropout layer with Adam optimizer and non-linearly decreasing learning rate from 0.001 to 0.0001. We used the *softsign* activation function for the input, hidden and output layers. The dataset was normalized in the range -1 to 1 to optimize the training process. The network architecture was trained with 30 batch size and 50 epochs. It was treated as a multivariate regression problem for time-series forecasting. The dynamics model training took approximately three minutes on a laptop with Python-based environment, a 64 bit Linux-based operating system, 32 GB RAM (with 32 GB virtual RAM) and Intel core i7-10750H CPU@2.60GHz. The trained neural network predicts only a single time step ahead state vector (x_{t+1}) with an input vector that includes the pressure signal (τ_t) associated with the next state vector, the current state vector and the associated pressure signal (x_t, τ_{t-1}), and the past state vector and the associated pressure signal (x_{t-1}, τ_{t-2}).

$$\mathcal{F}_\phi(x_{t+1}|\mathcal{X}, \tau) \text{ where}$$
$$\mathcal{X} = [x_t, x_{t-1}], \text{ and } \tau = [\tau_t, \tau_{t-1}, \tau_{t-2}] \quad (1)$$

The dynamics model is shown in Eq. (1). During model evaluation or use in the training environment, current predictions of the trained dynamics model are used as feedback for next instant state prediction, accumulating error over time and therefore, contributing to the training-to-reality gap.

IV. Proposed Control Architecture

This section presents the offline policy training using an RL algorithm for the desired task in a training environment with the data-driven dynamics model, and the two proposed online control strategies to recover desired accuracy.

A. Offline Policy Training

PPO algorithm [5] exploits monotonic on-policy improvements, while demonstrating improved sample efficiency in its class of algorithms, with minimal requirements for hyperparameter tuning. For soft robots, its ability to incorporate adaptive exploration, flexibility in hyperparameter tuning, and scalability to complex environments is particularly useful. In this work, we have used this algorithm for offline policy training.

1) Action and Observation Space: Based on the information presented in Sect. III-C, the input and output dimension is 45 and 9, respectively, as shown in Eq. (1). For the control policy training, we wrapped our dynamic model in the training framework presented by openai gym [60] and employed the algorithmic routine by Haarnoja et al. in [61] for PPO implementation. The observation space of the learning agent was a continuous space and consisted of the transition state (x_{t+1}) predicted by the dynamics model $(\mathcal{F}_\phi)$ and the distance of the third-module tip from the goal and the obstacle. The observation space was normalized in the range -1 to 1 based on the minimum and maximum values (in individual axes in each module) taken from the dataset used for the dynamics model training. Similarly, the action space of the learning agent was also a continuous space vector of nine signals. Each value in the action space ranged from -0.2 to 0.2.

2) Task Description: The goal of the task was to achieve high precision in reaching a chosen goal-point in 3D space while avoiding collision with the obstacle, with a controller acting in the dynamics domain of the soft arm. High-precision is assessed in terms of a percentage error i.e., error divided by the soft arm's characteristic length. Based on this, the task objective was set to impose a percentage error of $\leq 1\%$, i.e., $\leq 5\ mm$ of acceptable distance error, for a soft arm of length $598\ mm$, between the tip of the soft arm's distal module (denoted by x_t^3 at a time instant t) and the chosen goal-point. At every instant t, G_{dist} and O_{dist}, the distances of x_t^3 from the goal-point and the obstacle respectively, were calculated. Here, G_{dist} was always computed using a simple Euclidean distance formula between current tip and desired goal-point, while O_{dist} was a Euclidean distance of a 3D point from a cuboid plane (P_{obs}) computed using $\frac{\vec{n}\cdot\vec{V}}{|\vec{n}|}$, where $\vec{n}$ is a vector normal to P_{obs}, $\vec{V}$ is a vector from x_t^3 to an arbitrary point on P_{obs}.

The reward function (given in Eq. (2)) for this task was composed of three parts: 1) **Mind the boundary;** It limited the soft arm search space, ensuring faster convergence by avoiding time spent in space too far from the desired goal. The boundary in Eq. (2) is represented by B and its limits are described in Sect. III-A. It also added a constant -2.0 per-step penalty to encourage policy search for shortest path to the goal-point. 2) **Avoid collision;** At every step, O_{dist} was calculated. If it ranged from $20\ mm$ to $10\ mm$, a warning was generated and a proportional penalty was added to the overall reward, but the environment did not reset. The environment reset only if the collision flag was up and, if it was, the environment reset with a substantial penalty. The training environment did not include contact modeling; The collisions were detected mathematically, at every instant, in two ways: a) intersection between a line segment $l_1 = line(x_t, x_{t-1})$ and a finite plane P_{obs}, or b) $O_{dist} \leq 10\ mm$. 3) **Distance to the goal;** At each step, G_{dist} was also computed, and its negative added as a continuous penalty to the overall reward. The goal-point was considered reached if $G_{dist} \leq 5\ mm$. Note, the first and second penalties were quite sparsely distributed, so the third penalty mainly drove the policy training, nonetheless, resulting in successfully learning the task in the offline training environment.

$$\text{reward} = \begin{cases} -100.0 & x_t^3 \notin B \\ -5.0 * (20.0 - O_{dist}) & 10 < O_{dist} \leq 20 \\ -10000.0 & \vec{l_1} \cdot \vec{n} \neq 0,\ O_{dist} \leq 10 \\ -G_{dist} & G_{dist} > 5.0 \end{cases} \tag{2}$$

3) Training: We used a single hidden dense layer with 128 neurons for the critic network and two hidden dense layers with 64 neurons each for the actor network. For activation in the actor network, we used *softsign* in the input, hidden and output layers. A linearly decreasing learning rate from 0.01 to 0.001 and 0.003 to 0.0001 was selected for the critic and the actor, respectively. The other hyperparameters were as follows: episode length to 150 timesteps, batch size to 5, epochs to 10, and number of policy updates per episode to 6.

To speed up the process, the gym-based training environment with the dynamics model was vectorized and 4 processes (one training environment per process) were launched in parallel to share the rollouts for the policy training. On a laptop with Python-based environment, 64 bit Linux-based operating system, 32 GB RAM (with 32 GB virtual RAM), Intel core i7-10750H CPU@2.60GHz, and NVIDIA GeForce RTX 4080 GPU, the policy training took approximately $3\ hrs$ for a total of five million timesteps (35000 episodes with maximum 150 timesteps per episode). There was no substantial change in the reward after 15000 episodes, however, the number of timesteps needed to reach the goal-point with desired accuracy continued to decrease. The significance of this point is highlighted in Sect. VI.

The RL-policy $(\pi_\theta(\tau_t|x_t))$ aimed to maximize the reward function given in Eq. (2), by attempting to reach the goal-point as fast as possible while avoiding the obstacle. The policy took continuous actions in the range -0.2 to 0.2, while the dynamics model accepted actions in the range -1 to 1. The range -1 to 1 corresponded to the pressure from 0 to $1.5\ bars$. The gym-based tracking environment kept track of the action taken by the agent at the previous step. The current action proposed by the agent was added to the previous action and passed on to the dynamics model. A constraint applied in the training environment ensured that the overall action passed on to the dynamics model was not above the threshold

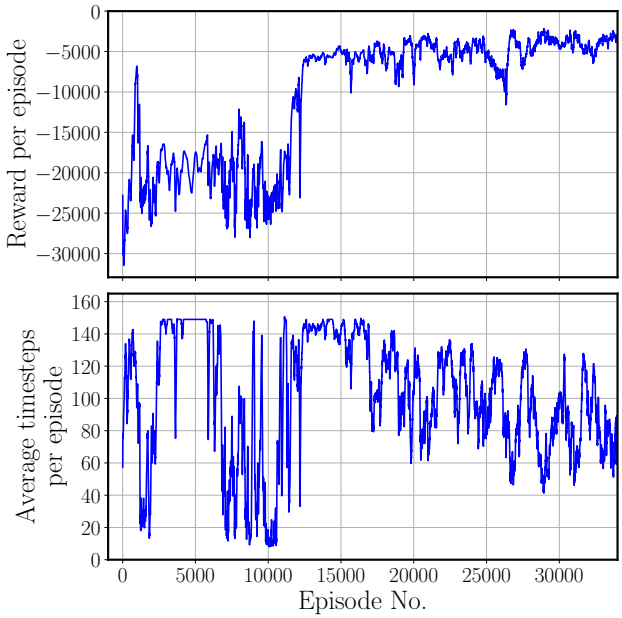

Fig. 3. The figure shows offline policy training for the desired task within the training environment. The training took three hours on average. The policy training was performed using the reward function as described in Eq. (2). The subfigure on the top refers to the reward per episode, while the one on the bottom refers to the average number of timesteps per episode. We set the maximum length of an episode to 150 timesteps.

$(1.5\ bars)$ as it might compromise the safety of the robot during policy testing. Another constraint was applied to ensure that the difference between the action passed to the model in the previous step and current step was not above the safety threshold: if it was, the environment did not take the action and added the usual penalty. The safety threshold was set to $0.15\ bars$.

B. Bayesian Optimization Assisted Coaching (BOAC)

This scheme is based on IL by coaching, inspired from [6] where there is an oracle responsible for generating trajectories for a desired task. A student policy is trained by using these trajectories. The student policy then attempts to predict actions as good as the oracle's on the training set. This approach works particularly well if there exists a significant difference in the information of the environment available to the oracle and the student policy [8]. In our setting, the RL-trained policy (π_θ) acted like the oracle while the student policy (π_i) was a new policy trained and optimized based on the trajectories generated by executing π_θ with the soft arm.

IL by coaching also requires a coach as presented by Hal Daumè et al. [6]. The coach can be a human or a synthetic agent, responsible for providing the student policy with easy-to-reach goals and incrementally raising the level to match the desired goals based on the progress as seen from a value function. In our case, the coach was a synthetic agent based on a k-means clustering algorithm, responsible for providing easy-to-follow trajectories to reach the desired goal-point with reduced accuracy, and incrementally raising the accuracy to match desired precision. The progress was tracked based on a Mean Squared Error (MSE) computed from the currently-followed trajectory and coach-proposed trajectory. For a given goal-point and obstacle placement in the training environment, the RL-trained policy can produce a trajectory that avoids the obstacle and reaches the goal-point with the desired accuracy. During the policy evaluation (and also in the training environment), different trajectories can be obtained, even when executing the policy in the deterministic mode, by changing the initial conditions of the dynamics model.

So, different deterministic trajectories were accumulated in a data buffer $\mathcal{D}_d$ and k-means clustering algorithm was applied on them. The K-means clustering algorithm generated Lx3 clusters where L was equal to the average length of the trajectories. The coach was trained for 10 different seeds, randomly chosen initial centroids, and 300 iterations with a tolerance of 10^-4 using an Expectation-Maximization-style algorithm [62]. The algorithm took less than a second to train on the work station defined in the Sect. III-A. The output cluster was a new path made up of the centroids of clusters whose points have been visited the more often by the policy in deterministic mode. The path was treated as a sequential series of target states (x_t^d) to be reached by the student policy to reach a desired goal-point.

The schematic flow of this approach is presented in Fig. 4 and the algorithm 1. In the algorithm, d_{π_θ} represents a distribution of n episodes as given in Eq. (4) where each episode consists of T state-action pairs $(x_t, \pi_\theta(x_t))$. The state-action pairs were converted into a timeseries in a sequence, as shown in the Eq. (3), where x_{t+1} was a state transition from current state (x_t) based on the action $(\pi_\theta(x_t))$. Each state-action pair has an associated reward that was also calculated from the testing environment with reduced desired accuracy. Based on these rewards, a total return can be computed for the n^{th} episode as $R(n)$. N episodes with maximum total return were sampled from d_{π_θ} and stored in the buffer $\mathcal{D}$. This buffer was used to train and optimize the student policy.

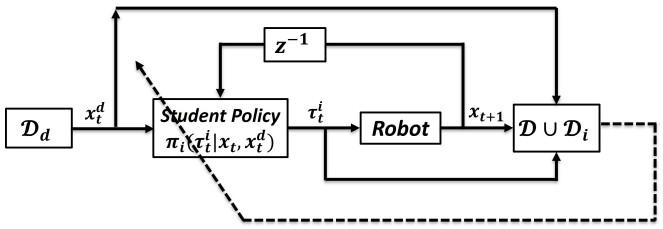

Fig. 4. The figure shows the schematic flow of BOAC. Gaussian process-based student policy π_i was trained and optimized periodically using a data buffer $\mathcal{D}$ to eliminate the training-to-reality gap. The policy takes the desired (x^d) and the current robot state (x) to predict an action (τ) to reach desired state.The dotted line underscores the batchwise training of the student policy based on the joint buffer $\mathcal{D} \cup \mathcal{D}_i$.

$$\Big(x_{t+1}, x_t, \pi_\theta(x_t)\Big) \leftarrow \Big(x_t, \pi_\theta(x_t)\Big) \quad \text{where } t = 1 : T-1 \tag{3}$$

$$d_{\pi_\theta} = \left\{\left(x_{t+1}^{(n)}, x_t^{(n)}, \pi_\theta(x_t^{(n)})\right)\right\}_{t=1}^{T-1} \quad \text{where } n = 1, 2, 3, ... \tag{4}$$

Algorithm 1 Bayesian Optimization Assisted Coaching (BOAC)

Initialize $\mathcal{D},\ \mathcal{D}_i,\ \mathcal{D}_d,\ \pi_0$
$\mathcal{D} \leftarrow$ Sample N trajectories from d_{π_θ}
Train π_0 on $\mathcal{D}$
while not $\mathbb{G}_f$:
    Coaching trajectories: K-Means Clustering ($\mathcal{D}_d$)
    while not $\mathbb{L}_f$:
        Coaching episode: $\{x_t^d\}_{t=0}^{T}$
        for $i = \mathbf{n}_B : \mathbb{T}_B$ **do** :
            (state: x_{t+1}, action: τ_t^i) $\leftarrow \pi_i(x_t, x_t^d)$
            Reform: $\mathcal{D}_i \leftarrow (x_{t+1}, x_t^d, \tau_t)$ (as in Eq. (3))
            $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
            Re-train: π_{i+1} on $\mathcal{D}$ using MSE loss

Algorithm 1 and Fig. 4 outline this approach once the coach-proposed trajectories are acquired. π_0 in the Algorithm 1 represents the preliminary version of the student policy as trained on $\mathcal{D}$ using MSE as a surrogate loss function. Other elements in the BOAC include $\mathbb{G}_f$, $\mathbb{L}_f$, $\mathcal{D}_i$, (π_i), n_B and T_B which are the global goal flag (for the desired goal), local goal flag (surrogate loss below 5 mm), instantaneous data buffer to store state-action reformed pairs, i^{th} instant of student policy, batch size for periodic student policy optimizations and total number of batches inside an episode, respectively. Note that the student policy was based on a Gaussian process and it acted as a local inverse dynamics model of the underlying soft arm.

C. Gaussian Process-based Recurrent Cerebellar Architecture (GPRCA)

This scheme aims to train an adaptive plant-compensator based on recurrent cerebellar architecture [9]. It was originally proposed to compensate for the three-dimensional vestibulo-ocular reflex to solve the motor-error problem. Porrill et al. [9] presented it as a converging solution to the modular control of systems with high degrees of freedom. The architecture is shown in Figure 2b in [9]. This architecture computes the training signal (motor-error) for the compensator as $\mathbf{e}(t) = \hat{\mathbf{x}}(t) - \mathbf{x}(t)$ where $\hat{\mathbf{x}}(t)$ is an observation coming from the plant model, and $\mathbf{x}(t)$ is the actual observation.

In our case, the control policy was learned in a training environment with the dynamics model ($\mathcal{F}_\phi$) of the robot as in Eq. (1). The model was an approximate depiction of a deformable robot with virtually infinite degrees of freedom and it did not account for the intrinsic or extrinsic uncertainties in the observations. We thus used [9] as an adaptive compensator to adapt the trained policy to the real environment. We revamped this recurrent architecture to suit our current implementation as shown in Fig. 5. The dynamics model took feedback directly from the soft arm, as shown in Fig. 5, ensuring the information being fed to it was $\mathcal{X} = [x_t, x_{t-1}]$, and consequently, the difference between the training environment and the soft arm response was captured, at the current instant without any drift, to train the compensator.

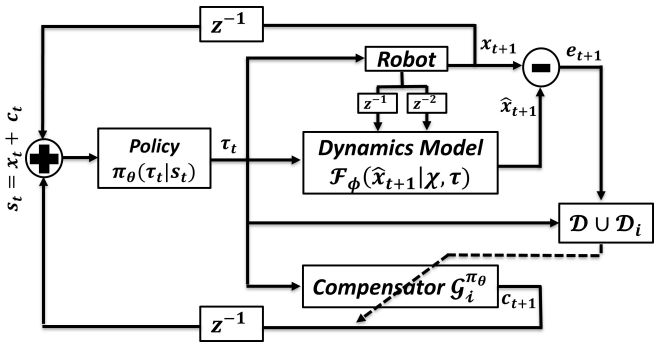

Fig. 5. The figure shows the schematic flow of GPRCA. The approach employs a Gaussian Process $\mathcal{G}_\mathbf{i}^{\pi_\theta}$ to bridge the training-to-reality gap with $e(t)$ as the training signal and actions proposed by the RL-trained policy as an input. The output of this process is a compensatory signal for the observation from the soft robot based on the robot dynamics model used in the original training environment. The dotted line underscores the batchwise training of the compensator based on the joint buffer $\mathcal{D} \cup \mathcal{D}_i$.

At an instant $\mathbf{t} = \mathbf{0}$, the RL-trained policy (π_θ) generated an action based on the arm's current resting position. The action was used to generate the next instant observation of the arm and the training environment. The discrepancy between the training environment and the soft arm observation generated the error signal. The rollout dataset (Eq. (5)) was used to train the Gaussian Process ($\mathcal{G}_\mathbf{i}^{\pi_\theta}$), which sent a compensatory signal in order to bridge the gap (the gap is visible in Fig. 7 without a compensatory signal). In the rollout dataset, $e_t = \hat{x}_t - x_t$ is the sensory-error signal where $\hat{x}_t = \mathcal{F}_\phi(\pi_\theta(s_{t-1}))$ with $\mathcal{X} = [x_t, x_{t-1}]$ and, x_t is the observation from the robot under the same action $\pi_\theta(s_{t-1})$.

$$\mathcal{D} = \left\{ \Big(\pi_\theta(s_{t-1}),\ e_t \Big) \right\}_{t=1}^{T}$$
$$\text{where } e_t = \hat{x}_t - x_t, \text{ and } \hat{x}_t = \mathcal{F}_\phi(\pi_\theta(s_{t-1})) \tag{5}$$
$$s_{t-1} = x_{t-1} + c_{t-1},\ x \in \mathcal{X}^R \text{ and } c \in \mathcal{G}^{\pi_\theta}$$

The compensator predicted $\mathbf{c}_t$ given the input signal (action predicted by the RL-policy); $\mathbf{c}_t$ was then added to the observation for the next instant. The new compensated observation was used to predict the action for the following instant, and so on. Based on the current action proposed by π_θ, $\mathcal{G}^{\pi_\theta}$ produced the compensatory signal for the next instant, as shown in Fig. 5 and Algorithm 2. In the algorithm, n_B is the batch-size for training the compensator (it may be considered as a hyperparameter, we set it to five timesteps) and $\mathbb{L}_E$ is the length of the episode executed. This scheme was executed until it reached the desired accuracy (i.e., until the global goal flag ($\mathbb{G}_f$) was raised), as opposed to gradually increasing the desired accuracy as in Algorithm 1 where there were also local goal flags ($\mathbb{L}_f$).

V. Results

This section presents the evaluation scenarios and the results achieved with the RL-policy and proposed control strategies.

A. Evaluation Scenarios

In the first set of experiments, the RL-policy was tested with the soft arm in open and closed loop setting to achieve task

Algorithm 2 Gaussian Process based Recurrent Cerebellar Architecture (GPRCA)

Initialize $\mathcal{D}$, $\mathcal{G}_0^{\pi_\theta}$
Get Dynamics Model: $\mathcal{F}_\phi$, Policy: π_θ
while not $\mathbb{G}_f$:
  for $i = \mathbf{n}_B : \mathbb{L}_E$ **do** :
    $c_t \leftarrow \mathcal{G}_i^{\pi_\theta}(\pi_\theta(s_{t-1}))$
    $\hat{x}_{t+1} \leftarrow \mathcal{F}_\phi(\pi_\theta(s_t))$
    $x_{t+1} \leftarrow$ **robot**$(\pi_\theta(s_t))$
    $\mathcal{D}_i \leftarrow (\tau_t,\ e_{t+1})$ (as in Eq. (5))
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
    Re-train $\mathcal{G}_{i+1}^{\pi_\theta} \leftarrow \mathcal{D}$

execution with desired accuracy, without using the proposed control strategies, demonstrating a performance gap. In the second set of experiments, labelled as **scenario 0 or S0**, proposed control schemes were employed to bridge the exhibited performance gap. In the final set of experiments, proposed strategies were evaluated for additional four scenarios, showcasing their adaptability to damage incidents.

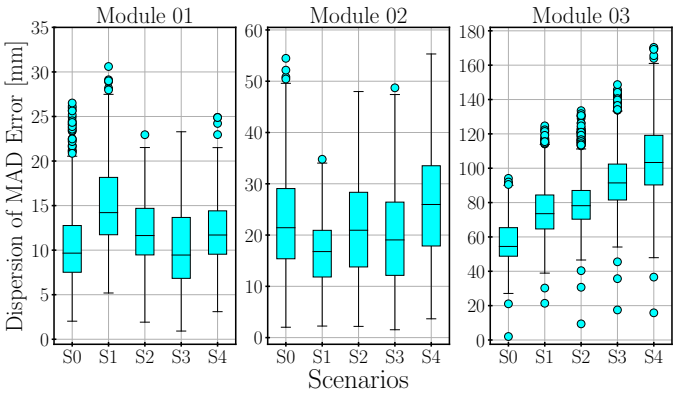

Fig. 6. The figure presents distribution of a dataset representing the performance gap across different evaluation scenarios. The gap distribution was formed by executing an actuation-space trajectory, as used for the trials in Sect. III-B, with the soft arm and its dynamics model several times. Mean Absolute Distance (MAD) error (in millimeters) was computed between the dynamics model in the training environment and the response of the individual modules in the soft arm for scenarios 0 to 4.

In the three-module soft arm, each module is independently actuated with three pneumatic chambers. In **scenario 1 or S1**, we disrupted the pressure supply of one of the chambers in the first module (connected to the base of the soft arm) by deploying a manual pneumatic rotary knob for pressure control. Similarly in **scenario 2 or S2**, we restored the pressure supply for the chamber in module 1 and disrupted a chamber in module 2. A similar pattern was also repeated in **scenario 3 or S3** where a chamber in module 3 was disrupted, while the chamber in module 2 was restored. In **scenario 4 or S4**, we disrupted two chambers, one in module 1 and another in module 2. These disruptions forced the soft arm to employ redundant limbs to compensate for the change in its behavior. The clear difference in performance among different scenarios is shown in Fig. 6, which highlights that each subsequent scenario tends to pose a bigger performance gap than the preceding one.

B. RL-policy Testing

1) Open-loop Testing: The trained policy was tested in an open-loop setting with the soft arm, where the instantaneous positions of the soft arm did not influence the action selected by the policy. We observed that replicating the sequence of actions taken in the training environment to reach the desired goal did not result in the soft arm successfully reaching the goal-point because of the training-to-reality gap. We conducted 17 trials in this setting, which are summarized in Fig. 7.

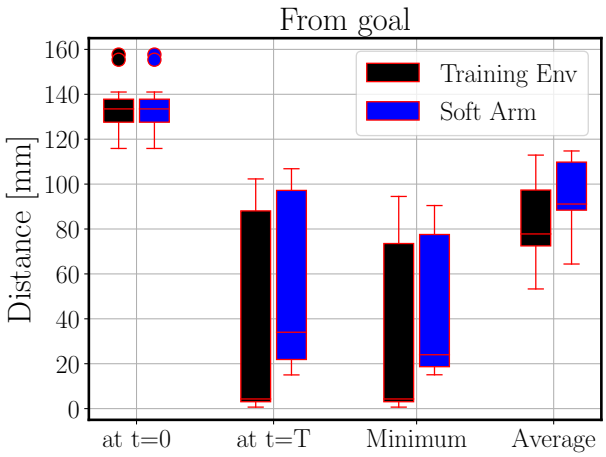

Fig. 7. The figure shows the open-loop testing of π_θ. The boxplot presents four quantities calculated from 17 trials: start-point, final point, minimum, and average distance from the goal. In all the trials, even with different initial conditions the training environment achieved a 94% success rate with the desired accuracy ($\leq 5\ mm$). However, in this setting, policy testing with the soft arm, the success rate dropped to 0% in terms of achieving desired accuracy. Nevertheless, in 47% of the trials, the soft arm managed to approach the goal-point within a range of 15 mm to 25 mm. In approximately 20% of the trials, a collision occurred, and in the remaining cases, the soft arm settled at a distance greater than 25 mm from the goal-point.

2) Closed-loop Testing: For closed-loop testing, the dynamics model in the testing environment was replaced with the soft arm. Instantaneous positions of the soft arm, and distances to the target and obstacle (calculated in real-time) were fed to the policy for action-selection. Based on the results with the open-loop setting, we set the desired accuracy to 15 mm. We consideried the policy successful if it managed to get the robot to a distance $\leq 15\ mm$ from the goal-point without collision. The result with this setting for a total of 21 trials is shown in Fig. 8.

C. Online Optimization for the Training-to-Reality Gap

It is clear from the training-to-reality gap (S0 from the Fig. 6) that we need a solution that can generalize well over the stochastic nature of the robot and the discrepancy due to the dynamics model. Although the RL-trained policy was capable of doing this, it lowered the accuracy of task execution. We now present two control schemes to recover the desired accuracy.

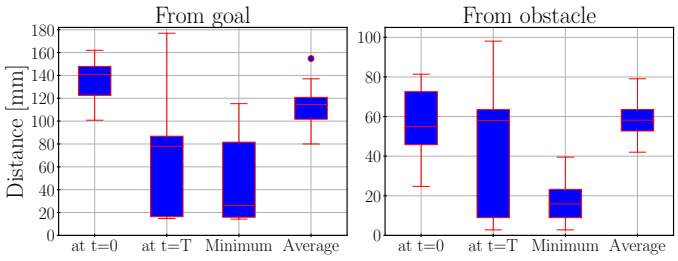

Fig. 8. The figure shows the closed-loop testing of π_θ with the soft arm. Both boxplots illustrate four quantities derived from 21 trials: start-point, final point, minimum, and average distance from the goal (left subplot) and the obstacle (right subplot). In a closed-loop setting, the observed outcomes included successfully reaching the goal with the newly defined accuracy threshold (38%), experiencing a collision (33%), or terminating the episode without reaching the goal-point or facing collision (29%). On average, the outcomes took 63, 15, and 150 timesteps, respectively.

1) BOAC for Optimization: The algorithmic flow of this scheme is introduced in the Algorithm 1. A Gaussian process [63] based student policy was trained using the data in the buffer $\mathcal{D}$. The buffer initially has N trajectories sampled from d_{π_θ}. The value of N may be considered as a hyperparameter here because having an insufficient number of these trajectories may mean that the underlying behavior of the oracle is not elicited, and too many may cause an increase in the training time. The value can, in any case, be decided by the hit-and-trial method. In our case, we chose seven trajectories ($N = 7$) with high return to train the initial version of the student policy (π_0).

Figs. 9a and 9b show the results with this scheme for the desired goal-point $[-30.0, -120.0, -620.0]$, obstacle position $x \in [-60.0, 60.0]$, $y \in [-50.0, -55.0]$ and $z \in [-700.0, -570.0]$ and boundary for the robot operation restricted to dimension with $x \in [-130.0, 90.0]$, $y \in [-150.0, 50.0]$ and $z \in [-700.0, -570.0]$. We executed eight optimization trials dedicated to three subsequent coaching profiles (for 15 mm, 10 mm and 5 mm, respectively). We recorded the results for various different goal-points to evaluate the effectiveness of this controller in the training-to-reality gap. The compilation is shown in Table II under BOAC for S0.

2) GPRCA for Optimization: The algorithmic flow of this scheme is illustrated in the Algorithm 2. We set the episode duration to 100 timesteps for this scenario. We tested it with the same goal as in Sect. V-C1 i.e., $[-30.0, -120.0, -620.0]$, the position and dimension of the obstacle were also kept the same. However, given the stochastic nature of the policy, the boundaries for robot operation were relaxed to $x \in [-170.0, 80.0]$, $y \in [-170.0, 50.0]$ and $z \in [-700.0, -570.0]$. To draw a conclusion on the comparison of Algorithm 1 and 2, we also executed this scheme for eight iterations of the online optimization. The results with this scheme are as shown in Fig. 10a, and 10b. We were able to achieve the goal-point in the third trial. In the first two trials, the scheme ran for the complete episode length without collision. The results for more goal-points for this scenario are reported in Table II under GPRCA for S0.

D. Online Optimization for the Damage Recovery

So far, the results for overcoming the training-to-reality gap have been presented using Algorithm 1 and 2. The following trials were aimed at evaluating the performance of the proposed algorithms for scenarios 1 to 4. The RL-trained policy (π_θ) was not re-trained for any damage incident to the soft arm. However, for each new task setting (i.e., different goal-point, obstacle location or the search boundary), a new RL-policy was trained.

1) BOAC for Task Recovery: BOAC was executed for damage scenario 1 while keeping the obstacle position, exploration boundary, and goal-point the same as in Sect. V-C1. The results achieved after 15 optimization trials are shown in Figs. 9c, and 9d. We discuss the response of BOAC in Sect. VI, and the results with more goal-points and damage scenarios are reported in Table II.

2) GPRCA for Task Recovery: The training environment with the dynamics model of the soft arm is kept the same, but the soft arm undergoes various changes due to the damage scenarios. The compensator was trained in real time with each different scenario to elicit adaptation to the new soft arm behavior by learning the modal-mismatch from the initial training environment to the current behavior of the soft arm. The experimental set-up remained the same, including obstacle position, exploration/boundary restriction, goal-point, and desired accuracy.

To adapt to damage scenario 1, GPRCA was executed with the same number of time-steps as in S0 (100 timesteps per optimization trial) and the results are shown in Figs. 10c, and 10d. For damage scenarios 2, 3, and 4, the soft arm undergoes substantial changes not only in terms of reduced workspace access but also the strength diminishes as the robot starts to throb (see Sect. VI). Consequently, to adapt to damage scenarios 2, 3, and 4, we executed the online training of the compensator for 150 time-steps (as done originally in the offline training environment). The results achieved for scenarios 2, 3 and 4 are shown in Fig. 11. The results with this scheme for different goal-points and damage scenarios are reported in Table II.

VI. Discussion

Table I reports the decrease in performance when a control solution derived in a non-data-driven [10] or data-driven [11]–[13] model setting is tested with the soft robot. The studies, reported in the table, recorded the performance variation or degradation as a result of the performance gap (from the simulation or training environment to the robot) without presenting any solution for it. Our target here is to impose high-precision task recovery while overcoming the training-to-reality gap. IL, traditionally, is a sample-efficient approach but the results are usually qualitative; However, combining IL with RL decision-making capability can yield quantitative results, rendering it an excellent candidate for tackling the problem in question. BOAC takes inspiration from this approach. Another novel approach to adaptability can be learning the performance gap and using it as a compensatory agent, as was presented in [9]. GPRCA employs similar strategy. Additionally, the

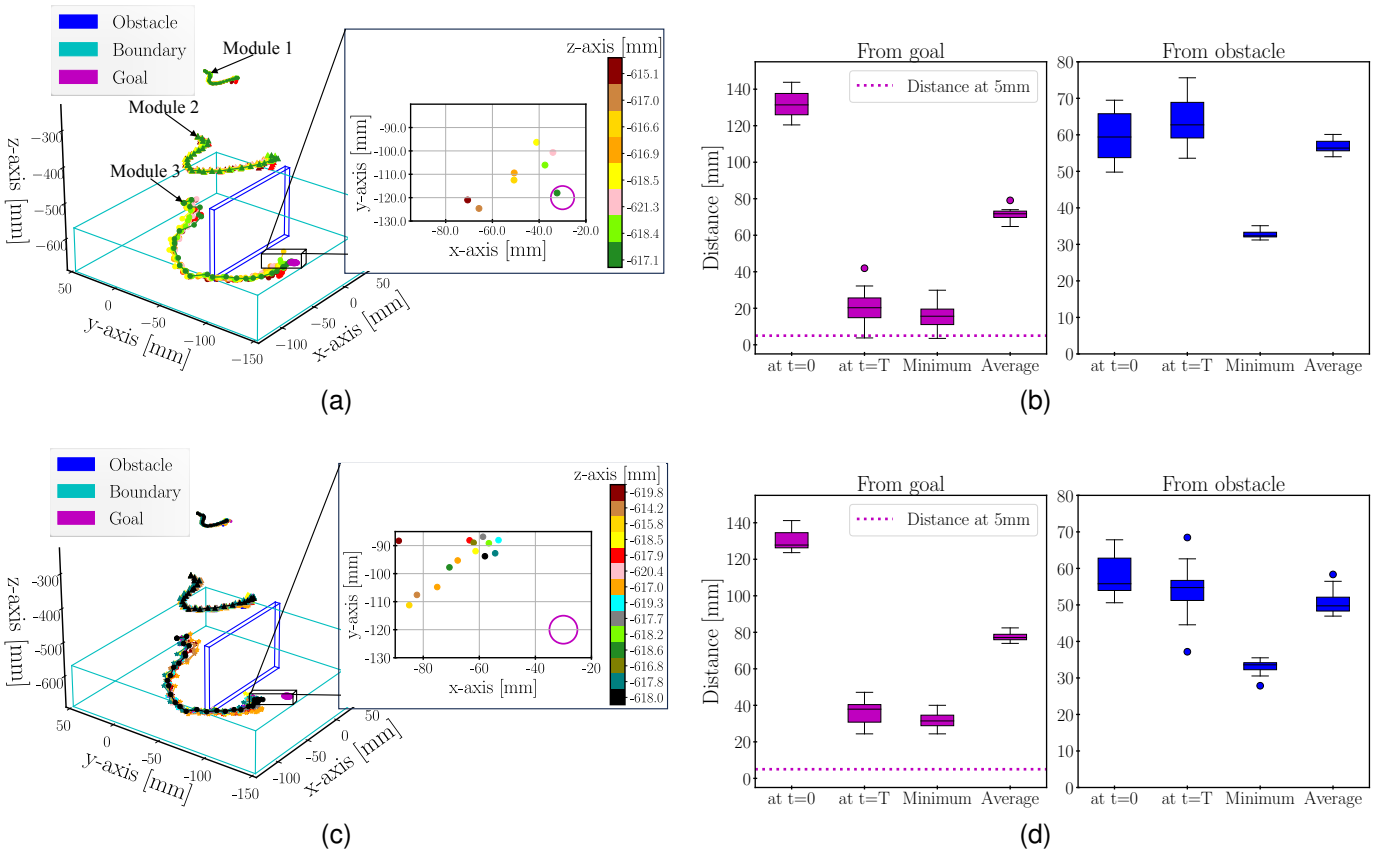

Fig. 9. The figure presents the results of BOAC implementation for S0 and S1. The 3D trajectories in (a) and (c) represent the response of the soft arm for S0 and S1, across 8 and 15 trials, respectively. These figures also include a zoomed-in 2D top view of the desired goal point with a 5 mm radius circle around it. The zoomed-in image only shows the coordinates along the x and y axes reached at the end of each trial and the colorbar shows their z-coordinates. The boxplots in (b) and (d) highlight the distance of the tip from the goal and the obstacle (left and right sub-figure, respectively) for scenarios 0 and 1, after 8 and 15 trials, respectively. Each trial in both scenarios is 30 timesteps (3 sec) long.

practicality of such adaptive approaches for soft robots require sample efficiency. Therefore, BO was employed in BOAC and GPRCA. The resultant time and accuracy of our approaches have been compared with similar studies found in the literature (refer to Sect. II-C).

Table III reports the average time (offline and online training time) and accuracy for RL, BOAC and GPRCA across all the conducted trials where the goal was reached (with or without damage) for all the goal-points and obstacle positions. On average, GPRCA takes longer in online training compared to BOAC, but both manage to get the sufficient accuracy. Although the online training time seems longer for GPRCA, it actually took fewer episodes than BOAC. This is because BOAC exploits on the coach-proposed trajectories, which are always of the same length (30 timesteps as in Fig. 9, 21 for **G1** as in Table II, and so on). GPRCA, on the other hand, is executed freely with chosen timesteps per episode. In many cases, restarting the episode more often is considered less favorable than letting an optimization run for longer timesteps. So, BOAC performs effectively if a smoother performance is required as shown in the Fig. 9 but the user may have to restart the optimization episode more often. GPRCA performs effectively in the other situation (when restarting the optimization episode more often is less favorable Figs. 10 and 11). Additionally, executing GPRCA for higher timesteps results in the compensator learning the gap better. In the trials shown in Figs. 10 and 11, by continuing to execute GPRCA after the goal-point has been reached with the desired accuracy, the compensator continues to learn effectively from the modal-mismatch and manages to reach the goal faster in the subsequent trials.

Fig. 6 highlights that the performance gap increases with each successive scenario. In addition, the soft arm also undergoes workspace reduction as the pressure supply to the pneumatic control chambers is interrupted. In the subsequent goal-points and scenarios, the optimization process in BOAC has slowed down, as can be seen in Table II. As a result, it is also possible that the goal may not be reached as the student policy may saturate or simply start diverging due to error compilation after a significant number of trials, as observed in S2 of **G1, G2** and S0 of **G3** (error percentage is $\geq 1\%$.). Since GPRCA trains the compensator ($\mathcal{G}^{\pi_\theta}$) from scratch based on the modal-mismatch at that point, the soft arm is able to reach the goal-points, even in later goal-points and scenarios. However, the number of optimization trials increases, as shown in Table II. For the goal-point **G3** in S3 and S4, the error percentage is $\geq 1\%$ even with a significant number of iterations. This could be attributed to either the soft arm's current damage, making the goal unreachable, or the significant divergence between the policy generated in the

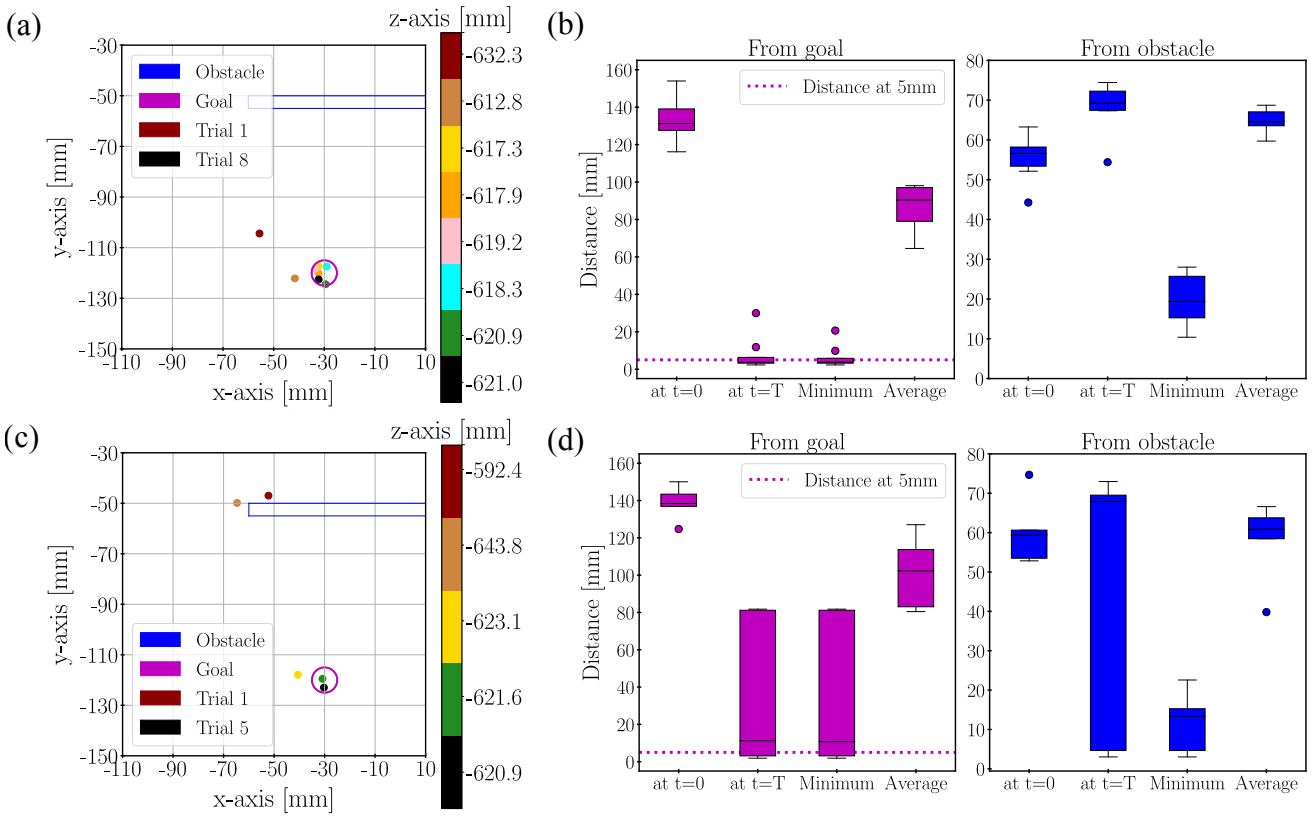

Fig. 10. The figure shows the results of algorithm 2 implementation, using episode length of 100 timesteps, for S0 and S1 for a total of 8 and 5 trials, respectively. The points reached at $t = T$ for all the trials are shown in (a) and (c) for S0 and S1, respectively. Same as before, the boxplots in (b) and (d) represent the distribution of the dataset resulted from the GPRCA trials for S0 and S1. On average, S0 and S1 reached the goal-point in 73 and 77 timesteps after 2 and 3 trials, respectively.

initial training environment and the one needed for the robot's current setting.

We conducted an additional set of experiments (included in the supplementary material of this study) with a two-module soft arm for trajectory tracking problem, with and without external loads attached to the tip of both modules. The policy was trained using an RL algorithm with a data-driven model with no knowledge of the external loading. Consequently, the policy performance degraded with the new soft arm setting due to an increase in the stochasticity and a decrease in the workspace of the two-module soft arm, as it was also seen when we deliberately damaged the three-module soft arm. BOAC and GPRCA managed to improve accuracy in trajectory tracking, with external loading, reducing the Mean Absolute Error (MAE), on average, by 84% and 93%, respectively, without retraining the RL-policy with the new soft arm setting.

We listed imbalanced morphology due to manufacturing inaccuracies, varying material properties, and incomplete depletion of the pneumatic chambers as the factors responsible for the soft robot's stochasticity. This variability combined with the drift and error-accumulation over finite horizon by the dynamics model are the root causes for the training-to-reality gap. However, there are other factors responsible for the task performance degradation. For instance, the ability of a control solution is greatly affected due to the task execution speed as reported in [24] where the gap for kinematic control to dynamic control solution exhibited almost 50% increase in error (MAE changed from $12-27\ mm$ to $22-44\ mm$). while our approaches bridge the gap, they do not present solution for the setting where the task constraints have changed, such as different goal-point, obstacle location or task domain (e.g., kinematics or dynamics). To tackle new task settings, a more generic policy (e.g., meta or multi-task policy) or an additional state- or action-exploration-based policy search algorithm may be required.

VII. Conclusions

We have highlighted the challenges in controlling soft robots arising from their inherent stochastic behavior, which is influenced by different elements such as elastic material properties, flexible shape, variable initial conditions and manufacturing inaccuracies. RL-based algorithms enable training intrinsically stable control policies, making them a strong candidate for soft robots' control problem. However, despite their adaptive nature, stochasticity hinders their direct application to soft robots, reducing the task-precision achievable in deploying them. This issue is exacerbated in applications that require high precision. We thus developed two strategies, built on top of the trained RL-policy, that leverage the data-efficient nature of the BO and critical problem-solving capability of the PPO algorithm. We successfully demonstrated that our schemes achieve the desired accuracy while bridging the training-to-

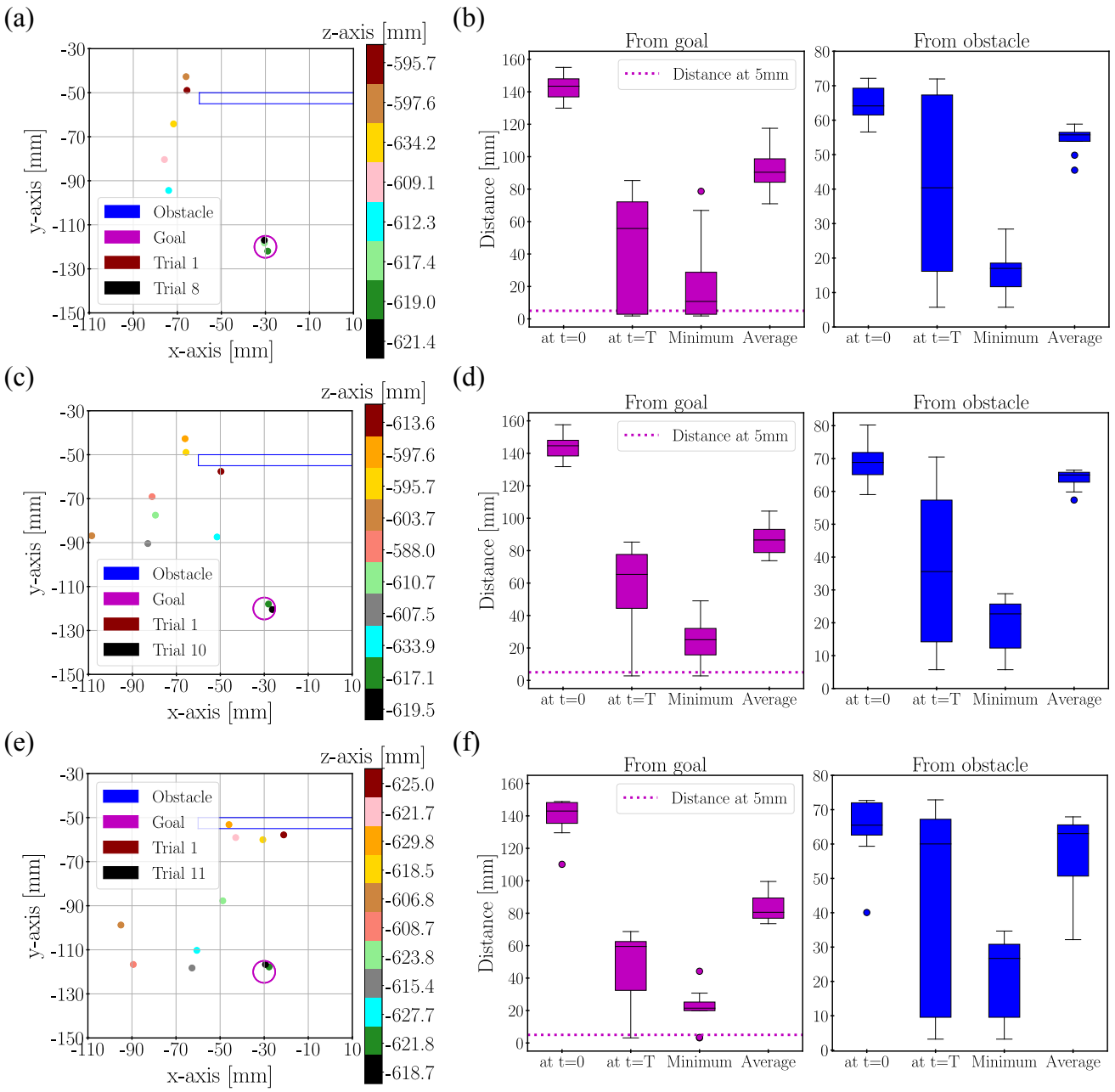

Fig. 11. The figure shows the results of algorithm 2 implementation, using episode length of 150 timesteps, for S2, S3 and S4 for a total of 8, 10, and 11 trials, respectively. The points reached at $t = T$ for all the trials are shown in (a), (c), and (d) for S2, S3, and S4, respectively. The goal for S2 is reached in the sixth trial, for S3 in the ninth, and for S4 in the tenth, with an average of 97, 120 and 106 timesteps, respectively.

reality gap. Notably, they effectively overcome stochasticity despite encountering a range of behavior-altering situations unaccounted for during policy training, including intentional damage incidents and external loads, all without needing to retrain the RL-policy from scratch. The results were compiled for additional goal-points in the 3D space against all damage scenarios. We plan to expand this work to adapt to changes in the soft robot's behavior caused by temporally dependent material properties (wear and tear, aging, etc.) and various external conditions (temperature, pressure, humidity, etc.). We believe that this research will help contribute to making the field of soft robotics more approachable, reliable, and adaptable.

ACKNOWLEDGMENTS

This work was carried out in the BRain-Inspired Robotics Lab (BRAIR Lab), The BioRobotics Institute, Scuola Superiore Sant'Anna in collaboration with the Soft Robotics Lab, National University of Singapore. It is funded by the European Union's Horizon 2020 Research and Innovation Programme within the framework of the project SMART (Soft, Self-responsive Smart Materials for Robots) under Marie Skłodowska-Curie Actions (MSCA), Innovative Training Network (ITN) with grant agreement no. 860108 and the project PROBOSCIS with grant agreement no. 863212.

TABLE II
RESULTS FOR ADDITIONAL GOAL-POINTS AND OBSTACLE LOCATIONS WITH ALL DAMAGE SCENARIOS

Goals	G1: $[+50.0, -100.0, -620.0]$					G2: $[-50.0, -100.0, -625.0]$					G3: $[-45.0, -100.0, -645.0]$				
Obstacle pos	$z = [-700.0, -610.0]$					$z = [-700.0, -610.0]$					$z = [-700.0, -570.0]$				
Parameters Definition	P1: Total no. of trials P2: Timesteps per trial P3: Goal Flag P4: Total Collisions P5: Final distance from goal at the end of trials in $[mm]$														
Algorithm: Bayesian Optimization Assisted Coaching (BOAC)															
Params	S0	S1	S2	S3	S4	S0	S1	S2	S3	S4	S0	S1	S2	S3	S4
P1	8	9	14	–	–	8	14	18	–	–	18	–	–	–	–
P2	21	21	21	–	–	24	24	24	–	–	27	–	–	–	–
P3	True	True	False	–	–	True	True	False	–	–	False	–	–	–	–
P4	0	0	0	–	–	0	0	0	–	–	5	–	–	–	–
P5	4.39	4.93	7.1	–	–	5.2	6.1	9.0	-	-	11.1	–	–	–	–
Algorithm: Gaussian Process-based Recurrent Cerebellar Architecture (GPRCA)															
Params	S0	S1	S2	S3	S4	S0	S1	S2	S3	S4	S0	S1	S2	S3	S4
P1	3	3	5	6	9	3	4	6	8	13	4	5	7	14	–
P2	86	92	89	94	75	80	74	87	87	89	82	89	83	98	–
P3	True	True	True	True	True	True	True	True	True	True	True	True	True	False	–
P4	0	0	0	0	3	0	1	1	2	4	0	0	2	5	–
P5	5.4	4.7	3.5	5.5	4.4	4.2	4.1	4.8	5.4	5.2	2.55	4.68	5.53	10.38	–

TABLE III
AVERAGE TIME AND ACCURACY ACROSS ALL THE EXPERIMENTS (INCLUDING DIFFERENT GOAL-POINTS AND SCENARIOS) WITH THE THREE-MODULE SOFT ARM

Parameter Vs. Approach	RL	BOAC	GPRCA
Offline Training Time $[hrs]$	Approx. 3	-	-
Online Training Time $[sec]$	-	30.9 ± 23.3	57.85 ± 34.4
Final Accuracy $[mm]$	42.6 ± 25.3	6.56 ± 10.5	4.41 ± 1.84

REFERENCES

[1] C. Laschi, T. G. Thuruthel, F. Lida, R. Merzouki, and E. Falotico, "Learning-based control strategies for soft robots: Theory, achievements, and future challenges," *IEEE Control Systems Magazine*, vol. 43, no. 3, pp. 100–113, 2023.

[2] 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.

[3] A. Wilson, A. Fern, and P. Tadepalli, "Incorporating domain models into bayesian optimization for rl," 12 2010, pp. 467–482.

[4] A. Younes and A. Yushchenko, "Toward faster reinforcement learning for robotics applications by using gaussian processes," vol. 2171, 11 2019, p. 190007.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 07 2017.

[6] H. He, H. III, and J. Eisner, "Imitation learning by coaching," *Advances in Neural Information Processing Systems*, vol. 4, pp. 3149–3157, 01 2012.

[7] A. Attia and S. Dayan, "Global overview of imitation learning," 2018. [Online]. Available: https://arxiv.org/abs/1801.06503

[8] M. S. Nazeer, C. Laschi, and E. Falotico, "Soft dagger: Sample-efficient imitation learning for control of soft robots," *Sensors*, vol. 23, no. 19, p. 8278, 2023.

[9] J. Porrill, P. Dean, and J. Stone, "Recurrent cerebellar architecture solves the motor-error problem," *Proceedings. Biological sciences / The Royal Society*, vol. 271, pp. 789–96, 05 2004.

[10] R. K. Katzschmann, M. Thieffry, O. Goury, A. Kruszewski, T.-M. Guerra, C. Duriez, and D. Rus, "Dynamically closed-loop controlled soft robotic arm using a reduced order finite element model with state observer," in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, 2019, pp. 717–724.

[11] T. G. Thuruthel, E. Falotico, M. Manti, and C. Laschi, "Stable open loop control of soft robotic manipulators," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1292–1298, 2018.

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

[13] 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.

[14] R. III and B. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *I. J. Robotic Res.*, vol. 29, pp. 1661–1683, 11 2010.

[15] B. Jones and I. Walker, "Kinematics for multisection continuum robots," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 43–55, 2006.

[16] 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.

[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] L. U. Odhner and A. M. Dollar, "The smooth curvature model: An efficient representation of euler–bernoulli flexures as robot joints," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 761–772, 2012.

[19] D. Cao and R. Tucker, "Nonlinear dynamics of elastic rods using the cosserat theory: Modelling and simulation," *International Journal of Solids and Structures - INT J SOLIDS STRUCT*, vol. 45, pp. 460–477, 01 2008.

[20] D. C. Rucker and R. J. Webster III, "Statics and dynamics of continuum robots with general tendon routing and external loading," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1033–1044, 2011.

[21] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 3982–3987.

[22] T. Bieze, F. Largilliere, A. Kruszewski, Z. Zhang, R. Merzouki, and C. Duriez, "Finite element method-based kinematics and closed-loop control of soft, continuum manipulators," *Soft Robotics*, vol. 5, 04 2018.

[23] G. Mengaldo, F. Renda, S. Brunton, M. Bächer, M. Calisti, C. Duriez, G. Chirikjian, and C. Laschi, "A concise guide to modelling the physics of embodied intelligence in soft robotics," *Nature Reviews Physics*, vol. 4, 08 2022.

[24] T. George Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Learning dynamic models for open loop predictive control of soft robotic manipulators," *Bioinspiration Biomimetics*, vol. 12, 08 2017.

[25] M. T. Gillespie, C. M. Best, E. C. Townsend, D. Wingate, and M. D. Killpack, "Learning nonlinear dynamic models of soft robots for model

predictive control with neural networks," in *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, 2018, pp. 39–45.

[26] A. Centurelli, A. Rizzo, S. Tolu, and E. Falotico, "Open-loop model-free dynamic control of a soft manipulator for tracking tasks," in *2021 20th International Conference on Advanced Robotics (ICAR)*, 2021, pp. 128–133.

[27] G. Chirikjian and J. Burdick, "A modal approach to hyper-redundant manipulator kinematics," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 3, pp. 343–354, 1994.

[28] E. Coevoet, A. Escande, and C. Duriez, "Optimization-based inverse model of soft robots with contact handling," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1413–1419, 2017.

[29] 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.

[30] M. A. Graule, C. B. Teeple, and R. J. Wood, "Contact-implicit trajectory and grasp planning for soft continuum manipulators," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 9401–9408.

[31] A. Abu Alqumsan, S. Khoo, and M. Norton, "Robust control of continuum robots using cosserat rod theory," *Mechanism and Machine Theory*, vol. 131, pp. 48–61, 01 2019.

[32] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, "Neural network and jacobian method for solving the inverse statics of a cable-driven soft arm with nonconstant curvature," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 823–834, 2015.

[33] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control strategies for soft robotic manipulators: A survey," *Soft Robotics*, vol. 5, 01 2018.

[34] T. George Thuruthel, E. Falotico, M. Cianchetti, and C. Laschi, *Learning Global Inverse Kinematics Solutions for a Continuum Robot*, 01 2016, vol. 569, pp. 47–54.

[35] Y. Ansari, M. Manti, E. Falotico, Y. Mollard, M. Cianchetti, and C. Laschi, "Towards the development of a soft manipulator as an assistive robot for personal care of elderly people," *International Journal of Advanced Robotic Systems*, vol. 14, no. 2, p. 1729881416687132, 2017.

[36] Y. Ansari, M. Manti, E. Falotico, M. Cianchetti, and C. Laschi, "Multi-objective 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.

[37] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," 2021. [Online]. Available: https://arxiv.org/abs/2102.03861

[38] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, 03 2018.

[39] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[40] 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. [Online]. Available: https://doi.org/10.1098/rsos.171628

[41] X. Zhang, F. Chan, T. Parthasarathy, and M. Gazzola, "Modeling and simulation of complex dynamic musculoskeletal architectures," *Nature Communications*, vol. 10, no. 1, pp. 1–12, 2019. [Online]. Available: https://doi.org/10.1038/s41467-019-12759-5

[42] 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.

[43] M. A. Graule, C. B. Teeple, T. P. McCarthy, G. R. Kim, R. C. St. Louis, and R. J. Wood, "Somo: Fast and accurate simulations of continuum robots in complex environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3934–3941.

[44] M. A. Graule, T. P. McCarthy, C. B. Teeple, J. Werfel, and R. J. Wood, "Somogym: A toolkit for developing and evaluating controllers and reinforcement learning algorithms for soft robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4071–4078, 2022.

[45] E. Coevoet, T. Morales-Bieze, F. Largilliere, Z. Zhang, M. Thieffry, M. Sanz Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt, and C. Duriez, "Software toolkit for modeling, simulation and control of soft robots," *Advanced Robotics*, vol. 31, pp. 1208–1224, Nov. 2017. [Online]. Available: https://hal.inria.fr/hal-01649355

[46] E. Ménager, P. Schegg, 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*, 2022.

[47] C. Frazelle, J. Rogers, I. Karamouzas, and I. Walker, "Optimizing a continuum manipulator's search policy through model-free reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5564–5571.

[48] R. Morimoto, S. Nishikawa, R. Niiyama, and Y. Kuniyoshi, "Model-free reinforcement learning with ensemble for a soft continuum robot arm," in *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*, 2021, pp. 141–148.

[49] J. Liu, J. Shou, Z. Fu, H. Zhou, R. Xie, J. Zhang, J. Fei, and Y. Zhao, "Efficient reinforcement learning control for continuum robots based on inexplicit prior knowledge," 2020. [Online]. Available: https://arxiv.org/abs/2002.11573

[50] 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)*, 2019, pp. 5133–5139.

[51] P. Oikonomou, A. Dometios, M. Khamassi, and C. S. Tzafestas, "Task driven skill learning in a soft-robotic arm," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1716–1723.

[52] H. T. Kalidindi, T. G. Thuruthel, C. Laschi, and E. Falotico, "Cerebellum-inspired approach for adaptive kinematic control of soft robots," in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, 2019, pp. 684–689.

[53] G. Fang, Y. Tian, Z.-X. Yang, J. M. P. Geraedts, and C. C. L. Wang, "Efficient jacobian-based inverse kinematics with sim-to-real transfer of soft robots by learning," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 6, pp. 5296–5306, 2022.

[54] M. Dubied, M. Y. Michelis, A. Spielberg, and R. K. Katzschmann, "Sim-to-real for soft robots using differentiable fem: Recipes for meshing, damping, and actuation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5015–5022, 2022.

[55] C. Johnson, T. Quackenbush, T. Sorensen, D. Wingate, and M. Killpack, "Using first principles for deep learning and model-based control of soft robots," *Frontiers in Robotics and AI*, vol. 8, 05 2021.

[56] H. T. Kalidindi, T. G. Thuruthel, C. Laschi, and E. Falotico, "Cerebellum-inspired approach for adaptive kinematic control of soft robots," in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2019, pp. 684–689.

[57] I. Koryakovskiy, M. Kudruss, H. Vallery, R. Babuska, and W. Caarls, "Model-plant mismatch compensation using reinforcement learning," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, 01 2018.

[58] M. Manti, A. Pratesi, E. Falotico, M. Cianchetti, and C. Laschi, "Soft assistive robot for personal care of elderly people," in *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2016, pp. 833–838.

[59] N. Zlatintsi, A. Dometios, N. Kardaris, I. Rodomagoulakis, P. Koutras, X. S. Papageorgiou, P. Maragos, C. Tzafestas, P. Vartholomeos, K. Hauer, C. Werner, R. Annicchiarico, M. Lombardi, F. Adriano, T. Asfour, A. Sabatini, C. Laschi, M. Cianchetti, A. Güler, and R. Lopez-Tarazon, "I-support: A robotic platform of an assistive bathing robot for the elderly population," *Robotics and Autonomous Systems*, vol. 126, p. 103451, 02 2020.

[60] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[61] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[62] N. Sammaknejad, Y. Zhao, and B. Huang, "A review of the expectation maximization algorithm in data-driven process identification," *Journal of process control*, vol. 73, pp. 123–136, 2019.

[63] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," in *Advances in Neural Information Processing Systems*, 2018.

Muhammad Sunny Nazeer was born in Pakistan (April 14, 1994). He completed his Bachelor's degree in Aeronautical Engineering (Avionics) at the National University of Sciences and Technology, Pakistan, in 2016. He worked for two years as a Design Engineer (2016 - 2018) in an aviation industry. Then in September 2018, he did a "European Master's in Advanced Robotics Plus (EMARO+)" under the Erasmus Mundus Joint Master's Degree (EMJMD) program. The first year was spent in Warsaw University of Technology (Sep 2018 – Aug 2019) and the second year in Ecole Centrale de Nantes (Sep 2019 – Aug 2020). For his Master's thesis, he joined Team RAINBOW, Inria/IRISA, Rennes and worked on a deep learning-based control system for intuitive and effective control of a team of drones (Feb 2020 – Aug 2020). Currently, he is working as an Early Stage Researcher (ESR) under Marie Skłodowska-Curie Actions (MSCA) SMART- Innovative Training Network on the control and behavior of self-healing soft robots at the BioRobotics Institute of Scuola Superiore Sant'Anna, Italy and Department of Excellence in Robotics and AI, Scuola Superiore Sant'Anna, Pisa, Italy.

Cecilia Laschi is Provost's Chair Professor of robotics at the National University of Singapore, leading the Soft Robotics Lab. She is Co-Director of CARTIN – Centre for Advanced Robotics Technology and Innovation. She is on leave from the Scuola Superiore Sant'Anna, Italy, The BioRobotics Institute. She is best-known for her research in soft robotics, an area that she pioneered and contributed to develop at an international level. She explores marine applications of soft robots and their use in the biomedical field, with a focus on eldercare. She is Editor-in-Chief of Bioinspiration & Biomimetics and Specialty Chief Editor of Soft Robotics in Frontiers in Robotics & AI and Editorial Board member of Science Robotics and IEEE Robotics & Automation Letters. She is an IEEE Fellow and founded the IEEE International Conference on Soft Robotics (RoboSoft). She is Co-Chair of the Gordon Research Conference on Robotics 2024. She co-founded the spin-off company RoboTech.

Egidio Falotico graduated in computer sciences from the University of Pisa, Italy, in 2008 and completed his Ph.D. in Biorobotics at the Scuola Superiore Sant'Anna (SSSA), Pisa, Italy, in 2013, and a Ph.D. in cognitive sciences from the University Pierre et Marie Curie, Paris, France, in March 2013. He is currently an Assistant Professor at The BioRobotics Institute, SSSA. He is the author or co-author of more than 100 international peer-reviewed papers and he regularly serves as a reviewer for more than 10 international ISI journals. He served as PI for SSSA in EU-funded projects (Human Brain Project, Proboscis, Growbot), focusing on the development of brain-inspired algorithms for robot control and on machine learning models for soft robot control. His research interests focus on neurorobotics, i.e., the implementation of brain models from neuroscience in robots.